Minimal Critical Sequences in Model-based Safety and Security Analyses: Commonalities and Differences

Théo Serru, Nga Nguyen, Michel Batteux, Antoine Rauzy

▶ **To cite this version:**

Théo Serru, Nga Nguyen, Michel Batteux, Antoine Rauzy. Minimal Critical Sequences in Model-based Safety and Security Analyses: Commonalities and Differences. ACM Transactions on Cyber-Physical Systems, 2023, 7 (3), pp.17. 10.1145/3593811 . hal-04175993

HAL Id: hal-04175993
https://hal.science/hal-04175993

Submitted on 2 Aug 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Minimal Critical Sequences in Model-Based Safety and Security Analyses: Commonalities and Differences

THÉO SERRU, ETIS laboratory - UMR8051, France and Airbus Protect, France
NGA NGUYEN, Léonard de Vinci Pôle Universitaire, Research Center, France
MICHEL BATTEUX, IRT SystemX, France
ANTOINE RAUZY, Norvegian University of Science and Technology, Norway

Discrete event systems are increasingly used as a modeling tool to assess safety and cybersecurity of complex systems. In both cases, the analysis relies on the extraction of critical sequences. This approach proves to be very powerful. It suffers however from the combinatorial explosion of the number of sequences to look at. To push the limits of what is feasible with reasonable computational resources, extraction algorithms use cutoffs and minimality criteria.

In this article, we review the principles of extraction algorithms and we show that there are important differences between critical sequences extracted in the context of safety analyses and those extracted in the context of cybersecurity analyses. Based on this thorough comparison, we introduce a new cutoff criterion, so-called footprint, that aims at capturing the willfulness of an intruder performing a cyberattack. We illustrate our presentation by means of three case studies, one focused on the analysis of failures and two focused on the analysis of cyberattacks and their effects on safety. We show experimentally the interest of the footprint criterion.

CCS Concepts: • **Security and privacy → Formal security models**.

Additional Key Words and Phrases: Safety, Cybersecurity, Critical Sequences, Model-Based Safety Analyses, Model-Based Security Analyses

ACM Reference Format:
Théo Serru, Nga Nguyen, Michel Batteux, and Antoine Rauzy. 2022. Minimal Critical Sequences in Model-Based Safety and Security Analyses: Commonalities and Differences. 1, 1 (April 2022), 20 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Industrial systems are increasingly embedding software and communication features. As a consequence, safety analyses for these systems must consider not only the effects of mechanical failures, but also those of cyberattacks [44]. Discrete event systems, such as stochastic Petri nets [28], Figaro [7] or AltaRica [2], provide a powerful framework to perform behavioral simulations of systems, both from a safety and from a cybersecurity perspective. The cornerstone of analyses relying on this framework is the generation of sequences of events leading to a critical state (so-called critical sequences). These sequences make it possible to identify the weaknesses of the system under study and give hints on how to improve its architecture to remedy these weaknesses. One of the main limitations of this approach stands however

Authors' addresses: Théo Serru, theo.serru@ensea.fr, ETIS laboratory - UMR8051, Cergy, France, 95000 and Airbus Protect, Blagnac, France, 31700; Nga Nguyen, nga.nguyen@devinci.fr, Léonard de Vinci Pôle Universitaire, Research Center, 92916 Paris La Défense, France; Michel Batteux, Michel.Batteux@irt-systemx.fr, IRT SystemX, Palaiseau, France; Antoine Rauzy, antoine.rauzy@ntnu.no, Norvegian University of Science and Technology, Trondheim, Norway.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
Manuscript submitted to ACM

in the combinatorial explosion of the number of sequences to look at. To tackle this problem, minimality criteria and cutoffs are applied to keep only the most relevant sequences and to reduce the exploration space. The relevant sequences correspond to the ones highlighting the major weaknesses of the system (e.g. the shortest or likeliest sequences) and the behavior of the stakeholders (e.g. behavior of the system when a failure occurs, behavior of the attacker, etc.). Algorithms to extract sequences in discrete event systems share this technique with those used to extract cutsets in fault trees [37]. In both cases, this is the only practical way to apply formal modeling and simulation approaches to industrial-scale systems within feasible amounts of computational resources. This is indeed a heuristic approach that relies on the assumption that the application of cutoffs does not discard important sequences. Consequently, a natural and central question is whether appropriate minimality and cutoffs criteria are the same for safety and for cybersecurity analyses.

In this article, we make a thorough review of commonalities and differences between critical sequences in both domains. A key observation is that critical sequences representing cyberattacks tend to be longer than those representing failures, which worsens the combinatorial explosion problem. However, conversely to safety critical sequences that result, most of the time, of a combination of independent failures whose order does not matter [19, 21]—the sequence $f_3 \rightarrow f_1 \rightarrow f_2$ produces the same effect as the sequence $f_1 \rightarrow f_2 \rightarrow f_3$—cyberattacks are sequences of intentional actions, performed in a well defined order to reach specific (sub-)objectives —e.g. the sequence $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow b_1 \rightarrow b_2$ aims at achieving two objectives a and b, which both need series of actions to be performed in order, namely $a_1 \rightarrow a_2 \rightarrow a_3$ on the one hand, $b_1 \rightarrow b_2$ on the other hand. When extracting automatically critical sequences representing the latter, it is possible to take advantage of this property. We introduce here the notion of footprint that aims at capturing the willfulness of cyberattacks. We show this notion can be used as a powerful cutoff to get rid of sequences obtained by shuffling actions performed to achieve sub-objectives. e.g. $a_1 \rightarrow b_1 \rightarrow a_2 \rightarrow b_2 \rightarrow a_3$.

The above ideas have been implemented in tools extracting critical sequences provided by two integrated modeling environments supporting the AltaRica language (SimfiaNeo [27] and AltaRica Wizard [3]).

To illustrate our discussion, we present in this article three case studies implemented in these environments. The first of these case studies is a “pure” safety analysis. The two others deal with cyberattacks and their effects on safety.

The contribution of this article is thus threefold. First, it reviews the use of discrete event systems to assess the safety related consequences of cyberattacks. Second, it introduces the notion of footprint as a new cutoff criterion to limit the combinatorial explosion of the number of critical sequences to look at. Third, it illustrates the discussion by means of case studies stemmed from industrial practice and shows the interest of footprints to assess these case studies.

The remainder of the article is organized as follows. Section 2 presents the context of this work and puts it in perspective with related works. Section 3 recalls definitions of discrete event systems, executions and sequences. Section 4 introduces formally minimality and cutoff criteria and discusses the application of the latter to both failure and cyberattack sequences. A definition of the footprint criterion is notably given in this section. Section 5 presents an algorithm dedicated to the extraction of minimal sequences with cutoffs, and demonstrates the benefits of using footprint. Section 6 illustrates the proposed approach by means of three case studies. Finally, section 7 concludes the article.

2 CONTEXT OF THIS WORK

2.1 Model-Based Risk Assessment in Industry

The industry must ensure that its products are sufficiently protected against failures, cyberattacks and any other threats potentially harming the system, its users or the environment. Consequently, safety and security risk assessments must be performed prior operating the products. To ease the maintainability and to be less prone to human errors, model-based analyses replace progressively document-centric ones [6]. Model-based safety or security analyses consist essentially of four steps. First, failures or threats that can affect the components of the system are identified. Second, a model is designed that reflects the architecture of the system and how failures and threats propagate. Third, critical scenarios of failures or cyberattacks are extracted from the model and safety and cybersecurity related performance indicators are calculated. Fourth, results of these calculations are analyzed to decide whether the system under study meets its safety or security requirements. Thanks to the extracted scenarios, analysts can identify architecture's weaknesses and implement safety/security countermeasures (e.g. redundancy, backup components, firewall, intrusion detection systems (IDS), etc.). The process is iterated until the system meets requirements. Compliance with requirements does not mean that all critical scenarios must be eliminated. Rather, one must ensure that the most probable scenarios have been evaluated and that the remaining ones have a sufficiently low impact or are very unlikely to occur. The main limitation of this approach stands in the combinatorial explosion of the number of scenarios to look at.

2.2 Related Works

Several engineering disciplines are using formal modeling and simulation and face the combinatorial explosion problem. We shall now review some of these works, which are directly related to the approach we propose in this article.

2.2.1 Fault Trees. Fault Trees are by far the most popular tool to assess the safety of industrial systems, see e.g. [40]. Their use is recommended by most of the safety standards, e.g. IEC 61508 (Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-related Systems), IEC 62279 (railway applications), IEC 61513 (nuclear industry), ISO26262 (automotive industry), ARP4761 and ARP4754 (avionic industry), etc.

From a mathematical standpoint, a fault tree is a set of (stochastic) Boolean equations [37]. The basic events of a fault tree represent individual failures of components of the system, while its top event encodes the combinations of these failures that induce a failure of the system as a whole. Such a combination is called a cutset. A cutset is minimal if none of its proper subsets is a cutset. Basic events are assumed to be statistically independent. Consequently, the order in which they occur does not matter.

The extraction of minimal cutsets plays a central role in fault tree analyses. Minimal cutsets are actually used both for qualitative purposes—as they represent scenarios of failure–and for quantitative purposes—as they can be used to calculate probabilistic risk indicators (probability of failure, importance measures, safety integrity levels…). Reference [37] describes state-of-the-art algorithms to extract minimal cutsets and to calculate these indicators.

In practice, industrial size fault trees may have a huge number of minimal cutsets (over several millions) and indeed an even larger number of non-minimal ones. Attempting to extract all of these minimal cutsets would be pointless for at least three reasons: first, they would not fit in the computer memory, even if advanced encoding techniques such as Minato's ZBBD [33] are used. Second, the analyst would not be able to review them individually. Third, it is, in general, the case that a tiny proportion of them (say a few thousands) concentrates the probability of failure of the system. To put it in another way, most of the failure scenarios can be safely ignored, as their probability is negligible. Consequently, state-of-the-art algorithms do not aim at extracting all the minimal cutsets, but rather look for the most probable ones.

To do so, they apply probabilistic cutoffs during the exploration in order to filter on the fly all cutsets whose probability is lower than a certain threshold (defined by the analyst).

As the present article focuses on safety and cybersecurity, it is worth mentioning attack trees. This method has been proposed in the 1990s to represent cyberattacks graphically [45]. Attack trees are strongly inspired from fault trees. Qualitative analyses are very similar in both cases. The calculation of probabilistic indicators is however slightly different [10]. To the best of our knowledge, attack trees are by no means used as extensively in the context of cybersecurity analyses as fault trees are in the context of safety analyses, probably in reason of the insufficient expressiveness of the formalism for the former context.

The mathematical and algorithmic developments on fault trees and minimal cutsets have nevertheless been a key source of inspiration for our work.

Note that the success of the modeling language AltaRica is partly due to the development of algorithms that compile AltaRica models into fault trees, see e.g. [35]. Note also that there have been some attempts to order basic events of minimal cutsets *a posteriori* [8] so to take into account dependencies.

2.2.2 Model-Checking. Model-checking techniques, see e.g. [1, 12] for reference monographs, rely on the representation of the behavior of the system under study by means of a state automaton (as we do in the present work). Properties of the system, such as the existence of a critical sequence, are then translated into properties of executions of the automaton, themselves expressed in (some) temporal logic.

These techniques face indeed the combinatorial explosion of the number of states and executions of the automaton representing the behavior of the system [11]. To tackle this problem, several powerful techniques have been developed, such as symbolic model-checking [31], bounded model-checking [5] and on-the-fly model-checking [20]. Without entering into too technical details here, these techniques are well suited to test the existence of a critical sequence, possibly to extract such a sequence (if any), but not to extract all of the critical sequences verifying a certain criterion (minimality, cost, probability…). This makes them quite difficult to use in the context of the present article.

On the fly model-checking can be combined with partial order reduction [16] which aims at reducing the search space when the execution order of events does not change the validity of the property to be checked. The notion of footprint introduced in this article relies actually on a similar idea although it implements it differently. It is worth noting that partial order reduction is based on the work of Mazurkiewicz [29] on trace theory. The idea is to quotient the set of executions of the automaton by means of some independence relation on its transitions. It suffices then to consider only one representative for each equivalence class of executions. In a way, we apply a similar idea by considering only minimal sequences, although the implementation is again very different.

To complete our tour of model-checking techniques, we must mention here probabilistic model-checking [25] that aims at introducing probabilities of transitions. The model-checker PRISM [14] implements this approach. Again, the key difference with our own approach is that we want to generate all the critical sequences verifying a certain criterion, and not to check for the existence of such sequence (or to extract one of them).

Note to conclude this section that some authors proposed to use model-checking in the context of safety or security assessment, see e.g. [24, 26, 32], with however the same limitations as discussed above.

2.2.3 Extraction of Critical Sequences. The extraction of critical sequences from discrete event systems has been already explored by a few researchers.

The most noticeable work on this topic is probably the development of the tool FigSeq (Figaro Sequence Generator) [7] that extracts critical sequences from Figaro models. The latter describe implicitly large Markov chains. The objective

of the extraction of critical sequences is actually to avoid generating the underlying Markov chains when assessing probabilities of failure. Cutoffs on length and probabilities of sequences are used to reduce the combinatorial explosion of the number of sequences to look at. The work by Brameret & al. [9] showed, in the context of AltaRica models, that the same result can be achieved (with a greater efficiency) by generating partial Markov chains.

The Figaro environment has been experimentally used to combine safety and cybersecurity assessments of control systems [22], without modifying the existing tools (see also [23]). To the best of authors' knowledge, this work has not been pursued, probably because of the combinatorial explosion issues and because probabilistic cutoffs can hardly be applied in the security context.

Note to conclude this section that integrated modeling environments supporting the AltaRica language, such as SimfiaNeo and AltaRica Wizard, provide tools to extract minimal sequences. These tools have been used so far to support safety analyses, but no article has been published so far on their implementation.

3 DISCRETE EVENT SYSTEMS

Although the ideas developed in the present article have been implemented in integrated modeling environments supporting the AltaRica language [2], they apply in the more general framework of discrete event systems. This is the reason we present them in this framework.

3.1 Definitions

Discrete Event Systems (DES) are formally defined as follows.

Definition 3.1 (Discrete Event Systems). A *discrete event system* is a five-tuple $\langle V, E, T, s_0, CS\rangle$, where:

- V is a finite set of variables. Each variable v of V takes its value in a finite or infinite set of constants, called the *domain* of v and denoted $dom(v)$. Each variable of V describes typically the state of a component of the system. The (global) state of the system is thus described by *valuations* of variables of V (which are members of the Cartesian product of their respective domains).
- E is a finite set of events;
- T is a finite set of transitions, i.e. of triples $\langle e, g, i\rangle$, where:
  - e is an event from E;
  - g is a Boolean condition on variables of V, called the *guard* of the transition;
  - i is an *instruction*, i.e. a mechanism that modifies the current values of variables.

  A transition $\langle e, g, i\rangle$ is *enabled* in the state s, if the valuation s satisfies the guard g, i.e. if $g(s) = true$. *Firing* the (enabled) transition $\langle e, g, i\rangle$ in the state s makes the system pass from the state s to the state $s' = i(s)$, which is denoted $s \xrightarrow{e} s'$.
- s_0 is the *initial state* of the system;
- CS is a Boolean condition on the values of the variables of V representing the critical states.

In practice, we shall consider that variables take their values into a finite set of symbolic constants, which are used to represent:

- The state of a component, e.g. working, failed;
- The presence of flow between two components;
- The access privilege of the attacker on a component, e.g. none, user or root;

- The capacity of the attacker to harm the confidentiality, integrity and availability of an asset (a component or a function critical from a functional or cybersecurity point of view);
- More generally, any information of interest regarding the functioning of the system, a failure or an ongoing cyberattack.

Events are used to represent failures, attacks, and reconfiguration actions. The latter can for instance put the system in a fail-safe mode, isolate a component, or turn on a security measure.

Transitions change the state of the system. Sequences of transitions can thus lead the system from its initial state to a state that is critical with respect to safety or cybersecurity.

The above definition assumes that transitions are deterministic in the following sense.

Definition 3.2 (Determinism). A transition t = $\langle e, g, i \rangle$ is deterministic if, starting from the state s, the transition t labeled with e will always end in the same state. In other words:

$$\forall e, s, s_k, s_j : \ (s \xrightarrow{e} s_k) \ and \ (s \xrightarrow{e} s_j) \Rightarrow (s_k = s_j)$$

Note the above definition of discrete event system is actually close to the notion of guarded transition systems [36] which is the underlying mathematical framework of AltaRica.

3.2 Semantics

Definition 3.3 (Executions). Let $M = \langle V, E, T, s_0, CS \rangle$ be a discrete event system. Then the set of *executions* on M is the smallest set such that:

- s_0 is an execution (in which no event is triggered);
- if $\sigma : s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_n} s_n$ is an execution, $n \geq 0$, and $\langle e, g, i \rangle$ is a transition that is enabled in the state s_n then $\sigma \xrightarrow{e} i(s_n)$ is an execution.

The state s is *reachable* from the initial state s_0 if there exists an execution $s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_n} s$, $n \geq 0$.
An execution $s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_n} s_n$, $n \geq 0$, is *critical* if:

- s_n satisfies CS;
- None of the s_i's, $0 \leq i < n$ satisfies CS.

Then we define the sequence associated with an execution as the word made of the events labeling an execution.

Definition 3.4 (Sequences). Let $M = \langle V, E, T, s_0, CS \rangle$ be a discrete event system and let $\sigma : s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_n} s_n$ be an execution of M. The sequence associated with σ is the word $e_1 \cdots e_n$.

As we assumed that transitions of discrete event systems are deterministic, the sequence associated with an execution characterizes fully this execution. In particular, we can speak about critical sequences.

Note that the reverse is not true: not all of the sequences correspond to executions.

A sequence $e_1 \cdots e_n$, $n \geq 0$, is *valid* if there exists an execution $s_0 \xrightarrow{e_1} s_1 \cdots \xrightarrow{e_n} s_n$. It is *invalid* otherwise.

4 FILTERING CRITICAL SEQUENCES

4.1 Rational

As stated above, discrete event simulation faces the combinatorial explosion of the number of critical sequences. This combinatorial explosion may have several sources that we shall illustrate by means of a series of stylized examples.

First, the number of (reachable) states of the system may grow quickly with the number of variables. As an illustration consider the following example.

Example 4.1. Consider a system with n components that may be working or failed. Assume that the components fail independently one another and that the system is failed if all these components are failed. Then, the number of reachable states is 2^n.

Second, even if the number of reachable states remains tractable, there may be a huge number of critical sequences. As an illustration, consider the following example.

Example 4.2. Consider a system with two components C and D with respectively m and n degradation levels. The degradation c_i (resp. d_i) makes the component C (resp. D) pass from the degradation level $i-1$ to the degradation level i. Initially, a component is at degradation level 0. It is failed when it reaches its maximum degradation level (m for C, n for D). The system is failed when its two components are failed. The number of reachable states of the system is thus simply $(m+1)\times(n+1)$. Now a critical sequence consists of the m degradations of C and the n degradations of D, in any order. It is easy to verify that there are $\binom{m+n}{m} = \binom{m+n}{n}$ such sequences and that this number grows very fast with m and n.

In absence of further information on the system, there is not much one can do against these two sources of combinatorial explosion. Note however that, due to their highly symmetric structures, the above examples could be dealt with by representing them by means of fault trees.

In practice, there are however many cases where it is possible to do something against the combinatorial explosion.

As a first illustration, consider the following example.

Example 4.3. Consider again the system of example 4.2, but assume now that the system is failed if one of its components is failed. There are still a huge number of critical sequences: basically, any sequence with m degradations of C and less than n degradations of D or with n degradations of D and less than m degradations of C is a critical sequence. The number of such sequences grows again very quickly with m and n. However, it is clear that only two critical sequences are of real interest: the sequence consisting of the m successive degradations of C, and the sequence consisting of n successive degradations of D.

It may not be possible to avoid fully to look at the other critical sequences, but the extraction algorithm should present only these two sequences to the analyst. In [38], the author calls this phenomenon *symmetric shuffle*.

There are also cases in which not all of the critical sequences are relevant. As a second illustration, consider the following example.

Example 4.4. A system consisting of n independent components that may fail independently. Assume that the system is failed if 2 out of these n components are failed. The number of critical sequences is thus $n\times(n-1)$. Now assume that the n components are separated into two groups containing respectively 2 and $n-2$ components and that the probability of failure of components of the first group is 10^{-2} while the probability of failure of components of the second group is 10^{-6}. Among the $n\times(n-1)$ critical sequences, there are thus:

- 2 sequences of probability 10^{-4};
- $4(n-2)\approx 4n$ sequences of probability 10^{-8};
- $(n-2)(n-3)\approx n^2$ sequences of probability 10^{-12};

It is clear that for not too big values of n ($n \ll 10^4$), the sequences of the first group concentrate the probability of failure of the system. For practical purposes, the others can thus be safely ignored.

The above stylized examples illustrate the ideas behind minimality and cutoffs: they are essentially means to filter critical sequences so to keep only the relevant ones, thereby reducing the exploration space. We shall now define formally these notions.

4.2 Minimality

The notion of minimal sequence has been defined by several authors [34, 38, 43] with the objective of avoiding the "symmetric shuffle" issue. To define what is a minimal sequence, we must introduce first the notion of subsequence.

Definition 4.5 (Subsequences). A sequence σ is a subsequence in another sequence σ', which is denoted $\sigma \subseteq \sigma'$, if all the events of σ occur in σ' in order.

As an illustration, consider again Example 4.3 for $m = 3$ and $n = 2$. The sequence $c_1c_2c_3$ is a subsequence of, for instance, the sequences $c_1d_1c_2c_3$ and $d_1c_1c_2d_2c_3$. Note that all these sequences are valid. It may be the case that a subsequence of a sequence is not a valid sequence. For instance, the sequence c_1c_3 is an invalid subsequence of the sequence $c_1c_2c_3$.

Minimal critical sequences are defined as follows.

Definition 4.6 (Minimal critical sequences). A critical sequence is minimal if none if its proper subsequences is a critical sequence.

Still in Example 4.3, the only two minimal critical sequences are $c_1 \cdots c_m$ and $d_1 \cdots d_n$.

4.3 Cutoffs for Safety Analyses

In safety analyses, critical sequences are successions of events (failures, repairs, reconfigurations, ...) leading the system to a failed state. Most of the time, failures are assumed to be statistically independent one another [19, 21]. If only such failures are taken into account, the order of events does not matter and one can stay in the realm of Boolean algebra (fault trees, minimal cutsets). If failures are dependent (e.g. common cause or cascading failures [19]) or if repairs and reconfigurations need to be considered, the order of events matters and minimal critical sequences must be extracted rather than minimal cutsets. Taking into account the order of events increase significantly the expressive power of the modeling formalism. It comes however with a significant price, as a much larger state-space needs to be explored, as a cutset of order k gives potentially rise to $k!$ sequences. For this reason, the application of cutoffs to limit the extraction to relevant sequences—i.e. those that involve only few failures and/or whose probability is above a certain threshold—is even more necessary than in the case of minimal cutsets.

The idea is to associate a cost to each event, to set the cost of a sequence as the aggregation of the costs of its events and to put a threshold on the cost of sequences the algorithm looks at. A notion of cost should thus to be taken in a very broad sense. The condition that a cost *Cost* must verify is that:

- It associates events with values taken in a domain D equipped with an order relation $\sqsubseteq$ and an aggregation function $\oplus$;

- It is monotonically increasing, i.e. for all valid sequences we, where w is a, possibly empty, sequence and e is an event, the following condition holds:

$$Cost\,(w) \quad \sqsubseteq \quad Cost\,(we) = Cost\,(w) \oplus Cost\,(e) \tag{1}$$

There are two immediate ways to implement costs.

The first one consists in associating defining costs over the domain $\left(\mathbb{R}^{+}, \leq, +\right)$, i.e. to associate a positive "weight" to each event and to set the weight of a sequence as the sum of the weight of its events. If the weight is set to 1 for all events, the weight of a sequence is simply its length. Alternatively, we may want to associate different weights to events, depending on their categories, e.g. associate 1 to failures, and 0 to other events.

The second one consists in defining costs over the domain $([0, 1], \geq, \times)$, i.e. to associate a probability to each event and to define the probability of a sequence as the product of the probabilities of its events.

Note that it is also possible to combine two or more costs defined as above in order, for instance, to put a threshold on both the number of failures involved in the sequence and a threshold on its probability.

In the context of safety analyses, combining cutoffs of lengths and probabilities of critical sequences is sufficient in the vast majority of cases. The probability of individual events is actually low in general. Consequently, the probability of a sequence decreases quickly with its size, as illustrated by Example 4.4. In practice, sequences involving more that 5 or 6 failures have usually an extremely low probability and can therefore be safely ignored. This makes it also possible to deal to a large extent with situations such as the one of Example 4.3: sequences that interleave events c_i's and d_j's will be quickly discarded because their length is too large or their probability is too low.

Note to conclude this section that, as pointed out by Collet & al. [13], one can often obtain a good approximation of the probability Q of failure of the system by applying the so-called *rare event approximation*:

$$Q \quad \approx \quad \sum_{w \in \Gamma;\, p(w) \geq \tau} p(w) \tag{2}$$

where Γ denotes the set of minimal critical sequences and τ is a conveniently chosen cutoff on the probability of the critical sequences.

4.4 Cutoffs for Cybersecurity Analyses

At a first glance, sequences describing cyberattacks are quite similar to those describing failure scenarios: they are successions of actions from the intruder possibly interleaved with actions representing counter-measures and reconfiguration of the system. There are, however, significant differences between the two types of sequences.

First, the former tend to be much longer than the latter, hence increasing the combinatorial explosion issue.

Second, it turns very hard in practice to associate a sound probability (of success) of an action of the intruder. Discussions about the use of probabilities in cybersecurity can be found here [4, 30]. Consequently, thresholds on lengths and probabilities make much less sense, while a solution based on only formal and sound .

Third and more importantly, actions of the intruder are deliberate. They do not occur at random like mechanical failures. Right on the contrary, they are performed intentionally to reach a certain objective, or sub-objective, in a precise order. We are typically in the situation of Example 4.2 where the attack consists in two independent sub-goals that need to be achieved to make the attack successful.

The above remarks led us to introduce a new type of cutoff, hereafter called *footprint*.

Let $M : \langle V, E, T, s_0, CS \rangle$ be a DES and let we be a valid sequence of M where w is a sequence of M and e is event of E. After the firing of the last transition (the one labeled with e), we can distinguish four disjoint subsets of T:

- T_{00}: the transitions not enabled before the firing of e and still not enabled after;
- T_{01}: the transitions not enabled before the firing of e but enabled after;
- T_{10}: the transitions enabled before the firing of e but not enabled after; and finally,
- T_{11}: the transitions enabled before the firing of e and still enabled after.

By definition, we have $T = T_{00} \cup T_{01} \cup T_{10} \cup T_{11}$.

In a critical sequence extraction process, we are not interested in transitions belonging to T_{00} and T_{10} as they are not enabled after the firing of e.

The transitions of T_{01} are the transitions that are made possible by the firing of e, that are the direct consequence of e. In a way, they represent actions that are connected to the action e, that aim at achieving the same (sub-)goal.

On the contrary, the transitions of T_{11} represent actions that are not related to e. In a way, they represent actions that are not connected to the action e, that aim at achieving another (sub-)goal.

Consequently, we want to prioritize, during critical sequence extraction process, the transitions of T_{01} over those of T_{11}. Hence, the idea of associating with each event a *footprint* F defined as follows:

$$F(e) \quad = \quad \begin{cases} 0 & \text{if (transition labeled with e)} \in T_{01} \\ 1 & \text{if (transition labeled with e)} \in T_{11} \end{cases} \tag{3}$$

The footprint of the sequence is then defined as the sum of the footprints of its events. This cutoff is thus defined on the domain $(\mathbb{N}, \leq, +)$. For the sake of completeness, we define in the initial state s_0, T_{00} and T_{01} as the empty set, T_{10} as the set of transitions not enabled in s_0, and finally T_{11} as the set of transitions enabled in s_0.

In Example 4.2, the footprint of sequences $c_1 \cdots c_m d_1 \cdots d_n$ and $d_1 \cdots d_n c_1 \cdots c_m$ is 2. All other sequences, that interleave c_i's and d_j's have a footprint larger than 2. By setting the threshold to 2, we can thus filter minimal critical sequences so to keep only the relevant ones.

Setting the "right" threshold for the footprint results indeed of a heuristic reasoning. If the analyst sets it too low, e.g. 1 in Example 4.2, some important critical sequences will be missed. On the contrary, if the footprint is set to a too high value, the cutoff will have no effect. The point is that it is always possible to take an iterative approach: first, extract critical sequences with a maximum footprint of 1, then those with a maximum footprint of 2, and so on.

As for the other cutoffs, the objective is to spend as efficiently as possible the "computation budget" we have, by prioritizing the analysis on the attack sequences that are the most likely to happen. By fixing these attacks, e.g. by introducing safeguards, we can improve the robustness of the system. At the end of this process, the attacks that will remain are either too improbable or too difficult to perform.

5 IMPLEMENTATION

We can now describe how the ideas developed in the previous section can be implemented in a concrete algorithm to extract critical sequences.

5.1 Algorithm

Algorithm 1 extracts minimal critical sequences whose cost is lower than a given cutoff c.

Algorithm 1 Algorithm to Extract Minimal Critical Sequences

Input: A discrete event system $\langle V, E, T, s_0, CS\rangle$
Input: A cost $Cost$ and a cutoff c defined on a domain $(D, \sqsubseteq, \oplus)$
Output: Γ the set of minimal critical sequences

```
1:  Begin
2:      σ = s_0
3:      Γ = ∅
4:      ExtractCriticalSequences(⟨V, E, T, s_0, CS⟩, c, σ, Γ)
5:      return Γ
6:  End
7:  ExtractCriticalSequences(⟨V, E, T, s_0, CS⟩, c, s_0 --w--> s_n, Γ)
8:  Begin
9:      if Cost(w) ⊐ c then
10:         return
11:     end if
12:     if CS(s_n) then
13:         if IsMinimal(w) then
14:             Γ ← Γ ∪ {w}
15:         end if
16:         return
17:     end if
18:     for all transitions ⟨e, g, i⟩ of T that are enabled in s_n do
19:         s_{n+1} ← i(s_n)
20:         ExtractCriticalSequences(⟨V, E, T, s_0, CS⟩, c, s_0 --w--> s_n --e--> s_{n+1}, Γ)
21:     end for
22: End
```

It calls the recursive function **ExtractCriticalSequences**. At each call, this function checks whether the cost of the current sequence exceeds the cutoff. If so, the function exits. If not, it checks if the state is critical. If so and the current sequence is minimal, it is stored. If the state is not critical, enabled transitions are fired, one after the other, and the algorithm is called recursively after each of these firings.

The function **IsMinimal** is implemented by checking valid subsequences of the current sequence. The latter is minimal if none of these valid subsequences is critical.

Algorithm 1 makes clear why selecting a suitable cutoff is so important: it makes it possible to reduce drastically the search space by stopping the exploration as soon as the cutoff is exceeded. Algorithm 1 shows also that if considering minimal sequences only reduces the number of sequences that are stored and improves the readability of the results, it does not prune the exploration.

A key property of Algorithm 1 is that it is complete in the following sense.

Property (Completeness of Algorithm 1). *Let $M : \langle V, E, T, s_0, CS\rangle$ be a discrete event system and let $Cost$ be a cost and c be a cutoff defined on a domain $(D, \sqsubseteq, \oplus)$, then Algorithm 1 produces all minimal critical sequences w of M such that $Cost(w) \sqsubseteq c$.*

Proof. By construction, without the condition on costs, Algorithm 1 extracts all minimal critical sequences. It corresponds actually to a depth-first exploration. Now, by the condition defined in equation 1, if it stops the recursive exploration on some sequence w because $Cost(w) \sqsupset c$, then all the sequences wv, that could have been obtained

from w by pursuing the exploration, verify $Cost(wv) \sqsupseteq Cost(w) \sqsupset c$ and must therefore be discarded. It follows that Algorithm 1 discards no sequence whose cost is less than c. □

Algorithms derived from Algorithm 1 have been implemented in SimfiaNeo and AltaRica Wizard. These algorithms implement several cutoffs, including the footprint presented in Section 4.4. The following sections show that this cutoff makes it possible to push the limits of what is feasible within reasonable computation resources, by reducing significantly the size of the exploration space.

5.2 Cutoffs Illustration

To illustrate the reduction of the exploration space, that is made possible by the cutoff on footprints, let us consider the simple network presented in Figure 1.

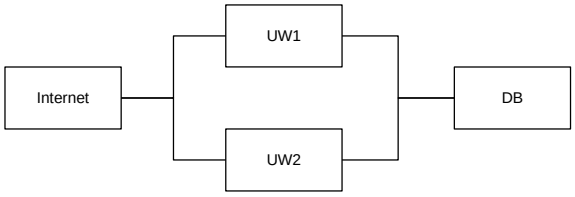

Fig. 1. Simple Network System With Two Workstations and a Database

The network contains an Internet server, two user workstations (UW) and a database server (DB). We consider a few events for each component to ease the representation of the sequences. The events are:

- **Internet**: send malware to UW1 (smal1), send malware to UW2 (smal2);
- **UW1**: privilege escalation (priv1), forward malware to DB (fmal1);
- **UW2**: privilege escalation (priv2), forward malware to DB (fmal2);
- **DB**: malware delete DB (del).

The event leading to the critical state is the deletion of the database by an attacker (triggered by del). We are in the situation of Example 4.3 where the two sequences C and D are:

C = smal1 priv1 fmal1 del

D = smal2 priv2 fmal2 del

In Table 1 we show several sequences that are extracted from the case study and the corresponding footprint and number of events (i.e. the order). To help understanding how is calculated the footprint, transitions taken from the set T_{11} are bolded.

Table 1. Comparison of the Footprint, Order and Minimality for Several Sequences

Sequence	Footprint	Order	Minimal?
smal1 priv1 fmal1 del	1	4	Yes
smal2 priv2 fmal2 del	1	4	Yes
smal1 priv1 **smal2** priv2 **fmal1** del	3	6	No
smal1 priv1 **smal2 fmal1** priv2 **del**	4	6	No

The two last sequences of Table 1 illustrate how two sequences with the same events can have a different footprint. Therefore, setting the right value will keep only the minimal critical sequences with the lowest footprint.

Then, Figure 2 compares a sequence extraction with the algorithm using footprint (left part (a)), and with the algorithm using only the length (right part (b)).

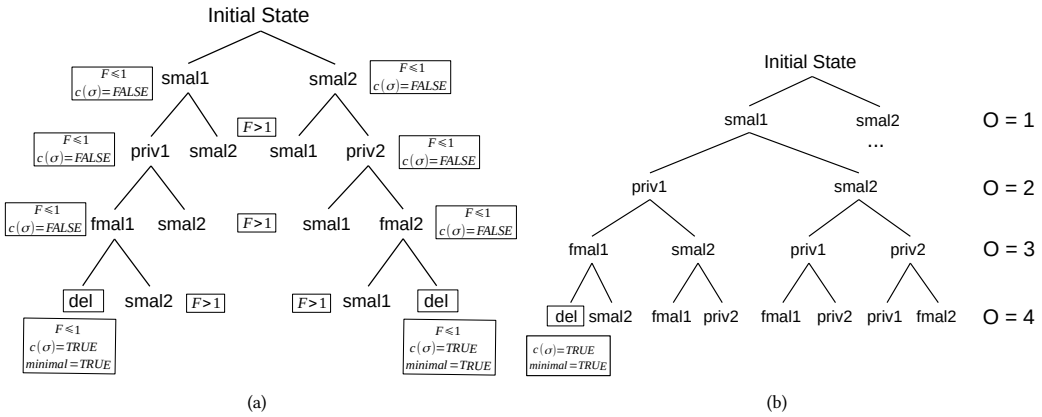

Fig. 2. (a) Execution of Algorithm 1 with the footprint $F \leq 1$ as a cutoff (b) Execution of Algorithm 1 with the order $o \leq 4$ as a cutoff

The left part of the figure (a) illustrates the execution of the algorithm for the network model and with the cutoff set to the footprint equal to one (i.e. $F = 1$). Considering sequences of footprint 1, we can see that the exploration is quickly restricted, i.e. it stops if the action executed is not from the set T_{01}. The right part of the figure (b) shows the exploration with the filter only on the length of the sequence (length ≤ 4). The figure reveals that the algorithm using footprint will explore less sequences than the algorithm using only the length: 8 sequences for the footprint instead of 16 sequences for the length. Furthermore, the second algorithm, using only the length, only tackles the explosion described in Example 4.2, where the former algorithm, using footprint, also handles Example 4.3. With this illustration, we show that, in theory, the length is not a good cutoff for cyberattack sequences contrarily to footprint.

6 EXPERIMENTAL VALIDATION

This section will provides concrete results, justifying theoretical elements for the relevance of minimality and cutoffs, presented in the previous sections. These results are based on three case studies with the SimfiaNeo tool, to show the interest of the footprint criterion. This tool extracts all critical sequences, and applies, during the generation, minimality and cutoffs on the order and footprint.

6.1 High-Integrity Pressure Protection System

The first case study is based on High-Integrity Pressure Protection System (HIPPS). Such systems are classically used for oil and gaz or chemical plants, to prevent the over-pressurization of a line or a vessel. We depict the system in Figure 3. In this system, three pressure transmitters (PT) are sending pressure data to acquisition modules (AM). If two out of three acquisition modules detect an overpressure, they send the information to the logic solver (LS). The logic solver then sends an order to close the two solenoid valves (SV), which are in active redundancy, to activate the two shutdown valves (SDV). Finally, autotest (AT) components are added to detect the failure of the acquisition module

and the logic solver. In this situation, the system would enter a fail-safe mode and close the valves. We consider three different kinds of failures. Failures for the pressure transmitters, the logic solver and the autotest components. Failures on demand for solenoid and shutdown valves Finally, false positive failures for pressure transmitters. The resulting model contains 16 components and 29 events, detailed in Table 2.

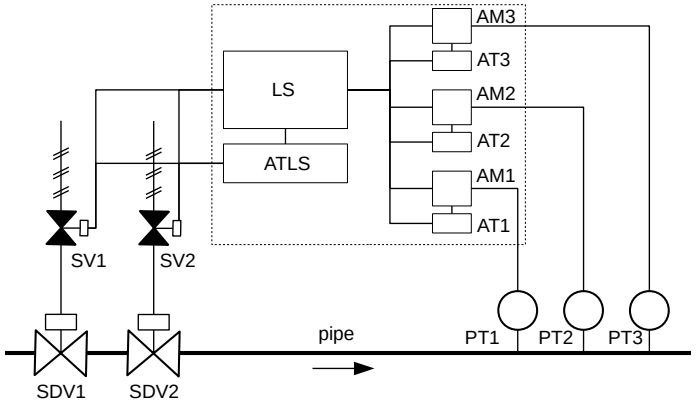

Fig. 3. High-Integrity Pressure Protection System

Table 2. Events of the HIPPS Case Study

Component	**Event**
Pressure Transmitter	Failure, False Negative
PT Acquisition Module	Failure
AutoTest	Failure
Logic Solver	Failure
Solenoid Valve	Open, Close, Opening, Closing, Refuse to open, Refuse to close
Shutdown Valve	Failure on demand
Pipe	Overpressure

The safety critical state of this case study is the propagation of an overpressure in the pipe's output (a vessel or a tank for instance). As most of the failures are independent, the state space is very large, even for such a small system. Therefore, we decide to cutoff the exploration up to the order 8. It resulted in the generation of 287 480 sequences, lowered to 241 minimal critical sequences (MCS). Minimality is not enough to provide exploitable results from the safety assessment. This is the main reason safety assessments are almost always probabilistic. Thus, we extracted failure parameters from guides or used crafted ones (detailed in Table 3) to illustrate the interest of probabilistic studies. The global probability of failure of the system is $5.4004 \times 10^{-8}\ h^{-1}$ and the six most probable sequences have a probability of $9 \times 10^{-9}\ h^{-1}$. It is clear that these six sequences concentrate the probability of failure of the system, as indicated in Example 4.4.

With this system, we show that purely qualitative assessment makes no sense, as there are too many combinations of failures.

Table 3. Failure Parameters for HIPPS Case Study

Component	**PT**		**AM**	**AT**	**LS**	**Valves**
Failure mode	Failure	False negative	Failure	Failure	Failure	Failure On Demand
Probability distribution	exponential	constant	exponential	exponential	exponential	constant
Failure parameter	$1.3 \times 10^{-6}\ h^{-1}$	$1 \times 10^{-5}\ h^{-1}$	$1 \times 10^{-6}\ h^{-1}$	$1 \times 10^{-6}\ h^{-1}$	$16 \times 10^{-6}\ h^{-1}$	3×10^{-2} per demand
Source	[17]	none	none	none	[17]	[18]

6.2 Autonomous Vessel

This second case study is the navigation system of an autonomous vessel introduced in [41]. Here, the analysis aims to generate the sequences of cyberattacks leading to a safety critical state. The architecture of the system is depicted in Figure 4 with the following components: an autonomous navigation system (ANS), a gobal positioning system (GPS), en electronic chart display and information system (ECDIS), an autonomous identification system (AIS), a RADAR, an emitter/receiver satellite (SAT), and a shore control center (SCC).

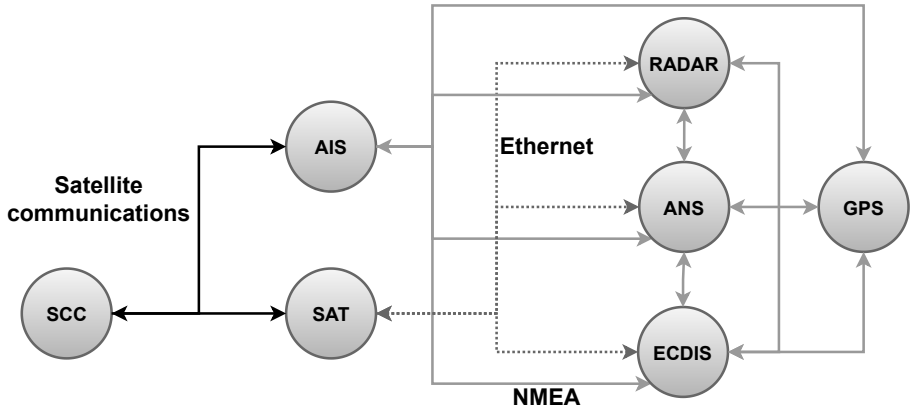

Fig. 4. Architecture of the Autonomous Ship's Navigation System

The safety critical state is the deviation, by the attacker, of the vessel from its route, without the radar activated (i.e. without a collision avoidance system). We modeled the vessel using AltaRica. This model represents the architecture depicted in Figure 4 and the behavior of every components is expressed in terms of DES. Table 4 contains all the cyberattack events that can affect the components.

From this model, we obtain 593 sequences of length 8 to 14, and 25 minimal sequences. Among these sequences, many of them contain the same events in a different order. For instance, Table 5 shows some sequences that are generated for this case study. They contain the same events (defined in Table 4) but in a different order, resulting in a different footprints.

The 3 sequences start with the same set of events: SAT receives an email infected with an infected payload, this payload is transferred to ECDIS, virus is downloaded during the chart update. Then we have two subsequences that are performed in any order; C: tempering of ECDIS maps (c_1), leading to the deviation of the vessel (c_2); D: ECDIS lateral

Table 4. Events of the Autonomous Vessel Case Study

Component	Event
ECDIS	Attacker access ECDIS, attacker exploits Apache vulnerability, attacker tempers with maps, attacker delete maps, download virus during chart update, lateral movement from ECDIS to radar via Ethernet
AIS	Malware deploys on AIS, privilege escalation, lateral movement to ECDIS, attacker send false information to ANS
ANS	Ship deviates from route, loss of incoming signal
SCC	Corrupted SCC send malicious package to AIS
RADAR	Attacker accesses radar, attacker deletes radar targets, radar spoofing, attacker deactivates radar
GPS	GPS jamming, GPS spoofing
SAT	Receive email with infected payload, infected payload transferred to ecdis

Sequence 1	Sequence 2	Sequence 3
SAT.receive_email_with_infected_payload	SAT.receive_email_with_infected_payload	SAT.receive_email_with_infected_payload
SAT.infected_payload_transfered_to_ecdis	SAT.infected_payload_transfered_to_ecdis	SAT.infected_payload_transfered_to_ecdis
ECDIS.download_virus_during_chart_update	ECDIS.download_virus_during_chart_update	ECDIS.download_virus_during_chart_update
ECDIS.attacker_tempers_with_maps	ECDIS.lateral_movement_from_ecdis_to_radar_via_ethernet	ECDIS.lateral_movement_from_ecdis_to_radar_via_ethernet
ANS.ship_deviates_from_route	RADAR.attacker_accesses_radar	RADAR.attacker_accesses_radar
ECDIS.lateral_movement_from_ecdis_to_radar_via_ethernet	RADAR.attacker_deletes_radar_targets	ECDIS.attacker_tempers_with_maps
RADAR.attacker_accesses_radar	ECDIS.attacker_tempers_with_maps	ANS.ship_deviates_from_route
RADAR.attacker_deletes_radar_targets	ANS.ship_deviates_from_route	RADAR.attacker_deletes_radar_targets

Table 5. Example of Sequences of Different Footprint Highlighted by Different Colors

movement to RADAR (d_1), attacker accesses RADAR (d_2), attacker deletes RADAR targets (d_3). We are in the situation described in Example 4.2 with two sub-objectives obtained by executing actions from the sets $C: \ c_1c_2$ and $D: \ d_1d_2d_3$ that lead to the deviation of the vessel without the collision avoidance (radar) activated. The shuffle is illustrated by Sequence 3 in Table 5 where the attacker penetrates the system, downloads the virus, and then executes the two sub-sequences in any order: $d_1d_2c_1c_2d_3$. As presented in Section 4.4 and Table 1, we avoid this shuffle by executing the subsections in order $c_1c_2d_1d_2d_3$ and $d_1d_2d_3c_1c_2$, as in Sequence 1 and 2. Setting a footprint $F \leq 2$ allows to generate these sequences while discarding the shuffle. It also allows to be closer to the reality, as the two sub-sequences (C and D) are taken from a real experiment [46]:

(1) *During an initial probe, Naval Dome sent a virus-laden email over the ship's satellite link to the captain's computer, which is regularly connected to ECDIS for chart updates. During the very next chart update, the virus transferred itself to ECDIS where it immediately installed itself and began to work. Once in place, the virus altered the vessel's position during a night voyage, deceiving the officer of the watch.*
(2) *Naval Dome used the local Ethernet switch interface that connects the radar to ECDIS, the voyage data recorder and bridge alert system to successfully enter the radar workstation. After doing so, Naval Dome succeeded in deleting radar targets from the vessel's bridge radar screen, effectively blindfolding the vessel.*

One can see here that the sequences with the lowest footprint are the ones where the scenarios from [46] are executed one after another, i.e. when they are closer to the attacker's behavior.

By extracting minimal critical sequences of footprint $F \leq 2$ we end-up with 10 sequences.

6.3 Automotive

Our last case study is taken from the European project EVITA [15]. This project aimed at designing an architecture for automotive on-board networks where security-relevant components are protected against tampering, and sensitive data are protected against compromise. The architecture of the system is taken from [39] and depicted in Figure 5.

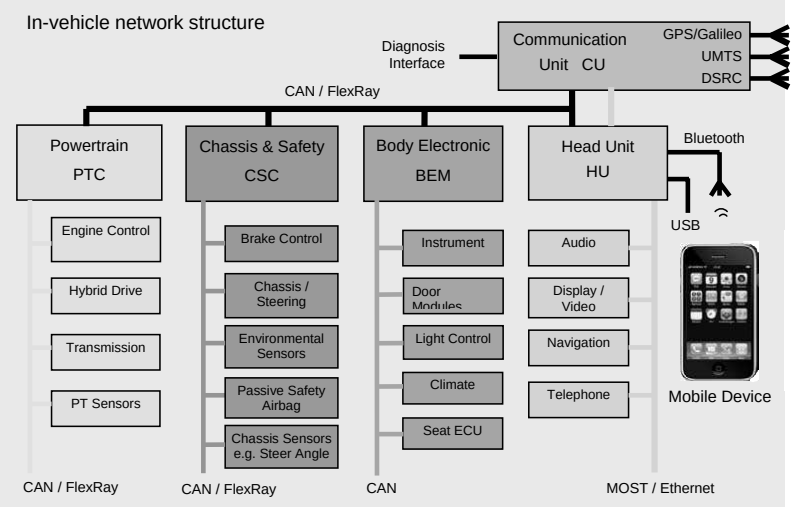

Fig. 5. Evita Use-Case Reference Architecture [39]

In the deliverable D2.3 [39], some cyberattacks scenarios (called "dark side scenarios") have been identified and attack trees (AT) were built. From the scenarios having an effect on the system's safety, we extracted cyberattacks and modeled them in AltaRica. In the model, we remove components not involved in the scenarios, and we add two "security" components to generate more complex outputs. Then, the resulting model contains 14 components, 21 links and 60 events (the model is displayed in [42]).

A first critical state we consider results from the attack on the active brake function. This function activates the brake when an emergency brake message from a neighbor car is received. Here, 401 625 critical sequences are generated in 14 minutes and 35 seconds. Minimality lowered the number of sequences to 116 and the time to 12 minutes and 34 seconds. At last, limiting the footprint $F \leq 2$ gives 72 sequences in less than 1 second.

The second critical state results from the unwanted activation of the brake by an attacker. Here, we get 133 429 critical sequences in 14 minutes 58 seconds, and we get 45 minimal sequences in 11 minutes and 15 seconds. Finally, among these 45 minimal critical sequences, 44 of them have a footprint $F <= 2$. Once again, the computation time reduced to 1 second.

As a conclusion, this case study gives encouraging results. Indeed, we have a great decrease in the state space when considering only minimal sequences and a great reduction in computation time when considering footprint. Then, the fact that only one sequence is avoided with the footprint filter shows that many minimal sequences have a low footprint. It also shows that the sequence we want to obtain in security shall have a low footprint, and that this cutoff is a great opportunity to reduce the state space explosion during computation.

6.4 Assessment results

Table 6 summarizes the state-space reduction we obtain for all the case studies, using SimfiaNeo.

Table 6. Number of Sequences Generated With or Without State Space Reduction for the Three Case Study

Case study	HIPPS	Autonomous Vessel	Evita TE1	Evita TE2
Critical sequences	287480 (order 8)	593 (order 15)	401625 (order 10)	133429 (order 10)
Computation time	2min48sec	1sec	14min35sec	14min58sec
MCS	241	25	116	45
Computation time	47s	1sec	12min34sec	11min15sec
MCS with cutoffs	6 with $P \geq 10^{-9}/h$	10 with $F \leq 2$	72 with $F \leq 2$	44 with $F \leq 2$
Computation time	N/A	instantaneous	<1sec	1sec

The benefits of having these numerical results are twofold. First, it highlights the state-space explosion and shows that the generated sequences cannot be exploited as it is (they are too numerous). In Section 4, we give a heuristic justification of the relevance of using the footprint, and proved that the filter is sound, along with minimality and cutoff for failure sequence. Then, Table 6 demonstrates the significant reduction of state-space and sequences generated when using cutoff and minimality. An analyst shall safely exploit these results and update the architecture of the system to mitigate them before launching another generation. Finally, the table also shows that it is relevant and effective to use a different cutoff for failures and cyberattacks.

7 CONCLUSION AND FUTURE WORKS

In this article, we reviewed the main commonalities and differences between the generation of attack and failure scenarios in safety and security assessments. First, we introduced Discrete Event Systems (DES), as a common formalism allowing to model Cyber-Physical Systems (CPS) and their behavior in case of failures or cyberattacks. A CPS modeled with DES would allow to generate sequences of events leading to a safety critical state. With this definition, we analyzed the commonalities and, more importantly, the fundamental difference between the occurrence of failure and cyberattack events. Indeed, critical sequences representing cyberattacks are more likely to contain dependent events (occurring in a defined order) and to be longer than those representing failures. Conversely, critical sequences representing failures of systems tend to be shorter and independent (events occurring in a random order). This awareness allowed to define an effective strategy to reduce the state-space when exploring and printing sequences of failures and cyberattacks. Minimality and cutoffs are the principal tools to lower this complexity. Therefore, aside from classical cutoffs used in safety assessments (e.g. order and probabilities) we introduced the notion of footprint in the attack sequences. Footprint captures the willfulness of the attack and shall be used in extraction algorithms. We provide proofs of the relevance and soundness of this cutoff and propose an algorithm to generate the sequences with minimality and cutoffs. This is illustrated by three case studies, one focused on the analysis of failures and two on the analysis of cyberattacks and their effects on safety. Relevance of the footprint criterion is emphasized by the reduction in computation time allowed by its use — from several minutes to a second.

In the future, we wish to pursue this work in three different ways. The first one is to challenge the tools, with the implemented algorithm on a more complex industrial case study, with more components, attacks and reconfiguration. The second one is to evaluate the use of trace theory to reduce the state-space. As we deal with sequence of (sometimes independent) events we could use the notion of equivalence class to take off some of the combinatorial. The third one is

to consider the impact of cyberattacks on the system's functions, to compare different strategies allocating functions to components.

ACKNOWLEDGMENT

This research project is funded by CY Initiative d'Excellence and Airbus Protect. We would specially like to thank Laurent Sagaspe, Raphael Blaize and Emmanuel Arbaratier for their support to this work.

REFERENCES

[1] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model-Checking.* MIT Press, Cambridge, MA, USA.

[2] Michel Batteux, Tatiana Prosvirnova, and Antoine Rauzy. 2019. AltaRica 3.0 in 10 Modeling Patterns. *International Journal of Critical Computer-Based Systems* 9, 1–2 (2019), 133–165. https://doi.org/10.1504/IJCCBS.2019.098809

[3] Michel Batteux, Tatiana Prosvirnova, and Antoine Rauzy. 2022. A Guided Tour of AltaRica Wizard, the AltaRica 3.0 Integrated Modeling Environment. In *32nd European Safety and Reliability Conference (ESREL 2022) (Proceedings of the 32nd European Safety and Reliability Conference (ESREL 2022))*, Maria Chiara Leva, Edoardo Patelli, Luca Podofillini, and Simon Wilson (Eds.). Dublin, Ireland, 2246–2253. https://www.rpsonline.com.sg/proceedings/esrel2022/html/S09-09-308.xml

[4] Steven M. Bellovin. 2006. On the Brittleness of Software and the Infeasibility of Security Metrics. *IEEE Security & Privacy* 4, 4 (July 2006), 96–96. https://doi.org/10.1109/MSP.2006.101 Conference Name: IEEE Security & Privacy.

[5] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. 2003. Bounded Model Checking. In *Advances in Computers.* Vol. 58. Academic Press, Waltham, MA, USA, 117–148. https://www.cs.cmu.edu/~emc/papers/Books%20and%20Edited%20Volumes/Bounded%20Model%20Checking.pdf

[6] Benjamin S. Blanchard and Wolter J. Fabrycky. 2008. *Systems Engineering and Analysis.* Pearson, Upper Saddle River, NJ 07456, USA.

[7] Marc Bouissou and Yannick Lefebvre. 2002. A path-based algorithm to evaluate asymptotic unavailability for large Markov models. In *Proceedings of the Reliability and Maintainability Symposium.* Seattle, WA, USA, 32–39. https://doi.org/10.1109/RAMS.2002.981616

[8] Marco Bozzano and Adolfo Villafiorita. 2003. *Integrating Fault Tree Analysis with Event Ordering Information.* Technical Report. CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA. 8 pages. https://es-static.fbk.eu/tools/FSAP/dissemination/papers/esrel-irst03.pdf

[9] Pierre-Antoine Brameret, Antoine Rauzy, and Jean-Marc Roussel. 2015. Automated generation of partial Markov chain from high level descriptions. *Reliability Engineering & System Safety* 139 (July 2015), 179–187. https://doi.org/10.1016/j.ress.2015.02.009

[10] Carlos E. Budde, Christina Kolb, and Mariëlle Stoelinga. 2021. Attack Trees vs. Fault Trees: Two Sides of the Same Coin from Different Currencies. In *Quantitative Evaluation of Systems (Lecture Notes in Computer Science)*, Alessandro Abate and Andrea Marin (Eds.). Springer International Publishing, Cham, 457–467. https://doi.org/10.1007/978-3-030-85172-9_24

[11] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2001. Progress on the State Explosion Problem in Model Checking. In *Informatics: 10 Years Back, 10 Years Ahead*, Reinhard Wilhelm (Ed.). Springer, Berlin, Heidelberg, 176–194. https://doi.org/10.1007/3-540-44577-3_12

[12] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. 2018. *Model Checking (second edition).* MIT Press, Cambridge, MA, USA.

[13] Jérôme Collet. 1996. Some remarks on rare-event approximation. *IEEE Transactions on Reliability* 45, 1 (March 1996), 106–108. https://doi.org/10.1109/24.488924

[14] Department of Computer Science, University of Oxford. 2022. PRISM - Probabilistic Symbolic Model Checker. http://www.prismmodelchecker.org/

[15] EVITA Project. 2011. EVITA: E-safety vehicle intrusion protected applications. https://www.evita-project.org/

[16] Patrice Godefroid. 1996. *Partial-Order Methods for the Verification of Concurrent Systems.* Springer Berlin, Heidelberg, Berlin, Germany. 143 pages. https://link.springer.com/book/10.1007/3-540-60761-7

[17] Stein Hauge and Tor Onshus. 2010. *Reliability data for safety instrumented systems.* Sintef, Trondheim, Norway.

[18] UK Health and Safety Executive. 2017. Failure Rate and Event Data for use within Risk Assessments. https://www.hse.gov.uk/landuseplanning/failure-rates.pdf

[19] Hiromitsu Kumamoto and Ernest J. Henley. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists.* Wiley-IEEE Press, Piscataway, NJ, USA. https://ieeexplore.ieee.org/book/5264399

[20] Gerard J. Holzmann. 2003. *The SPIN Model Checker: Primer and Reference Manual.* Addison Wesley, Boston, MA 02116, USA.

[21] John J. Andrews and Bob Moss. 2002. *Reliability and Risk Assessment, 2nd Edition | Wiley* (2nd edition ed.). Portland, OR, USA. https://www.wiley.com/en-ie/Reliability+and+Risk+Assessment%2C+2nd+Edition-p-9781860582905

[22] Siwar Kriaa, Marc Bouissou, and Youssef Laarouchi. 2015. A new safety and security risk analysis framework for industrial control systems. *Journal of Risk and Reliability* 233, 2 (2015), 151–174. https://doi.org/10.1177/1748006X18765885

[23] Siwar Kriaa, Ludovic Pietre-Cambacedes, Marc Bouissou, and Yoran Halgand. 2015. A survey of approaches combining safety and security for industrial control systems. *Reliability Engineering and System Safety* 139 (2015), 156–178. https://doi.org/10.1016/j.ress.2015.02.008

[24] Orna Kupferman and Moshe Y. Vardi. 2001. Model Checking of Safety Properties. *Formal Methods in System Design* 19, 3 (Nov. 2001), 291–314. https://doi.org/10.1023/A:1011254632723

[25] Marta Kwiatkowska, Gethin Norman, and David Parker. 2018. Probabilistic Model Checking: Advances and Applications. In *Formal System Verification: State-of the-Art and Future Trends*, Rolf Drechsler (Ed.). Springer, Cham, 73–121. https://doi.org/10.1007/978-3-319-57685-5_3

[26] Timo Latvala. 2003. Efficient Model Checking of Safety Properties. In *Model Checking Software*, Thomas Ball and Sriram K. Rajamani (Eds.), Vol. 2648. Springer, Berlin, Germany, 74–88. https://doi.org/10.1007/3-540-44829-2_5

[27] Mathilde Machin, Laurent Sagaspe, and Xavier de Bossoreille. 2018. SimfiaNeo, Complex Systems, yet Simple Safety. In *Embeded Real Time Software and System conference*. Toulouse, France, 4. https://www.erts2018.org/uploads/program/ERTS_2018_paper_9.pdf

[28] Marco Ajmone Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana A. Franceschinis. 1998. Modelling with Generalized Stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review* 26, 2 (Aug. 1998), 2. https://doi.org/10.1145/288197.581193

[29] Antoni Mazurkiewicz. 1996. Introduction to Trace Theory. In *The Book of Traces*. WORLD SCIENTIFIC, 3–41. https://doi.org/10.1142/9789814261456_0001

[30] John McHugh. 2006. Quality of protection: measuring the unmeasurable?. In *Proceedings of the 2nd ACM workshop on Quality of protection*. ACM, Alexandria Virginia USA, 1–2. https://doi.org/10.1145/1179494.1179495

[31] Kenneth L. McMillan. 1993. *Symbolic Model Checking*. Kluwer Academic Publisher, New York, NY, USA.

[32] Saoussen Mili, Nga Nguyen, and Rachid Chelouah. 2019. Transformation-based Approach to Security Verification for Cyber-Physical Systems. *IEEE Systems Journal* 13 (December 2019), 3989 – 4000. Issue 4. https://doi.org/10.1109/JSYST.2019.2923818

[33] Shin-Ichi Minato. 1993. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In *Proceedings of the 30th ACM/IEEE Design Automation Conference, DAC'93*. IEEE, Dallas, Texas, USA, 272–277. https://doi.org/10.1145/157485.164890

[34] Pierre-Yves Piriou, Jean-Marc Faure, and Jean-Jacques Lesage. 2016. A formal definition of Minimal Cut Sequences for dynamic, repairable and reconfigurable systems. In *2016 European Safety and Reliability Conference (ESREL 2016)*. Glasgow, United Kingdom, 9. https://hal.archives-ouvertes.fr/hal-01325898/file/ESREL2016_PiriouFaureLesage_V3.pdf

[35] Tatiana Prosvirnova and Antoine Rauzy. 2015. Automated generation of Minimal Cutsets from AltaRica 3.0 models. *International Journal of Critical Computer-Based Systems* 6, 1 (2015), 50–79. https://doi.org/10.1504/IJCCBS.2015.068852

[36] Antoine Rauzy. 2008. Guarded Transition Systems: a new States/Events Formalism for Reliability Studies. *Journal of Risk and Reliability* 222, 4 (2008), 495–505. https://doi.org/10.1243/1748006XJRR177

[37] Antoine Rauzy. 2020. *Probabilistic Safety Analysis with XFTA*. AltaRica Association. http://www.altarica-association.org/members/arauzy/Publications/pdf/Rauzy2020-XFTABook.pdf

[38] Antoine B. Rauzy. 2011. Sequence Algebra, Sequence Decision Diagrams and Dynamic Fault Trees. *Reliability Engineering & System Safety* 96, 7 (July 2011), 785–792. https://doi.org/10.1016/j.ress.2011.02.005

[39] Alastair Ruddle, David Ward, Benjamin Weyl, Muhammad Sabir Idrees, Yves Roudier, Michael Friedewald, Timo Leimbach, Andreas Fuchs, Sigi Gurgens, Olaf Henniger, Rieke Roland, Matthias Ritscher, Henrik Broberg, Ludovic Apvrille, Renaud Pacalet, and Gabriel Pedroza. 2010. *Security requirements for automotive on-board networks based on dark-side scenarios, Deliverable D2.3*. Contract EVITA. Telecom ParisTech. https://hal.telecom-paris.fr/hal-02286288

[40] Enno Ruijters and Mariëlle Stoelinga. 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review* 15-16 (Feb. 2015), 29–62. https://doi.org/10.1016/j.cosrev.2015.03.001

[41] Théo Serru, Nga Nguyen, Michel Batteux, and Antoine Rauzy. 2023. Modeling Cyberattack Propagation and Impacts on Cyber-Physical System Safety: An Experiment. *Electronics* 12, 1 (2023). https://doi.org/10.3390/electronics12010077

[42] Théo Serru, Nga Nguyen, Michel Batteux, Antoine Rauzy, Raphael Blaize, Laurent Sagaspe, and Emmanuel Arbaretier. 2022. Generation of Cyberattacks Leading to Safety Top Event Using AltaRica: an Automotive Case Study. In *Congrès Lambda Mu 23 " Innovations et maîtrise des risques pour un avenir durable " - 23e Congrès de Maîtrise des Risques et de Sûreté de Fonctionnement, Institut pour la Maîtrise des Risques*. Paris-Saclay, France, 8. https://hal.archives-ouvertes.fr/hal-03814648

[43] Zhihua Tang and Joanne B. Dugan. 2004. Minimal cut set/sequence generation for dynamic fault trees. In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*. Los Angeles, CA, USA, 207–213. https://doi.org/10.1109/RAMS.2004.1285449

[44] Jean-Marc Thiriet and Stéphane Mocanu. 2018. Some Considerations on Dependability Issues and Cyber-Security of Cyber-Physical Systems. In *The 7 th IEEE International Conference on Smart Communications in Network Technologies (SACONET'18)*. El Oued, Algeria, 6. https://hal.archives-ouvertes.fr/hal-01909025

[45] J. D. Weiss. 1991. A System Security Engineering Process.. In *Proceedings of the 14th National Computer Security Conference*, Vol. 249. 572–581.

[46] Martyn Wingrove. 2018. 'Impregnable' radar breached in simulated cyber attack. https://www.rivieramm.com/news-content-hub/news-content-hub/impregnable-radar-breached-in-simulated-cyber-attack-25158.