Towards an unified definition of Minimal Cut Sequences

Pierre-Yves Chaux, Jean-Marc Roussel, Jean-Jacques Lesage, Gilles Deleuze, Marc Bouissou

Towards a unified definition of Minimal Cut Sequences

Pierre-Yves Chaux * **Jean-Marc Roussel** *
Jean-Jacques Lesage * **Gilles Deleuze** ** **Marc Bouissou** **

* *Automated Production Research Laboratory (LURPA), ENS Cachan, Cachan, FRANCE (e-mail: {FirstName.LastName}@ lurpa.ens-cachan.fr).*
** *Électricité De France, Recherche et Dévelopement (EDF R&D), Clamart, FRANCE (e-mail: {FirstName.LastName}@ edf.fr).*

Abstract: The growing complexity of systems increases the complexity of reliability modelling and implies the need to model both the dynamics of systems and the reparability of components. The qualitative reliability analyses aim at finding the minimal representation of all the scenarios of failures and repairs of the components, leading to the system failure. This minimal representation is the set of Minimal Cut Sequences. In order to provide a formal definition of these specific scenarios, whatever the risk analysis model used, this paper proposes coherence rules for dynamic and repairable systems whose dysfunctional scenarios are modelled by a finite automaton.

Keywords: Minimal Cut Sequences, Dynamic Systems, Repairable Components, Qualitative studies, Coherence Rules, Finite Automaton.

1. INTRODUCTION

Qualitative reliability analyses of dynamic systems have been a challenge for many years, even if efficient models allowing representing systems where the order of component failure occurrences have an effect on the system failure were proposed. Especially the size of studied systems have made necessary to define a minimal representation of all scenarios that lead to the system failure rather than expressing all of them. This minimal representation is done through what have been called *Minimal Cut Sequences (MCS)*, translating the minimal cuts from static systems to dynamic ones. Obtaining MCS for dynamic systems composed of repairable components is a major challenge because of mainly two difficulties. The first one is that reliability models give an implicit description of the sequences of fault and repair events that are possible, rather than expressing them explicitly. The second one is that these models describe an infinity of sequences, since modeled systems have repairable components.

In order to perform qualitative reliability analyses of binary dynamic and repairable systems, whatever the initial reliability model used, a unified formal definition of Minimal Cut sequences is proposed in this paper, as well as a way to compute them. Since this definition needs to be independent of the model, we have choosen to use Finite Automata (FA) to represent all scenarios that may lead to system failure. In order to limit the qualitative analysis to the Minimal Cut Sequences whithout loss of exhaustivity, it must be proven that they are necessary and sufficient to represent the whole set of scenarios that lead to the system failure. To do that, this paper proposes a transposition of the coherence rules, initially defined for static systems, for dynamic and repairable systems. Afterward, these rules are used as a way to generate larger sets of dysfunctionnal sequences from a shorter one leading to a formal definition of the Minimal Cut Sequences set.

2. PROBLEM STATEMENT

2.1 Binary, dynamic and repairable systems

The following study focuses on *binary dynamic and repairable systems*. A system is considered as binary if its failure can be modeled by a Boolean variable. A dynamic system is a system where the order of the component failures has an impact on the system failure (by opposition to static systems); a repairable system is composed of at least one repairable component.

2.2 Necessity of a formal definition for Minimal Cut sequences

A *Cut* was initially defined for binary coherent static systems in Birnbaum et al. (1961) as a set of failed components that leads to the system failure. The structure function of such systems has been represented by a Boolean function which can be modeled using fault trees (Fussell (1976)). A *minimal cut* has then been defined as a prime implicant (McCluskey Jr (1956)) of this structure function (Rauzy (2001)) on coherent systems.

Dynamic Fault Trees (DFT) (Dugan et al. (1992)) is one of the models which extend fault trees to perform the modeling and analysis of the reliability of dynamic systems. As the coherence rules were not defined for dynamic systems, several semi-formal definitions, mostly adapted from the minimal cuts definition, were given

for *Minimal Cut Sequences*. (Tang and Dugan (2004)) defined MCS as minimal cuts whose component failures have been ordered. In (Merle et al. (2011)) an algebraic framework based on the Boolean algebra associated with temporal operators is proposed to express a dynamic structure function which can be simplified using associated theorems. Some differences can be observed on results when both definitions are applied on the same system such as the one presented in (Boudali and Dugan (2005)). While the first one allows the computation of a super-set of the MCS set, some of which being not possible in the model, the second one only gives an algebraic representation of MCS without enumerating them. Since both definitions are based on the non-repairable component hypothesis, they cannot be applied for dynamic repairable systems. Another definition have been given in (Walker et al. (2007)), where a MCS is a reduced element from the Base Temporal Form, computed from a Qualitative Fault Tree.

The *Boolean logic Driven Markov Processes (BDMP)* formalism (Bouissou and Bon (2003)) is a fault tree based model which allows the representation of the reliability of dynamic repairable systems while modelling reconfigurations at the level of components and/or sub-systems. (Bouissou (2006)) proposed a first definition using FA of minimal cut sequences where *a MCS is considered as a sequence leading to the system failure where no sub-sequence leading to the system failure can be found*. Since we have shown in (Chaux et al. (2012)) that some of the MCS cannot be found by the previous algorithms, this paper proposes a new MCS definition and a way to compute them exhaustively.

In an informal way, the Minimal Cut Sequences can be defined as the minimal set of sequences of minimal length that are necessary and sufficient to describe the whole set of cut sequences.

2.3 Formal Framework

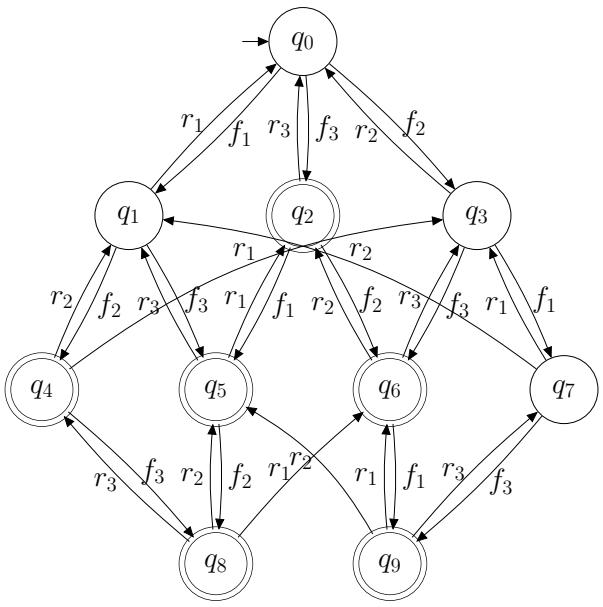

Fig. 1. Example of a FA model of a system

The structure function of a reliability model can be described as the relation between the scenarios of failure and repair events described by the model and the resulting state of the system (faulty/working). In this study, we assume that a structure function of a dynamic repairable system can always be represented by a finite automaton such as the one presented in fig.1. This hypothesis is for example validated for the DFT model using a failure automaton (Coppit et al. (2000)), for the BDMP model (Chaux et al. (2011)) or for Altarica models using the accessibility graph (Rauzy (2002)).

This system is composed of 3 components $\{1, 2, 3\}$, the system is faulty if 3 is faulty or if the last fault of 1 has occurred before the last fault of 2 and if both components are faulty. Each component of the system can only be faulty or operational and is associated to only one repair and one fault event. In the initial state of the system, all components are operational.

This work focusses on the minimal deterministic automaton (as defined in Hopcroft (2008)) representing the structure function of a dynamic repairable system (Chaux et al. (2012)). This automaton can be defined by the 5-tuple $< \Sigma, Q, q_0, Q_M, \delta >$, where :

- Σ is the finite alphabet of repair and failure events,
- Q is the finite set of states of the automaton,
- q_0 is the single initial state,
- Q_M is the set of marked states,
- δ is the structure function.

The marked states represents the states where the system is faulty. As this automaton is minimal, there is only one initial state, and no equivalent states (Hopcroft (2008)). Furthermore, each transition of the transition function δ between two distinct states is labelled with only one event of failure or repair. $\delta(q_i, u) = q_j$ represents that the q_j state is reached from the q_i state when the event u occurs; $\delta(q_i, \sigma) = q_k$ represents that q_k is the state reached from q_i at the end of the sequence σ.

The automaton of fig.1 describes two languages. The *dysfunctional language* ($\mathcal{L}_D$) that is generated by the automaton, is the language composed of all *dysfunctional sequences*. $\mathcal{L}_D$ contains all possible scenarios in the system, that might, or not, lead to the system failure. The *failure language* ($\mathcal{L}_F$) is the language marked by the automaton, composed of all *failure sequences*, containing all scenarios leading to the system failure. An additional language can be defined as the set of *cut sequences* ($\mathcal{L}_{CS}$). A cut sequence is defined in this work as a failure sequence that leads to a marked state of the FA while going through only non-marked states. $\mathcal{L}_{CS}$ is composed of all sequences leading to the first system failure. The relation between those languages is:

$$\mathcal{L}_{CS} \subset \mathcal{L}_F \subset \mathcal{L}_D \tag{1}$$

In order to identify which components are faulty at the end of a sequence σ, a $FC(\sigma)$ (Failed Components) function is also defined.

$$FC : \mathcal{L}_D \rightarrow C$$
$$FC(\sigma) = \left\{ c_i \in C | Proj(\sigma/\Sigma_{c_i})_{|Proj(\sigma/\Sigma_{c_i})|} \in \Sigma_{c_i,F} \right\} \tag{2}$$

The equation looks at the projections of the sequence σ onto each of the component alphabets ($\Sigma_{c_i} = \{f_i, r_i\}$). If the last event of the projection $(Proj(\sigma/\Sigma_{c_i})_{|Proj(\sigma/\Sigma_{c_i})|})$

is a failure event then the component is faulty at the end of the sequence σ. If there is no event from the component or if the last event is a repair event then the component is operational in the last state reached by σ. For example if we consider the sequence $f_1 f_2 r_1$, the projected sequence on the component 1 alphabet is $f_1 r_1$, since the last event is a repair, component 1 is operational. Since the last event that occurs on component 2 is a failure, 2 is the only failed component in the last state reached by σ, what is denoted as $FC(f_1 f_2 r_1) = \{2\}$.

3. COHERENCE OF DYNAMIC REPAIRABLE SYSTEMS

Our strategy for determining the minimal description of the failure language, i.e. the set of minimal cut sequences, is the same as the one that has been used for static systems by using the concept of coherence. A transposition of the coherence rules for dynamic and repairable systems is given in the following.

The coherence of a system has initially been defined for static systems (Birnbaum et al. (1961)) as a relation between sets of faulty components and the resulting system fault. This relation is composed of two concepts. First the semi-coherence of a system is intuitively describing that, if a system is working (failed) with a given set of faulty components, it has to remain operational (failed) with one less (more) faulty component. Furthermore the coherence of a semi-coherent system expresses that a system must be faulty (working) if all (none) components are faulty.

In the field of qualitative analysis of static systems, the coherence of a system is used for the computation of a minimal representation of the system failure, i.e. the cut sets (Rauzy (2001)). In the following, we define the paradigm of coherence for dynamic and repairable systems whose structure function is represented by a finite automaton as stated in section 2.3.

3.1 Semi-coherence and coherence of a dynamic system

The failure of a dynamic system depends on both the set of faulty components at the end of a given sequence of the dysfunctional language and the order in which these components have failed. Therefore semi-coherence needs to be defined as a relation between the growth of a sequence length, the set of faulty components at the end of the considered sequence and the preservation of the system state (faulty or working).

The main goal of this study being qualitative reliability analysis, we will only consider how a failure sequence can be modified to make its length larger while allowing the preservation of the system failure.

3.2 System failure preservation by adding a single event

Let us consider a failure sequence $\sigma \in \mathcal{L}_F$.The first way to increase the length of σ is to add an event into this sequence. The obtained sequence is denoted σ'. This sequence is described in the model (and thus possible) if it belongs to the dysfunctional language $\mathcal{L}_D$. The resulting set of faulty components $FC(\sigma')$ at the end of σ' may be larger or smaller than the original set of faulty components $FC(\sigma)$ (but it cannot be equal since the added event is either a repair or a failure event from one of the components). *Under the hypothesis that a failure event of a component cannot prevent the system failure*, only the case where the new set $FC(\sigma')$ has one more faulty component than $FC(\sigma)$ ($FC(\sigma') \supset FC(\sigma)$) preserves the system failure.This property that must be verified for all failure sequences in order for the system to be considered as semi-coherent is formalized by proposition 3.

$$\forall \sigma \in \mathcal{L}_F, \forall (\sigma_1, \sigma_2) \in (\Sigma^*)^2 | \sigma = \sigma_1 \sigma_2, \forall u \in \Sigma_F : \left. \begin{array}{l} \sigma' = \sigma_1 u \sigma_2 \\ \sigma' \in \mathcal{L}_D \\ FC(\sigma') \supset FC(\sigma) \end{array} \right\} \Rightarrow \sigma' \in \mathcal{L}_F \quad (3)$$

This property can be illustrated by considering the failure sequence $f_1 f_2$ from the fig.1 and by adding the f_3 event into this sequence. Since the system is considered as being coherent, all obtained sequences must be failure sequences; this is verified and illustrated by fig.2.

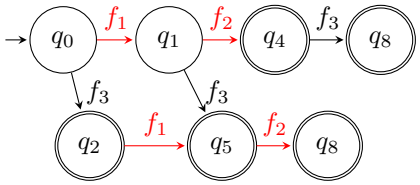

Fig. 2. Adding f_3 into the failure sequence $f_1 f_2$

In the same way it is possible to envisage the addition of an event inside a non-failure dysfunctional sequence. If the added event reduces the set of faulty components at the end of the sequence (repair event), the resulting sequences are also non-failure dysfunctional sequences.

3.3 System failure preservation by an ordered distribution of events

A second way to increase the length of a given faulty sequence σ while keeping an identical set of failed components is to "shuffle" it with another sequence δ. Only the possible resulting dysfunctional sequences σ' are considered. Such a shuffle can be expressed by proposition 4, where $\sigma' \supseteq \sigma$ expresses that σ is a sub-sequence of σ'.

$$\sigma \in \mathcal{L}_D, \delta \in \Sigma^*, \left\{ \begin{array}{l} \sigma' \in \mathcal{L}_D \\ |\sigma'| = |\sigma| + |\delta| \\ \sigma' \supseteq \sigma \\ \sigma' \supseteq \delta \end{array} \right. \quad (4)$$

Since both sequences σ and σ' have the same set of failed components ($FC(\sigma') = FC(\sigma)$) the objective is to define how a sequence δ can be shuffled with σ in order to still have a faulty (or working) system at the end of the newly built sequences σ'. This conservation of the system state (faulty or working) between two sequences σ and σ' is expressed by proposition 5.

$$(\sigma, \sigma') \in (\mathcal{L}_D)^2, \left\{ \begin{array}{l} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_F \end{array} \right. \quad (5)$$

The first step, for shuffling a given sequence δ with the considered σ sequence, is to identify where the events from δ can be inserted inside σ without effect on the state of the

system at the end of the sequence or on the set of failed components. This case can be expressed by proposition 6.

$$\begin{array}{l}\forall\sigma \in \mathcal{L}_D, (\sigma_1,\sigma_2,\sigma_3) \in (\Sigma^*)^3 | \sigma = \sigma_1\sigma_2\sigma_3 \\ \forall(\sigma_{21},\sigma_{22}) \in (\Sigma^*)^2 | \sigma_2 = \sigma_{21}\sigma_{22}, \\ \forall\delta \in \Sigma^* \forall\sigma' \in \{\sigma_1\sigma_{21}\delta^*\sigma_{22}\sigma_3\}, \\ \sigma' \in \mathcal{L}_D \Rightarrow \begin{cases} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_F \end{cases}\end{array} \quad (6)$$

The subword σ_2 of the σ sequence, where the inclusions of events from δ will not have any effect, is then formally expressed as the subword of σ where δ can be inserted any number of times without having any effect on any of the obtained sequences σ'. This can be illustrated on the automaton from fig.1 considering $\sigma = f_1 f_2$ and $\delta = f_3 r_3$.

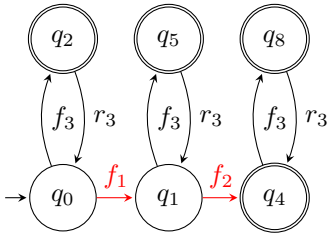

Fig. 3. Adding the $f_3 r_3$ sequence into the $f_1 f_2$ sequence .

As shown on fig.3, $f_3 r_3$ can be inserted any number of times anywhere inside the sequence σ. Therefore the subword σ_2 where the shuffle with δ will not have any effect is: $\sigma_2 = \sigma, \sigma_1 = \epsilon, \sigma_3 = \epsilon$.

After a subword σ_2 has been identified for a given couple (σ, δ), the semi-coherence condition is that any dysfunctional sequence σ' obtained by shuffling δ inside σ_2 must allow to conserve the state of failure of the system at the end of the σ' sequence. If σ leads to a system failure (respectively a working system state), then all σ' must lead to a system failure (respectively an working system state). The relation between the couple (σ, δ) and the shuffled sequences σ' is expressed by the proposition 7.

$$\begin{array}{l}\sigma = \sigma_1\sigma_2\sigma_3 \in \mathcal{L}_D, \delta \in \Sigma^*, \\ \forall\sigma' \in \mathcal{L}_D, \begin{cases} \sigma' = \sigma_1\sigma'_2\sigma_3 \\ |\sigma'_2| = |\sigma_2| + |\delta| \\ \sigma'_2 \supseteq \sigma_2 \\ \sigma'_2 \supseteq \delta \end{cases} \\ \Rightarrow \begin{cases} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \Rightarrow \sigma' \in \mathcal{L}_F \end{cases}\end{array} \quad (7)$$

On the example from fig.3 where $f_3 r_3$ is to be shuffled with $f_1 f_2$, all possible resulting sequences are shown on fig.4. Since the system from fig.1 is considered as being coherent, all the last states reached by the σ' sequences must be marked, what is verified in this case.

Using the two previously described methods to increase the length of a failure sequence, it is possible to consider the shuffle with a sequence δ that increases the set of faulty components if it is decomposed into several additions of event or shuffling of sequences.

3.4 Functional modelling of semi-coherence

In the previous sections we described how the coherence for dynamic repairable systems can be used as a way

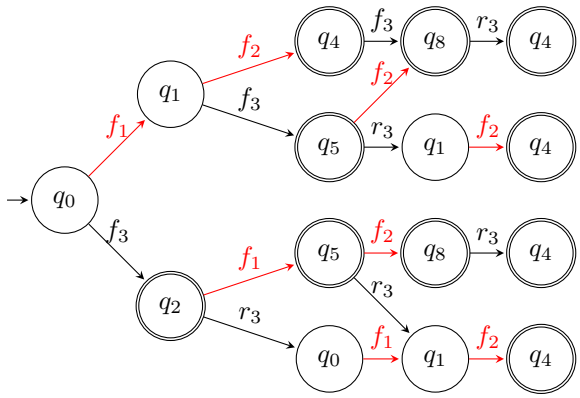

Fig. 4. Generated sequences by ordered distribution of the $f_3 r_3$ sequence into the $f_1 f_2$ sequence.

to generate a new set of failure sequences from a given failure sequence, by adding failure events or by shuffling the considered sequence with another one.

In order to express this generation, the coherence rule is seen as a function. This function, denoted $CR(S)$ allows from a set a failure sequences to generate a larger set of failure sequences by any number of event additions or sequences shuffling. The function can be expressed by the proposition 8.

$$\begin{array}{l} CR : \mathcal{L}_F \xrightarrow{3,6,7} \mathcal{L}_F \\ \quad : S \xrightarrow{3,6,7} S', (S, S') \in (\mathcal{L}_F)^2, S' \supseteq S \end{array} \quad (8)$$

Since any sequence σ from the initial set S can be shuffled with $\delta = \epsilon$, then all sequences from S are also in the S' set. The larger set which can be used as an initial set of sequences is the whole $\mathcal{L}_F$ language, which is also its own image by the CR function.

4. KERNEL OF THE FAILURE LANGUAGE, MINIMAL CUT SEQUENCES SET

In this section, the concepts of *Kernel of the failure language* and of *Language of Minimal cut sequences* are defined. They are finite and minimal representations of respectively the failure language $\mathcal{L}_F$ and of the cut sequences $\mathcal{L}_{CS}$, which are both of infinite size for repairable systems.

4.1 Kernel of the failure language: definition and properties

We define the *Kernel of the failure language* ($Kern(\mathcal{L}_F)$) as the minimal set of failure sequences that are necessary and sufficient to generate the whole failure language using the coherence property defined in proposition 8. A formal definition of $Kern(\mathcal{L}_F)$ can therefore be given as:

$$\begin{cases} Kern(\mathcal{L}_F) \xrightarrow{CR} \mathcal{L}_F, \text{ with } Kern(\mathcal{L}_F) \subseteq \mathcal{L}_F \\ \nexists S \subset Kern(\mathcal{L}_F), S \xrightarrow{CR} \mathcal{L}_F \end{cases} \quad (9)$$

This definition is composed of two parts. The first one expresses that the kernel is a subset of the failure language from which the whole failure language can be generated using the coherence property. The second one expresses that the kernel is the minimal set that allows such a generation.

Several theorems, that will be useful for the computation of the Kernel, can now be given.

Theorem 1. Every sequence of the kernel can only be generated by itself.

$$\begin{array}{l}\forall \sigma \in Kern(\mathcal{L}_F), \nexists \sigma' \neq \sigma \\ \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S\end{array} \tag{10}$$

Proof. Let σ be a sequence of the kernel $Kern(\mathcal{L}_F)$ which can be generated using another shorter sequence σ':

$$\exists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S$$

It is then possible to remove σ from $Kern(\mathcal{L}_F)$ and still be able to generate the full failure language $\mathcal{L}_F$:

$$Kern(\mathcal{L}_F) \setminus \{\sigma\} \xrightarrow{CR} \mathcal{L}_F$$

In this case, the kernel do not comply with the minimality property expressed in its definition (proposition 9).

Theorem 2. The kernel of a given failure language $Kern(\mathcal{L}_F)$ is unique.

Proof. Let $Kern_1(\mathcal{L}_F)$ and $Kern_2(\mathcal{L}_F)$ be two distinct kernels as defined by (9). Let us consider two disjointed kernels ($Kern_1(\mathcal{L}_F) \cap Kern_2(\mathcal{L}_F) = \emptyset$). Let σ be a sequence belonging to only one kernel ($\sigma \in Kern_1(\mathcal{L}_F), \sigma \notin Kern_2(\mathcal{L}_F)$). If $\sigma \notin Kern_2(\mathcal{L}_F)$ then $\exists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S$. If $\sigma \in Kern_1(\mathcal{L}_F)$ then $\nexists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S$. These two propositions are contradictory then the two kernels cannot be disjoint $Kern_1(\mathcal{L}_F) \cap Kern_2(\mathcal{L}_F) \neq \emptyset$. Moreover, as $\sigma \notin Kern_2(\mathcal{L}_F)$ then $\exists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S$, then by definition σ which can be generated cannot be in $Kern_1(\mathcal{L}_F)$ since a shorter sequence can generate it (theorem 1) then $\sigma \notin Kern_1(\mathcal{L}_F)$. If no sequence can be in only one of the kernels then $Kern_1(\mathcal{L}_F) = Kern_2(\mathcal{L}_F)$.

Theorem 3. No sequence from $Kern(\mathcal{L}_F)$ can go through a state of the automaton twice.

Proof. Let σ be a sequence from $Kern(\mathcal{L}_F)$ which goes through one state twice. Let $\sigma = \sigma_1\sigma_2\sigma_3$ with $\sigma_2 \neq \epsilon$ be a decomposition of σ such that the state reached by σ_1 be the same state as the one reached by the $\sigma_1\sigma_2$ sequence. In this case, σ can be generated using the shorter sequence $\sigma_1\sigma_3$ as σ_2 can be inserted inside σ after σ_1 and before σ_3 any number of times without changing $FC(\sigma)$ or the resulting system failure. Since σ can be generated by the shorter sequence $\sigma_1\sigma_3$, it cannot be an element of the kernel $Kern(\mathcal{L}_F)$ (theorem 1).

Theorem 4. $Kern(\mathcal{L}_F)$ is a finite language.

Proof. Accordingly to theorem 3, no sequence σ of the kernel can go through the same state twice. All σ sequences are sequences of the language generated by a finite automaton. Since a finite automaton is composed of a finite number of states, the maximum length of a σ sequence of the kernel is $|\sigma| \leq (Card(Q) - 1)$ where Q is the set of states of the automaton.

4.2 Minimal cut sequences set: definition and properties

Cut sequences are the sequences of the failure language ($\mathcal{L}_F$) whose only the last reached state is marked (representing the system failure). The set of all these sequences is called the Cut sequences language $\mathcal{L}_{CS}$. The minimal representation of $\mathcal{L}_{CS}$ is called the language of minimal cut sequences $\mathcal{L}_{MCS}$. A minimal cut sequence σ is then defined as a sequence belonging to $\mathcal{L}_{MCS}$ and can be defined by the following proposition:

$$\begin{cases}\mathcal{L}_{MCS} \xrightarrow{CR} S, S \supseteq \mathcal{L}_{CS} \text{ with } S \subseteq \mathcal{L}_F \text{ and } \mathcal{L}_{MCS} \subseteq \mathcal{L}_{CS} \\ \nexists S' \subset \mathcal{L}_{MCS} \xrightarrow{CR} S, S \supseteq \mathcal{L}_{CS} \text{ with } S \subseteq \mathcal{L}_F\end{cases} \tag{11}$$

Like for the definition of the kernel, the MCS language definition is composed of two parts. The first one expresses that it is possible to generate at least all cut sequences from the MCS, and the second part expresses the minimality of $\mathcal{L}_{MCS}$. Since the generation of longer sequences using a cut sequence with the coherence function is not limited to cut sequences, the minimal cut sequences language cannot be defined as the set that allows the exact generation of the cut sequences language.

In order to benefit from the proven theorems on the kernel of the failure language, an additional theorem is needed:

Theorem 5. Any sequence from the Minimal cut sequences language $\mathcal{L}_{MCS}$ is a sequence of the kernel $Kern(\mathcal{L}_F)$.

$$\sigma \in \mathcal{L}_{MCS} \implies \sigma \in Kern(\mathcal{L}_F) \tag{12}$$

Proof. Let $\sigma \in \mathcal{L}_{MCS}$ be a MCS which is not an element of the kernel ($\sigma \notin Kern(\mathcal{L}_F)$). If $\sigma \notin Kern(\mathcal{L}_F)$ then $\exists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_{CS}, \{\sigma'\} \xrightarrow{CR} S, \sigma \in S$, since it can be generated by the cut sequence σ', $\mathcal{L}_{MCS} \setminus \{\sigma\}$ can be used to generate the whole cut sequence language $\mathcal{L}_{CS}$. In this case the initial $\mathcal{L}_{MCS}$ does not comply with the minimality criteria expressed by proposition 11.

Therefore the minimal cut sequences language is a subset of the kernel:

$$\mathcal{L}_{MCS} \subseteq Kern(\mathcal{L}_F) \tag{13}$$

Since $\mathcal{L}_{MCS}$ is also a subset of $\mathcal{L}_{CS}$ a second definition of the minimal cut sequences language can be given as:

$$\mathcal{L}_{MCS} = \mathcal{L}_{CS} \cap Kern(\mathcal{L}_F) \tag{14}$$

As $\mathcal{L}_{MCS}$ is a subset of a finite language, $\mathcal{L}_{MCS}$ is also finite. Since it can be defined as the intersection of two unique languages, $\mathcal{L}_{MCS}$ is also unique.

Summarizing, for a coherent dynamic repairable system, a unique and finite set of failure sequences can represent all cut sequences. This set of sequences, defined as the **Minimal Cut Sequences**, is the most compact explicit description of cut sequences that are necessary and sufficient to represent all sequences that lead to the first system failure.

5. MCS COMPUTATION, PRACTICAL RESULTS

The goal of this section is to give a brief description on how the minimal cut sequences language can be computed from an automaton. Seeing that a sequence can be generated using different ways using the CR function, CR is non-injective. For example $f_1f_3f_2r_1$ can be generated by adding f_2 in $f_1f_3r_1$ or by shuffling f_1r_1 with the failure sequence f_3f_2. CR being not injective, CR is not bijective, and consequently the inverse function CR^{-1} that could allow the direct computation of the MCS using the

whole cut sequences language, does not exist. That is the reason why we propose a computation process that aims at reducing a starting set of sequences to the MCS language by removing the sequences which can be computed using the CR function. This computation process is composed of 3 main steps:

(1) First, a finite set of failure sequences that may belong to the MCS language is generated by exploring the automaton. This set contains all non-looped sequences (theorem 3) whose only the last state reached is marked (CS definition).
(2) A second finite set of failure sequences that may be used to compute a sequence from the first set using the CR function is generated. This set contains all failure sequences whose size is strictly lower than the longest sequence obtained in the first step.
(3) Every sequence σ from the first set is compared to every shorter sequence σ' from the second set to evaluate if it can be generated using σ' with the CR function. If the generation is possible, σ cannot be an element of the MCS language and is therefore removed from the first set.

At the end of the process only the sequences that are elements of the MCS language are left in the first set of generated sequences. The worst case complexity of this algorithm is $\mathcal{O}(q!q^q)$ if there is one transition between each different states and if the longest non looped sequence is of length q. On the general case, the algorithm complexity is far less.

On the example of fig.1 only the sequences f_3 and f_1f_2 are identified as Minimal Cut Sequences among the 161 non-looped sequences included into this automaton $\mathcal{L}_{MCS} = \{f_3, f_1f_2\}$. In this case, the kernel (which can be computed in a similar way) is only composed of MCS $Kern(\mathcal{L}_F) = \mathcal{L}_{MCS} = \{f_3, f_1f_2\}$, but in the general case, those two sets of sequences are different.

6. CONCLUSIONS

In this paper we proposed a formal definition of the minimal cut sequences. Considering that all scenarios of failure and repair events describing the safety of a binary dynamic and repairable system can always be modeled by a finite automaton, this definition is based on the languages theory. In order to define the minimal cut sequences, the concept of coherence for dynamic and repairable systems has been proposed. Since the coherence provides a way to compute a set of failure sequences starting from a shorter failure sequence, two minimal representations have been expressed. While the kernel provides a finite and unique representation of the infinite language of sequences that lead to the system failure, the minimal cut sequences language is defined as a restriction of the kernel to the only sequences that are necessary to characterize at least the first system failure.

REFERENCES

Birnbaum, Z., Esary, J., and Saunders, S. (1961). Multi-component systems and structures and their reliability. *Technometrics*, 3(1), 55–77.

Boudali, H. and Dugan, J. (2005). A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering and System Safety*, 87(3), 337–349.

Bouissou, M. (2006). Détermination efficace de scenarii minimaux de défaillance pour des systèmes séquentiels. In *15ème colloque de fiabilité et maintenabilité, Lille (France)*.

Bouissou, M. and Bon, J. (2003). A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes. *Reliability Engineering and System Safety*, 82(2), 149–163.

Chaux, P.Y., Roussel, J.M., Lesage, J.J., Deleuze, G., and Bouissou, M. (2011). Qualitative analysis of a bdmp by finite automaton. In *Advances in Safety Reliability and risk management*, 2055–2057. Taylor & Francis Ed.

Chaux, P.Y., Roussel, J.M., Lesage, J.J., Deleuze, G., and Bouissou, M. (2012). Systematic extraction of Minimal Cut Sequences from a BDMP model. In *Proc. of the 21th European Safety & Reliability Conf. (ESREL'12), Helsinki (Finland)*.

Coppit, D., Sullivan, K., and Dugan, J. (2000). Formal semantics of models for computational engineering: a casestudy on dynamic fault trees. In *11th International Symposium on Software Reliability Engineering, 2000. ISSRE 2000. San Jose (USA)*, 270–282.

Dugan, J., Bavuso, S., and Boyd, M. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3), 363–377.

Fussell, J. (1976). Fault tree analysis: concepts and techniques. *Generic Techniques in System Reliability Assessment*.

Hopcroft, J.E. (2008). *Introduction to Automata Theory, Languages, and Computation, 3/E*. Pearson Education India.

McCluskey Jr, E. (1956). Minimization of boolean functions. *Bell System Technical Journal*, 35, 1417–1444.

Merle, G., Roussel, J.M., and Lesage, J.J. (2011). Algebraic Determination of the Structure Function of Dynamic Fault Trees. *Reliability Engineering and System Safety*, 96(2), 267–277.

Rauzy, A. (2001). Mathematical foundations of minimal cutsets. *IEEE Transactions on Reliability*, 50(4), 389–396.

Rauzy, A. (2002). Mode automata and their compilation into fault trees. *Reliability Engineering and System Safety*, 78(1), 1–12.

Tang, Z. and Dugan, J. (2004). Minimal cut set/Sequence generation for dynamic fault trees. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS Los Angeles (USA)*, 207–213.

Walker, M., Bottaci, L., and Papadopoulos, Y. (2007). Compositional temporal fault tree analysis. In *Computer Safety, Reliability, and Security: 26th International Conference, SAFECOMP 2007, Nurmberg, Germany, September 18-21, 2007, Proceedings*, volume 4680, 106. Springer.