Qualitative analysis of a BDMP by finite automata

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

▶ **To cite this version:**

Pierre-Yves Chaux, Jean-Marc Roussel, Jean-Jacques Lesage, Gilles Deleuze, Marc Bouissou. Qualitative analysis of a BDMP by finite automata. 20th European Safety & Reliability Conf. (ESREL'11), Sep 2011, Troyes, France. pp. 2055-2057. hal-00782748

HAL Id: hal-00782748
https://hal.science/hal-00782748

Submitted on 30 Jan 2013

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.

Qualitative analysis of a BDMP by Finite Automaton

Pierre-Yves Chaux [1,2] & Jean-Marc Roussel [1] & Jean-Jacques Lesage [1]
& Gilles Deleuze [2] & Marc Bouissou [2]

[1] *LURPA - 61, av. du président Wilson 94235 Cachan cedex, France*
{*pierre-yves.chaux, jean-jacques.lesage, jean-marc.roussel*}*@lurpa.ens-cachan.fr*

[2] *EDF R&D - 1, av. du Général de Gaulle 92140 Clamart, France*
{*pierre-yves.chaux, marc.bouissou, gilles.deleuze*}*@edf.fr*

ABSTRACT: The Boolean Driven Markov Processes (BDMPs) were developed by EDF to conduct predictive risk modelling and assessment on critical systems. A BDMP model is a description of the combinations of failures that makes a system to fail like it is done with Fault tree models. The calculation of all the sequences of events that conduct to the global failure is complex as they are implicitly represented by a BDMP that generates a large state space. This paper provides a semantic analysis of BDMP models conducted within the languages and finite automaton theories in order to develop a systematic way to obtain an extensive representation of all scenarios contained in a BDMP. This representation is obtained by using an algorithm given in this paper which constructs an equivalent finite automaton of a BDMP. This FA representation leads to the possibility of extracting all the minimal scenarios of failure implicitly described by a BDMP.

1 INTRODUCTION

Many studies which have been carried out on predictive risk modelling and assessment target two complementary objectives (Henley and Kumamoto 1981). On the one hand, quantitative analyses aim at calculting the failure rate of the modelled system or of one of its subsystems. On the other hand the qualitative analyses aim at determining the scenarios which lead to the failure of the whole system. While those scenarios are in most cases only constituted by basic components failures, they may also include repairs of these components.

The Boolean Driven Markov Processes (BDMP) were created to include all the Electricité De France expertise in the construction and analysis of reliability models. While staying close to Static Fault Tree models (Stamatelatos, Vesely, Dugan, Fragola, Minarick, and Railsback 2002), BDMPs extend the fault tree capacities by allowing to model both the failures and repairs of basic components. This extension also enables the description of the redundancies between components and between complex sub-systems constituted by a number of components, which can be redundant one from another.

The main goal of this study is to conduct a qualitative analysis of a BDMP while setting aside all its capacities to model and conduct quantitative analysis on the reliability of a system. The qualitative analysis consists in enumerating all the sequences of repair and failure events that are implicitly described by a BDMP. To explicitly describe those combinations, this study is conducted within the languages and finite automata (FA) theories. For that, each scenario of failures and repairs is translated into a sequence of events. The set constituted by those sequences (or words) is a language.

Because the language generated by a BDMP is regular, it can be represented by a FA. This paper describes an algorithm which is used to construct the FA "equivalent" to a BDMP in the away that this FA generates the same language as the one which is implicitly described in the BDMP. The qualitative studies can now be conducted on the FA rather than on the BDMP itself. This allows us to gain all the benefits of handling a formal model on which many results were published.

The section 2 will present a short example on the safety modelling of a system using the BDMP in order to describe its modelling capacities. In section 3, we will describe how the scenarios that are implicitly describes by a BDMP can represented by the generated language of a finite automaton. After proposing a semantical analysis of the BDMP model in section 4,

the generation mechanism of this automaton will be described in section 5.

2 MODELLING A SYSTEM SAFETY USING A BDMP

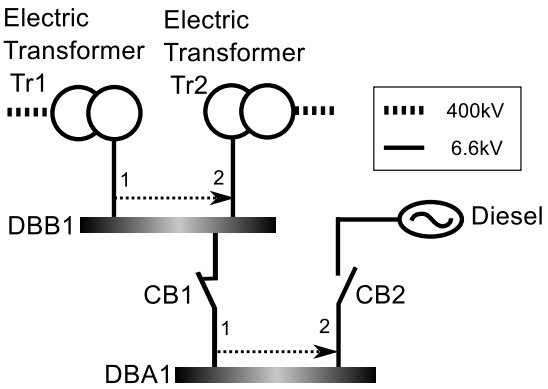

Figure 1: The power supply of electric board used for reactor safety functions LHA

In order to illustrate our works and describe how the scenarios conducting to the global failure are described in a BDMP model, the example of Figure 1 has been chosen. The main function of the system is to provide the power supply to the components which support the cooling and the control functions of a nuclear reactor core. The distribution board DBA1 is one of the two distribution boards that achieve this function. The modelling of the failure scenarios of the system combines several difficulties that are present in most reliability assessment studies.

- All the components can be repaired.
- Two cold redundancies have to be modelled, one between the two possible supplies for the DBA1 board (DBB1 and the diesel generator) and another between the supplies of the DBB1 board (electric transformers 1 and 2). If DBB1 can't be supplied by transformer Tr1 then transformer Tr2 is supplying DBB1. If DBA1 can't be supplied by DBB1 through the circuit breaker CB1 then the diesel generator is started and the circuit breaker CB2 is closed.
- As most components can only fail during their operation (active mode), the diesel generator can also fail while in stand-by (dormant mode).

BDMPs were developed by M. Bouissou to offer to the engineers a formalism to model complex failure mechanisms while using a user-friendly environment (Bouissou and Bon 2003). Based on the Fault Tree model (Stamatelatos, Vesely, Dugan, Fragola, Minarick, and Railsback 2002), the BDMP offer the same capacities to model redundancies as the Spare gates from the Dynamic Fault tree (Dugan, Bavuso, and Boyd 1992). But it also expend their capacities:

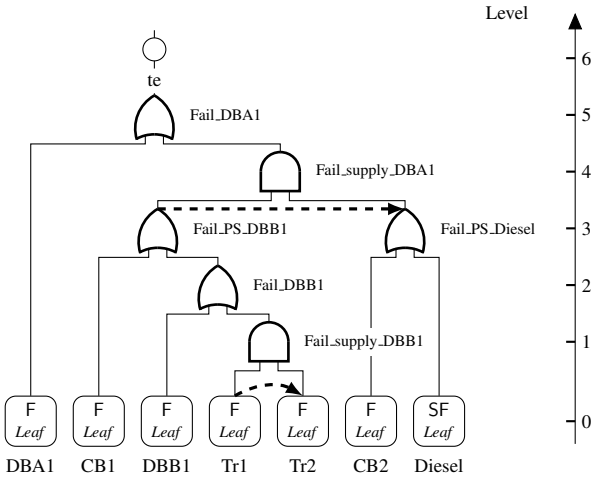

Figure 2: The BDMP of the studied system

- For the modelling of the redundancies *between complex sub-systems* (not only between components);
- For the modelling of the repair events of components where the DFT are modelling non-repairable components only, even if an extension of DFT for the modelling of repair procedures can be found in (Bobbio and Raiteri 2004).

The BDMP shown in figure 2 models the scenarios of failure and repairs events of the power supply system as:

- The Top event (TE) at the root of the tree represents the global failure of the system (main function is faulty: the LHA distribution board is powering the safety components of the core);
- the Fault Tree structure describes the static combinations of components faults that make the whole system to fail by using AND and OR logical gates;
- the triggers describe the redundancy mechanisms: A trigger is a link between the failure state of a subsystem and the mode of another. On the example, the trigger between the transformers Tr1 and Tr2 indicates that if the transformer Tr1 (which is the primary component) fails then the transformer Tr2 (the secondary system at the destination of the trigger) is activated. When the transformer Tr1 is repaired, the system returns into its initial configuration where the function is provided by Tr1.

The system is faulty when LHA is faulty OR when LHA is not powered. The power supply of LHA failed if LGD AND the power supply from the diesel generator failed.

3 QUALITATIVE ANALYSIS OF A BDMP

The qualitative analysis of a BDMP is mostly like those done on Dynamic Fault Trees (DFT). On both cases, the goal is to extract sequences of failure and repair events that lead to the whole system faillure, those sequences are called *Cut sequences*. This extraction is to be done on models that implicitly define sequential relations between the occurrences of failure events (and repair events for BDMP models). Many methods for the quantitative analysis of DFT have been developed and published. Most of these methods are translating the DFT model into formalisms which can be more or less effective to conduct this kind of analysis.
As (Dugan, Bavuso, and Boyd 1992),(Coppit, Sullivan, and Dugan 2000) are developing translations of a DFT into Markov processes, (Boudali and Dugan 2005) presented a translation into continuous time Bayesian networks, (Bobbio and Raiteri 2004) one into Well Formed Petri Nets and more recently one into Interactive I/O Markov chains (Boudali, Crouzen, and Stoelinga 2007). The main goal of those translations is to conduct quantitative analysis. Among those, only the one using Well Formed Petri Nets seems to be able to support qualitative studies but they are making the generation of the marking graph too complex to extract the cut sequences.

Specifically designed to allow the qualitative studies of a DFT, (Tang and Dugan 2004) developed a method based on the separation between an equivalent static fault tree and the sequential constrains generated by the dynamic gates of a DFT. The cut sets of the equivalent static fault tree are calculated using Zero-suppressed Boolean decision diagrams then the sequential constrains (which represent the dynamic behavior of a DFT) are applied on the cuts sets to obtain a set of cut sequences. This method, which is not fully automated, return a superset of the cut sequences of a DFT, then it is an approximation even if the results are conservatives.

(Merle, Roussel, and Lesage 2011) recently developed a fully algebraical approach to conduct both qualitative and quantitative analyses on the DFT model. this method is based on the preliminary calculation of the structural function. This calculation, which is fully algebraical, is then used to determine a canonical form of the structural function, from it, the cut sequences are automatically extracted.

All those works are conducted under the hypothesis that basic components cannot be repaired. Because a BDMP models both the failure and repair mechanisms, a specific translation method is to be developed. To do that, the scenarios of failure and repair events implicitly described by a BDMP will be seen as sequences from a language as this language can be represented by a finite automaton.

All scenarios implicitly described by a BDMP are constituted by the failure and repair events of the leaves and can be represented by a language $\mathcal{L}$ constituted by word on an alphabet Σ. The Σ alphabet is the union of the failure alphabet Σ_f and the repair alphabet Σ_r. Then $\mathcal{L}$ contains all the failure and repair sequences described by a BDMP. As all those sequences are described by the BDMP only a subset of those actually conduct to the global failure of the system (the main function of the system is lost) while the others are non-critical (the main function of the system is still up).

Theoreme 1. *The language $\mathcal{L}$ containing all scenarios described by a BDMP is regular.*

Proof. Let $\mathcal{L}_1 = \{u \mid u \in \Sigma\}$ be the language describing the BDMP scenarios after only one occurrence of an event. As $\mathcal{L}_1$ is a union of singletons, by definition $\mathcal{L}_1$ is regular. Let $\mathcal{L}_n = \{u \mid |u| \leq n, u = va\}$ with $|v| \leq (n-1), a \in \Sigma$.
Suppose that $\mathcal{L}_n$ is regular: $\mathcal{L}_{n+1} = \{u \mid |u| \leq n, u = va\}$ with $v \in \mathcal{L}_n, a \in \Sigma$. $\mathcal{L}_{n+1}$ is regular because it is constituted by the union of words of $\mathcal{L}_n$ with words which are concatenations betweens words of $\mathcal{L}_n$ and words of $\mathcal{L}_1$. □

Theorem 1 makes the usage of the Kleene theorem possible: as $\mathcal{L}$ is regular, there is a finite automaton (FA) generating this language.

4 SEMANTIC ANALYSIS OF A BDMP

A BDMP can be defined by a 4-tuple
$< L, G, T, te >$ where:

$L = \{L_i\}$ is the set of leaves. Those can be of two type: the F leaves which allow the description of a failure which can only occur in an active mode are in L_F while SF leaves allow the description of a failures which can occur in both active and dormant mode are in the L_{SF} set. The L set can then be decomposed as $L = L_F \bigcup L_{SF}$.

$G = \{G_i\}$ is the set of static gates. That can be of kinds OR or AND respectively placed in G_{OR} and G_{AND} ($G = G_{OR} \bigcup G_{AND}$).

Using the L and G sets, the node set of a BDMP is defined as $N = L \bigcup G$ the union of leaves and gates. An extended node N^* set is also defined by adding the Top Event te to the node set N ($N^* = N \bigcup \{te\}$). The state of a node is defined using a 2-tuple of booleans (F_{N_i}, M_{N_i}) where F_{N_i} represents the failure of the node ($True$ if N_i is faulty) and M_{N_i} represents the mode of the node ($True$ if in an active mode, $False$ if in a dormant mode).

Each gate G_i posseses at least two inputs which are defined by $Inputs(G_i) \subseteq N$. A function

$UpStream(N_i) \subseteq (G \bigcup \{te\})$ is necessary. This function is used to calculate the set of gate that are using a node N_i as an input : $UpStream(N_i) = \{G_j \in (G \bigcup \{te\} | N_i \in Inputs(G_j)\}$

$T = \{T_i\}$ is the set of the triggers of a BDMP. A trigger T_i has an origin which is denoted $Orig(T_i)$ and a destination denoted $Dest(T_i)$. As both of these elements are node from N, a trigger T_i is defined by a 2-tuple $(Orig(T_i), Dest(T_i)) \in N^2$. To calculate the mode of each node, a function $UpTrigger(N_i) \subseteq N$ is also needed, this functions returns the set of node that are at the origin of the triggers that have N_i as their destination: $UpTrigger(N_i) = \{Orig(T_j) \in N | T_j \in T, Dest(T_j) = N_i\}$

The Top event te is the representation of the failure of the modelled system, it has only one input and no outputs ($Inputs(te) \subseteq N, Card(Inputs(te)) = 1$).

On the example of figure 2:

- $L = \{L_{DBA1}, L_{CB1}, L_{DBB1}, L_{Tr1}, L_{Tr2}, L_{CB2}, L_{Diesel}\}$
- $G = \{G_{Fail\_DBA1}, G_{Fail\_supply\_DBA1}, G_{Fail\_PS\_DBB1}, G_{Fail\_PS\_Diesel}, G_{Fail\_DBB1}, G_{Fail\_PS\_DBB1}\}$
- $T = \{T_1, T_2\}$
  $T_1 = (G_{Fail\_PS\_DBB1}, G_{Fail\_PS\_Diesel})$
  $T_2 = (L_{Tr1}, L_{Tr2})$

While the section 4.1 is describing how the fault of each node is calculated in the fault tree, the section 4.2 will describe the calculation of the mode and the section 4.3 will describe the behavior of the leaves.

4.1 *Logical description of the faut tree structure*

The role of the static fault tree structure, which is constructed using the leaves as basic componants, is to define the relation between F_{te} the global failure boolean and the F_{L_i} the failures of the leaves. This relation is defined using AND and OR gates. To calculate this relation all the failure of the gates must be known as the failure of the leaves are result of their behavior.

For each gate $G_i \in G$ the failure F_{G_i} can be calculated knowing the gate type :

- For an OR gate, $F_{G_i} = \bigvee_{N_j \in Inputs(G_i)} F_{N_j}$
- For an AND gate, $F_{G_i} = \bigwedge_{N_j \in Inputs(G_i)} F_{N_j}$

As the fault tree structure is acyclic, the failure booleans can be calculated at each level, going from the leaves to the root level. The failure boolean at the root level F_{te} is representing the top event of the tree. On the example of the figure 2:
$F_{te} = F_{DBA1} \vee \big((F_{CB1} \vee F_{DBB1} \vee (F_{Tr1} \wedge F_{Tr2}))$
$\wedge (F_{CB2} \vee F_{Diesel})\big)$

4.2 *Logical description of the trigger structure*

The main role of trigger is to define the mode M_{L_i} of the leaves of a BDMP by the failures of other leaves.

A trigger $T_i \in T$ of a BDMP is a description of a redundancy between two components or sub-systems. Below its origin is the main sub-system of the redundancy which must be faulty to make the subsystem below the destination of the trigger active. Then the parents of N_i in the fault tree structure and the set of the nodes which are at the origin of a trigger to N_i must be known to calculate the mode M_{N_i} of a node N_i.

The calculation of the mode M_{N_i} of a node $N_i \in N^*$ depends on the situation of the node.

- Case 1: The top event is alway in an active mode.
  $M_{te} = True$.
- Case 2: The node is a gate input and is not the destination of triggers.
  $(UpStream(N_i) \neq \emptyset \wedge UpTrigger(N_i) = \emptyset)$:
  $M_{N_i} = \bigvee_{G_j \in UpStream(N_i)} M_{G_j}$.
- Case 3: The node is a gate input and is the destination of triggers.
  $(UpStream(N_i) \neq \emptyset \wedge UpTrigger(N_i) \neq \emptyset)$:
  $M_{N_i} = \big(\bigvee_{G_j \in UpStream(N_i)} M_{G_j}\big) \wedge \big(\bigwedge_{N_k \in UpTrigger(N_i)} F_{N_k}\big)$.
- Case 4: The node is not a gate input and is the destination of triggers.
  $(UpStream(N_i) = \emptyset \wedge UpTrigger(N_i) \neq \emptyset)$:
  $M_{N_i} = \bigwedge_{N_k \in UpTrigger(N_i)} F_{N_k}$.
- Case 5: The node is not a gate input and is not the destination of triggers.
  $(UpStream(N_i) = \emptyset \wedge UpTrigger(N_i) = \emptyset)$:
  $M_{N_i} = False$.

In order to define all the mode boolean of the leaves, all the modes of the nodes are to be calculated beginning by the hightest level. As the BDMP is an oriented acyclic graph each element can be calculated. The result is the leaves mode functions which only depends on the leaves failures ($\forall L_i \in L, M_{L_i} = f(F_{L_j \in L})$). Using this methodology on the example of figure 2 all the M_{L_i} can be calculated from the Top Event to the leaves:

- $M_{te} = True$
- $M_{DBA1} = True$
- $M_{CB1} = True$
- $M_{DBB1} = True$
- $M_{Tr1} = True$

- $M_{Tr2} = F_{T1}$
- $M_{CB2} = (F_{CB1} \vee F_{DBB1} \vee (F_{Tr1} \wedge F_{Tr2}))$
- $M_{Diesel} = (F_{CB1} \vee F_{DBB1} \vee (F_{Tr1} \wedge F_{Tr2}))$

4.3 *Event based description of a leaf*

To define the mechanism described by leaf L_i of a BDMP, the leaf is seen in this study as a automaton which can evolve using two kind of events. On the one hand the events used to describe the failures and repair mechanism are in the alphabet $\Sigma_f \bigcup \Sigma_r$, on the other hand those used for switching mode are in the alphabet Σ_m. The failure and repair events are generated by the leaf when the component they represent sustains a fault or a repair while the mode switching events are generated by the BDMP structure. The automaton representing the generic behavior for both kind of leaves is shown in figure 3.

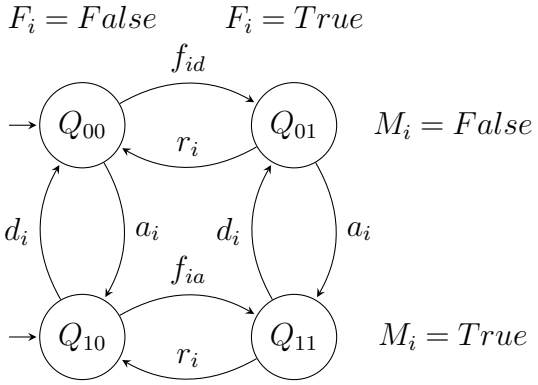

Figure 3: Generic behavior of a leaf L_i

Four states can be identified from the operation state of a component (faulty, non-faulty) and its mode (active, dormant). The initial state is chosen between Q_{00} and Q_{10} as at the initialization of the model all leaves are non-faulty. Q_{00} is the initial state if the leaf is initially in a dormant mode and Q_{10} is the initial state if the leaf is initially in an active mode. Some evolutions between those four states are generated by the events of failures and repairs occurrences, where:

- f_{id} is the failure event while in a dormant mode.
- f_{ia} if the failure event while in an active mode.
- r_i is the repair event of the leaf which can occur during active and dormant mode.

All the other evolution are conducted by the mode switching events which are :

- a_i if the event that make the leaf swich from a dormant mode to an active mode
- d_i is the event that make the leaf swich from an active mode to a dormant mode

Another way to describe a state of a leaf i is to associate a tuple of two Boolean values (M_i, F_i) . M_i represents the mode of the leaf ($True$ if the leaf is in its active mode) while F_i is set to $True$ when the leaf is faulty. While describing the behavior of a SF leaf, the automaton of figure 3 is able to represent the behavior of a F leaf by removing the transition from Q_{00} to Q_{01} on the occurrence of event f_{di}.

5 GENERATING THE "EQUIVALENT" AUTOMATON OF A BDMP

5.1 *The Finite State Automata formalism used to describe a BDMP*

A Finite Automaton (FA) is defined by the tuple $< \Sigma, Q, q_0, Q_M, \delta >$ where:

- Σ is the entry alphabet of the automaton,
- Q is the state set of the automaton,
- q_0 is the initial state,
- Q_M is the set of marked states of the automaton ($Q_M \in Q$),
- δ is the transition set: $\delta = \{< q_i, u, q_j > | (q_i, q_j) \in Q^2, u \in \Sigma\}$

The entry alphabet Σ is the set containing all the failure events (f_{ia}, f_{id}) and all the repair events (r_i) from the leaves $L_i \in L$ of the BDMP.

Each state $q \in Q$ is a state of the automaton which is associated with a state of the BDMP. A BDMP state is defined by the vector $\mathbf{F}_{q_i}$, containing all the fault level from the leaves. As the maximum number of BDMP states is $n = 2^{Card(L)}$ where L is the leaf set, The maximum number of states in the automaton is also limited by n.

When a BDMP is initialized all its leaves are non-faulty, this BDMP state will be associated with the automaton initial state q_0.
The marked state set Q_M is chosen to represent the set of BDMP states where the system has failed. This choice allows to calculate the sequences of failure defined by a BDMP as the marked language of the automaton.

The transition function δ represents the evolutions between two states of the automaton on the occurrences of an event from the alphabet Σ.

As Σ and q_0 are easily deducted from a given BDMP; Q,Q_M and δ must be calculated. In order to automatically generate the FA which represents the BDMP scenarios, a specific algorithm is developed in the following sections of this paper.

5.2 *The generation mecanism*

The FA generation mostly consists in generating the state set Q, the marked state set Q_M and the transition function δ. This generation starts by using the initial state q_0 and then exploring the BDMP states to generate the automaton states. This is done by Algorithm 1.

Algorithm 1 Generate states and transitions of the "equivalent" FSA of a BDMP

Require: $\Sigma = \Sigma_f \bigcup \Sigma_r$: Symbol alphabet ,$L = \{L_i\}$: leaf dictionnary(contain leaf type),$\{M_{Li}\}$ Boolean mode function of leaf L_i, F_{te} Top event boolean function, $\Delta : (S\mathbf{F}, \Sigma) \rightarrow S\mathbf{F}$: function representing the event effects on a failure vector .

1: # Generating initial state
2: $\mathbf{F}_{q0} \leftarrow [F_{Li} = False | L_i \in L]$
3: $S\mathbf{F} \leftarrow \mathbf{F}_{q0}$
4: $Q \leftarrow q_0$
5: **for** each q_i in Q **do**
6: # Step 1 : evaluate $\mathbf{M}_{qi}$
7: $\mathbf{M}_{qi} = [M_{Lj} | L_j \in L]$
8: # Step 2 : generate sensible events SE in state q_i.
9: $SE = Sensible\_events(L, \mathbf{F}_{qi}, \mathbf{M}_{qi})$
10: # Step 3 : Generate next state q_n
11: **for** each $u \in SE$ **do**
12: $\mathbf{F}_{qn} = \Delta(\mathbf{F}_{qi}, u)$
13: **if** $\mathbf{F}_{qn} \notin S\mathbf{F}$ **then**
14: $S\mathbf{F} \leftarrow \mathbf{F}_{qn}$
15: $Q \leftarrow q_n$
16: **if** $F_{te}(F_{L_k}, F_{L_k} \in \mathbf{F}_{q_n}) = True$ **then**
17: $Q_M \leftarrow q_n$
18: **end if**
19: **end if**
20: $\delta \leftarrow < q_i, u, q_n >$
21: **end for**
22: **end for**
23: **return** $< \Sigma, Q, q_0, Q_m, \delta >$

A BDMP state is completely defined by the fault states (faulty, non-faulty) of all its leaves and is associated with an automaton state $q_i \in Q$. It will be represented by a Boolean vector $\mathbf{F}_{q_i}$ of length $Card(L)$. Each component of this vector is the value of the fault Boolean F_{L_i} of each leaf $L_i \in L$. All those BDMP states are placed in the set $S\mathbf{F}$ representing the state set of a BDMP. The faulty states of a BDMP ($TE = True$) are associated with marked states of the automaton, those automaton states are placed in the marked state set $Q_M \in Q$.

In order to apply the evolution associated with the occurrence of an event from Σ on a BDMP state, a $\Delta : (S\mathbf{F}, \Sigma) \rightarrow S\mathbf{F}$ function need to be defined. Δ is a representation of the effect of a failure or repair event on the leaves according the the automaton representation of a leaf proposed in section 4.3.

A mode vector $\mathbf{M}_{q_i}$ is also defined. This Boolean vector contains all the mode M_{L_i} from the leaves $L_i \in L$ and is associated with a automaton state $q_i \in Q$. This vector $\mathbf{M}_{q_i}$ is obtained by updating the mode of each leaf using the boolean function associated with each M_{L_i} and the fault values in the vector $\mathbf{F}_{q_i}$ associated with q_i.

Algorithm 2 $Sensible_e vents(L, \mathbf{F}_{qi}, \mathbf{M}_{qi})$

Require: $L = \{L_i\}$: leaf dictionnary(contain leaf type), $\mathbf{F} = [F_{Li}]$ faillure vector, $\mathbf{M} = [M_{Li}]$ mode vector.

1: $SE := \emptyset$
2: **for** each $L_i \in L$ **do**
3: **if** $\overline{F_{Li}} \wedge \overline{M_{Li}}$ **then**
4: **if** $L_i \in L_{SF}$ **then**
5: $SE \leftarrow \{f_{id}\}$
6: **end if**
7: **else if** $F_{Li} \wedge \overline{M_{Li}}$ **then**
8: $SE \leftarrow \{r_i\}$
9: **else if** $\overline{F_{Li}} \wedge M_{Li}$ **then**
10: $SE \leftarrow \{f_{ia}\}$
11: **else if** $F_{Li} \wedge M_{Li}$ **then**
12: $SE \leftarrow \{r_i\}$
13: **end if**
14: **end for**
15: **return** SE

After the generation of initial state q_0 associated with the BDMP state $\mathbf{F}_{q_0}$ where all fault Booleans are $False$ (lines 1-4), the algorithm generates the next states from the current state $q_i \in Q$ using the "FOR" loop between lines 5 and 22. The first step is to update the mode vector $\mathbf{M}_{q_i}$ associated with current state (line 7). As both the failure ($\mathbf{F}_{q_i}$) and mode vectors ($\mathbf{F}_{q_i}$) are known, all the current states of the leaves are known. In order to calculate SE, the Σ subset that contains the events the BDMP is sensible to, the algorithm 2 is used in line 9. This second algorithm generate SE by seeking the events each leaf is sensible to according to their automaton model shown in section 4.3. When the SE set is known the "FOR" loop between lines 11 and 21 calculate for each tuple ($q_i, u \in SE$) the next state q_n. Using the Δ function in the line 12 of the algorithm 1, the next BDMP state $\mathbf{F}_{q_n}$ is calculated. In order to know if q_n is already present in Q (line 13), this new BDMP state is compared to the existing ones which are present in the $S\mathbf{F}$ set. If q_n is a new automaton state then q_n is placed in Q (line 15) and $\mathbf{F}_{q_n}$ in $S\mathbf{F}$ (line 14). By updating the top event function F_{te} at the new state, the global fault state is known. If q_n is a state where the top event function is $True$ (the system is faulty) then q_n is also placed in the marked state set Q_m (lines 16-18). Even if q_n was already known it is necessary to place the transition $< q_i, u, q_n >$ in the transition function δ (line 20). When the algorithm has completed the generation, the five elements of the tuple describing this automaton are returned. The convergence of

the algorithm is satisfied by a finite maximum number of explored states ($2^{Card(L)}$).

5.3 *The "equivalent" FSA of the studied example*

In order to generate the "equivalent" automaton of a BDMP several elements must be known. The $L = \{L_i\}$ set of the leaves of the BDMP presented in figure 2, each leaf must also be associated with its type, depending on its behavior:

Leaf	Type	associated events
L_{DBA1}	F	f_{0a}, r_0
L_{CB1}	F	f_{1a}, r_1
L_{DBB1}	F	f_{2a}, r_2
L_{Tr1}	F	f_{3a}, r_3
L_{Tr2}	F	f_{4a}, r_4
L_{CB2}	F	f_{5a}, r_5
L_{Diesel}	SF	f_{6d}, f_{6a}, r_6

From these informations extracted from the BDMP, the algorithm 1 generates the "equivalent" automaton. This FA is composed by 128 states and 852 transitions. Among those states, 103 are marked and represents all the situations where the whole system failed. As the number of state is exactly equal to 2^n where $n = Card(L) = 7$ is the number of leaves in the BDMP, all possible situations are reachable from the initial state.

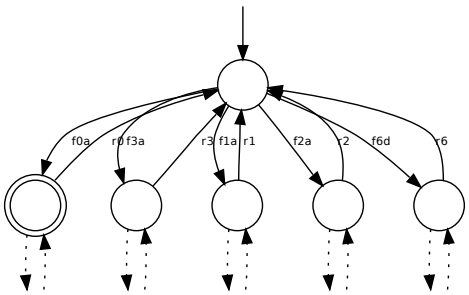

Figure 4: The first states of the 'equivalent' FA of the studied system

A formal verification of the semantical consistency between the FA and the BDMP is not possible but some simple characteristics of the BDMP behavior can be easily verified on the FA. Figure 4 shows the initial state and the first next states of the equivalent FA. From the initial state only 4 components can fail, those are the DBA1 board, the CBA1 circuit breaker, the DBB1 board and the Tr1 transformer in their active mode and the diesel generator in its dormant mode, moreover all repairs can only occur after the associated component failed.

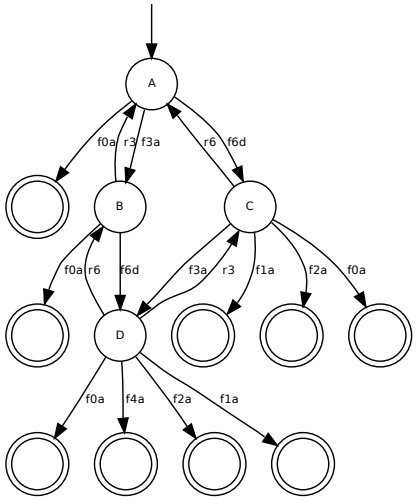

Figure 5: FA extracted from the equivalent FA of the studied example

The first usage of this equivalent automaton is to extract the minimal cut sequence of the BDMP model. Another usage can be to assist the choice of a maintenance strategy. To illustrate this usage, an extracted FA from the equivalent one is used. This FA, shown in figure 5, associates the number of failures which are critical for the system with the failure state of the transformer Tr1 (f_{3a}, r_3 events) and the diesel generator (f_{6d}, r_6 events). Considering the scenario where both transformer Tr1 and the diesel generator failed, the current state after both failures is the state denoted D in figure 5. In this state, where 4 components are critical for the system, two repair events can occur. As the repair of the transformer Tr1 is setting the system in the C state where there are still 3 critical components, the repair of the diesel generator is setting the system in the B state where only one component is critical. So if there is only one maintenance team available, the priority must alway be given to the repair of the diesel generator even if the team already stated the corrective maintenance on the transformer Tr1.

6 CONCLUSIONS

In this paper a method to generate automatically the equivalent finite automaton to a BDMP was described. This equivalent automaton generates a language that contains all the possible scenarios of failure and repair events that are implicitly described by a BDMP. In order to do so, a complete study of the semantic of a BDMP was necessary.

This FA makes many kinds of qualitative analysis possible on the BDMP model. This paper presents one example of those by giving a way to elaborate

a corrective maintenance strategy between two components by looking which repair makes the system in a more secure state.

Our currents works focuses on :

- giving the minimality criterias needed to define what are the cut sequences on the BDMP model. This definition is particularly important to conduct quantitative studies as they were formally defined in the literature.
- formalizing the leaves that describes a failure mechanism that can only append where the leaf is switching mode. This kind of leaf (that was ignored in this paper) are allowing the BDMP model to describes much more complex failure mechanisms.
- developing and testing the usage of techniques adapted to handle large state spaces. The model-checking techniques may be used to obtain the cut sequences from a BDMP with more leaves.

REFERENCES

Bobbio, A. and D. Raiteri (2004). Parametric fault trees with dynamic gates and repair boxes. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, pp. 459–465.

Boudali, H., P. Crouzen, and M. Stoelinga (2007). A compositional semantics for Dynamic Fault Trees in terms of Interactive Markov Chains. *Lecture Notes in Computer Science 4762*, 441.

Boudali, H. and J. Dugan (2005). A new Bayesian network approach to solve dynamic fault trees. In *Reliability and Maintainability Symposium*, Volume 17.

Bouissou, M. and J. Bon (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.

Coppit, D., K. Sullivan, and J. Dugan (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. Proceedings*, pp. 270–282.

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

Henley, E. and H. Kumamoto (1981). *Reliability engineering and risk assessment*. Prentice-Hall Englewood Cliffs (NJ).

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

Stamatelatos, M., W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback (2002). Fault tree handbook with aerospace applications.

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