Formal Verification of Safety Analysis Models of Repairable and Reconfigurable Systems

Elodie Kobeissi, Pierre-Yves Piriou, Jean-Marc Faure

Formal Verification of Safety Analysis Models of Repairable and Reconfigurable Systems

Elodie Kobeissi[1], Pierre-Yves Piriou[2], Jean-Marc Faure[3]
[1]LURPA, ENS Cachan, Univ. Paris-Sud, Université Paris-Saclay, F-94235 Cachan, France (e-mail : elodie.kobeissi@ens-cachan.fr)
[2]Electricité de France, R&D, 78400 Chatou, France (e-mail : pierre-yves.piriou@edf.fr)
[3]LURPA, ENS Cachan, Univ. Paris-Sud, Supmeca, Univ. Paris-Saclay, 94235 Cachan, France (e-mail : jean-marc.faure@ens-cachan.fr)

Abstract: This paper proposes a method to formally check whether formal properties hold on a dynamic model which has been designed by experts for Model Based Safety Analysis/Assessment. As repairable and reconfigurable systems are considered, this model is assumed to be described in the Generalized Boolean-logic Driven Markov Processes (GBDMP) modelling framework. Translation rules are given to obtain a formal model that describes correctly the evolution of the initial model with the semantics of the verification tool. The approach is exemplified on a simple case of standby redundancy.

Keywords: Model Based Safety Analysis/Assessment, Formal verification, Dynamic properties, Generalized BDMP, Stability, Computation Tree Logic.

1. INTRODUCTION

Model Based Safety Analysis/Assessment (MBSA) techniques are aiming to compute dependability attributes (instantaneous and asymptotic reliability and availability, minimal cut sequences, for instance) by analysing models which have been previously designed by safety/dependability experts (Bouissou 2003, Lipaczewski 2015). Accurate and trustworthy computation results require that the initial models are flawless; unfortunately, as any model based on expertise, these models may contain flaws coming from ambiguous or misunderstood specifications, human errors, etc. The overall aim of this work is to contribute to tackle out (or at least limit) this issue by proposing an approach of formal verification of models that will be used later for MBSA.

The modelling framework which was selected for this work is GBDMP (Generalized Boolean-Logic Driven Markov Processes). This framework has been developed for MBSA of dynamic, repairable, and reconfigurable systems (Piriou and al, 2017); it particularly enables fine modelling of different reconfiguration strategies and of the failure of these strategies. Construction rules of well-formed GBDMP models have also been defined in the above reference to ensure syntactic consistency. However, these rules do not guarantee that a well-formed model satisfies dynamic properties, i.e. properties that consider evolutions of the model during time. Verification of such properties require to translate the model into the language of a verification tool then to check whether the properties hold or not on the model (Figure 1). Last, only logical time is considered in this work because functional correctness must be checked before time correctness; hence, the properties to check will be stated in Computation Tree Logic (CTL). Formal verification will be performed by using the widespread NuSMV model-checker; therefore, the formal model of the GBDMP will be a transition system in the language of this tool.

The outline of the paper is the following. A state of the art on formal verification of models for MBSA is given in the next section. The GBDMP modelling framework is shortly presented in section 3 while section 4 focuses on translation of a GBDMP model into the NuSMV language. Examples of positive and negative verification of dynamic properties are detailed in section 5. Finally, concluding remarks and perspectives for future works are drawn up in the last section.

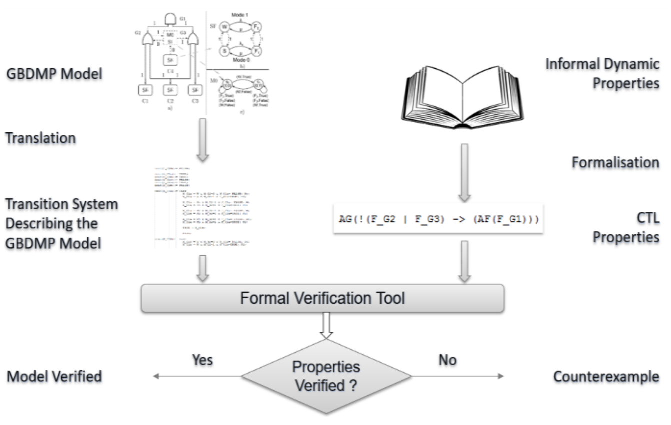


Fig. 1. Principle of the work

2. RELATED WORKS

Several worthwhile results have been obtained in the field of formal verification of MBSA models since the beginning of the 2000s. Checking correctness of a fault tree is the objective of the work presented in (Schäfer 2003); the semantics of fault trees is expressed in Duration Calculus with Liveness and phase automata that model the behaviour of the system components are introduced to model-check.

The objective of (Thums and Schellhorn 2003) is similar, but the semantics of fault trees is given in CTL and the components behaviour is modelled with timed automata. Moreover, two kinds of gates are defined: decomposition gates with the Boolean semantics and cause consequence gates, where time may elapse between the causes and the consequence. A last contribution in this domain that deserves to be mentioned is the work reported in (Ortmeier and Schellhorn 2007) where fault trees are described in Interval Temporal Logic (ITL) and state charts model the behaviour of the system components.

To allow modelling of failures that depend on the order of fault events, a modelling framework named State/Event Fault Trees (SEFTs) is proposed in (Kaiser and al, 2007). This framework subsumes fault tree, deterministic state machines and Markov chains. Analysis of models developed in this frame is performed by translating the component models into Deterministic and Stochastic Petri Nets (DSPNs). This framework is implemented in the ESSaRel (Embedded Systems Safety and Reliability Analyser) tool. Last, (Bozzano et al. 2015) proposes recently to check properties of AltaRica OCAS models by using the NuSMV verification tool while (Sharvia and Papadopoulos 2015) focuses on verification of HiP-HOPS models with the same tool. Despite their interest, none of these works has addressed models for dynamic, repairable, and reconfigurable systems. This study is aiming at filling this gap.

3. BRIEF REMINDER ON GBDMP

3.1 Overall description

A GBDMP model is a 3-tuple composed of:

- an extended fault tree that describes the physical and functional structure of the considered critical process; this fault tree comprises classical gates and leaves as well as switches to introduce explicitly reconfigurations of the process.
- a set of Switched Markov Processes (SMP) which represent the functional and dysfunctional behaviours of the components of the critical process; a SMP is associated to every leaf $l \in L$, where L is the set of leaves of the fault tree.
- a set of Moore machines which model reconfiguration strategies; a Moore machine is associated to every switch $s \in S$, where S is the set of switches of the fault tree.

The three components of this tuple are interconnected. Hence, syntactic rules to build well-formed GBDMP models have been stated; the input/output alphabets of a Moore machine must be consistent with the inputs/outputs of the associated switch, for instance. It will be assumed in what follows that the model to formally verify is well-formed. For space reasons, it is not possible to give a complete syntax of GBDMP; detailed presentations can be found in (Piriou and al, 2017) and (Piriou and al, 2016).

3.2 Example

The example of Fig. 2 is composed of two groups of redundant components {C1a, C1b, C1c} and {C2a, C2b} in a series. The nature of these components does not matter; the only assumptions are that each component can be activated and deactivated and may fail and be repaired when active and inactive. The configuration of each group is managed by a controller (D1 for the first group and D2 for the second one) that performs a reconfiguration strategy:

- for the first group, C1a and C1b are by default active and C1c inactive. Then, if one of the active components fails while the other two ones are not faulty, the failed component is deactivated, and the inactive component activated.
- for the second group, if C2a is not faulty, C2a and C2b must be respectively active and inactive, whatever the dysfunctional state (faultless/faulty) of C2b; if C2a is faulty, C2a and C2b must be respectively inactive and active.

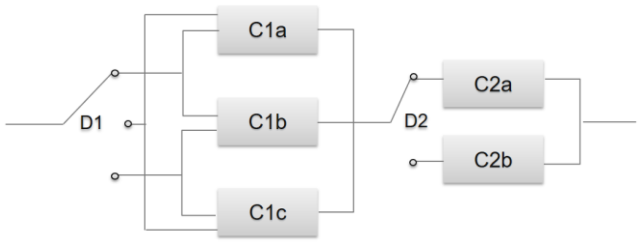


Fig. 2. Standby redundancy example

The GBDMP model of this example, with the above reconfiguration strategies, is shown at Fig. 3. In the fault tree, the leaves C1a, C1b and C1c are linked by dashed arrows to the switch S1; this means that the SMP associated to these leaves send/receive variables to/from the Moore machine. This latter is associated to this switch to model the first strategy where the activation status (active/inactive) of the components C1a, C1b and C1c depends on the failure statuses of these components. A similar explanation can be given for the leaves C2a and C2b and the switch S2. It must be noted nevertheless that no variable is sent to S2 from the leaf C2b because the activation status of C2a does not depend on the failure status of C2b, for the second strategy.

Moreover, the success of a reconfiguration strategy is not guaranteed because the component where it is implemented, a Programmable Logic Controller for instance, may fail. Two failure modes of this controller will be considered hereafter:

- Frozen: no more switching is possible, i.e. the configuration of the process remains the same;
- Bad contact: there is no connection between D1 (D2) and the group {C1a, C1b, C1c} ({C2a, C2b}).

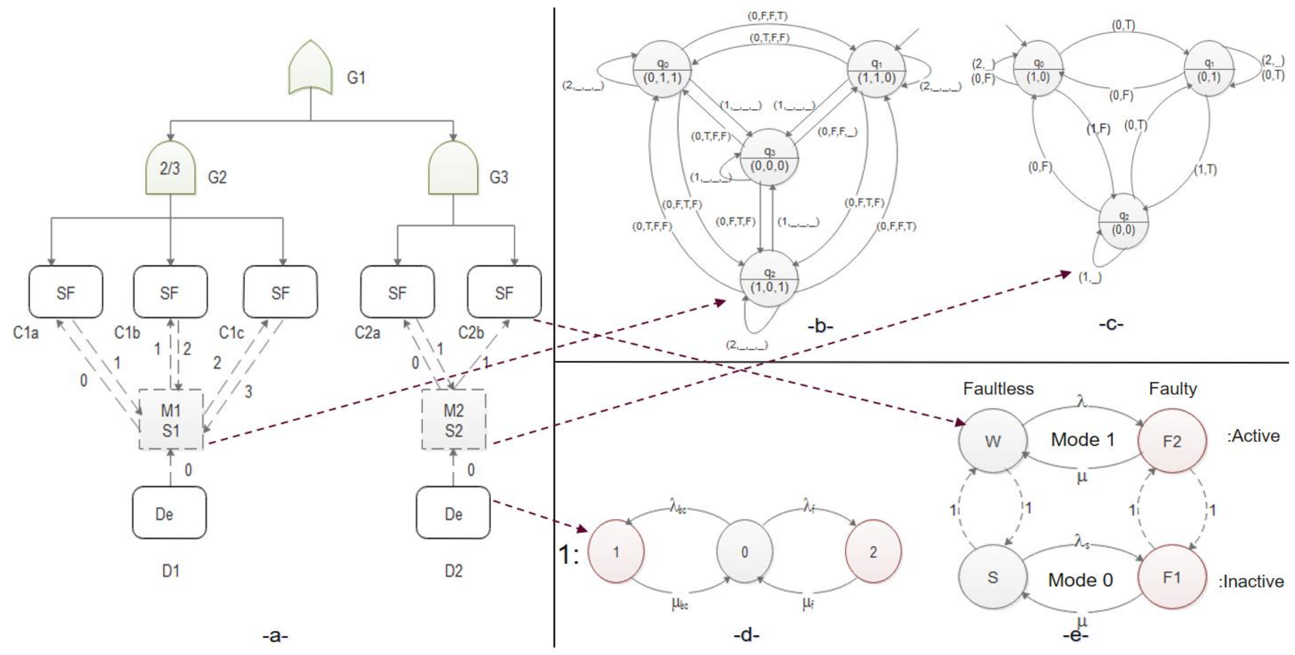


Fig. 3. GBDMP model of the system of Figure 2, a) Extended Fault Tree; b) Moore Machine associated to switch S1; c) Moore Machine associated to switch S2; d) SMP associated to leaf De; e) SMP associated to leaf SF

This possible issue is modelled by the link between the leaf D1 (D2) and the switch S1 (S2). The activation status of a component of the groups {C1a, C1b, C1c} and {C2a, C2b} depends on the failure status of the reconfiguration controller.

3.3 Evolutions of a GBDMP model

A GBDMP model describes a stochastic system whose components may fail and be repaired. However, when focus is put on formal verification of functional (and not quantitative) dynamic properties of this model, the random transitions of its SMP must be replaced by deterministic transitions labelled by the appropriate event. Hence, the transitions that model failures (repairs) will be labelled with failure (repair) events, and not failure (repair) rates, in what follows. With this modelling, the global state of a GBDMP model is completely defined by two sets of variables:

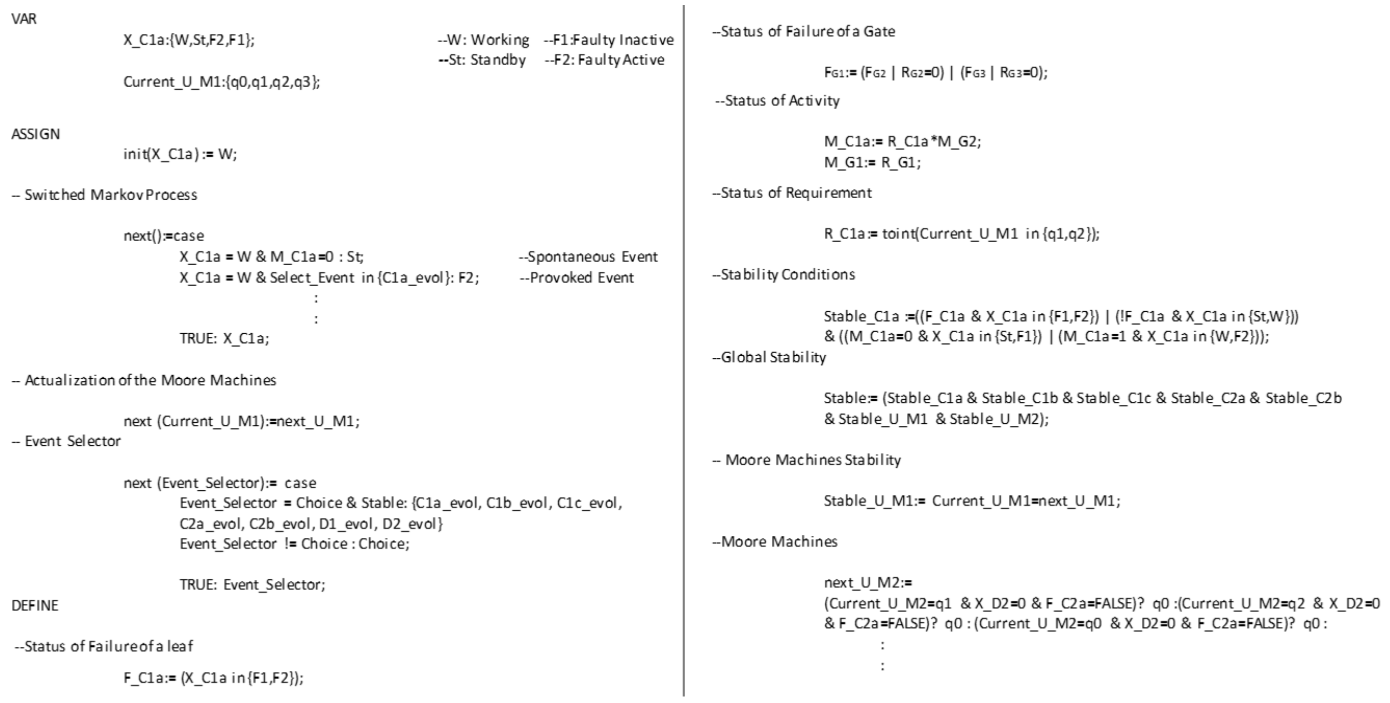
- The set of state variables of the Moore machines. The state variable of the Moore machine associated to the switch $s \in S$ will be noted U_s.
- The set of state variables of the Switched Markov Processes. The state variable of the SMP associated to the leaf $l \in L$ will be noted X_l.

Moreover, for each node (leaf or gate) n of the fault tree, three other variables can be computed from these two sets of state variables, as detailed in (Piriou, 2017):

- F_n: Boolean variable that represents the failure status of the node n (F_n=True/False means that n is faulty/faultless).
- R_n: Boolean variable that represents the requirement status of the node n (R_n=True/False means that n is required/not required to perform the process function).
- M_n: Integer that represents the activity status of the node n (M_n=k means that n is in the kth activity mode).

All these variables (state variables and variables associated to a node) are interdependent and must be computed from the dependency graph that can be built from the extended fault tree. Therefore, the state space of a GBDMP model includes two kinds of states: *stable* states and *unstable* states. A state is stable when the state variable of every SMP is consistent with the failure and activity statuses of the corresponding leaf l (if F_l is True and M_l equal to k for instance, the SMP associated to l must be in a failure state of the kth activity mode), unstable otherwise.

A GBDMP model evolves from state to state on occurrence of two types of events: *spontaneous* events and *provoked* events. A spontaneous event occurs only when the current state is stable; it corresponds to a failure or repair event or an operator request, like a phase change. The spontaneous events are associated to the transitions in solid lines in the SMP. A provoked event occurs when the state is unstable; it is the consequence of a spontaneous event, like the transition of the activity mode of a spare component from inactive to active. The provoked events are associated to the transitions in dashed lines in the SMP. The spontaneous events are asynchronous (their occurrence dates are different) whereas the provoked events which are consequences of a given spontaneous event occur synchronously (at the date of the spontaneous event which caused them).



```
VAR
            X_C1a:{W,St,F2,F1};                     --W: Working   --F1:Faulty Inactive
                                                    --St: Standby  --F2: Faulty Active
            Current_U_M1:{q0,q1,q2,q3};


ASSIGN
            init(X_C1a) := W;

-- Switched Markov Process

            next():=case
                  X_C1a = W & M_C1a=0 : St;                   --Spontaneous Event
                  X_C1a = W & Select_Event in {C1a_evol}: F2;  --Provoked Event
                                    :
                                    :
                  TRUE: X_C1a;

-- Actualization of the Moore Machines

            next (Current_U_M1):=next_U_M1;
-- Event Selector

            next (Event_Selector):= case
                  Event_Selector = Choice & Stable: {C1a_evol, C1b_evol, C1c_evol,
                  C2a_evol, C2b_evol, D1_evol, D2_evol}
                  Event_Selector != Choice : Choice;

                  TRUE: Event_Selector;
DEFINE

 --Status of Failure of a leaf

            F_C1a:= (X_C1a in {F1,F2});
```

```
--Status of Failure of a Gate

            FG1:= (FG2 | RG2=0) | (FG3 | RG3=0);
 --Status of Activity

            M_C1a:= R_C1a*M_G2;
            M_G1:= R_G1;
--Status of Requirement

            R_C1a:= toint(Current_U_M1 in {q1,q2});

--Stability Conditions

            Stable_C1a :=((F_C1a & X_C1a in {F1,F2}) | (!F_C1a & X_C1a in {St,W}))
            & ((M_C1a=0 & X_C1a in {St,F1}) | (M_C1a=1 & X_C1a in {W,F2}));
--Global Stability

            Stable:= (Stable_C1a & Stable_C1b & Stable_C1c & Stable_C2a & Stable_C2b
            & Stable_U_M1 & Stable_U_M2);

-- Moore Machines Stability

            Stable_U_M1:= Current_U_M1=next_U_M1;

--Moore Machines

            next_U_M2:=
            (Current_U_M2=q1 & X_D2=0 & F_C2a=FALSE)? q0 :(Current_U_M2=q2 & X_D2=0
            & F_C2a=FALSE)? q0 : (Current_U_M2=q0 & X_D2=0 & F_C2a=FALSE)? q0 :
                    :
                    :
```

Fig. 4. Part of the NuSMV code for the model of figure 3

4. TRANSLATING GBDMP MODELS IN NUSMV LANGUAGE

NuSMV [Cimatti 2002] is a symbolic verification tool that checks properties on a model in the form of a transition system in a specific syntax. The aim of this section is to give the main translation rules to obtain this model from a well-formed GBDMP model. A part of the result obtained for the GBDMP of figure 3 is given at figure 4.

Only the variables that define the global state (state variables of the Moore machines and SMP) are first defined in the VAR section. Translation of the evolutions of these variables is straightforward by using the operator 'next' that defines the next state. The status variables of the nodes are then computed and the stability condition is obtained, for every leaf of the fault tree, from the values of the state variables and the status variables of this leaf, as follows:

Stable_X_l := ((M_l =0 & X_l in {set of states in mode 0}) | … | (M_l =k & X_l in {set of states in mode k})) & ((F_l & X_l in {set of failure states}) | (!F_l & X_l in {set of non-failure states}))

The global state of the GBDMP model is stable when the Boolean variable Stable which is the conjunction of every Stable_X_l and Stable_Um variable is True.

The main issue in this translation is modelling of the spontaneous events which are asynchronous in the GBDMP framework while NUSMV considers by default that two events may occur simultaneously. To avoid incorrect evolutions, an event selector must be introduced (Figure 5). The role of this automaton is to prevent simultaneous failure/repair events, from the SMP associated to the leaves of the fault tree, to occur. From the initial state, Choice, when the global state of the GBDMP model is stable, only one state of the event selector can be activated and then only one spontaneous event can occur. When a non-initial state of this automaton is active, a variable l_evol where l is the name of the corresponding leaf, is then set; this variable means that the SMP associated to this leaf can evolve and must be introduced as a guard in every transition of the SMP. The initial state of the event selector becomes active again once the SMP has evolved. To sum up, introducing this automaton and these guards in the NUSMV code permits to describe correctly the evolutions of the GBDMP model with the semantics of the verification tool.

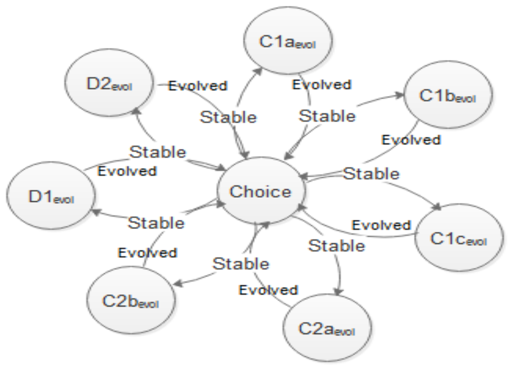


Fig. 5. Event selector for the model of figure 3

5. FORMAL VERIFICATION OF DYNAMIC PROPERTIES

5.1 Notations

Seven properties that must be satisfied by the GBDMP model will be given hereafter. They concern the behaviour of the components, redundant components of the critical process and controllers where the reconfiguration strategies are implemented, which are modelled by SMP, the reconfiguration strategies themselves, modelled with Moore machines, and the top event of the extended fault tree. These properties will be exemplified on the example of figure 3 by using CTL expressions. Hence, the notations of the state and path quantifiers in this logic must be reminded:

- X represents the next state of the current state,
- F means "for at least one state (there exists in the Future) of a path",
- G means "for every state (Globally) of a path",
- A means "along All paths" from the current state,
- E means "along at least one (there Exists) path" from the current state.

With these notations, EF Φ, where Φ is a logic expression, means "there exists a path where there exists at least one state where Φ is verified" and AG "for every path and for every state of the path, Φ is verified", for instance.

5.2 *Properties presentation*

5.2.1 Properties on the behaviour of the components

The first property concerns the behaviour of a redundant component (C1a, C1b, C1c, C2a, C2b in the example); the three following properties the behaviour of a controller where a reconfiguration strategy is implemented.

Property 1: A redundant component can be activated (1a) and deactivated (1b); it can also fail (1c) and be repaired (1d).

This property can be formalized by the four CTL formulas below, where EF (Φ & AX! Φ) means that there exists in the future a state where Φ is true and for all immediate next possible states, Φ is false:

$\text{EF}\ (M_l=0\ \&\ \text{AX}\ M_l>0)$ (1a)

$\text{EF}\ (M_l>0\ \&\ \text{AX}\ M_l=0)$ (1b)

$\text{EF}\ (!F_l\ \&\ \text{AX}\ F_l)$ (1c)

$\text{EF}\ (F_l\ \&\ \text{AX}!F_l)$ (1d)

where M_l represents the activity status of the leaf l associated to the component and F_l the failure status of this component; it is assumed that $M_l=0$ means that the component is inactive.

Property 2: A controller may fail in two ways (two failure modes) (2a) and be repaired when failed (2b).

For the component associated to the leaf D1 for instance, this property is formalized by the following two expressions:

$\text{EF}\ (!F_{D1}\ \&\ \text{AX}\ ((F_{D1}\ \&\ (X_{D1}=2\ |\ (X_{D1}=1)));$ (2a)

$\text{EF}\ ((F_{D1}\ \&\ (X_{D1}=2\ |\ (X_{D1}=1))\ \&\ \text{AX}\ (!F_{D1}));$ (2b)

where $X_{D1}=1(2)$ means that the failure mode is bad contact (frozen).

Property 3: When the failure mode is bad contact, every component of the group of redundant components controlled by the controller is inactive.

For the example of figure 2, this property is formalized by two CTL expressions (one for each group of components).

$\text{AG}\ ((X_{D1}=1\ \&\ \text{Stable}) \rightarrow (M_{C1a}=0\ \&\ M_{C1b}=0\ \&\ M_{C1c}=0));$ (3-1)

$\text{AG}\ ((X_{D2}=1\ \&\ \text{Stable}) \rightarrow (M_{C2a}=0\ \&\ M_{C2b}=0));$ (3-2)

It must be noted that the variable Stable is introduced in these statements because the property holds only in the stable states of the model. This remark can be made for the other following properties.

Property 4: When the failure mode is frozen, every component of the group of redundant components which is controlled by the controller remains in its current state.

$\text{AG}\ (X_{D1}=2 \rightarrow (\text{Stable\_}X_{C1a}\ \&\ \text{Stable\_}\ X_{C1b}\ \&\ \text{Stable\_}\ X_{C1c}));$ (4-1)

$\text{AG}\ (X_{D2}=2 \rightarrow (\text{Stable\_}\ X_{C2a}\ \&\ \text{Stable\_}\ X_{C2b}));$ (4-2)

5.2.2 Properties on the reconfiguration strategies

To check whether the two reconfiguration strategies which have been defined at 3.2 have been correctly modelled in the GBDMP, the properties 5 and 6 are to be verified. Property 5 will be expressed in an *event*-oriented form whereas property 6 will be given in a *state*-oriented form to show that informal specifications can be formalised in these two forms. In the expression of property 5, the construction "A [! Stable U p]" is used to specify the assertion "p will be True in the next stable state".

Property 5: If the controller D1 is faultless, by default (no component is faulty), C1a and C1b are arbitrarily activated (5a). Then, if one of the active components fails while the other two are faultless, disable the failed component and activate the inactive component ((5b-1) to (5b-3)).

$\text{AG}\ ((!F_{D1}\ \&\ \text{Stable}\ \&\ !F_{C1a}\ \&\ !F_{C1b}\ \&\ !F_{C1c}) \rightarrow (M_{C1a}=1\ \&\ M_{C1b}=1\ \&\ M_{C1c}=0));$ (5a)

$\text{AG}\ ((!F_{C1a}\ \&\ \text{AX}\ F_{C1a})\ \&\ (M_{C1a}=1\ \&\ !F_{C1b}\ \&\ !F_{C1c}\ \&\ !F_{D1}) \rightarrow \text{AX}\ (\text{A}\ [!\text{Stable U}\ (M_{C1a}=0\ \&\ M_{C1b}=1\ \&\ M_{C1c}=1)]));$ (5b-1)

$\text{AG}\ ((!F_{C1b}\ \&\ \text{AX}\ F_{C1b})\ \&\ (M_{C1b}=1\ \&\ !F_{C1a}\ \&\ !F_{C1c}\ \&\ !F_{D1}) \rightarrow \text{AX}\ (\text{A}\ [!\text{Stable U}\ (M_{C1a}=1\ \&\ M_{C1b}=0\ \&\ M_{C1c}=1)]));$ (5b-2)

$\text{AG}\ ((!F_{C1c}\ \&\ \text{AX}\ F_{C1c})\ \&\ (M_{C1c}=1\ \&\ !F_{C1a}\ \&\ !F_{C1c}\ \&\ !F_{D1}) \rightarrow \text{AX}\ (\text{A}\ [!\text{Stable U}\ (M_{C1a}=1\ \&\ M_{C1b}=1\ \&\ M_{C1c}=0)]));$ (5b-3)

Property 6: If the controller D2 is faultless and the model stable, whatever the dysfunctional state of C2b, if C2a is not faulty, then C2a and C2b are respectively active and inactive (6a); if C2a is faulty, C2a and C2b are respectively inactive and active (6b).

$\text{AG}\ ((!F_{D2}\ \&\ \text{Stable}\ \&\ !F_{C2a}) \rightarrow (M_{C2a}=1\ \&\ M_{C2b}=0));$ (6a)

$\text{AG}\ ((!F_{D2}\ \&\ \text{Stable}\ \&\ F_{C2a}) \rightarrow (M_{C2a}=0\ \&\ M_{C2b}=1));$ (6b)

5.2.3 Property on the top event

The top event of the extended fault tree is False when the critical system of figure 2 performs correctly its function, True otherwise. The following property can then be stated.

Property 7: If every controller is faultless, the global function is performed if at least two components among {C1a, C1b, C1c} are faultless and at least one component among {C2a, C2b} is also faultless.

This property can be formalized as follows by introducing a Boolean variable TE (Top Event); this variable is merely the complement of the failure status of the top gate of the fault tree.

$$AG\ ((!F_{D1}\ \&\ !F_{D2}\ \&\ Stable\ ) \rightarrow (count\ (!F_{C1a}, !F_{C1b}, !F_{C1c}) \geq 2\ \&\ count\ (!F_{C2a}, !F_{C2b}) \geq 1) \leftrightarrow !TE\ ); \quad (7)$$

where count is a function which counts the number of variables that are True in a set of Boolean variables.

5.3 Detection of errors in the GBDMP model

As the model of figure 3 is correct, every property which has been defined at the previous section holds on this model. To check whether errors during modelling can be detected by the verification tool, two modifications have been brought separately:

- The top gate (OR in the correct model) has been replaced by a AND gate, which is obviously a basic error.
- The label of the transition from q3 to q1 in the Moore machine M1 has been changed.

The first error is detected because Property 7 does not hold anymore; a counterexample that shows that it is possible in this case to have a False TE while two components in the group {C1a, C1b, C1c} are faulty (Figure 6) is provided by the model checker. Such a counterexample surely eases analysis of the GBDMP model and detection of the modelling error.

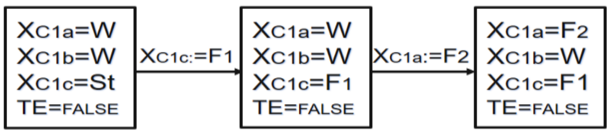


Fig.6. Counter example for the first modelling error.

When only the second error is introduced, both properties 5a and 7 are not satisfied. A counterexample shows that the three components of the group remain inactive after the controller D1 has been repaired. Analysis of the Moore machine that describes the strategy implemented in this controller permits to detect the erroneous label. In figure 7, the states in solid lines correspond to stable states of the system, while the dotted ones correspond to unstable states.

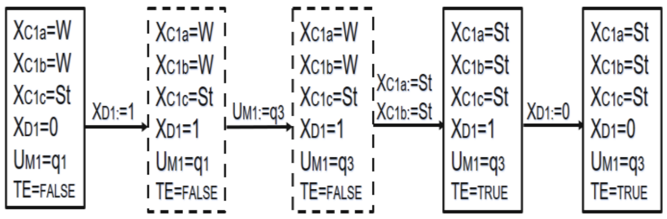


Fig.7. Counter example for the second modelling error.

6. CONCLUSIONS

This paper has shown that modelling errors in a model developed for MBSA of repairable and reconfigurable systems can be detected by using formal verification techniques. Solutions to translate this model in the language of the selected verification tool while keeping the initial semantics have been proposed and some examples of errors that can be detected have been given. The experiments show that the approach scales well. Future work is aiming at facilitating verification by non-experts in temporal logic; development of libraries of CTL formulas which are built automatically from the knowledge of the structure of the critical process and the reconfiguration strategies is under investigation.

REFERENCES

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

Bozzano, M, Cimatti, A., Lisagor, O., Mattarei, C., Mover, S., Roveri, M., and Tonetta, S. (2015). Safety assessment of AltaRica models via symbolic model-checking. Science of Computer Programming, 98, 464–83.

Cimatti, A., E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, & A. Tacchella (2002). Nusmv 2: An opensource tool for symbolic model checking. Computer Aided Verification, 359–364.

Kaiser, B., Gramlish, C., and Förster, M. (2007). State/event fault trees — a safety analysis model for software controlled systems. Reliability Engineering & System Safety, 92 (11), 1521–37.

Lipaczewski, M., Ortmeier, F., Prosvirnova, T., Rauzy, A., Struck, S. (2015). Comparison of modeling formalisms for safety analyses: SAML and AltaRica. Reliability Engineering & System Safety, 140, 191–199.

Ortmeier, F., and Schellhorn, G. (2007). Formal fault tree analysis - Practical experiences. Electronic Notes in Theoretical Computer Science, 185, 139–51.

Piriou, P-Y., Faure, J-M., Lesage, J-J. (2017). Generalized Boolean logic Driven Markov Processes: A powerful modelling framework for Model-Based Safety Analysis of dynamic repairable and reconfigurable systems, Reliability Engineering & System Safety, Volume 163, Pages 57-68.

Piriou, P-Y., Faure, J-M., Lesage, J-J. (2016). From safety analysis of reconfigurable systems to design of fault-tolerant control strategies. 3rd Int. Conf. on Control and Fault-Tolerant Systems, SysTol'16, 609-614.

Schäfer, A. (2003). Combining real-time model-checking and fault tree analysis. FME 2003: Formal Methods, 522–41.

Sharvia, S. and Papadopoulos Y. (2015). Integrating model checking with HiP-HOPS in model-based safety analysis. Reliability Engineering & System Safety, 135, 64–80

Thums, A., and Schellhorn, G. (2003). Model-checking FTA. FME 2003: Formal Methods, 739–57.