

BIEBER Pierre* – **CASTEL Charles*** – **CHOLVY Laurence*** – **DEMMOU Hamid**** –
KEHREN Christophe* – **MEDJOUDJ Malika**** – **RIVIERE Nicolas**** –
SEGUIN Christel* – **VALETTE Robert****

**ONERA*

F-31055 Toulouse Cedex 4 – France

{firstname.lastname@onera.fr}

***LAAS-CNRS*

F-31077 Toulouse Cedex 4 - France

{firstname.lastname@laas.fr}

Résumé en français et en anglais :

This paper deals with scenarios that lead a dynamic system to a critical situation. It aims at clarifying the logical properties of a representation of such scenarios. It identifies and compares two approaches for scenario representation. The first one is declarative; it proposes to represent scenarios by a particular subset of formulae of temporal logic. The second one is constructive; it puts in evidence that scenarios are also encoded inside proof structures such as proof of sequent in Linear Logic.

Cet article porte sur les scénarios qui conduisent un système dynamique à une situation critique. Il a pour but de préciser les propriétés logiques des représentations de tels scénarios. Il identifie et compare deux approches de représentation. La première est déclarative. Elle propose de modéliser les scénarios par un sous-ensemble particulier de formules de logique temporelle. La seconde est constructive; elle montre que les scénarios sont aussi encodés à l'intérieur de structures de preuve comme les preuves de séquent en logique linéaire.

Mots clés en français et en anglais :

Représentation de scénario, logique temporelle linéaire, réseaux de Petri.

Scenario representation, linear temporal logic, Petri nets.

1 – Introduction

This paper presents the work done during the past year during a research project supported by the CNRS federation FERIA and involving researchers from the LAAS-CNRS and ONERA. Entitled "A comparative study of methods and tools for the construction of scenarios" the project has focused, this past year, on a comparative study of scenario representations. Before detailing the work, let us present the motivation of the project.

Frequently, embedded systems (in cars, trains, planes *etc.*) are not only in charge of controlling some system composed of mechanic and hydraulic parts, they are also in charge of monitoring it and of coordinating its behaviour with other mechatronical systems. This means that when some event, affecting the safety (of the passengers in general) occurs, some reconfiguration of the system is executed in order to maintain the vehicle in a safe degraded state. Sometimes, the reconfiguration fails and the system is driven to a feared state with dramatic consequences for the passengers. In early design stages, the designers have to know and understand how the systems can reach such feared states because they precisely have to imagine reconfiguration policies which avoid them. The qualitative description of the scenarios which lead the system from a "normal" state to a feared one is therefore very important. It is clear that a probabilistic quantification of the scenarios is also important, but this topic will not be addressed in this paper.

Classical safety analysis is based on fault trees. They are a static representation of all the ways a system can fail. The root corresponds to the system failure and each leaf to an elementary fault. Each

intermediary node is a boolean function ("and" or "or") and a fault tree can be seen as a monotone boolean function describing how the system fails. The major benefit of fault trees is that a set of minimal cutsets (similar to the prime implicants of the boolean function) can be derived. These minimal cutsets (as well as the prime implicants) are a representation of all the minimal scenarios of failure.

Unfortunately, fault trees are not relevant for dealing with systems subject to reconfigurations because, for these systems, the occurrence order of the events has to be taken into account and in classical logic, the two basic operators are commutative. For example, in a 2-redundant system if f_i is the failure of component i and r_i the fact that component i has been repaired, the sequences $(f_1;f_2;r_1)$ and $(f_1;r_1;f_2)$ are completely different (in the first case the feared state is reached and not in the second one) although the corresponding prime implicants $(f_1 \wedge f_2 \wedge r_1)$ and $(f_1 \wedge f_2 \wedge r_1)$ are identical. It is why our first task has been to compare various ways of describing scenarios. Our second step will be to try to define a notion similar to that of prime implicants and then to develop algorithms to derive them from a system model.

Next section aims at clarifying, through an example, the kind of systems we want to analyse and the expected properties of critical scenarios. We intend also to assess the impact of system modelling language over the scenarios definition. So we present in section three a first formalisation in Temporal Logic. In section four, we consider Girard Linear Logic framework. Finally we conclude by comparing the two approaches.

2 – Critical scenarios for a simple repairable system: example and expected properties

We consider a simple constellation of satellites to illustrate the concepts we want to deal with. We first present the system and its behaviour, both in nominal and in faulty state. Then we characterize the failure condition that we want to analyze. Finally we specify the scenarios that are useful to analyze the studied failure condition.

2.1 – The repairable satellite constellation

The system consists in three components: two satellites and one ground station. Each component may be active ("ok") or out of order ("ko" or "not ok"). All components are fail-silent: they do not provide any output when they are "ko". When they are ok, the satellites and the ground station have slightly different behaviours. An active satellite "i" sends continuously a tele-measure "out_i" to the ground station; when the ground station is ok, it provides the expected data "outs" if at least one tele-measure is available. Finally, the components are initially ok; then they may fail and be repaired at any time instant.

Such a behavior may be easily modelled by a finite state automaton as depicted in figure 1. In the initial state S_1 , all components are ok and consequently the ground station provides an output and the flag out_s is true. If the ground station fails (fault f_s occurs), then the system reaches the state S_3 where out_s becomes false. In S_3 , the ground station may be repaired (the transition r_s is taken) and the system goes back to state S_1 , or other faults may occur and so on.

The system shall provide the expected data as long as possible. So it reaches a critical situation when the ground station does not compute any more an output: the flag "outs" becomes false and it remains false. For instance, starting from the initial state, the sequences of transitions $f_s, f_s;r_s;f_s, f_s;r_s;f_s;r_s;f_s, \dots$ lead to this case. In such a repairable system, an infinite number of paths lead to this critical failure condition whereas safety engineers are interested in a finite, concise and complete representation of these various paths. Let us clarify what are the expected properties of this representation.

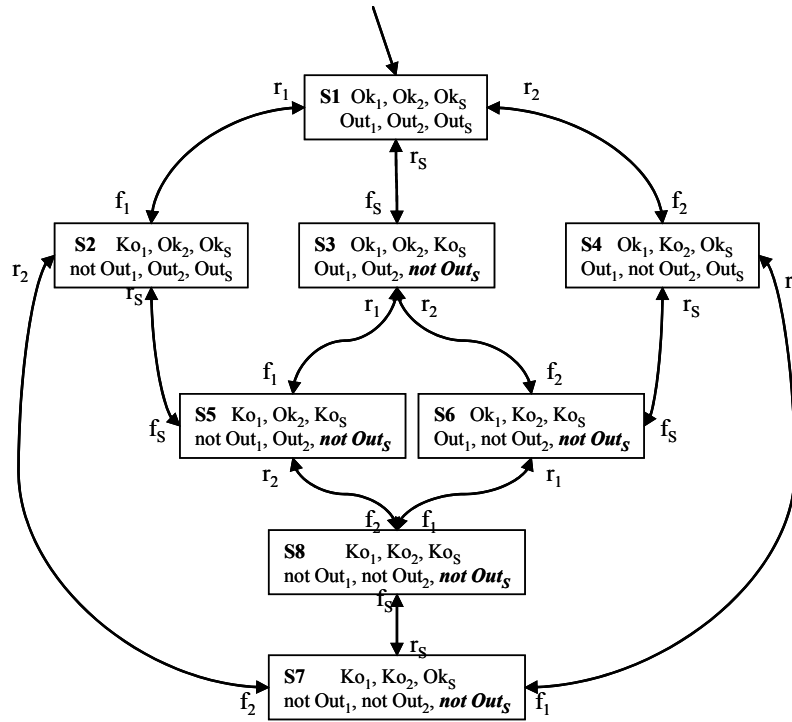


Figure 1: Automaton of the system with failure

2.2 – Properties of critical scenarios

We call “critical scenario CS for a failure condition FC and a system model SM ” a representative of a set of evolution paths that are extracted from SM and that lead from an initial configuration to a system state where FC holds.

For a given FC and a system model SM , we would like to have:

- a complete set of representatives (the set does not forget an evolution path);
- relevant representatives: each scenario shall contain necessary and sufficient information. Part related to useless sub-scenario shall be discarded;
- compact representatives so that we have only one representative for a subset of equivalent paths.

In order to find out the formal properties of critical scenarios, let us put in perspective the classical approach with ours. In the current practice, the system model SM is given by a set of fault trees SM_i . Each fault tree is dedicated to a failure condition FC_i , also called top level event of the tree. In such a case, there is no distinction between FC_i and SM_i . FC_i is a boolean formula that describes which combinations of basic events (leaves of the tree) lead to the failure condition.

A scenario CS_j leading to FC_i should correspond to an implicant of FC_i . An implicant of FC_i is a boolean formula that describes sufficient conditions to get FC_i . However, FC_i may have plenty of implicants. Indeed, relevant representatives for safety engineers are *prime implicants* that describe sufficient and necessary conditions to logically infer FC_i .

Prime implicants have a *compact canonical form*; they are products, i.e. conjunction of literals, positive (to denote the occurrence of a basic event) or negative (to denote that the negated event does not occur).

Finally, a set of prime implicants is *complete* for FC_i if the conjunction of the negation of all products implies that FC_i does not hold. In other words, the set of implicants contains all possible causes of FC_i .

From this, it appears that we have to clarify the formal relationship between CS_j , FC_i and SM . The relations are well defined when CS_j , FC_i and SM are formulae of propositional logic. What happens when these items are no longer boolean formulae but automata or formulae of a logic that can express dynamic aspects of a system? Does the choice of the modelling language impact the definition of the relationship? We will try to answer these questions by considering two different logical languages that were defined to express dynamic aspects of systems: Temporal Logic and Girard Linear Logic. In these logics, the notion of logical consequences is of course well defined and we will see that they can be used to define

relevance and completeness of a set of scenarios. Nevertheless, there is no agreement on the canonical form that will enable compact expression of scenario. So, before entering into the details of the retained formalisms, let us clarify some compactness conditions from an informal user point of view.

First, when observing our example, it appears that sequences of events $f_s, f_s;r_s;f_s, f_s;r_s;f_s;r_s;f_s, \dots$ lead to the failure condition: “ out_s is false and will remain false if no more transition is taken”. Indeed, from these sequences, we want to retain that the ground station failed (transition f_s). The repetition of prefix $f_s;r_s$ or some other prefix is not important for this scenario. Similarly, a suffix that does not contain reparation of the ground station (transition r_s) is useless. The only important fact is to express that *after* f_s, r_s *never* occurs. The former sequences catch the transition ordering explicitly and the absence of reparation after the failure implicitly.

Another critical scenario is satellites 1 and 2 permanent loss. In such a case, the ordering of f_1 and f_2 is not important. Thus, a *partial order* notation will provide a more compact representation than a set of totally ordered sequences.

We have also to take into account commonly agreed work hypotheses for system models. Thus, we assume that *only one transition can be fired at a time point*. This means that we cannot simultaneously consider the reparation of one item and the failure of another. This hypothesis seems reasonable since it is highly non probable that two independent events occur at the same time.

Finally, it is worth noting that in our example, the system state is only determined by the initial state and a sequence of event. We will indeed consider *deterministic systems*. Moreover, if the system behaviour depends on input variables, we assume that *a variable change corresponds to an event* that can be referred into the scenarios.

3 – Characterization of a scenario (critical) by a formula of LTL

3.1 – Linear Temporal Logic (LTL)

LTL [Manna-Pnuelli 1993] is an extension of classical logic with temporal operators such as G, F and X (representing “globally”, “finally” and “next” respectively). Formulae of this logic can be interpreted with respect to infinite sequences of system states generated by an automaton (like the automaton of figure 1). For our example, all sequences will begin by the initial state S_1 . A system state assigns a value to each system parameter like the boolean parameters ok_i or out_i in our example. Moreover, we add booleans like f_i or r_i in each state. Such a boolean is true when the corresponding event occurs. Some other temporal logics enable to deal directly with label of transitions ([Harel 1984], [Arnold-Niwinski 2001]). We choose LTL for the sake of conciseness because we are interested mainly by G, F and X operators.

The meaning of the temporal operators is illustrated in figure 2 hereunder and defined as follows.

A formula Xp is true in a state of the sequence if the formula p is true in the next state of the sequence.

Fp is true in a state if p is true in this state or will be true in at least one following state of the sequence.

Gp is true in a state if it is true in this state and it remains true in all following states of the sequence.

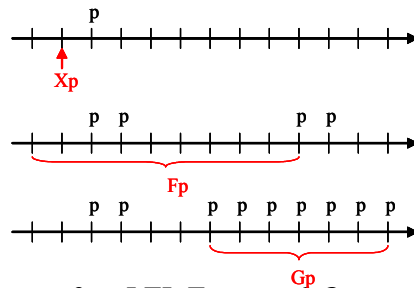


Figure 2 – LTL Temporal Operators

3.2 – From LTL logical consequence to logical properties of critical scenarios

We say that a sequence (S_1, S_2, \dots) satisfies a temporal formula p if p is true in the first state S_1 of the sequence. We note this $(S_1, S_2, \dots) \models p$.

Finally, we say that a temporal formula q is a logical consequence of a temporal formula p if all sequences that satisfy p , satisfy also q . We note this $p \models q$.

It is worth noting that these notions are classical in temporal logic and seem sufficient to formalize minimal implicants in the dynamic case. Let us assume that the automaton describing the system, a failure condition and a critical scenario can be encoded respectively by the temporal formulae SM , FC_i and CS_j . Thus CS_j is a relevant critical scenario with respect to SM and FC_i if:

- CS_j is a sufficient condition i.e FC_i is a logical consequence of the conjunction of the temporal formulae $CS_j \wedge SM$: $CS_j \wedge SM \models FC_i$.
- CS_j is a necessary condition i.e.
for all formulae CS_k , if $CS_k \wedge SM \models FC_i$ and $CS_k \models CS_j$ then $CS_j \models CS_k$

We can also express completeness with these notions. A set $\{CS_1, CS_2, \dots, CS_m\}$ of critical scenarios is complete with respect to a system model SM and a failure condition FC_i if $SM \wedge FC_i \models CS_1 \vee CS_2 \vee \dots \vee CS_m$.

So this framework seems to be a natural extension of the classical one. Moreover, tools like model-checkers [MacMillan 1997] enable to check whether a CS_j is a sufficient condition or whether a set of implicants is complete. Nevertheless, a hard point remains: the theoretical necessary condition cannot be easily verified. Moreover, it does not help to find out the format of a minimal critical scenario. So we propose a canonical form that catches user requirements that we describe in the previous section.

3.3 – Canonical form of critical scenario in LTL and application to the example

We postulate that LTL formulae that describe interesting critical scenarios are conjunction of the following more basic temporal formulae:

- $F a$: the event a occurs at least once in the present or future
- $\text{not } F a$: the event will never occur now or in the future. This is similar to $G(\text{not } a)$
- $F(a \wedge F b)$: event b occurs after event a has occurred
- $F(a \wedge \text{not } F b)$: event b will never occur after event a has occurred.

In our example, the failure condition “permanent loss of the ground station service” will be stated by the formulae $F(G(\text{not } out_s))$.

The first critical scenario is $F(f_s \wedge \text{not } F r_s)$. It corresponds to the permanent loss of the ground station.

The second scenario is $F(f_1 \wedge \text{not } F r_1) \wedge F(f_2 \wedge \text{not } F r_2)$. It corresponds to the permanent loss of both satellites.

3.3 –Discussion

We were able to encode the system model in SMV language (see for instance [Kehren-al 2004]). We used the SMV model checker to verify that both LTL formulae of critical scenario are all the implicants of the failure condition. Minimality remains unproved. However, the proposed format is quite compact. It enables the expression of partial order between events, without interleaving of useless prefix or suffix. It should be validated by wider experiments.

4 – Characterization of a scenario (critical) by a sequent in linear logic

This representation requires a description of the whole system under the form of a Petri net (the system model SM). The scenario is not simply a firing sequence between an initial and a final marking for two reasons. First in a firing sequence, all the firings are ordered. If the firing of t_1 precedes that of t_2 in the sequence, it does not mean that t_1 is a cause of t_2 . Second, in a firing sequence, it is necessary to consider reachable markings and therefore to specify the state of components which are not involved in the feared scenario. It is why the Petri net is translated into a set of Linear Logic formulas. The scenario is then a

Linear Logic sequent and the precedence relations (causality) between the transition firings are obtained by building and labelling a proof tree of the sequent [Rivière 2003].

4.1 – Translation of Petri net into Linear Logic

The Multiplicative Intuitionistic Linear Logic (MILL) [Girard 1987] fragment contains the necessary connectors to translate a Petri net into Linear Logic. It contains the multiplicative connective " \otimes " (conjunction of hypotheses) and the linear implication " \multimap ". There is no negation and the meta connective " $,$ " is commutative. The specificity of Linear Logic (with respect to classical logic) is that logical propositions are consumed when they are used for a deduction. Proving a sequent is verifying that the required hypotheses are available when they are used in a proof step.

An Atomic proposition P is associated with each place of the Petri net. It represents one token located in this place. Marking are denoted by monomial in \otimes . Transition firings are denoted by formulas of the form: $t : Pre(t) \multimap Post(t)$ where $Pre(t)$ and $Post(t)$ are monomials in \otimes . A reachability proof is expressed by the following sequent: $M_0, \lambda_0 \vdash M_n$ where M_0 is an initial partial marking, M_n a final one and λ_0 is a set of formulas denoting transition firings. If a transition t is fired n times then the corresponding formula has to be present in n exemplars in λ_0 . A partial marking is a set of tokens whose location is known. The presence of other tokens in other places is not excluded. It therefore represents a set of possible states. The final partial marking M_n is the failure condition, and the set λ_0 is a first representation of the critical scenario CS. In the sequent, the absence of events is not denoted.

4.2 – Proof of a sequent and labelled proof trees

A sequent proof tree is a syntactical proof. It is a set of rules proving that the connectives have been correctly introduced. The construction of the proof is based on an iterative step which consists in eliminating each transition firing in λ_0 after verifying that the required atoms have been produced. The precedence relations imposed by the structure and the markings of the Petri net correspond to the causal relations between the iterative steps. They are derived by labelling the atoms of the proof tree: each logical atom (token) corresponds to a causal relation between the formula (transition firing) which produces it and the one which consumes it. After having proven the sequent, it can be said that the feared state M_n (i.e. FC_i) is a logical consequence of λ_0 . In addition, it can be remarked that each transition in λ_0 depicts a fragment of the system behaviour. So λ_0 can also be seen as $CS_j \wedge SM$. The sequent can therefore be seen as an alternative expression of the sufficient condition given in section 3. The minimality of the scenario is easy to characterize: $\neg \exists \lambda_1, \lambda_1 \subset \lambda_0$ such that $M_0, \lambda_1 \vdash M_n$

4.3 – Modelling the system, the failure conditions and the scenarios

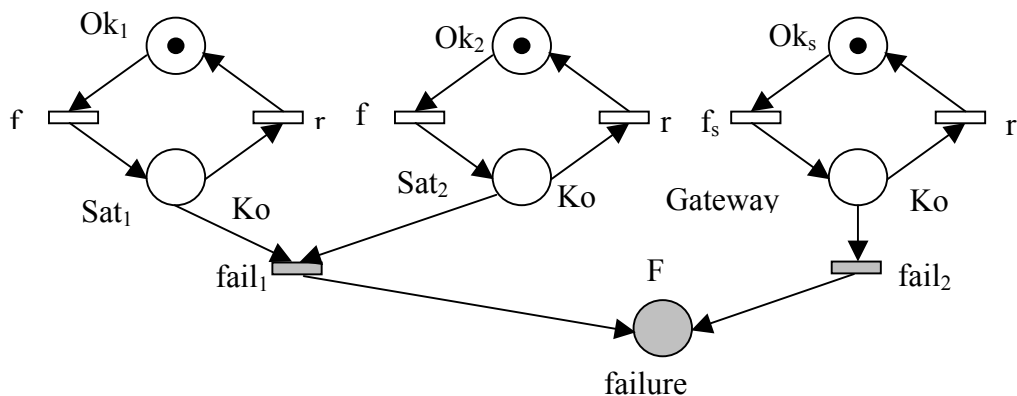


Figure 3: Petri net model of the system with failure

The Petri net is obtained by composing the three automata representing the two satellites and the gateway

(ground station). This corresponds to the upper part in figure 3. The reachable marking graph of this part coincides with the automaton in figure 1. As the Petri net has to describe the failures, it is necessary to add the two transitions $fail_1$ and $fail_2$ which are fired when the system is down as well as the place F corresponding to the feared state (failure).

Let us consider the scenario corresponding to the failure of the gateway. The initial partial marking in Linear Logic is Ok_s because the states of the two satellites are not relevant for this scenario. The set of transition firings is $(f_s, fail_2)$ and the sequent is $Ok_s, f_s, fail_2 \vdash F$

As noted above, the sequents do not explicit the precedence relations among the transition firings. These relations are derived from the sequent proofs which exhibit for each token the transition firing which has produced it and the one which has consumed it. For example, during the scenario corresponding to the gateway failure, a token is produced in place Ko_s by f_s and consumed by $fail_2$. There is thus a causal relation between these two events. The graph representing the precedence relations is given in figure 4. I_3 is an initial event and F_2 a final one.

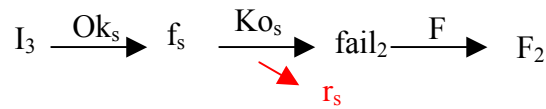


Figure 4: Precedence graph of the scenario corresponding to the gateway

Similarly, the graph representing the precedence relations of the scenario corresponding to the failure of the two satellites is given in figure 5.

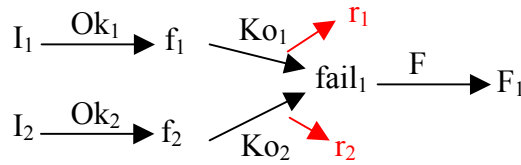


Figure 5: Precedence graph of the scenario corresponding to the two satellites

In the precedence graph, it is possible to introduce transitions which are conflicting with those in the scenario. For example in the first graph, r_s is conflicting with $fail_2$ (both consume the token in Ko_s) and in the second graph, transitions r_1 and r_2 are conflicting with $fail_1$. These conflicting transitions are a way of re-introducing the notion of absence of event.

4.4 – Discussion

Some points have to be underlined. In the scenario corresponding to the failure of the gateway, the states and events concerning the two satellites are not considered. This scenario can thus be seen as a minimal representation, for example with respect to a scenario corresponding to the failure of satellite Sat_1 , and then that of the gateway. This is a consequence of the fact that only the events with a direct causal relation with the feared state (within one scenario) are taken into account.

Another point is that the scenario in figure 5 corresponds to two firing sequences: $(f_1; f_2; fail_1)$ and $(f_2; f_1; fail_1)$. From the point of view of the causality they are two views of a unique scenario. This points out the importance of not representing scenarios by sequences.

Finally, From the Petri net model of the system, it can be seen that in some scenario steps, the transitions are conflicting with another ones. This means that the scenarios can be interrupted at these points. For instance, in the scenario involving the two satellites, the transition $fail_1$ is in conflict with the transitions r_1 and r_2 . This means that when this scenario occurs, if a repair is executed before the second failure, the feared state will be avoided.

5 – Conclusion

This paper has presented two approaches for scenario representation. Both associate a logic expression with an explicit representation of a set of precedence relations which correspond to a strong notion of causality. From this point of view they are closely related and formally represent the partial order which has to be verified by the events in order to lead the system to a feared state.

In the first approach, the description is a formula of Linear Temporal Logic which directly describes the set of precedence relations defining the partial order (by transitive closure). It is therefore similar to a prime implicant which is also a formula (but in classical logic). In the second approach, the description is a Linear Logic sequent which not only specifies the events but also a partial initial state and the feared state (also defined as a partial state). The precedence relations between the events are derived from the sequent proof.

Current work concerns the notion of minimality: how is it possible to formally identify events which are not strictly necessary to reach the feared state? It also concerns the generation of a scenario from either a Petri net model or a language based on product of automata such as AltaRica [Arnold-al 1999].

References :

- [Arnold-al 1999] A. Arnold, A. Griffault, G. Point and A. Rauzy. *The AltaRica formalism for describing concurrent systems*. Fundamenta Informaticae, 40(2-3): 109-124, 1999.
- [Arnold-Niwinski 2001] A. Arnold and D. Niwinski. *Rudiments of mu-calculus*. Studies in Logic and the Foundations of Mathematics. North-Holland, 2001
- [Kehren-al 2004] C. Kehren, C. Seguin, P. Bieber; C. Castel, C. Bounol, J.-P. Heckmann and S. Metge. *Advanced Multi-System Simulation Capabilities with AltaRica*, proceedings of Safety Critical System Conference, Rhodes Island, USA, august 2004.
- [Girard 1987] Girard J-Y. *Linear logic*. Theoretical Computer science, 50, P1-102, 1987
- [Harel 1984] D. Harel, “*Dynamic logic*”. Handbook of Philosophical Logic, D. Gabbay and F. Günthner Eds, D. Reidel 1984.
- [MacMillan 1993] K.L. MacMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993, ISBN 0-7923-9380-5.
- [Manna-Pnuelli 1993] Z. Manna, A. Pnuelli. *The temporal logic of reactive and concurrent systems*. Springer – Verlag, New-York, 1992, ISBN 0-387-97664-7.
- [Medjoudj 2004] Medjoudj M, Khalfaoui S, Demmou H, Valette R. *A method for deriving feared scenarios in hybrid systems*. Probabilistic Safety Assessment and Management (PSAM7- ESEREL04). Berlin, Germany, 14-18 June 2004.
- [PRA 99] Pradin-Chézalviel B, Valette R, Künzle L.A. *Scenario duration characterization of t-timed Petri nets using linear logic*. IEEE PNPM'99, 8th International Workshop on Petri nets and Performance Models, p208-217, Zaragoza, Spain, September 6 -10, 1999
- [Rivière 2003] Rivière N. *Modélisation et analyse temporelle par réseaux de Petri et logique linéaire*. Phd thesis. Institut National des Sciences Appliquées, Toulouse, France, 2003.