Improving dependability of controlled systems: a challenge for automation science and engineering

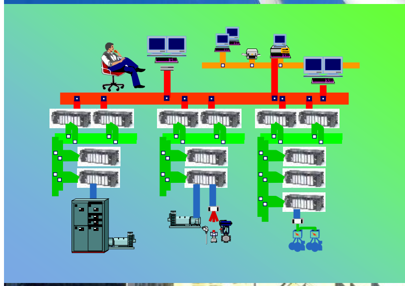
Jean-Marc Faure

with contributions from Jean-Jacques Lesage and other colleagues and PhD students of the Automation Engineering team of LURPA

http://www.lurpa.ens-cachan.fr/isa/

Dependable Control of Discrete event Systems

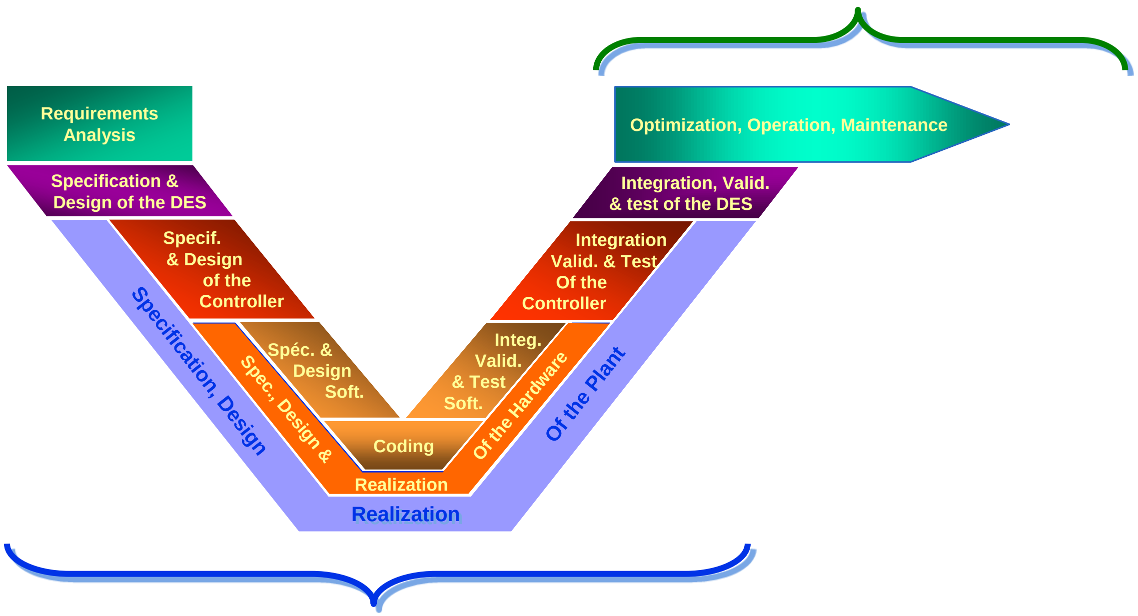
- **Dependability**
  - Dependability is the trustworthiness of a system which allows reliance to be justifiably placed on the service it delivers
  - Dependability attributes: safety, security, availability, reliability, maintainability
- **Objective of our works: to develop methods, models and tools that improve design, implementation and operation of mainly discrete control systems, so as to increase the overall dependability.**
- **Targets: from basic embedded logic controller to networked automation systems**
- **Application fields: critical systems (energy, transport, healthcare, complex mechatronic systems)**
- **Industrial partners: Alstom, Dassault Systems, EDF, ...**


Dependable Control of DES: the quest for the Holy Grail

On-line approaches (during operation)

- FDI, Diagnosis, Prognosis, …
- Dynamic reconfiguration, …



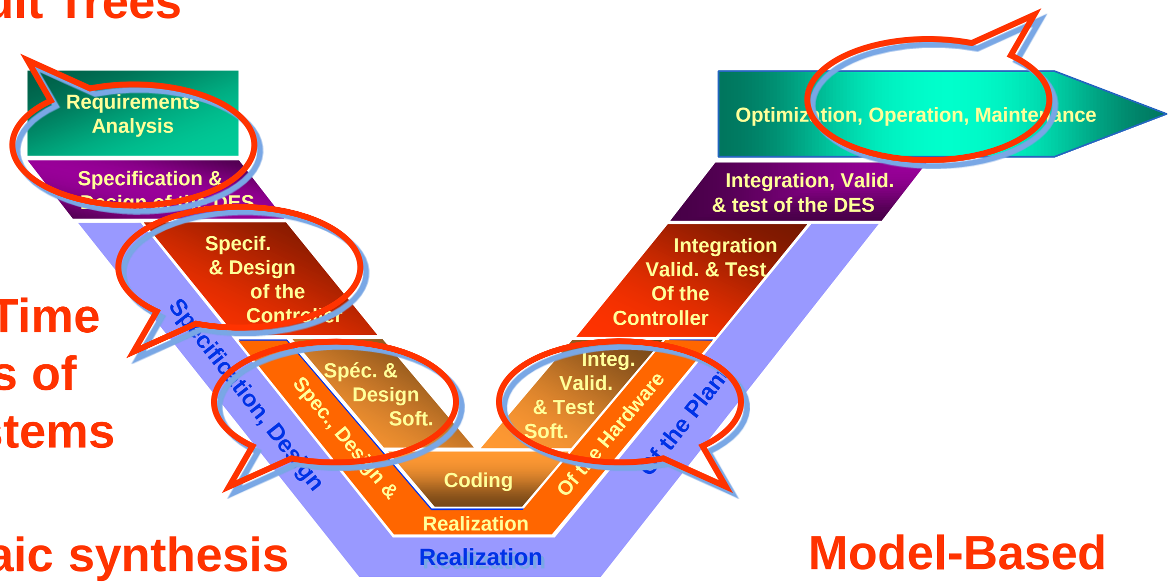
Off-line approaches (during specification, design, implementation and validation)

- Fault Prevention (Synthesis, …)
- Fault Forecasting (Fault Tree Analysis, …)
- Fault Tolerance (Physically or functionally redundant solutions, …)
- Fault Removal (Verification, Test, …)

Some recent PhD works

Algebraic Analysis of Dynamic Fault Trees

Model-Based Diagnosis

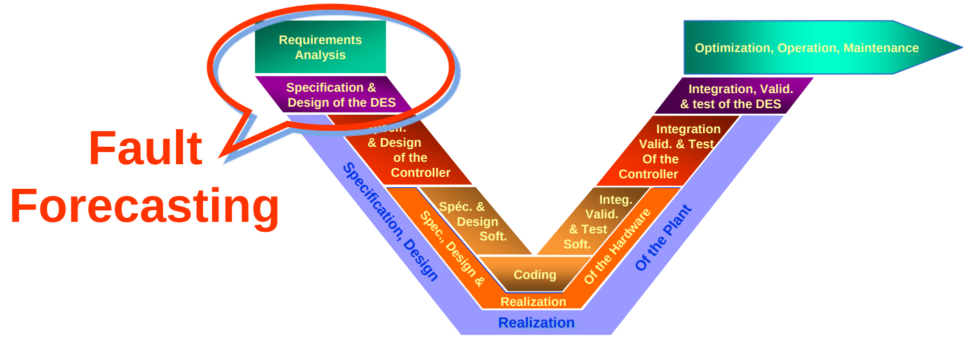


Evaluation of Time Performances of Networked Systems

Algebraic synthesis of logical controllers

Model-Based Conformance Test

Algebraic modeling and analysis of Dynamic Fault Trees

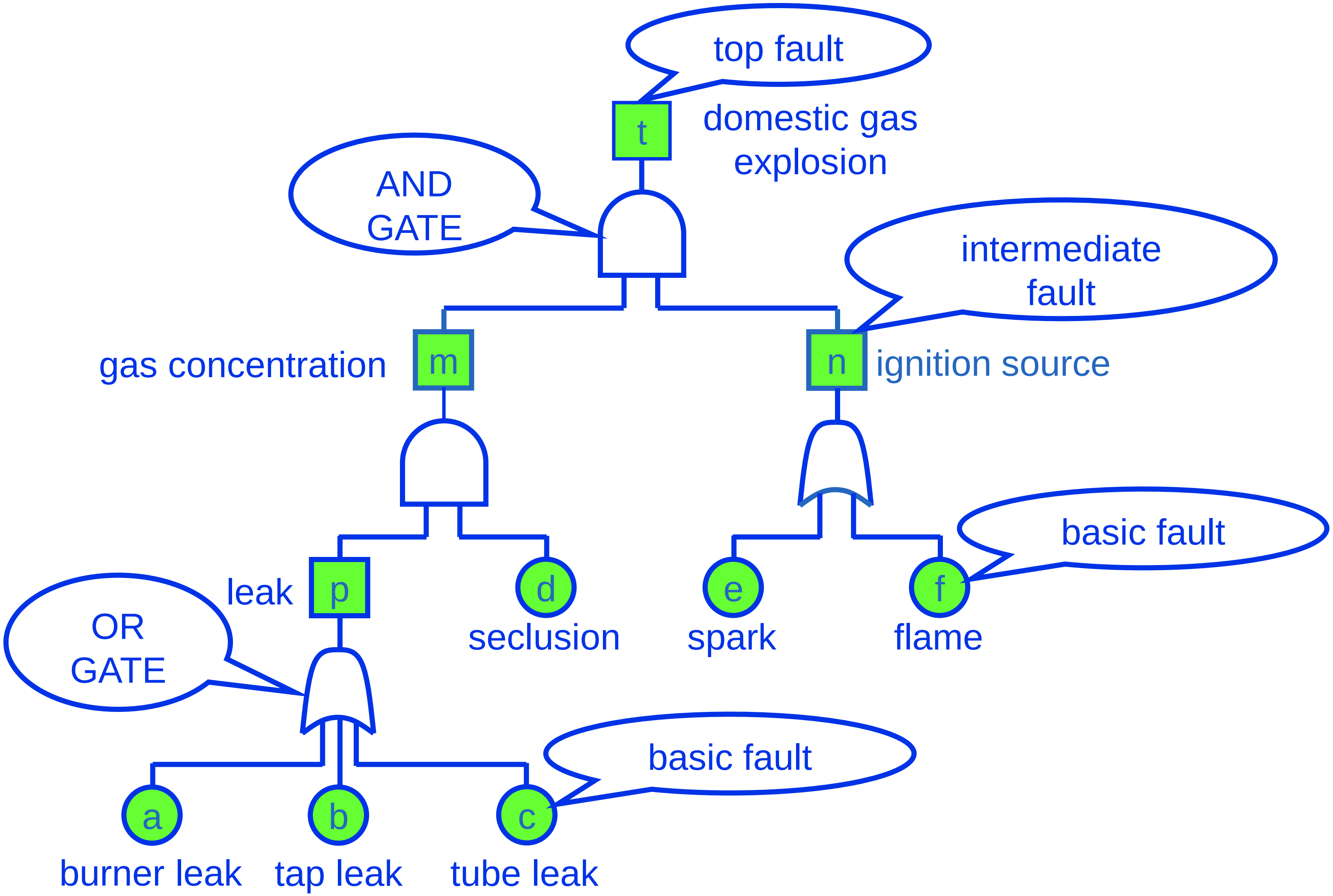


Guillaume MERLE

Jean-Marc ROUSSEL, Jean-Jacques LESAGE

La spirale, 1957 – Germaine Richier

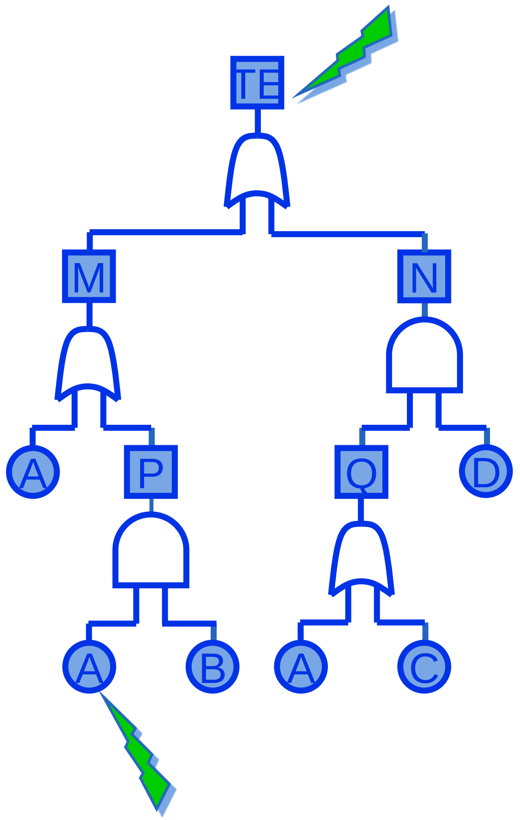
Fault Tree syntax



Fault Tree Analysis

The case of Static Fault Trees (gates: OR, AND, K/among/M)

Top fault (event)



Basic fault (event) : occurs with a given probability Pr{A}

BOOLEAN

ALGEBRA

STRUCTURE FUNCTION

TE= A + (A . B) + (A + C) . D
= A + (C . D)

Qualitative analysis (minimal cut sets)

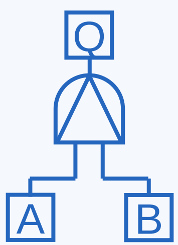
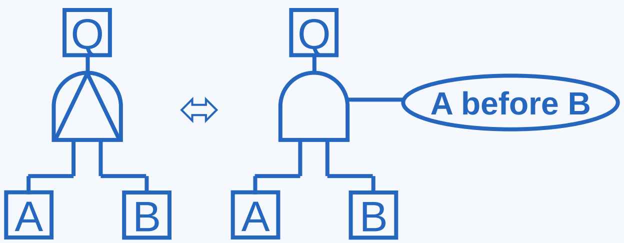
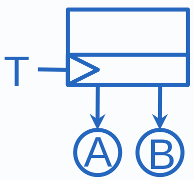
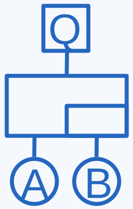
- direct: {A,C.D} (BDDs for complex SFTs)

Quantitative analysis (Pr{TE})

- direct: Pr{TE} = Pr{A + (C.D)}
  = Pr{A} + Pr{C.D}
  = Pr{A} + Pr{C} x Pr{D}

(evaluation methods for complex SFTs)

The case of Dynamic Fault Tree Analysis (SFT + gates PAND, FDEP and Spare)

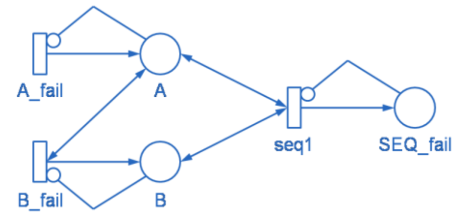
Symbol	Definition
PAND 	 [Fussel et al. 1976]
FDEP 	*Asserts a functional dependency – that the failure of the trigger event causes the immediate and simultaneous failure of the dependent basic events* [Dugan et al. 1992]
Spare 	*Output of gate occurs when the principal and all spares components have failed.* *2 states for each spare component (active/dormant) associated to 2 failure rates : $\lambda/\alpha\lambda$* *3 types of spares: Cold (α = 0), Warm (0 < α < 1), Hot (α = 1)* [Dugan et al. 2002]

No algebraic model for Dynamic Gates

- Structure Function of DFT undeterminable

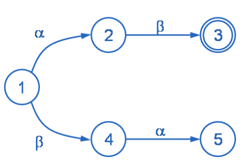
Qualitative analysis (minimal cut sequences)

- Extracted from Occurrence graph of SPNs

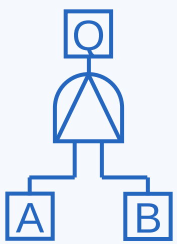
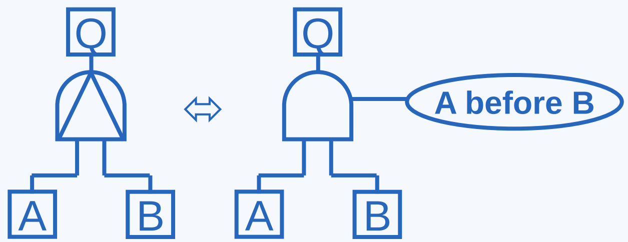
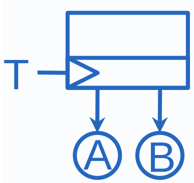
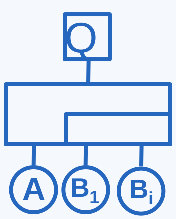


Quantitative analysis

- Continuous Time Markov Chains, Markov Decision Processes
- limited to exponential distributions, time consuming



Dynamic Fault Tree Analysis
(SFT + gates PAND, FDEP and Spare)

Symbol	Definition
PAND 	
FDEP 	*Asserts a functional dependency – that the failure of the* trigger event *causes the immediate and simultaneous failure of the* dependent basic events
Spare 	*Output of gate occurs when the principal and all spares components have failed.* *2 states for each spare component (active/dormant) associated to 2 failure rates : $\lambda/\alpha\lambda$* *3 types of spares: Cold (α = 0), Warm (0 < α < 1), Hot (α = 1)*

Dynamic gates expressing a priority:
- Sequential (PAND)
- Preemption-based (FDEP)

Needs: modeling of the order of occurrence of fault events

Results: Algebraic structure that allows determination of Structure Function and direct qualit. and quant. analysis

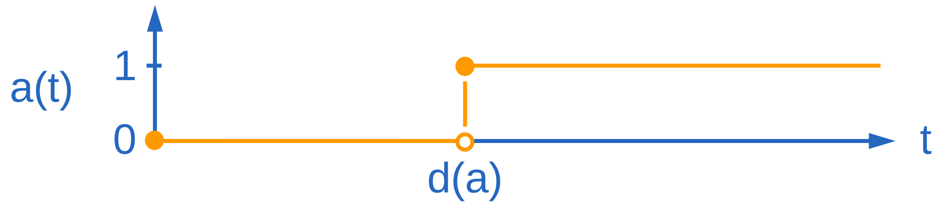
Dynamic gate expressing :
- Explicit duration of event
- Dependence between probabilities

(Pr{Bi} before A occurs < Pr{Bi] after A occurs)

Algebraic model of faults

- Defined on $\mathbf{R}^{+} \cup \{+\infty\}$ (faults = functions of time)
- Two values
  - 0: no fault
  - 1: fault
- non-repairable: single change of value

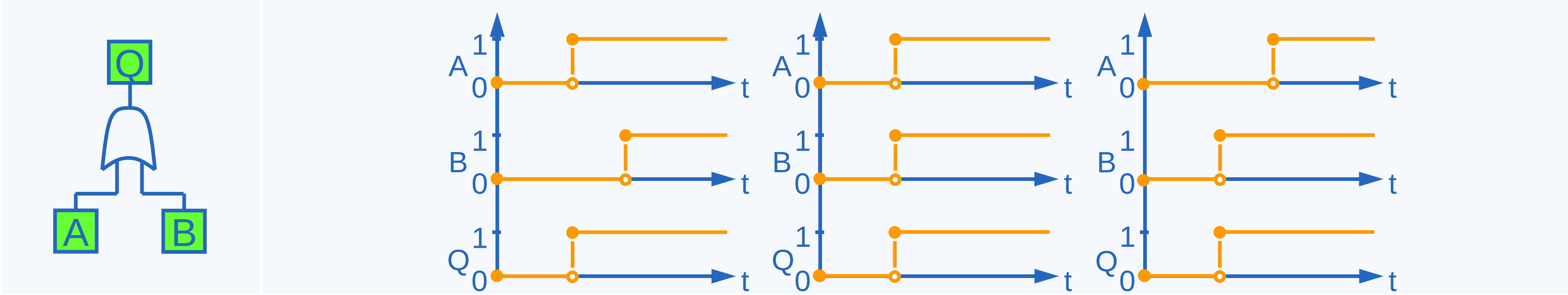
  $\Rightarrow$ date of occurrence d(a)



- Set of non-repairable faults F_{nr}
- Two specific faults $\perp$ and $\top$

Algebraic model of static gates (example OR gate)

- Expected behaviour



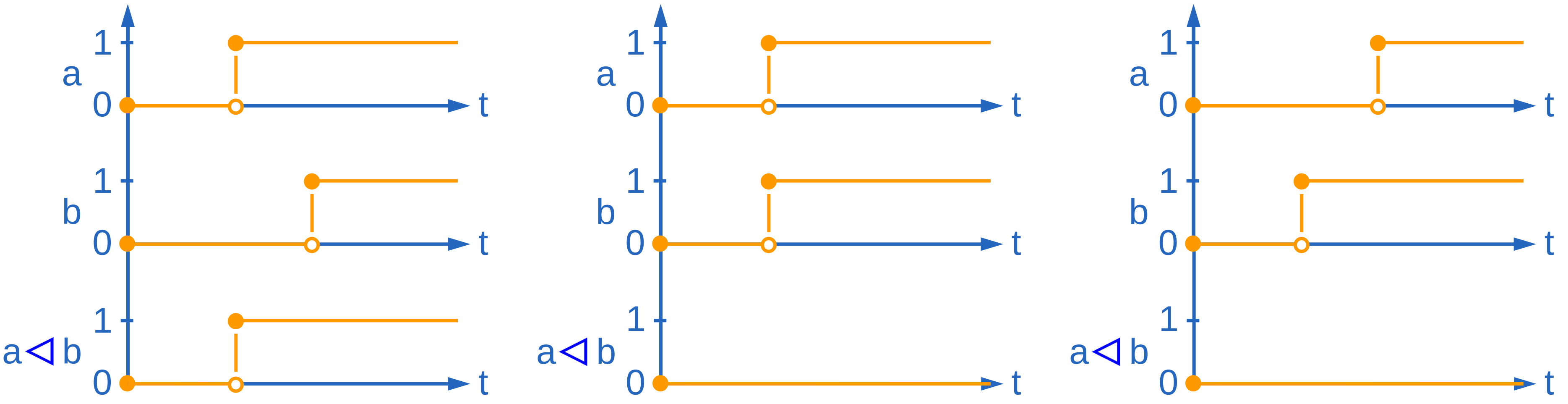
- Algebraic model

$$Q = A + B \qquad \text{with} \qquad \begin{aligned} + : F_{nr} \times F_{nr} &\to F_{nr} \\ (a,b) &\mapsto a+b \end{aligned} \qquad a+b = \begin{cases} a \text{ if } d(a) < d(b) \\ a \text{ if } d(a) = d(b) \\ b \text{ if } d(a) > d(b) \end{cases}$$

- $(F_{nr},+,.,\perp,\top)$ is an abelian dioïd, like $(\{0,1\},+,.,0,1)$
  - $\Rightarrow$ common theorems of Boolean algebra usable
  - $\Rightarrow$ structure function of static fault trees is determinable and simplifiable

Algebraic model of BEFORE operator

Expected behavior:

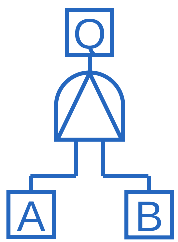
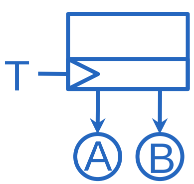
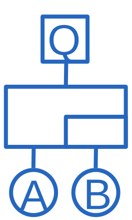


Algebraic model:

$$\triangleleft : F_{nr} \times F_{nr} \to F_{nr}$$
$$(a,b) \mapsto a \triangleleft b$$

$$a \triangleleft b = \begin{cases} a & \text{if } d(a) < d(b) \\ \perp & \text{if } d(a) \geq d(b) \end{cases}$$

Behavioral & probabilistic model of dynamic gates

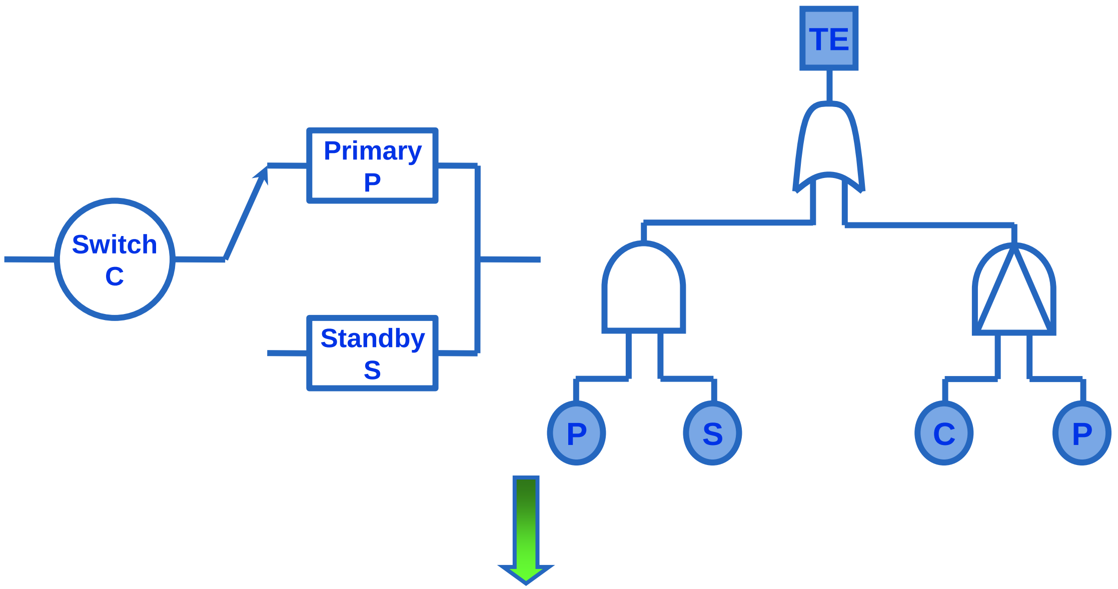
Gate symbol	Behavioral model	Probabilistic model
	$Q = (A.B).(A \trianglelefteq B)$ $= B.(A \trianglelefteq B)$	$\Pr\{Q\}(t) = \int_0^t f_B(u) F_A(u)\,du$
	$A_T = (A \trianglelefteq T) + T = A + T$ $B_T = (B \trianglelefteq T) + T = B + T$	$\Pr\{A_T\}(t) = F_A(t) + F_T(t) - F_A(t) \times F_T(t)$ $\Pr\{B_T\}(t) = F_B(t) + F_T(t) - F_B(t) \times F_T(t)$
	$\begin{cases} Q = B_a.(A \triangleleft B_a) + A.(B_d \triangleleft A) \\ B_d.B_a = \perp \end{cases}$ B may be active or dormant	$\Pr\{Q\}(t) = \int_0^t \left(\int_v^t f_{B_a}(u,v)\,du \right) f_A(v)\,dv + \int_0^t F_{B_d}(u) f_A(u)\,du$

Some theorems for development and simplification

Simplification Theorems		Development Theorems
$f+g=g+f$	$f+(f.g)=f$	$f+(g.h)=(f+g).(f+h)$
$f.g=g.f$	$f.(f+g)=f$	$f.(g+h)=(f.g)+(f.h)$
$f+(g+h)=(f+g)+f$	$f+\perp=f$	
$f.(g.h)=(f.g).f$	$f.\mathrm{T}=f$	
$f+f=f$	$f+\mathrm{T}=\mathrm{T}$	
$f.f=f$	$f.\perp=\perp$	

Simplification Theorems		Development Theorems
$f+(f\triangleleft g)=f$	$f\triangleleft f=\perp$	$f\triangleleft(g+h)=(f\triangleleft g).(f\triangleleft h)$
$g+(f\triangleleft g)=f+g$	$f\triangleleft\perp=f$	$(f+g)\triangleleft h=(f\triangleleft h)+(g\triangleleft h)$
$f.(f\triangleleft g)=f\triangleleft g$	$\perp\triangleleft f=\perp$	$f\triangleleft(g.h)=(f\triangleleft g)+(f\triangleleft h)$
$f+((f\triangleleft g).h)=f$	$f\triangleleft\mathrm{T}=\perp$	$(f.g)\triangleleft h=(f\triangleleft h).(g\triangleleft h)$
$(f\triangleleft g).(g\triangleleft h).(f\triangleleft h)=(f\triangleleft g).(g\triangleleft h)$		$(f\triangleleft g)\triangleleft h=(f\triangleleft g).(f\triangleleft h)$
$(f\triangleleft g).(g\triangleleft f)=\perp$		

Another example [Fussel, 1976]



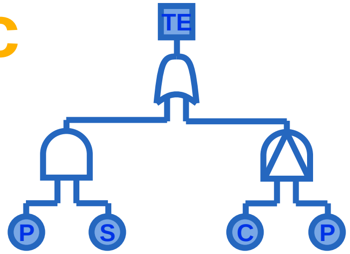
STRUCTURE FUNCTION

TE = (P . S) + (P . (C ◁ P))

Qualitative analysis

minimal cut sequences: {P,S},{S,P} and {C,P}

Quantitative analysis using our algebraic method (1)



quantitative analysis:

$$
\begin{aligned}
TE = (P.S) + P.(C \lhd P) \Rightarrow \Pr\{TE\} &= \Pr\{(P.S) + P.(C \lhd P)\} \\
&= \Pr\{P.S\} + \Pr\{P.(C \lhd P)\} - \Pr\{(P.S).(P.(C \lhd P))\} \\
&= \Pr\{P\} \times \Pr\{S\} + \Pr\{P.(C \lhd P)\} - \Pr\{S.(P.(C \lhd P))\} \\
&= \Pr\{P\} \times \Pr\{S\} + \Pr\{P.(C \lhd P)\} - \Pr\{S\} \times \Pr\{P.(C \lhd P)\} \\
&= \Pr\{P\} \times \Pr\{S\} + (1 - \Pr\{S\}) \times \Pr\{P.(C \lhd P)\}
\end{aligned}
$$

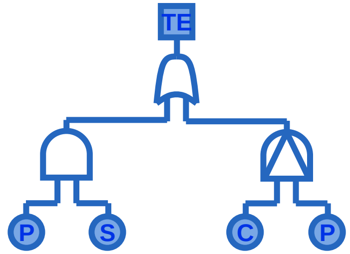
Pr{P} and Pr{S} are known whatever the distribution

Pr{P.(C◁P)} is known from the distributions of C and P

$$\forall (A,B), \Pr\{B.(A \lhd B)\} = \int_0^t f_B(u) F_A(u)\, du, \quad F_A(t) = \int_0^t f_A(v)\, dv$$

the method does not depend on the distribution

Quantitative analysis (2)



quantitative analysis for an exponential distribution:

$$\Pr\{P\}(t)=\int_0^t f_p(u)\,du=\int_0^t \lambda_p e^{-\lambda_p u}du=1-e^{-\lambda_p t}$$

$$\Pr\{S\}(t)=\int_0^t f_s(u)\,du=\int_0^t \lambda_s e^{-\lambda_s u}du=1-e^{-\lambda_s t}$$

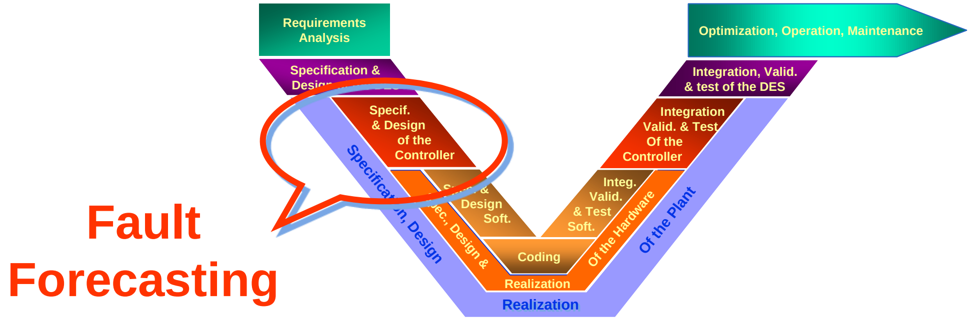
$$\Pr\{P.(C\lhd P)\}(t)=\int_0^t f_p(u)F_c(u)\,du=\int_0^t \lambda_p e^{-\lambda_p u}\left(1-e^{-\lambda_c u}\right)du$$

$$=\frac{\lambda_p}{\lambda_c+\lambda_p}e^{-(\lambda_c+\lambda_p)t}-e^{-\lambda_p t}+\frac{\lambda_c}{\lambda_c+\lambda_p}$$

$$\boxed{\begin{aligned}\Pr\{TE\}(t)&=\Pr\{P\}(t)\times\Pr\{S\}(t)+\left(1-\Pr\{S\}(t)\right)\times\Pr\{P.(C\lhd P)\}(t)\\&=\frac{\lambda_p}{\lambda_c+\lambda_p}e^{-(\lambda_c+\lambda_p+\lambda_s)t}-e^{-\lambda_p t}-\frac{\lambda_p}{\lambda_c+\lambda_p}e^{-\lambda_s t}+1\end{aligned}}$$

More detail: **Probabilistic Algebraic Analysis of Fault Trees with Priority Dynamic Gates and Repeated Events**, *G. Merle, J.-M. Roussel, J.-J. Lesage*, A. Bobbio, IEEE Trans. on Reliability, 59(1), pp. 250-261, March 2010

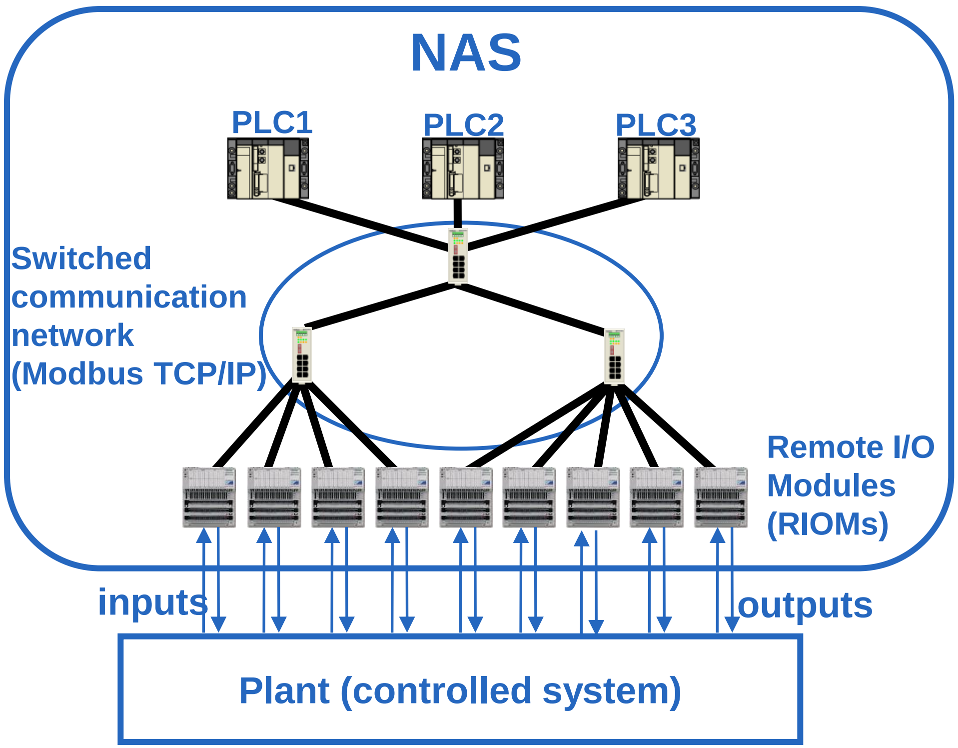
Evaluation of Time Performances of Networked Automation Systems by Iterative Proofs of Logic Properties

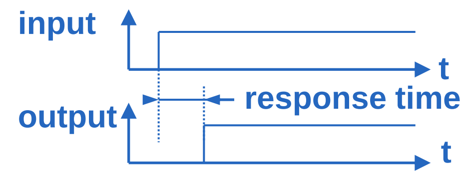


Silvain RUEL,

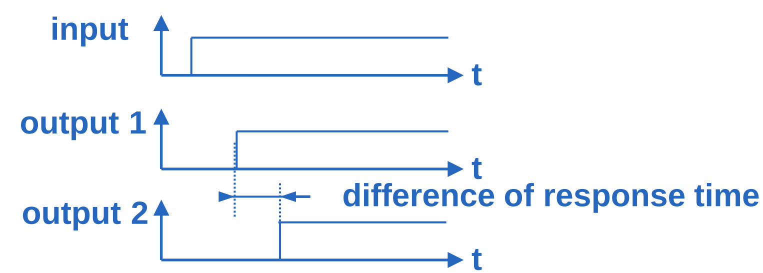
Olivier DE SMET, Jean-Marc FAURE

Time performances of networked automation systems





Response time: delay between an input event and the resulting output event



Difference of response time: delay between two output events resulting from the same input event

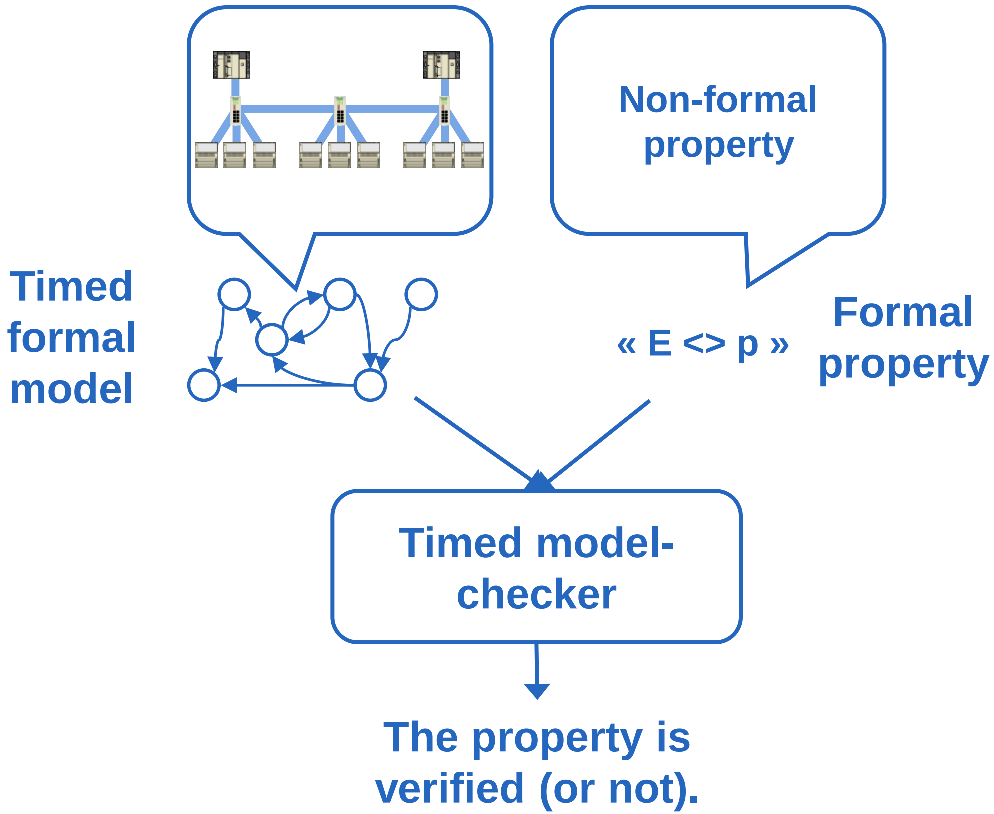
Aim of this study

- **Simulation techniques (based on analysis of Petri nets models of the NAS or on specific network simulator):**
  - **are non-exhaustive**
  - **then can provide a distribution of a time performance but not the bounds of this distribution.**

- **Is it possible to obtain these bounds by formal verification of timed models?**
  - **Exhaustive analysis technique**

- **Two issues to solve:**
  - **How to obtain numerical values from results of verification of logic properties?**
  - **How to avoid (limit) combinatory explosion?**

Formal verification of timed models



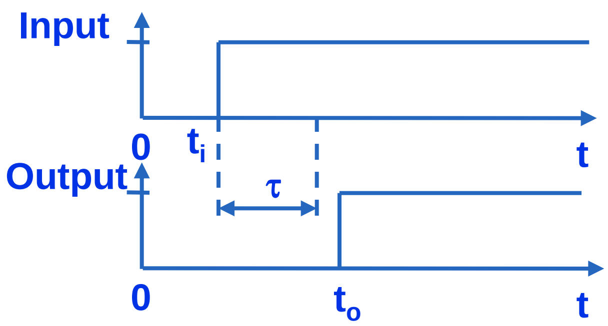
Five kinds of formal properties with the selected model-checker (UPPAAL)

- E<>p (possibility)
- E[ ]p (potentially always)
- A<>p (eventually)
- A[ ]p (invariantly)
- p→q (leads to)

Only logic properties can be checked; it is not possible to obtain a numerical value at the end of the verification

Contribution (1): parametric observer automaton

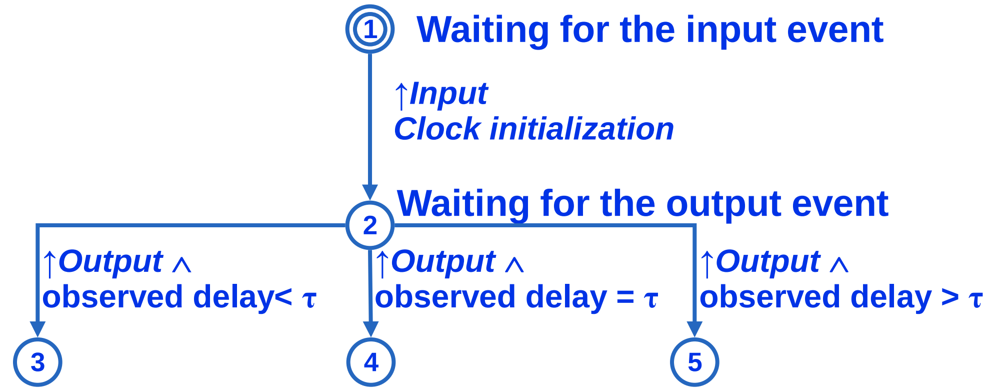
Basic idea



Three cases

- $t_o - t_i < \tau$
- $t_o - t_i = \tau$
- $t_o - t_i > \tau$

Parametric observer automaton structure



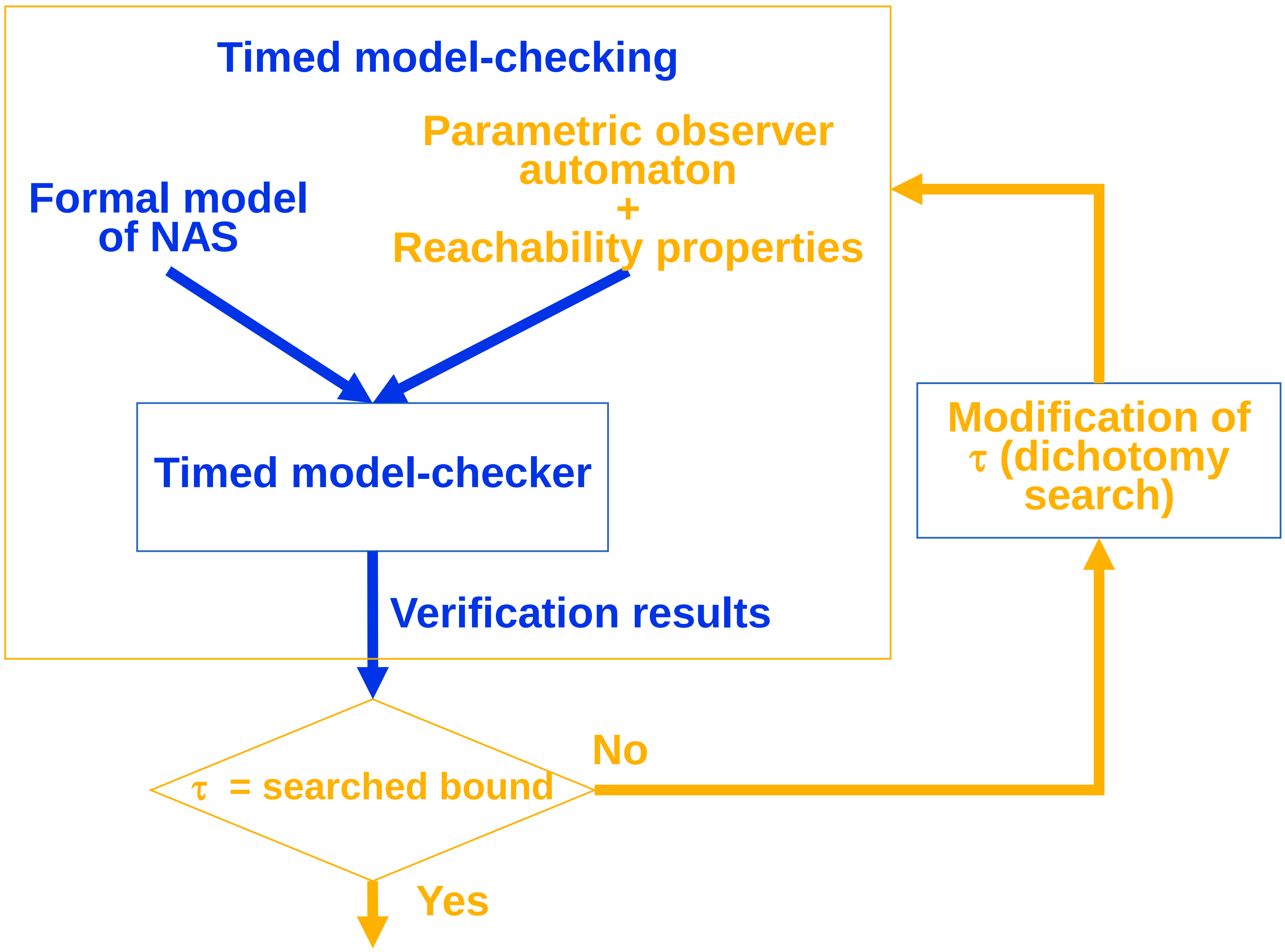
Associated reachability properties

- P1 : E<> OBS.3
- P2 : E<> OBS.4
- P3 : E<> OBS.5

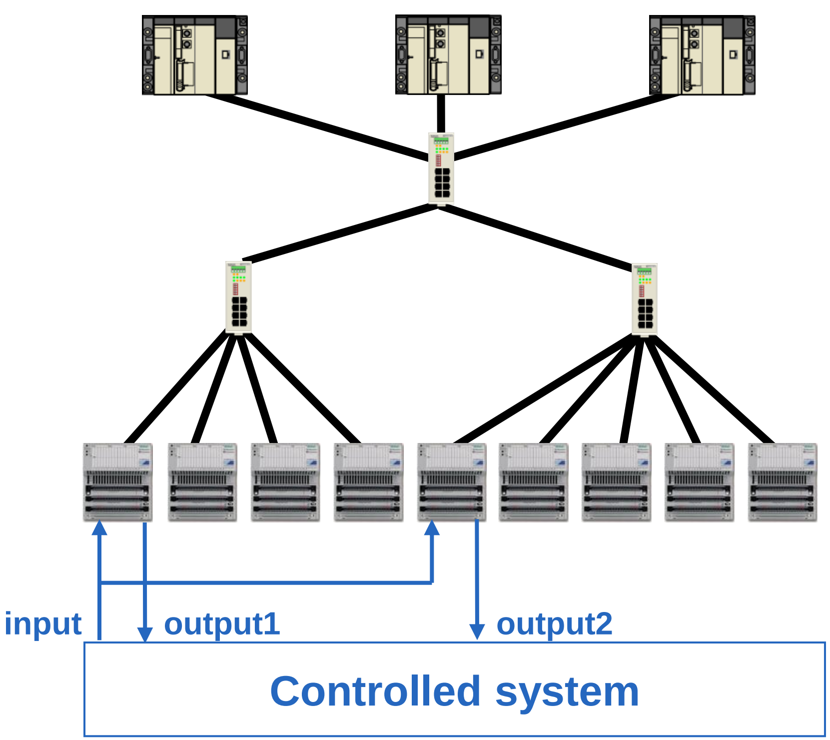
There exists at least one execution such that the state 3 (resp. 4, 5) is reached.

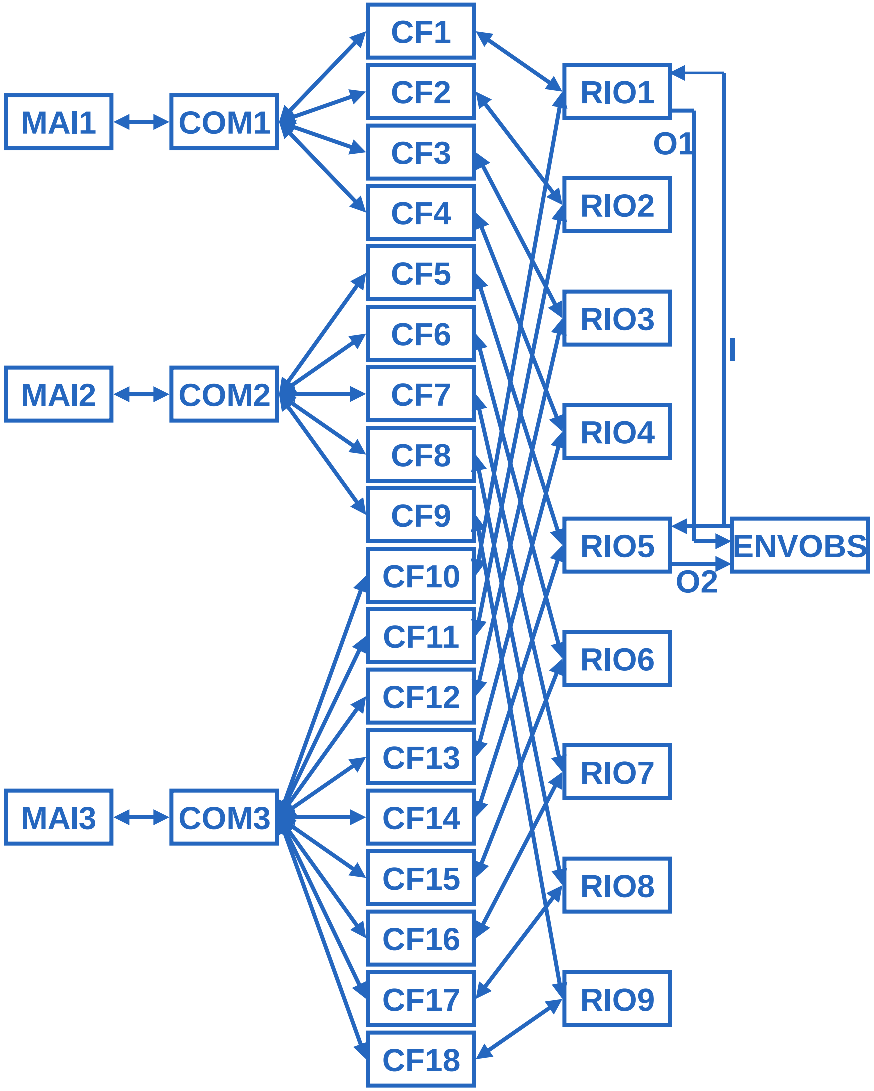
- τ is the upper bound iff P1 and P2 are verified and P3 is not verified
- τ is the lower bound iff P2 and P3 are verified and P1 is not verified

Contribution (2): Iterative proofs of logic properties



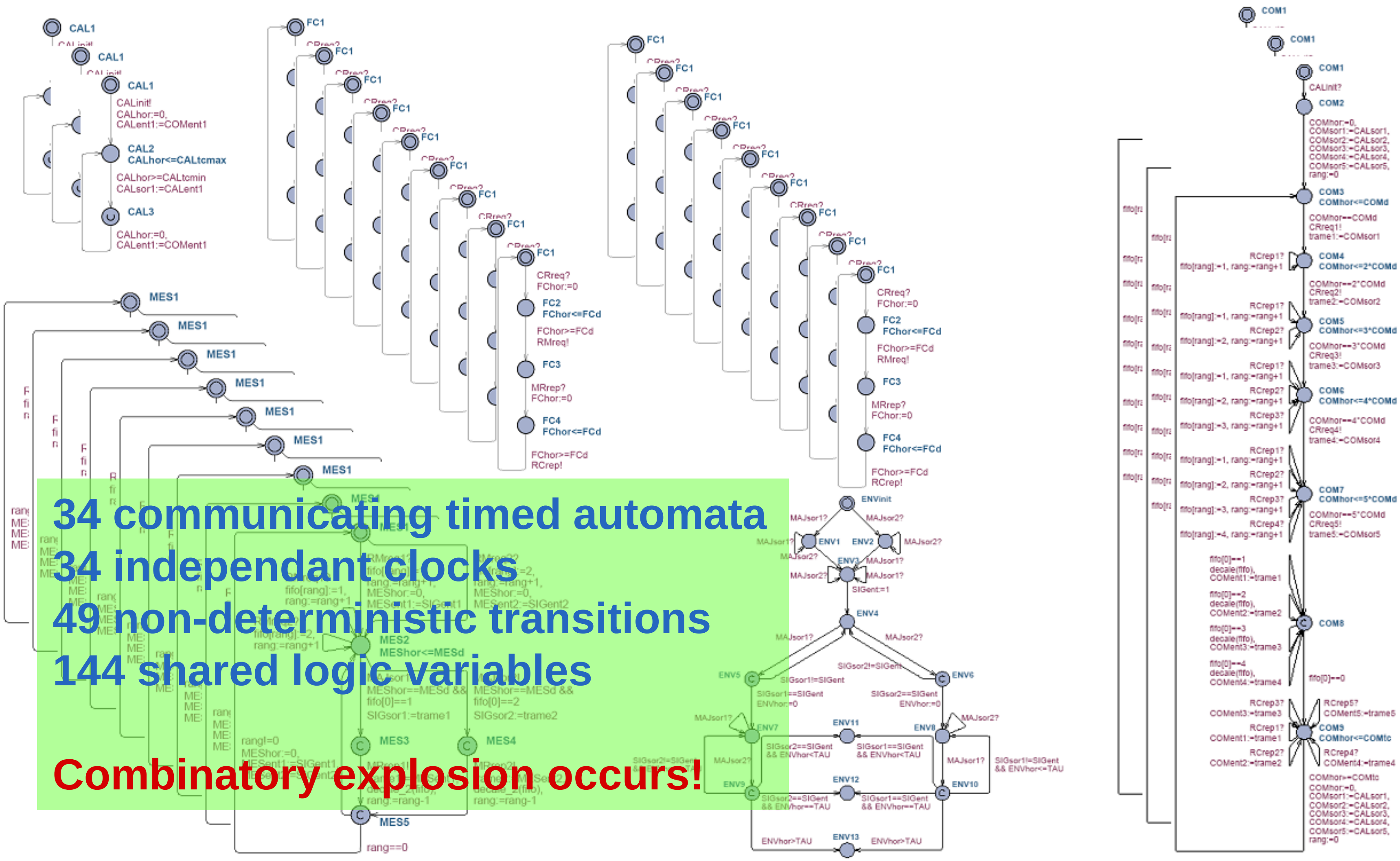
Scalability





Structure of the formal model:
A network of 34 communicating timed automata

Scalability



34 communicating timed automata
34 independant clocks
49 non-deterministic transitions
144 shared logic variables

Combinatory explosion occurs!

Two-stepped abstraction method

Initial model of NAS

↓

Simplification of the structure of the model (some components models are removed) ← **Time performance to study**

↓

Simplified model

↓

Modification of the models of the remaining components (to keep the impact of the removed models)

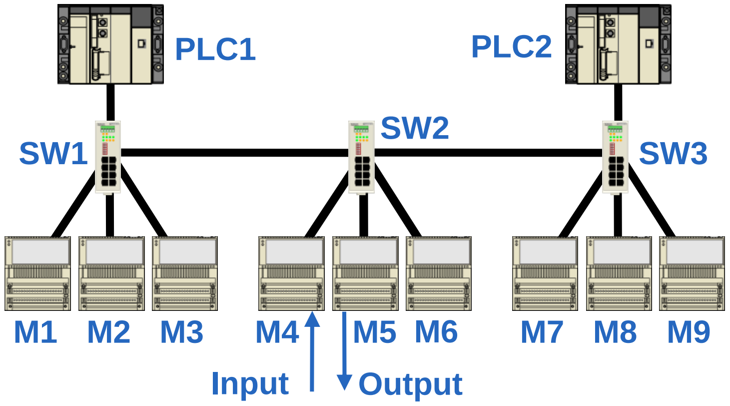
↓

Final abstract model

- **Abstraction is conservative w.r.t. the studied time performance**
- **No more combinatory explosion**
- **Verification results obtained in reasonable times (some seconds to some minutes)**

For more details:
Ruel S. et al. Building effective formal models to prove time properties of networked automation systems. WODES'08, pp. 334-339, Göteborg (Sweden), May 2008

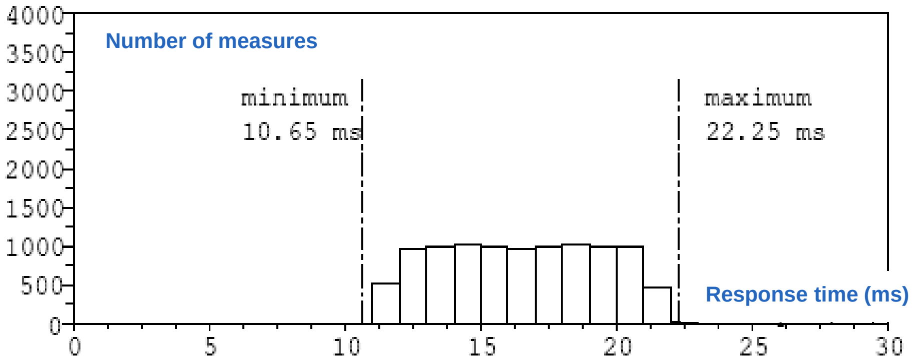
Experimental validation



Results obtained by iterative proofs

- RTmin = 9.49 ms
- RTmax = 23.13 ms

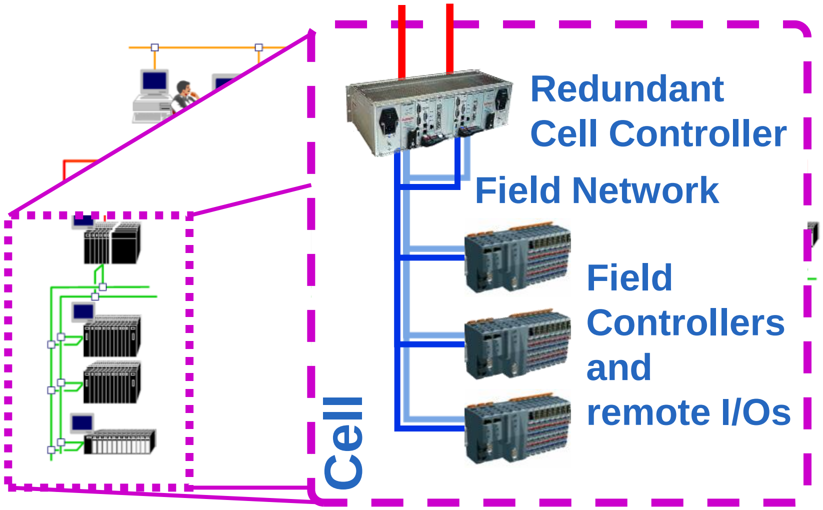
Measures



- **All the measured values are within the computed bounds.**
- **Small differences between the computed bounds and the minimum/maximum values of the distribution: 11% for the lower bound, 4% for the upper bound**

Other works on networked automation systems

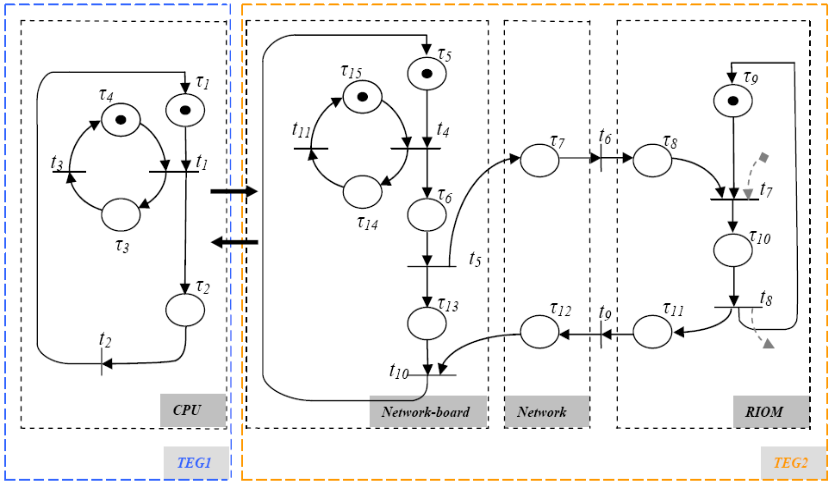
- **Formal verification of properties of redundant Ethernet Powerlink**
  - **Cooperation with Alstom Power (PhD work of Steve Limal)**
  - **Examples of properties**
    - **Each message is transmitted even if one medium fails.**
    - **Each failure must be detected.**
    - **The redundant extension must not trigger the CSMA/CD mechanism.**



Automation cell

Other works on networked automation systems

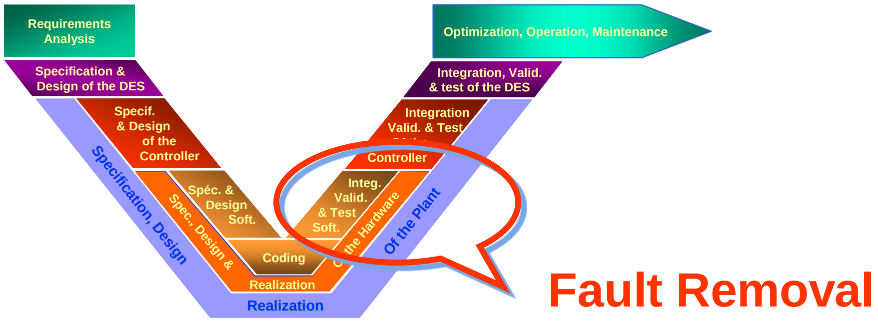
- **Analytic evaluation of the response time using the (max,+) algebra**
  - **PhD work of Boussad Addad)**
  - **The system is modeled as a set of Timed Event Graphs (Petri nets where every place has at most one upstream and one downstream transition).**
  - **The distribution and its bounds can be obtained from the analytic expression.**



For details:
Analytic Calculus of Response Time in Networked Automation Systems,
B. Addad, S. Amari, J-J. Lesage, IEEE Trans. on Automation Science and Engineering Vol. 7, Issue. 4, pp. 858-869, 2010.

LURPA

Conformance test of logic controllers from specifications in Grafcet language



Julien PROVOST,

Jean-Marc ROUSSEL, Jean-Marc FAURE

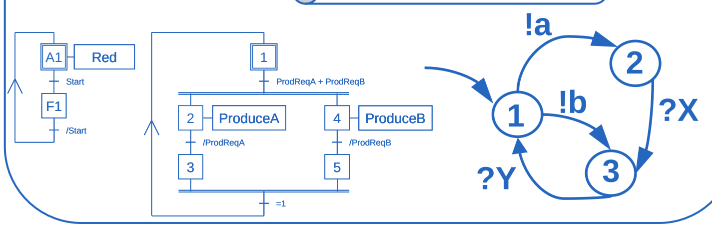
(In the frame of the TESTEC (Test of critical real-time embedded systems) project funded by the French Research Agency)

Aim of conformance test

Check whether an implementation, seen as a black-box with inputs-outputs, behaves correctly with respect to its specification

Specification

Textual description of the expected behavior



Implementation

Code + OS

PLC

or

ECU

or electronic board, VLSI circuits, communication protocol…

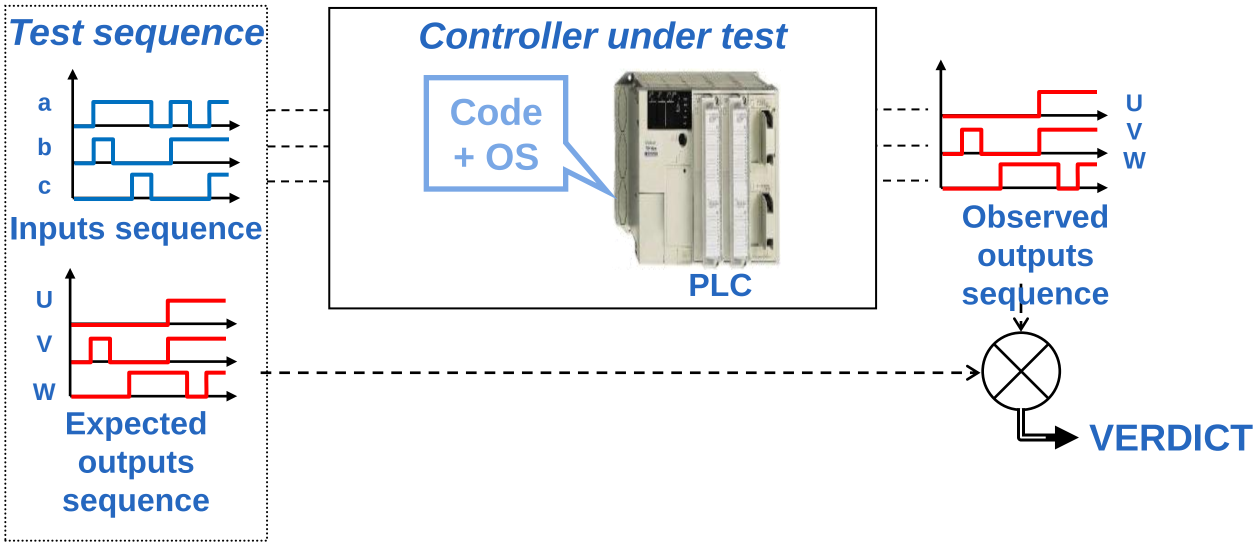
Conformance test

Verdict (conform/not conform) + example if not conform

Conformance test execution

The implementation under test is connected to a test-bench which generates an inputs sequence.

The observed outputs sequence is compared to the expected one.



How to build automatically the test sequence from the specification?

Constraints of this study

- **Specification in Grafcet language (IEC 60848 standard)**
- **Conformance test must be complete: every evolution from every state of the specification must be tested**
- **Non-invasive test**
- **Automatic construction of the test sequence**

- **Only non-timed models are considered**

2 scientific issues / 2 scientific contributions

How to obtain a formal model from a Grafcet specification?

- The Grafcet standard provides only textual descriptions of the evolution rules.
- A formal model is mandatory to build a complete test sequence.

A formal semantics of Grafcet in the form of FSM (Mealy machine)

- Parallel and transient evolutions are taken into account
- Relies on an intermediary model: Stable Location Automaton

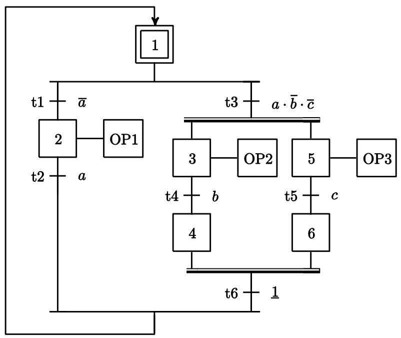
How to build a test sequence suitable for controllers with cyclic I/O scanning?

Definition of the SIC-testability concept

- To prevent from spurious test results

From Grafcet to FSM

Grafcet



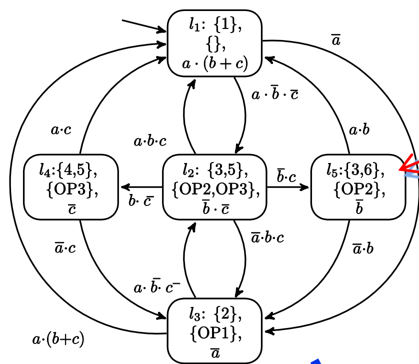
Evolution and stability conditions consistency

$$\forall l \in L$$

$$\left(\prod_{s\in S_{Act}(l)} X(s)\right)\cdot\left(\prod_{\substack{s\in S_G(g)\\ s\notin S_{Act}(l)}} \overline{X(s)}\right)\cdot E_{Stab(I_G)}(l)$$

$$\cdot\left(\sum_{\substack{t\in T_G(g)\\ S_U(t)\subset S_{Act}(l)}} E_{Cond(I_G,S_G)}(t)\right) = 0$$

Stable Location Automaton



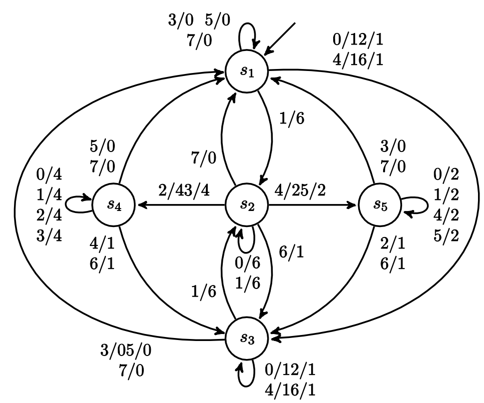
Set of active steps
Set of emitted outputs
Stability condition

Transition function of the Mealy machine

$$\forall s \in S_M, \forall i \in I_M$$

$$\left[\begin{array}{l} \text{if } \exists e \in Evol \left[\left\{\begin{array}{l} l_U(e) = s \\ E_{Evol(I_G)}(e)\cdot m_I(i) = m_I(i) \end{array}\right.\right] \\ \quad [\delta_M(s,i) = l_D(e)] \\ \text{else} \\ \quad [\delta_M(s,i) = s] \end{array}\right]$$

FSM

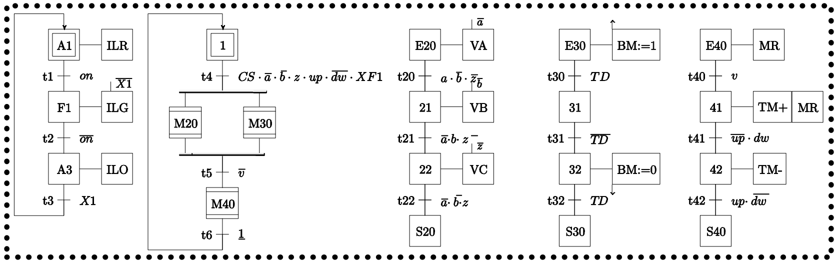


$n_{states} = n_{locations}$

$n_{transitions} = n_{locations} \cdot 2^{n_{IV}}$

Some figures

Grafcet



- 9 inputs and 10 outputs
- Several sub-graphs: 16 steps and 15 transitions
- Transition conditions are defined by Boolean expressions

850 ms

Stable Location Automaton

- Same numbers of inputs and outputs
- State machine: 64 locations and 389 evolutions
- Evolution conditions are defined by Boolean expressions

350 ms

FSM

- 2^9 input alphabet and 2^{10} output alphabet elements
- State machine: 64 states and 32,768 transitions
- Every transition is labeled by a couple (input,output)

Provost, J., et al. Translating Grafcet specifications into Mealy machines for conformance test purposes.
Control Engineering Practice (2010),
doi:10.1016/j.conengprac.2010.10.001

Test sequence construction from the equivalent FSM

The test sequence must be:

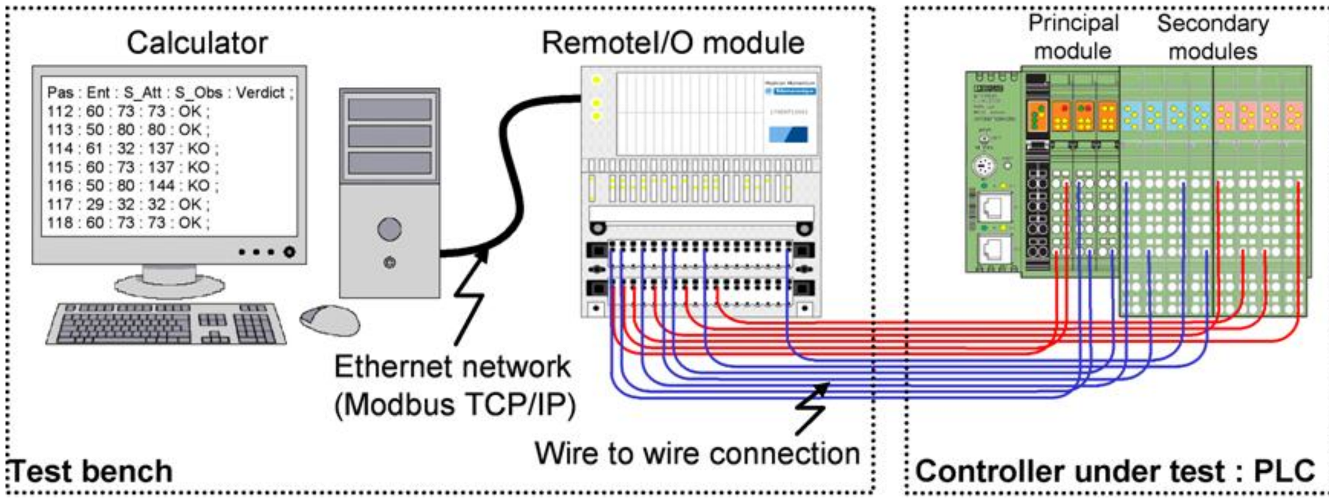
- **Initializable**: the first test step corresponds to a transition that starts from the initial state
- **Complete**: every transition must be tested at least once

Two optimization criteria

- **Classical approach**: minimization of the length (number of test steps)
  - Transition-Tour method, variant of the Chinese Postman problem
  - For the previous example, this sequence comprises 73,528 test steps and is computed in less than 2 s
  - Erroneous verdicts may occur with logic controllers with cyclic I/O scanning

- **Our proposal**: minimization of the number of MIC test steps
  - To avoid the previous issue
  - Definition of the concept of SIC-testability

Conformance test execution experiments with minimum-length sequences



First program

Correct program, model-checked

→Test bench may reject the program.
False errors are sometimes declared.

→Biased results

Second program

Erroneous program, with intentionally added errors

→Test bench may accept the program.
All errors are not always detected.

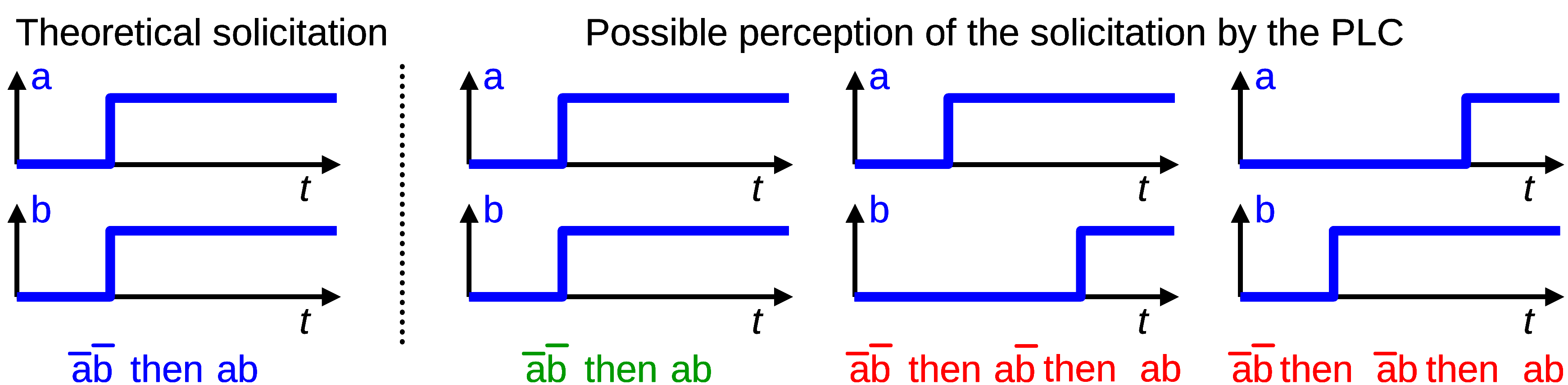
→Non-valid results

→ Lack of confidence in the conformance test verdicts

Results analysis

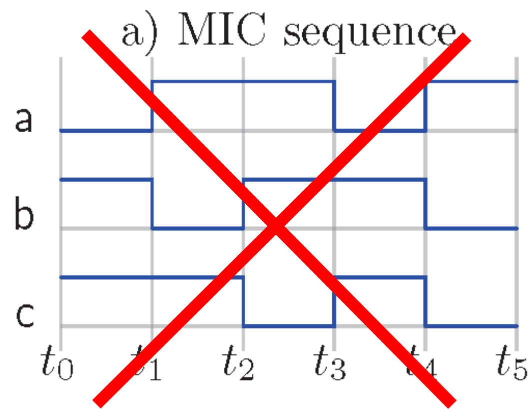
All verdict errors occur when several input values are changed simultaneously.

Synchronous events generated by the test-bench are seen as asynchronous by the implementation under test.

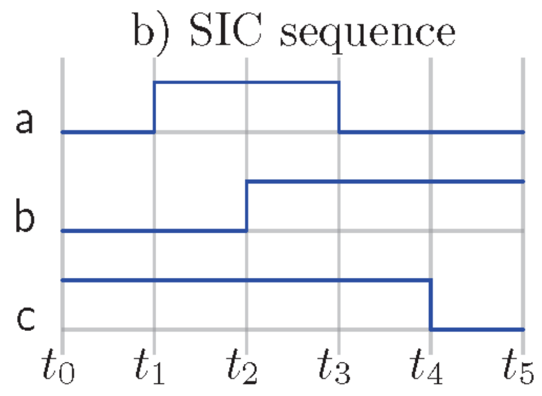


Solutions

- **Synchronize the test-bench and the controller under test**
  - Not always easy and not realistic (a plant is not synchronized with its controller)
- **Privilege SIC test sequences**



t_i



MIC (Multiple Input Changes) test sequence
Several logical inputs can change at the same date t_i

SIC (Single Input Change) test sequence
Only one logical input changes at any date t_i

Proposed method to build test sequences

- **Is the specification SIC-testable? (an initializable, complete and SIC test sequence can be built from the specification)**
  - If the specification is not SIC-testable, determine its SIC-testable part.

- **Build the test sequence**
  - For a SIC-testable specification, this sequence is obtained by solving a Traveling Salesman problem on a specific graph whose nodes are couples (state, inputs valuation)
  - For a non-SIC-testable specification, this sequence is composed of a SIC sequence to test its SIC-testable part followed by a MIC sequence to test the remaining transitions
  - For more details:

Provost, J., et al. SIC-testability of sequential logic controllers. WODES 2010, Berlin, pp. 203-208, August 30 - September 1, 2010

Provost, J., et al. Testing Programmable Logic Controllers from Finite State Machines specification. DCDS'11, pp. 3-8, Saarbrücken, Germany, June 15-17, 2011

Checking SIC-testability of a specification (1)

A SIC relation is defined between two inputs valuations

- v_I and v'_I satisfy a SIC relation: they differ in only one input

$$dim((v_I \backslash v'_I) \cup (v'_I \backslash v_I)) = 1$$

- Notation: $v'_I \ R_{Gray} \ v_I$

$$dim((v_I \backslash v'_I) \cup (v'_I \backslash v_I)) = 1 \Leftrightarrow v'_I \ R_{Gray} \ v_I$$

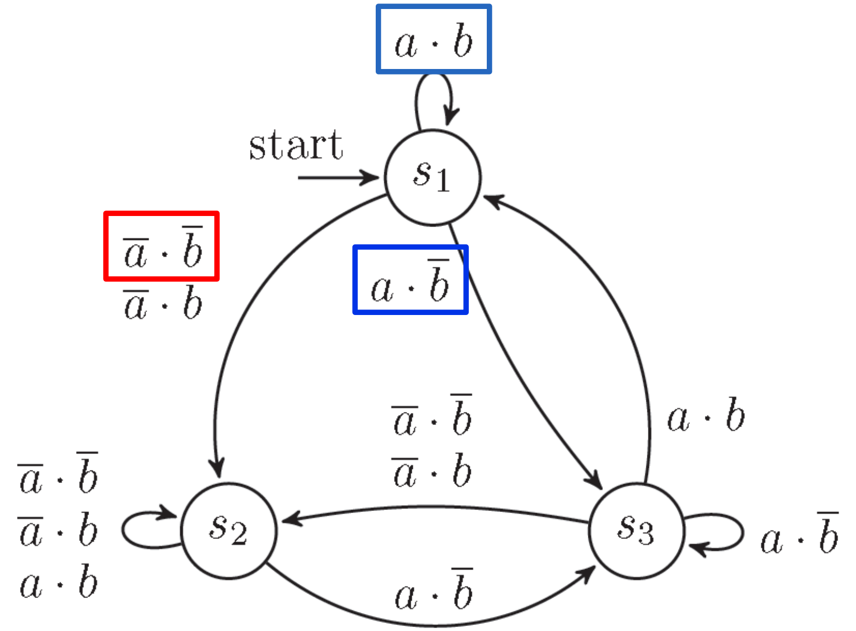
- Example

$$a \cdot b \quad R_{Gray} \quad a \cdot \overline{b} \quad ?$$

$$\begin{aligned} & dim((\{a,b\} \backslash \{a\}) \cup (\{a\} \backslash \{a,b\})) \\ & = dim(\{a\} \cup \emptyset) \\ & = 1 \end{aligned}$$

$$a \cdot b \quad R_{Gray} \quad \overline{a} \cdot \overline{b} \quad ?$$

$$\begin{aligned} & dim((\{a,b\} \backslash \emptyset) \cup (\emptyset \backslash \{a,b\})) \\ & = dim(\{a,b\} \cup \emptyset) \\ & = 2 \end{aligned}$$



→ This specification is not SIC-testable

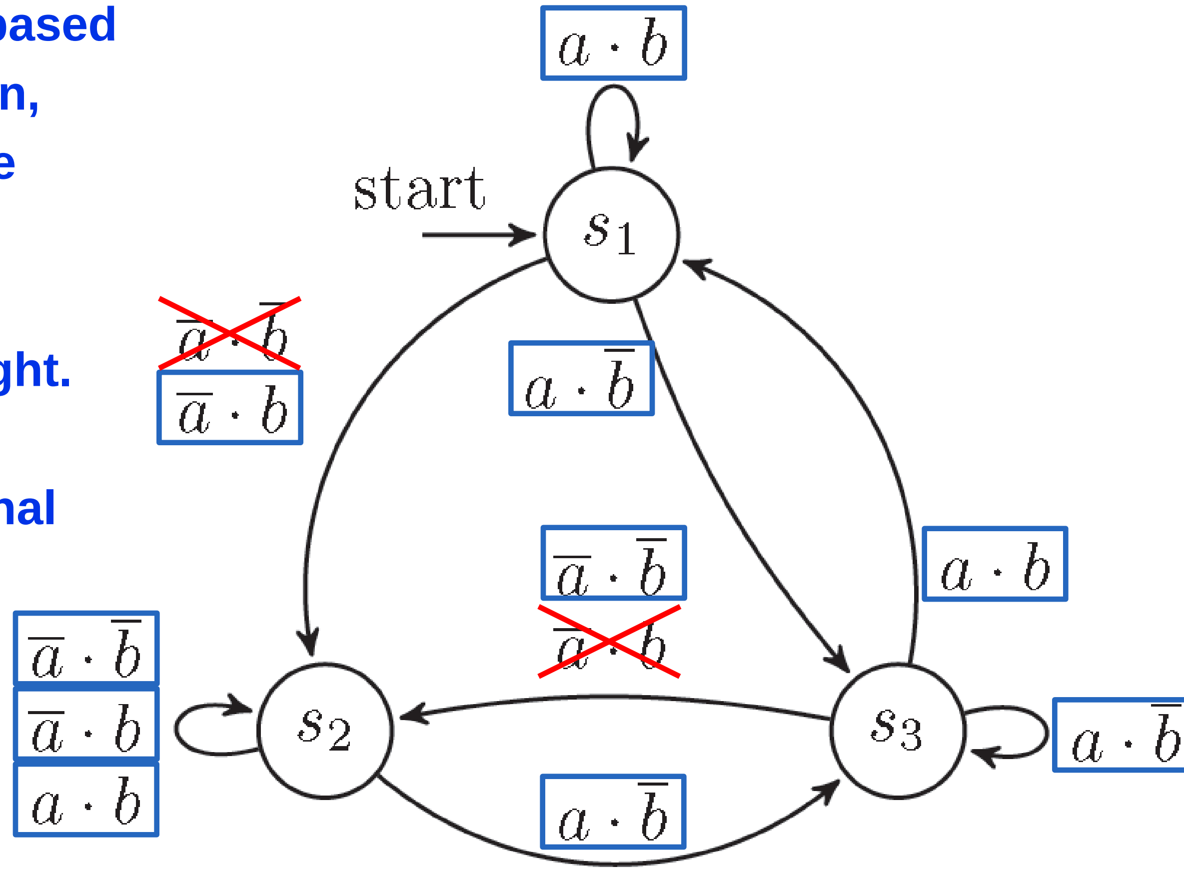
Checking SIC-testability of a specification (2)

SIC-testability checking is based on a fixed point computation, starting from the initial state

The SIC-testable part of the example is shown on the right.

Two transitions of the original model are not SIC-testable (cannot be included into a SIC test sequence):

$$(s_1, \overline{a} \cdot \overline{b})$$
$$(s_3, \overline{a} \cdot b)$$



A SIC test sequence can be generated for the SIC-testable part.

Conclusions

- **DES modeling and analysis techniques can definitely contribute to improve the dependability attributes (safety, security, availability, …) of automated systems**

- **However, be careful with:**
  - **Combinatory explosion when dealing with non-trivial systems**
    - **Abstraction mechanisms or algebraic approaches may lessen (remove) this issue**
  - **Construction of the formal models**
    - **A DES model may be mathematically sound but meaningless w.r.t. the real world**
  - **Industrial acceptance of the scientific results and scientific acceptance of the industrial constraints and practices (tailored-made languages, existing engineering environments, well-established know-how)**


La spirale, 1957 – Germaine Richier


Ecole Normale Supérieure de Cachan
photo : Bruno DENIS / LURPA-ENS CACHAN

Thank you !