Technical report on Conformance Test of Programmable Logic Controllers — Execution of Minimum-Length Test Sequences

Julien Provost, Jean-Marc Roussel, Jean-Marc Faure

Abstract

Conformance test is a functional test technique which is aiming to check whether an implementation, seen as a black-box with inputs/outputs, conforms to its specification. Numerous theoretical worthwhile results have been obtained in the domain of conformance test of finite state machines. The optimization criterion which is usually selected to build the test sequence is the minimum-length criterion. On the basis of several experiments, this technical report shows that this kind of test sequence may lead to erroneous test verdicts when the implementation is a Programmable Logic Controller (PLC) that executes a control program. The error rates that have been obtained for different configurations of the PLC under test are given as well as guidelines for conformance test of PLCs from finite state machines.

Index Terms

Conformance test, Formal Methods, Programmable Logic Controller, Test Sequence, Mealy machine, Test Verdict

I. Introduction

Programmable Logic Controllers (PLCs) are industrial automation components that are widely used to implement control functions, even in critical systems like power production and distribution, rail transport, chemical processes, water distribution, etc. This explains why numerous research works have been carried out since more than ten years to develop methods that avoid flaws to be introduced during the development of PLC software. These researches are based on two main approaches: model-based (model-driven) engineering [1]–[3] and formal verification and validation (V&V) techniques [4]–[7] or a combination of both [8]. Whatever the interest of the results obtained in these works, it must be noted that all of them are based on *models*. Formal V&V techniques for instance have been applied to models of the specification of the control logic, in the form of IEC 60848 models, state-charts, Signal Interpreted Petri Nets, Net Condition Event Systems [9]–[11] or of PLC programs, in IEC 61131-3 languages [12]–[15].

However, validation of a *real* PLC which executes a control program requires the conformance test of this component be performed, even if the specification and program models have been verified and validated, and a certified code generator has been used to produce the executable code. Conformance test is a functional test, i.e. the system under test, named implementation, is seen as a black-box (its internal structure is unknown) with observable inputs/outputs; the overall aim is to check whether this implementation behaves as specified (Figure 1). Conformance test of PLCs is advocated by certification bodies and standards [16], [17], which explains the growing interest of companies in several industrial domains for efficient hardware-in-the-loop techniques [18]–[20] to improve the existing practices.

A promising solution to develop such techniques is to benefit from the results of the researches of the Discrete Event Systems community in the domain of conformance test of formal models. In these works, the specification is a formal model: a Mealy (or finite state) machine [21], a transition system [22], [23], or a timed automaton [24] for instance. The first formalism has been selected for this study because it is well suited to the modeling of logic systems specifications; moreover, functional correctness must be tested before time correctness. Conformance of the implementation, which is assumed to be represented as another machine, to the specification requires a test objective be first defined. A classical test objective, when critical systems are considered, is to cross at least once each edge of the directed graph that represents the structure of the machine; this permits to check every state change from each state of the formal model. Then, the test sequence can be constructed from this model. Several algorithms have been developed to build test sequences that meet this objective; a solution that minimizes the length of the sequence, therefore the duration of the test if the duration of one test step is constant, was first presented in [25]. As industrial specifications are not expressed in formal languages but in standardized, tailored-made languages, translation rules of industrial specification languages into formal ones are to be developed in order to use these theoretical results for conformance test of PLCs; this issue has been solved in [26] where a method to obtain from a Grafcet [27] an equivalent Mealy machine is presented. The approach can be applied to other languages.

First version of the manuscript: February 5 2014; last update: March 31, 2014

J. Provost is with the Assistant Professorship for Safe Embedded Systems, Technische Universität München, DE-85748 Garching bei München, Germany. E-mail: julien.provost@tum.de

J.-M. Roussel and J.-M. Faure are with LURPA, ENS Cachan, FR-94230 Cachan, France. E-mail: jean-marc.roussel@lurpa.ens-cachan.fr, jean-marc.faure@lurpa.ens-cachan.fr

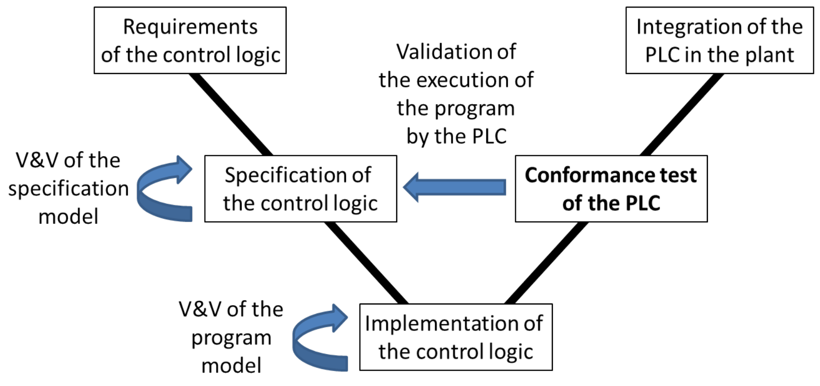

Requirements of the control logic
Validation of the execution of the program by the PLC
Integration of the PLC in the plant
V&V of the specification model
Specification of the control logic
Conformance test of the PLC
V&V of the program model
Implementation of the control logic

Fig. 1. Place of the work in the life-cycle of PLC software

Another issue remains, however: the optimization criterion which is usually selected to build the test sequence (minimum length) does not consider the technological features of the implementation. The aim of this technical report is to show, on the basis of several experiments, that using a minimum-length test sequence for conformance test of PLCs may lead to erroneous test verdicts, and to quantify the error rate for different configurations of the PLC under test.

The next section reminds how a minimum-length test sequence can be built from a specification expressed in the IEC 60848 Grafcet language. The execution of this sequence is addressed in the third section; as erroneous test verdicts can be yielded in some cases, two sets of experiments are presented to find the origin of this issue and to quantify the error rate. The obtained results are discussed and guidelines for conformance test of PLCs are given in the fourth section.

II. Background: test sequence generation from GRAFCET specification

The aim of the section is to define the notations used in this paper and to remind previous results. The notations and definitions will be illustrated through an example introduced in the following subsection.

A. *Example*

Several event-based formalisms, like state automata and finite state machines, have been proposed to model sequential logic systems; they have permitted to obtain worthwhile theoretical results in the domains of synthesis and analysis of DES (Discrete Event Systems). However, when design, implementation and test of PLCs are considered, it is often more convenient to select a modeling formalism whose inputs and outputs are logic variables, and not events, and where the transition conditions depend on combinations of input values. This is the case in this paper, where the specification of the PLC's behavior is represented by a Grafcet.

Grafcet is a standardized graphical specification language [27] which is widely used in several industrial domains, like railway transport, electrical power production, manufacturing industry, environment. A good scientific presentation of the main features of the Grafcet modeling framework can be found in [26], [28], [29]. A Grafcet model describes the expected behavior of a logic controller which receives logic input signals and generates logic output signals. A Grafcet (Figure 3) comprises steps, graphically represented by squares, and transitions, represented by horizontal lines; a step can only be linked to transitions and a transition only linked to steps. A step defines a partial state of the controller and can be active or inactive; hence, a Boolean variable, named step activity variable and noted Xi, for step i, can be defined for each step. Actions may be associated to a step; an action associated to a step is performed only when this step is active and then acts upon an output variable. A transition condition must be associated to each transition; this condition is a Boolean expression which may include input variables and step activity variables.

The example used in this paper is a parking gate. This gate should open (OG) when the remote control is activated (r) and should close (CG) once the vehicle (v) has left the area. Two sensors permit to detect whether the gate is fully opened (o) or fully closed (c). To avoid damaging the vehicle or hurting people, the gate should re-open if a vehicle or people are detected in the area while the gate is closing. The example is illustrated in Figure 2 and the specification of its behavior, defined in Grafcet, is given in Figure 3. This specification describes the control of the two outputs OG and CG: OG (respectively CG) is set when the step 11 (resp. 21) is active and reset when the step 10 (resp. 20) is active. It is composed of two graphs that communicate via the variable $X11$; OG has priority over CG, which is mandatory for safety reasons, because it is not possible to set CG once OG set (the transition from the step 20 to the step 21 cannot be fired when $X11$ is *true*).

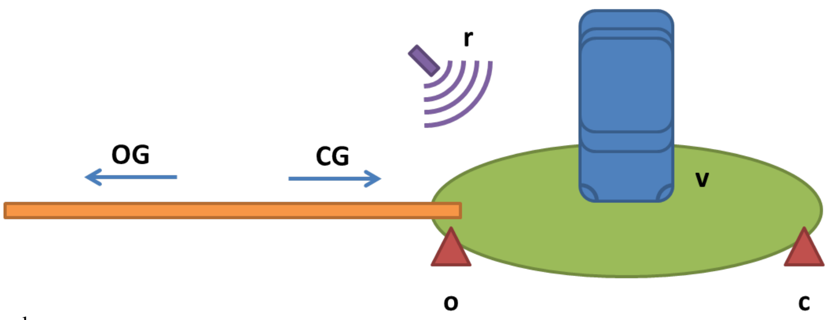
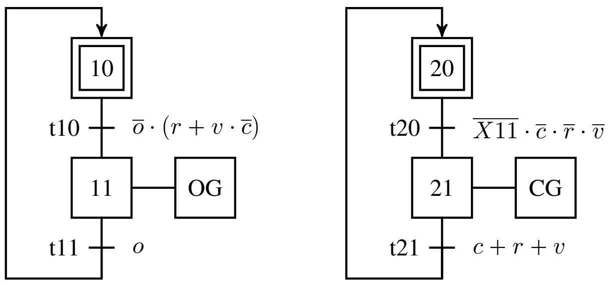


Fig. 2. Illustration of the example


10
t10
$\overline{o} \cdot (r + v \cdot \overline{c})$
11
OG
t11
o
20
t20
$\overline{X11} \cdot \overline{c} \cdot \overline{r} \cdot \overline{v}$
21
CG
t21
$c + r + v$

Logic inputs: $\{c, o, r, v\}$, Logic outputs: $\{CG, OG\}$
Notations: the symbols $+$, $\cdot$ and $\overline{}$ represent the logic operators OR (disjunction), AND (conjunction) and NOT (negation), respectively.

Fig. 3. Grafcet specification for the example given Fig. 2

B. Notations of the input and output combinations

A PLC owns n logic input variables and m logic output variables; the value of each of them is either *true* or *false*. 2^n input (2^m output) combinations can then be constructed from this set of input (output) variables by assigning a weight to each variable. An input combination c_I will be represented in three different manners in this paper:

- The first representation, noted symbol(c_I), is the more compact one; symbol(c_I) is indeed an integer that belongs to $[0, 2^n - 1]$ and is defined as follows:
  symbol(c_I) = $\sum_{i=0}^{n-1} c_I[n-1-i] \times 2^{(n-1-i)}$, where:
  - $c_I[n-1-i]$ is an integer that belongs to $[0, 1]$ and is equal to 1 if the i^{th} input variable is *true* and 0 otherwise,
  - $2^{(n-1-i)}$ is the weight of this variable[1].

  This representation will be used in the graphical and tabular descriptions of a Mealy machine.
- The second representation, noted minterm(c_I), is a Boolean expression. A minterm is the conjunction of all the n logic input variables in their positive or complemented form. This representation is very efficient for symbolic calculus.
- The third representation, noted $\mathbb{1}(c_I)$, is that of the set of the only variables which are *true* for the given combination.

The same rules apply for the output combinations c_O. Table I gives the correspondence between these representations for the example introduced in Figures 2 and 3.

C. From Grafcet to Mealy machine

Conformance test of Mealy machines is a mature technique that previously yielded numerous sound results; good syntheses on this topic are available in [21], [30]. This explains why this formalism was selected to represent formally the specification model.

1) Principle of the translation: The figure 4 illustrates the results of the two phases of the construction of the formal specification model; a detailed presentation of this construction method, without semantics loss, can be found in [26]. Only the main features of these phases are reminded below.

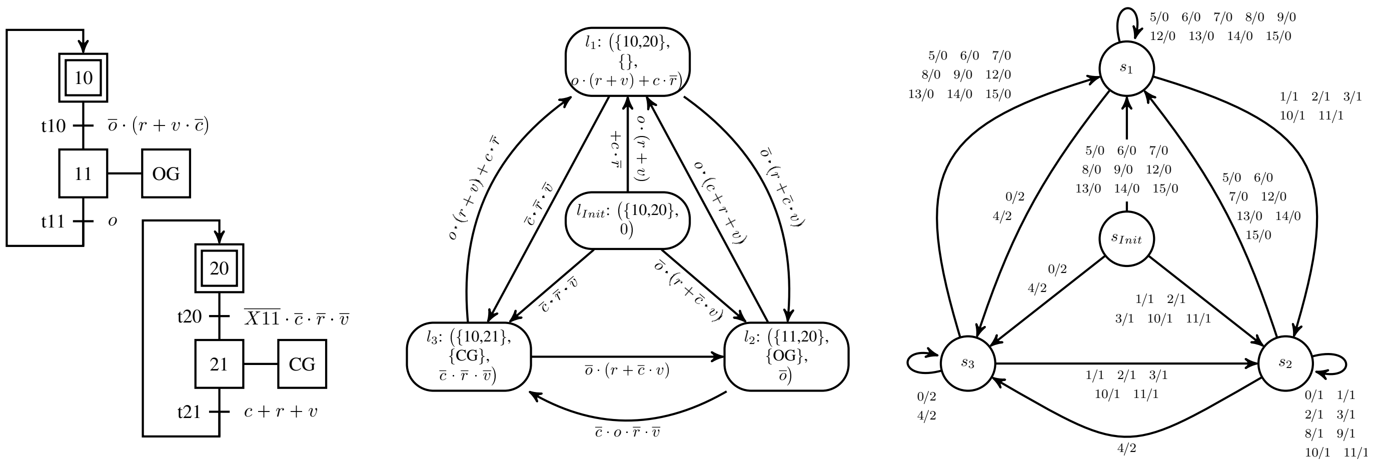
a) The Grafcet is first translated into an automaton, called Stable Location Automaton, that describes the reachable state space of the specification. A state (location) of this automaton represents a stable situation of the Grafcet as well as the outputs

[1]The weights are assigned arbitrarily to the variables.

TABLE I
EQUIVALENCE BETWEEN THE DIFFERENT REPRESENTATIONS OF THE INPUT AND OUTPUT COMBINATIONS

(logic inputs, weight)	(c,8)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	(o,4)	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	(r,2)	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	(v,1)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
symbol(c_I)		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
minterm(c_I)		$\overline{c}\cdot\overline{o}\cdot\overline{r}\cdot\overline{v}$	$\overline{c}\cdot\overline{o}\cdot\overline{r}\cdot v$	$\overline{c}\cdot\overline{o}\cdot r\cdot\overline{v}$	$\overline{c}\cdot\overline{o}\cdot r\cdot v$	$\overline{c}\cdot o\cdot\overline{r}\cdot\overline{v}$	$\overline{c}\cdot o\cdot\overline{r}\cdot v$	$\overline{c}\cdot o\cdot r\cdot\overline{v}$	$\overline{c}\cdot o\cdot r\cdot v$	$c\cdot\overline{o}\cdot\overline{r}\cdot\overline{v}$	$c\cdot\overline{o}\cdot\overline{r}\cdot v$	$c\cdot\overline{o}\cdot r\cdot\overline{v}$	$c\cdot\overline{o}\cdot r\cdot v$	$c\cdot o\cdot\overline{r}\cdot\overline{v}$	$c\cdot o\cdot\overline{r}\cdot v$	$c\cdot o\cdot r\cdot\overline{v}$	$c\cdot o\cdot r\cdot v$
$\mathbb{1}(c_I)$		$\{\}$	$\{v\}$	$\{r\}$	$\{r,v\}$	$\{o\}$	$\{o,v\}$	$\{o,r\}$	$\{o,r,v\}$	$\{c\}$	$\{c,v\}$	$\{c,r\}$	$\{c,r,v\}$	$\{c,o\}$	$\{c,o,v\}$	$\{c,o,r\}$	$\{c,o,r,v\}$

(logic outputs, weight)	(CG,2)	0	0	1	1
	(OG,1)	0	1	0	1
symbol(c_O)		0	1	2	3
minterm(c_O)		$\overline{CG}\cdot\overline{OG}$	$\overline{CG}\cdot OG$	$CG\cdot\overline{OG}$	$CG\cdot OG$
$\mathbb{1}(c_O)$		$\{\}$	$\{OG\}$	$\{CG\}$	$\{CG,OG\}$



10
t10
$\overline{o}\cdot(r+v\cdot\overline{c})$
11
OG
t11
o
20
t20
$\overline{X11}\cdot\overline{c}\cdot\overline{r}\cdot\overline{v}$
21
CG
t21
$c+r+v$
l_1: ({10,20}, {}, $o\cdot(r+v)+c\cdot\overline{r}$)
l_{Init}: ({10,20}, 0)
l_3: ({10,21}, {CG}, $\overline{c}\cdot\overline{r}\cdot\overline{v}$)
l_2: ({11,20}, {OG}, $\overline{o}$)
$o\cdot(r+v)+c\cdot\overline{r}$
$\overline{c}\cdot\overline{r}\cdot\overline{v}$
$o\cdot(r+v)+c\cdot\overline{r}$
$o\cdot(c+r+v)$
$\overline{o}\cdot(r+\overline{c}\cdot v)$
$\overline{c}\cdot\overline{r}\cdot\overline{v}$
$\overline{o}\cdot(r+\overline{c}\cdot v)$
$\overline{o}\cdot(r+\overline{c}\cdot v)$
$\overline{c}\cdot o\cdot\overline{r}\cdot\overline{v}$
s_1
s_{Init}
s_2
s_3
5/0 6/0 7/0 8/0 9/0 12/0 13/0 14/0 15/0
5/0 6/0 7/0 8/0 9/0 12/0 13/0 14/0 15/0
1/1 2/1 3/1 10/1 11/1
0/2 4/2
5/0 6/0 7/0 8/0 9/0 12/0 13/0 14/0 15/0
5/0 6/0 7/0 12/0 13/0 14/0 15/0
0/2 4/2
1/1 2/1 3/1 10/1 11/1
1/1 2/1 3/1 10/1 11/1
0/2 4/2
4/2
0/1 1/1 2/1 3/1 8/1 9/1 10/1 11/1

Fig. 4. Example of translation of a Grafcet into a Mealy machine. Left to right: Grafcet, Stable Location Automaton, Mealy machine

to set in this situation. Each transition is labeled with a Boolean expression on the input variables, condition to evolve from a location to another one.

b) This intermediary model is then translated into a Mealy machine. Each location is directly translated into a state of the Mealy machine but the transitions are expanded to obtain an enumerative representation; therefore a condition-labeled transition gives rise to a set of symbol-labeled transitions.

The formal definition of the final model, Mealy machine that represents the PLC specification model, is given hereafter.

2) Definitions: Formally, a Programmable Logic Controller (PLC) can be defined by a 3-tuple $(I, O, \mathcal{B})$ where:

- I is a non-empty set of logic inputs.
- O is a non-empty set of logic outputs.
- $\mathcal{B}$ is the behavior of the controller.

Since inputs are logic variables and not events, they can be combined together to give input combinations, respectively outputs can be combined to give output combinations. If the cardinality of I (respectively O) is $|I|$ (resp. $|O|$), there exist $2^{|I|}$ (resp. $2^{|O|}$) distinct input (resp. output) combinations c_I (resp. c_O). Let us note C_I the set of the input combinations and C_O the set of the output combinations.

Using this definition of input and output combinations, the behavior of a PLC can be represented by a Mealy machine $(S, s_{Init}, C_I, C_O, \delta, \lambda)$, where:

- S is a non-empty set of states.
- s_{Init} is the initial state, $s_{Init} \in S$.
- C_I is the input alphabet, $|C_I| = 2^{|I|}$.
- C_O is the output alphabet, $|C_O| = 2^{|O|}$.
- δ is the transition function, defined as follows:

$$\begin{array}{rll}\delta : & S \times C_I & \rightarrow S \\ & (s_s, c_I) & \mapsto s_t = \delta(s_s, c_I)\end{array} \tag{1}$$

- λ is the output function, defined as follows:

$$\begin{array}{rll}\lambda : & S \times C_I & \rightarrow C_O \\ & (s_s, c_I) & \mapsto c_o = \lambda(s_s, c_I)\end{array} \tag{2}$$

TABLE II
ENUMERATED REPRESENTATION OF THE TRANSITION AND OUTPUT FUNCTIONS: $(s, c_I) \mapsto (\delta(s, c_I), \lambda(s, c_I))$

symbol(c_I) / s: state	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_{Init}	$s_3, 2$	$s_2, 1$	$s_2, 1$	$s_2, 1$	$s_3, 2$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_2, 1$	$s_2, 1$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$
s_1	$s_3, 2$	$s_2, 1$	$s_2, 1$	$s_2, 1$	$s_3, 2$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$s_2, 1$	$s_2, 1$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$	$\textcircled{s_1, 0}$
s_2	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$s_3, 2$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$\textcircled{s_2, 1}$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$
s_3	$\textcircled{s_3, 2}$	$s_2, 1$	$s_2, 1$	$s_2, 1$	$\textcircled{s_3, 2}$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_2, 1$	$s_2, 1$	$s_1, 0$	$s_1, 0$	$s_1, 0$	$s_1, 0$

By definition, the behavior $\mathcal{B}$ of a PLC is deterministic: there is only one initial state and δ and λ are two functions. Moreover, to avoid misinterpretation errors during the test, the behavior $\mathcal{B}$ must be:

- completely defined: δ and λ are total functions[2];

$$\forall (s, c_I) \in S \times C_I, \begin{cases} \exists! \delta(s, c_I) \in S \\ \exists! \lambda(s, c_I) \in C_O \end{cases} \tag{3}$$

- limited to its reachable part;

$$\forall s \in S, \ \exists [c_I^0, \cdots, c_I^n] \in C_I^* \mid \begin{cases} s^1 = \delta(s_{Init}, c_I^0) \\ \forall k \geqslant 1, \ s^{k+1} = \delta(s^k, c_I^k) \\ s = s^n \end{cases} \tag{4}$$

- without transient evolution, i.e. no inputs change introduces successive changes of states or emitted outputs.

$$\forall (s, c_I) \in S \times C_I, \begin{cases} \delta(\delta(s, c_I), c_I) = \delta(s, c_I) \\ \lambda(\delta(s, c_I), c_I) = \lambda(s, c_I) \end{cases} \tag{5}$$

3) Illustration on the example: The above definitions and properties will be illustrated on the example presented in Figure 4. This example owns 4 logic inputs (c, o, r and v), and 2 logic outputs (OG and CG). Then, 16 input combinations (c_I) and 4 output combinations (c_O) can be defined. Since the state space comprises 4 states (s_{Init}, s_1, s_2 and s_3), the transition and output functions are defined on 64 couples (s, c_I). Table II gives an enumerated description of these functions. In this table each line corresponds to a state s and each column to an input combination c_I, represented by its symbol: symbol(c_I).

Each cell of the table contains the value of the couple $(\delta(s, c_I), \lambda(s, c_I))$. A circled couple means that the same state is both source and target of the transition (self-loop structure: $\delta(s, c_I) = s$). The behavior given in table II is deterministic and completely defined since every cell contains one and only one state name. This behavior does not contain any transient evolution since the value of each cell is either a circled value or leads to a cell with a circled value ($\delta(s, c_I) = s$ or $\delta(\delta(s, c_I), c_I) = \delta(s, c_I)$). For example, when the active state is s_1, for the input combination 0, the system evolves towards state s_3 ($\delta(s_1, 0) = s_3$). Since $\delta(s_3, 0) = s_3$, no other evolution is possible without a new change of input combination.

D. Test sequence generation

A test sequence is an ordered list of couples (input combination, expected output combination) which represents the external view of the expected behavior of a PLC that executes a correct control code. Formally, a test sequence is defined as follows:

$$\left[(c_I^0, c_O^0), (c_I^1, c_O^1), ..., (c_I^n, c_O^n)\right] \in (C_I \times C_O)^* \tag{6}$$

However, the input combinations c_I^k and the expected output combinations c_O^k are not independent. The expected output combination c_O is that associated to the transition which goes from a source state s_s to a target state s_t for the input combination c_I. Hence, an elementary conformance test step et is defined by the following 4-tuple:

$$et = (s_s, c_I, s_t, c_O) \in S \times C_I \times S \times C_O$$
$$\text{where} \begin{cases} s_t = \delta(s_s, c_I) \\ c_O = \lambda(s_s, c_I) \end{cases} \tag{7}$$

A test sequence TS is an ordered list of elementary test steps et which must be:

P1: initializable, i.e. the source state of the first test step is the initial state of the PLC's behavior model, and the input combination c_I^0 is such that the target state is stable:

[2] $\exists!$: There exists exactly one.

$$\begin{cases} s^0 = s_{Init} \\ \delta(\delta(s^0, c_I^0), c_I^0) = \delta(s^0, c_I^0) \\ \lambda(\delta(s^0, c_I^0), c_I^0) = \lambda(s^0, c_I^0) \end{cases} \tag{8}$$

P2: consistent, i.e. the source state of the $(k+1)^{th}$ elementary test step is equal to the target state of the k^{th} elementary test step.

$$TS = [(s^0, c_I^0, \delta(s^0, c_I^0), \lambda(s^0, c_I^0)), \cdots, (s^n, c_I^n, \delta(s^n, c_I^n), \lambda(s^n, c_I^n))] \mid \forall k \geqslant 0, s^{k+1} = \delta(s^k, c_I^k) \tag{9}$$

Moreover, if the test objective is to cross at least once each transition of the Mealy machine (usual objective when the control of critical systems is considered), the test sequence must be:

P3: complete, i.e. there is at least one test step for each element of the transition function:

$$\forall (s, c_I) \in (S \backslash s_{Init} \times I),\ (s, c_I, \delta(s, c_I), \lambda(s, c_I)) \in TS \tag{10}$$

An infinity of test sequences that satisfy the three properties **P1**, **P2** and **P3** may be constructed from a given Mealy machine. However, to reduce the duration of the test, an optimization criterion is commonly introduced; the sequence TS must be:

P4: minimum-length, i.e. there is no initializable, consistent and complete test sequence shorter than TS:

$$|TS| : \forall\ ts \text{ satisfying P1, P2 and P3}, |ts| > |TS| \tag{11}$$

The generation of an initializable, consistent, complete and minimum-length test sequence can be automated by using the transition tour method [25]. This optimization problem is a particular solution of a well-known problem in graph theory: the Chinese Postman Problem [31] – or pre-Eulerian path –. The general formulation of this problem is the following: Find a minimum-length closed path that traverses at least once each arc of a given graph. When a Mealy machine is considered, the graph is directed, but not weighted; the optimization problem is then simplified.

For the example given in Figure 4 this complete and minimum-length test sequence contains 43 test steps; the first 15 ones are given in the table III. It must be noted that the number of test steps of this sequence (43) is smaller than the number of transitions of the Mealy machine (48) even if all transitions are traversed at least once. This is possible because there is no transient evolution in the Mealy machine. Hence, two transitions (s_s, c_I, s_t, c_O) and (s_t, c_I, s_t, c_O) of the Mealy machine can be tested during only one experimental test step defined as (s_s, c_I, s_t, c_O) if the duration of this step is greater than several scanning cycles of the PLC. In that case, the inputs of the PLC under test are scanned several times during one experimental test step and it is possible to test a first transition which corresponds to a state change provoked by the input combination c_I then a second one, self-loop on the target state of the first transition for this input combination. This interesting feature will be developed in the section III-B when describing the execution of a test.

TABLE III
FIRST TEST STEPS OF THE MINIMUM-LENGTH TEST SEQUENCE FOR THE EXAMPLE DETAILED IN FIGURE 4.

s_s:	s_1	s_3	s_2	s_3	s_2	s_3	s_2	s_2	s_2	s_2	s_3	s_2	s_1	s_3	s_2	…
c:	0	1	0	1	0	0	1	1	0	0	0	1	0	0	1	…
o:	0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	…
r:	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	…
v:	0	1	0	0	0	1	1	0	0	0	0	1	0	1	0	…
s_t:	s_3	s_2	s_3	s_2	s_3	s_2	s_2	s_2	s_2	s_3	s_2	s_1	s_3	s_2	s_1	…

III. MINIMUM-LENGTH TEST SEQUENCE EXECUTION

In order to assess the interest of a minimum-length test sequence built from a Mealy machine for conformance test of real industrial logic controllers, a specific test bench has been developed. This test bench is briefly described at the beginning of this section. The experimental conditions of a conformance test are then given. As biased – rejection of a correct implementation – and non-valid – acceptance of a non-conform implementation – test results were obtained during some tests, two series of experiments were undertaken to pinpoint the origin of these erroneous results; they are described in the third and fourth subsections. The section is concluded by a discussion on this phenomenon and some strategies to improve the level of confidence in conformance test results by modifying the execution of the test or the construction of the test sequence.

A. *Presentation of the test bench*

The test bench used for this work has been developed from feedback on a previous platform that was designed for discrete event systems identification purposes [32].

This test bench (Figure 5) is built from an industrial remote input/output module (RIOM). This module (Schneider Electric Momentum 170ENT11001) has 16 logic inputs and 16 logic outputs that can be remotely set/reset from a computer via a Modbus TCP/IP connection. Each output of this RIOM is connected upstream from an input of the logic controller under test (blue wires in Figure 5), while each of its inputs is connected downstream from an output of the controller (red wires in Figure 5). This open solution provides high flexibility at a very low cost.

The logic controller under test, a modular PLC (Phoenix Contact ILC 170 ETH 2TX) was selected because of its high reactivity (its cycle time can be set to less than 5 ms) and the compliance of its programming languages with the IEC 61131-3 standard [33].

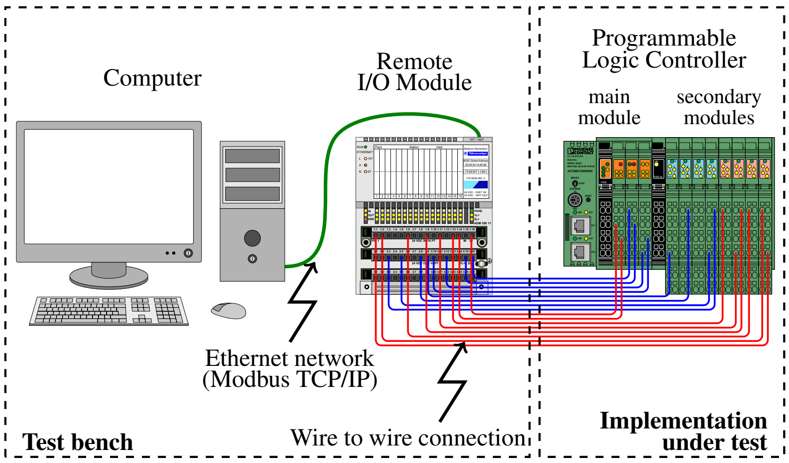

Computer
Remote I/O Module
Programmable Logic Controller
main module
secondary modules
Ethernet network (Modbus TCP/IP)
Wire to wire connection
Test bench
Implementation under test

Fig. 5. Experimental test bench

B. *Execution of a conformance test*

The code implemented in the PLC for the experiments was a Structured Text code obtained automatically from the Grafcet specification according to the method proposed in [34]. This solution allows either to generate an a priori correct code or to artificially introduce programming errors. Once programmed, the controller to be tested is disconnected from its programming environment during the entire test execution.

The execution of the code implemented in a PLC relies on the PLC's input scanning cycle. This scanning cycle can be configured so that the code is executed either in a cyclic or in a periodic mode. In the first case (cyclic mode), once the code has been executed by the PLC the logic outputs are immediately updated; thus, the execution time can vary from one cycle to another. In the second case (periodic mode), the code is executed at a fixed period; thus, once the code has been executed the PLC will wait for the end of the cyle before emitting logic outputs. Figure 6 illustrates the different steps of a periodic execution; a similar figure can be drawn for a cyclic execution by removing the waiting (W) phase. As illustrated in this figure, the minimum causality delay (response time) of a PLC is equal to one PLC cycle (if the logic inputs changed just before the input scanning), while the maximal causality delay is almost equal to two PLC cycles (if the logic inputs changed just after the input scanning).

Once the controller has been connected to the test bench and restarted, each elementary test step (s_s, c_I, s_t, c_O) is executed as follows:

- The test bench applies the input combination c_I to the logic inputs of the controller under test.
- Then, the test bench waits while the PLC executes its code and updates its outputs. To be sure that the outputs have been updated, this waiting time must be at least longer than two PLC cycles (maximal value of the causality delay)
- When the controller's outputs are updated, the test bench reads the output combination emitted by the PLC and compares it to the expected one c_O, then stores the results. The next experimental test step can start.

Hence, the duration of one experimental test step is at least equal to two PLC periods, for a periodic mode, or two times the maximal value of the PLC cycle, for a cyclic mode. This minimal bound guarantees that each output combination update, consequence of an input combination change, is observed. Moreover, if this duration is longer (three PLC cycles for instance), it is possible during one experimental test step that corresponds to a theoretical test step (s_s, c_I, s_t, c_O) to test two successive transitions of the Mealy machine: the transition defined by this step and the transition (s_t, c_I, s_t, c_O), self-loop on the target state of the first transition. In the case of the example of figure 4, it is possible for instance to test during one experimental test step the transition labelled $4/2$ from state s_2 to state s_3 and the self-loop with the same label on s_3. As the PLC scans its

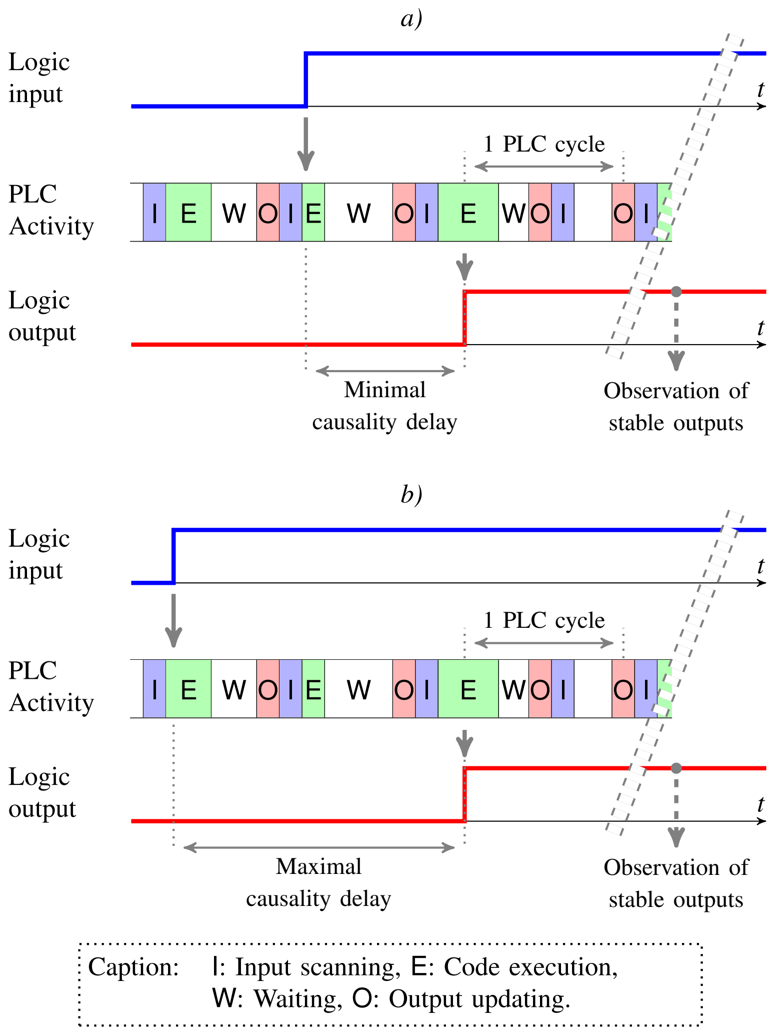

a)
Logic input
t
1 PLC cycle
PLC Activity
I E W O I E W O I E W O I O I
Logic output
t
Minimal causality delay
Observation of stable outputs
b)
Logic input
t
1 PLC cycle
PLC Activity
I E W O I E W O I E W O I O I
Logic output
t
Maximal causality delay
Observation of stable outputs
Caption: I: Input scanning, E: Code execution,
W: Waiting, O: Output updating.

Fig. 6. PLC periodic execution and maximal causality delay

inputs cyclically (in a periodic manner or with a variable scanning time), the input combination change that leads to the input combination c_I (4 in the example) provokes first a state change from s_s to s_t (s_2 to s_3 in the example); this state change can be observed after a delay equal to at most two PLC cycles. If the input combination is unchanged, the output combination computed during the following PLC cycle will be unchanged, which corresponds to a self-loop on s_t with the same input and output combinations ($4/2$ in the example). As the Mealy machine does not include any transient evolution, no more state or output combination change is possible; the output combination is stable. Then, defining a duration of each experimental test step greater or equal to three PLC cycles permits to reduce the length of the test sequence as noted in subsection II-D.

C. Robustness study – Experiment 1

As the first results were conclusive, robustness of the method was addressed by focusing on the influence of the features of the controller under test (cyclic or periodic I/O scanning and distribution of the inputs) on the test verdict.

When performing the conformance test of a controller with a cyclic, but not periodic, I/O scanning mode (this execution mode is the most commonly encountered in industrial applications because it improves the reactivity of the controller), erroneous (biased and non-valid) test results were obtained. A detailed analysis of the results files showed that these errors happened only when several inputs were changed simultaneously, from one test step to the following one. An assumption was then put forward: the synchronous changes of the outputs of the test bench are interpreted as asynchronous by the PLC in these cases. An example of interpretations that rely on this assumption is given in Figure 7, where two inputs are considered; as the electrical input signals applied to the input cards of a PLC are filtered and thresholded before being converted to logic variables used by the PLC code, theoretically synchronous signals sent by the test bench may give rise to synchronous or asynchronous logic variables. If this assumption is true, the input combination that is used by the PLC code is not the one planned in the test sequence and the observed output may be different from the expected one if the control code is correct; similarly, a flawed control code may then lead to an expected, while computed from an unplanned input, output.

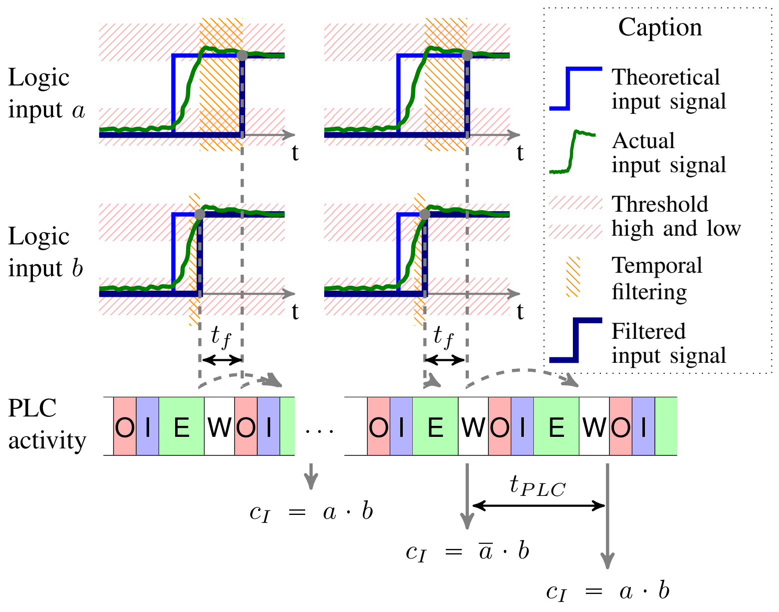


Fig. 7. Possible interpretations of simultaneous changes of 2 inputs; where t_f is the time delay between two filtered signals (obtained from theoretically synchronous signals), and t_{PLC} is the PLC cycle duration.

To check whether this assumption was true and to analyze the sensitivity of the controller to this kind of interpretation error, the following experiment was carried out:

- A set of 8 input signals that were varying simultaneously and in an identical manner (from 0 to 1, then from 1 to 0) with a period far greater (more than 10 times) than the maximal PLC cycle time, was sent by the test bench to the controller. Hence, the changes (rising and falling edges) of the 8 signals were a priori totally synchronous.
- When the controller detected the first change of one of its inputs, the current value of each input was copied on the corresponding output (8 outputs were used, each one associated to an input). Thus, the output combination was identical to the input combination at the moment of the first input change which had been detected.
- The test bench read the output combination and compared it with the expected one, what should be identical to the input combination, if no perception error had occurred. A counter was incremented when an error was detected, i.e. when the output combination differed from the input one.

Table IV gives the rates of interpretation errors according to the PLC I/O scanning mode and to the distribution of inputs on the PLC internal components (all inputs connected on the main module, all inputs connected on the secondary module, inputs distributed on the main module and on the secondary module). These figures were obtained from series of 200,000 experiments for each mode and each distribution.

These interpretation errors can be explained because the PLC cycle and the test bench cycle are not synchronized. When the inputs of the PLC are changed during the input reading phase of the PLC cycle, some changes may be detected but other ones not; synchronous changes are missed in this case. In the case of inputs distributed on several modules (last two lines of the table IV), the errors rate is increased owing to the communication protocol between the different modules of the PLC. This protocol is indeed designed to reduce the waiting time of the processor of the main module and not to ensure simultaneous reading of distributed inputs. Globally, this experiment showed that the assumption formulated previously to explain the erroneous tests was plausible.

TABLE IV
RATES OF INTERPRETATION ERRORS – EXPERIMENT 1

Distribution \ Configuration		Cyclic	Periodic 10 ms	Periodic 20 ms
main module	Change 0 to 1	0.062 %	0.022 %	0.014 %
	Change 1 to 0	1.918 %	0.886 %	0.455 %
secondary module	Change 0 to 1	0.104 %	0.040 %	0.020 %
	Change 1 to 0	2.536 %	0.993 %	0.550 %
allocated on the two modules	Change 0 to 1	29.17 %	39.62 %	20.23 %
	Change 1 to 0	38.55 %	43.46 %	21.89 %

D. Robustness study – Experiment 2

Nonetheless, the experiment 1 relied on a hypothesis on the changes of the input signals: their rising and falling edges were supposed synchronous. To remove this hypothesis, a second experiment was achieved with a slightly modified test bench. Instead of connecting each input of the PLC to a different output of the RIOM – as it is necessary to execute a conformance test –, all inputs of the PLC were connected to a *single* output of the RIOM (Figure 8). Then, the same sequence of measures was carried out.

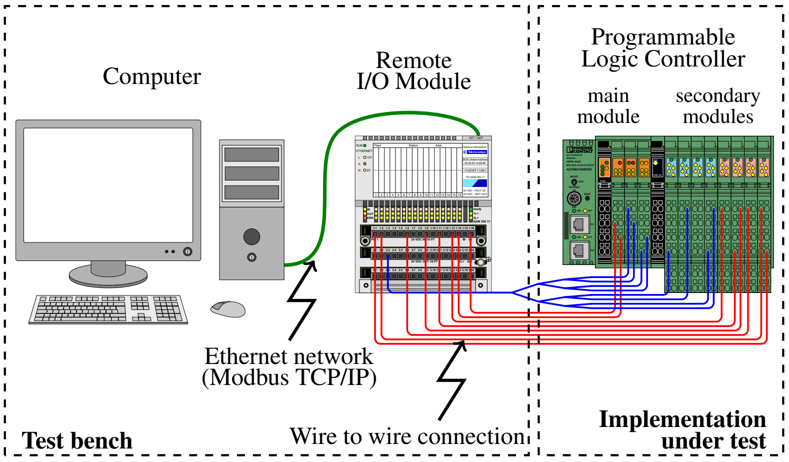

Computer
Remote I/O Module
Programmable Logic Controller
main module
secondary modules
Ethernet network (Modbus TCP/IP)
Wire to wire connection
Test bench
Implementation under test

Fig. 8. Experimental test bench – Experiment 2

Table V gives the rates of interpretation errors for this second configuration of the test bench. These figures were obtained from series of 500,000 experiments for each I/O scanning mode and each distribution.

The results of the last two lines of the tables IV and V are very similar and will not be discussed anymore. On the opposite, comparison of the results of the first four lines of the table V to those obtained in the first experiment permits to conclude that the previous hypothesis was not correct; the edges of the outputs of the test bench are not completely synchronous. This explains why most error rates are reduced when only one output of the test bench is used; the rate for the cell at the second line and the first column decreases from 1.918% to 0.237% for instance. However, this experiment confirms the validity of the assumption on the PLC behavior; even when only the same signal is applied to several inputs of the PLC, a priori synchronous changes of these inputs are sometimes perceived as asynchronous. The origin of the erroneous test results that have been obtained with a minimum-length sequence is therefore well identified.

TABLE V
RATES OF INTERPRETATION ERRORS – EXPERIMENT 2

Distribution \ Configuration		Cyclic	Periodic 10 ms	Periodic 20 ms
main module	Change 0 to 1	0.059 %	0.026 %	0.012 %
	Change 1 to 0	0.237 %	0.105 %	0.048 %
secondary module	Change 0 to 1	0.086 %	0.036 %	0.018 %
	Change 1 to 0	0.429 %	0.080 %	0.095 %
allocated on the two modules	Change 0 to 1	30.22 %	39.34 %	19.86 %
	Change 1 to 0	36.63 %	42.21 %	21.20 %

E. Discussion and Guidelines

Some issues may arise when applying theoretical results based on DES (Discrete Event Systems) models, like state automata or Mealy machines, to real PLCs. This has been already underlined in [35] and [36] when considering the implementation of a supervisor built according to the supervisory control theory on a PLC. The issue addressed in these references was the following one: *how to implement correctly a model that was constructed with the assumption that all events are asynchronous, i.e. only one event can occur at any date, on a PLC with cyclic (or periodic) I/O scanning?* I/O scanning implies indeed that asynchronous changes of the inputs of the PLC may be treated as synchronous if they occur during the same cycle.

The issue that was pinpointed in the previous experimental study is dual: synchronous events (input changes) may be detected by the control code as asynchronous. This happens when these changes occur during the input reading phase of the PLC cycle. As the inputs are not read instantaneously, some input changes may be detected at the i^{th} cycle while other ones will be detected only at the next cycle (Figure 7). This issue increases when the PLC cycle time decreases because the relative duration of the input reading phase is then increased – the duration of this phase is constant for a given number of inputs –

and the likelihood of erroneous interpretation increases; this explains why the error rates are greater in the four first lines of the tables IV and V for the cyclic mode – the cycle time was in the range of 4 to 5 ms during the experiment.

The previous experiments have shown that using a minimum-length (usual optimization criterion) test sequence obtained from a Mealy machine by the transition-tour method may yield spurious test results. The theoretical analysis relies indeed on the implicit assumption that all inputs changes generated by the test bench are correctly detected by the controller under test, even if several inputs are simultaneously modified, which is not always true.

To overcome this issue, several strategies can be considered:

- synchronization of the test bench with the controller under test;
- test execution for the configurations of the controller that lessen the error rate (periodic I/O scanning and no inputs distribution);
- multiple execution of the same test sequence and statistical analyses of the results.

The first strategy requires additional communication between the test bench and the controller and does not correspond to the real operation of a closed-loop system where the PLC is connected, but not synchronized, to a real plant. Therefore, the second and third strategies are more appropriate.

IV. Conclusion

Even if numerous theoretical results on conformance test of Mealy machines have been published, application to conformance test of PLCs is not completely straightforward because the technological features of these industrial components are not taken into account in the theoretical studies. This paper has highlighted in particular that biased or non-valid test results may be obtained when using a minimum-length test sequence because the I/O scanning cycle of the PLC provokes asynchronism between input events that are assumed synchronous.

A promising solution to tackle out this issue is to construct a test sequence where no synchronous input events are present by definition. However, it is not sure that an initializable, consistent and complete test sequence that satisfies this property, i.e. where only one input is changed from any test step to the following one, can always be constructed whatever the specification of the PLC.

References

[1] E. Estévez and M. Marcos, "Model based validation of industrial control systems," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 2, pp. 302–310, 2012.

[2] M. Witsch and B. Vogel-Heuser, "Towards a formal specification framework for manufacturing execution systems," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 2, pp. 311–320, 2012.

[3] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML - the glue for seamless automation engineering," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. IEEE, 2008, pp. 616–623.

[4] G. Frey and L. Litz, "Formal methods in PLC programming," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2431–2436.

[5] S. Patil, V. Vyatkin, and M. Sorouri, "Formal verification of intelligent mechatronic systems with decentralized control logic," in *17th International Conference on Emerging Technologies and Factory Automation (ETFA 2012)*, 2012.

[6] S. Preuße, H.-C. Lapp, and H.-M. Hanisch, "Closed-loop system modeling, validation, and verification," in *17th International Conference on Emerging Technologies and Factory Automation (ETFA 2012)*, 2012.

[7] M. Perin and J.-M. Faure, "Building meaningful timed models of closed-loop DES for verification purposes," *Control Engineering Practice*, 2012.

[8] C. Seidner and O. H. Roux, "Formal methods for systems engineering behavior models," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 4, pp. 280–291, 2008.

[9] X. Weng and L. Litz, "Verification of logic control design using SIPN and model checking: methods and case study," in *American Control Conference, 2000. Proceedings of the 2000*, vol. 6. IEEE, 2000, pp. 4072–4076.

[10] S. Klein, G. Frey, J.-J. Lesage, and L. Litz, "Supporting the changeability of SIPN-based logic control algorithms by verification and validation," in *Proc. of IMACS-IEEE int. conf. on Computational Engineering in Systems Applications*, 2003.

[11] V. Vyatkin and H.-M. Hanisch, "A modeling approach for verification of IEC 1499 function blocks using net condition/event systems," in *Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA'99. 1999 7th IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 261–270.

[12] G. Canet, S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen, "Towards the automatic verification of PLC programs written in Instruction List," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2449–2454.

[13] S. Lampérière-Couffin and J.-J. Lesage, "Formal verification of the sequential part of PLC programs," in *Workshop on Discrete Event Systems (WODES'00)*. Springer, 2000, pp. 247–254.

[14] V. Gourcuff, O. de Smet, and J.-M. Faure, "Efficient representation for formal verification of PLC programs," in *Proceedings of 8th International Workshop On Discrete Event Systems (WODES'06)*, Ann Arbor USA, 07 2006, pp. pp. 182–187.

[15] D. Soliman, K. Thramboulidis, and G. Frey, "Transformation of function block diagrams to uppaal timed automata for the verification of safety applications," *Annual Reviews in Control*, 2012.

[16] IEC 60880, *Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions*, 2nd ed. International Electrotechnical Commission, 2006.

[17] IEC 61850-10, *Communications Networks and Systems in Substations - Part 10: Conformance testing*, 2nd ed. International Electrotechnical Commission, 2005.

[18] D. Maclay, "Simulation gets into the loop," *IEE Review*, vol. 43, no. 3, pp. 109–112, 1997.

[19] F. Gu, W. S. Harrison, D. M. Tilbury, and C. Yuan, "Hardware-in-the-loop for manufacturing automation control: Current status and identified needs," in *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*. IEEE, 2007, pp. 1105–1110.

[20] N. Schetinin, N. Moriz, B. Kumar, A. Maier, S. Faltinski, and O. Niggemann, "Why do verification approaches in automation rarely use hil-test?" in *Industrial Technology (ICIT), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1428–1433.

[21] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines - a survey," in *Proceedings of the IEEE*, vol. 84, no. 8, 1996, pp. 1090–1123.

[22] J. Tretmans, “Model based testing with labelled transition systems,” in *Formal Methods and Testing*, ser. Lecture Notes in Computer Science, R. M. Hierons, J. P. Bowen, and M. Harman, Eds. Springer, 2008, vol. 4949, pp. 1–38.
[23] S. Pickin, C. Jard, T. Jéron, J.-M. Jézéquel, and Y. Le Traon, “Test synthesis from UML models of distributed software,” *Software Engineering, IEEE Transactions on*, vol. 33, no. 4, pp. 252–269, 2007.
[24] M. Krichen and S. Tripakis, “Conformance testing for real-time systems,” *Formal Methods in System Design*, vol. 34, no. 3, pp. 238–304, 2009.
[25] S. Naito and M. Tsunoyama, “Fault detection for sequential machines by transitions tours,” in *Proceedings of the IEEE Fault Tolerant Computer Symposium*, 1981, pp. 238–243.
[26] J. Provost, J.-M. Roussel, and J.-M. Faure, “Translating grafcet specifications into Mealy machines for conformance test purposes,” *Control Engineering Practice*, vol. 19, no. 9, pp. 947–957, 2011.
[27] IEC 60848, *GRAFCET specification language for sequential function charts*, 2nd ed. International Electrotechnical Commission, 2002.
[28] R. David, “Grafcet: a powerful tool for specification of logic controllers,” *IEEE Transaction on Control Systems Technology*, vol. 3, no. 3, pp. 253–268, 1995.
[29] H. Guéguen and N. Bouteille, “Extensions of Grafcet to structure behavioural specifications,” *Control Engineering Practice*, vol. 9, no. 7, pp. 743 – 756, 2001.
[30] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Eds., *Model-Based Testing of Reactive Systems, Advanced Lectures*. Springer, 2005, vol. 3472 of Lecture Notes in Computer Science.
[31] K. Mei-Ko, “Graphic programming using odd or even points,” *Chinese Mathematics*, vol. 1, pp. 273–277, 1962.
[32] B. Denis, O. De Smet, J.-J. Lesage, and J.-M. Roussel, “Dispositif et procédé d’analyse de performances et d’identification comportementale d’un système en tant qu’automate à évènements discrets et finis,” in *French Patent N° 01 110 933*, 2001.
[33] IEC 61131-3, *Programmable controllers - Part 3: Programming languages*, 2nd ed. International Electrotechnical Commission, 2003.
[34] J. Machado, B. Denis, J.-J. Lesage, J.-M. Faure, and J. Ferreira Da Silva, “Logic controllers dependability verification using a plant model,” in *3rd IFAC Workshop on Discrete-Event System Design, DESDes’06*, 2006, pp. 37–42. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00361815
[35] M. Fabian and A. Hellgren, “PLC-based implementation of supervisory control for discrete event systems,” in *Proceedings of the 37th IEEE Conference on Decision and Control, 1998.*, vol. 3. IEEE, 1998, pp. 3305–3310.
[36] F. Basile and P. Chiacchio, “On the implementation of supervised control of discrete event systems,” *Control Systems Technology, IEEE Transactions on*, vol. 15, no. 4, pp. 725–739, 2007.

Julien Provost Julien Provost received the Ph.D. degree from the École Normale Supérieure de Cachan, France, in 2011. He is currently Assistant Professor at Technische Universität München, Germany where he holds the Assistant Professorship for Safe Embedded Systems. His current research interest focuses on the development of formal methods for specification, verification and validation of safe-critical distributed Discrete Event Systems (DES).

Jean-Marc Roussel Jean-Marc Roussel received the Ph.D. degree in 1994. He is currently Associate Professor of Automatic Control at the École Normale Supérieure de Cachan (France). His research fields are modeling, synthesis and analysis of control systems with formal methods.

Jean-Marc Faure Jean-Marc Faure received the Ph.D. degree from the École Centrale de Paris in 1991. He is currently Professor of Automatic Control and Automation Engineering at the Institut Superieur de Mecanique de Paris and researcher at Ecole Normale Superieure de Cachan, France. His research fields are modeling, synthesis and analysis of Discrete Event Systems (DES) with special focus on formal verification and conformance test methods to improve dependability of critical systems.

J.-M. Faure is member of the IEEE and Associate Editor of the Journal T-ASE since 2012. He is chair of the steering committee of the IFAC workshop series ”Dependable Control of Discrete Systems” and has served in many committees of IFAC and IEEE conferences.