A Conformance Relation for Model-Based Testing of PLC

Anaïs Guignard, Jean-Marc Faure

A conformance relation for model-based testing of PLC

Anais Guignard * **Jean-Marc Faure** **

* *Ecole Normale Superieure de Cachan, 61 av. du president Wilson, 94235 Cachan, France (e-mail: anais.guignard@lurpa.ens-cachan.fr)*
** *Ecole Normale Superieure de Cachan, 61 av. du president Wilson, 94235 Cachan, France (e-mail: jean-marc.faure@lurpa.ens-cachan.fr)*

Abstract: This paper focuses on the execution of conformance testing of PLC with I/O scanning which executes a control code; it is assumed that the test sequence has been built from a finite state machine that represents the specified behavior. It is first shown that the conformance relation which allows to decide whether the implementation conforms to the specification or not must be defined from a sequence of observations, and not from only one observation, to take into account the delays introduced by the I/O scanning. Then, a second conformance relation is proposed to avoid that asynchronous detection of synchronous input changes leads to biased or non-valid verdicts; this new relation is defined from the analysis of concurrent transitions and accepted sequences in the specification model.

Keywords: Finite state machine, Programmable logic controller, Conformance test, Test sequence, Applications

1. INTRODUCTION

Model-based conformance testing is a formal analysis method that aims to check whether an implementation, seen as a black-box with inputs and outputs, behaves as specified. Several worthwhile theoretical results have been already obtained in this field by using specification models in the form of finite state machines (FSM) (Lee and Yannakakis (1996), Li and Kumar (2012)), transition systems (Brinksma and Tretmans (2001), Tretmans (1996)) or Petri nets (Bochmann and Jourdan (2009)). Globally, these works have shown how errors in a model of the implementation can be detected by observing its output sequences in response to input sequences.

The issue of conformance test of real devices, like Programmable Logic Controllers (PLC [1]) that execute a control code, has been addressed more recently (Provost et al. (2011b)). This approach benefits from the previous results on testing of FSM (or Mealy machines) and is decomposed in two phases:

- Construction of a test sequence from the specification model;
- Execution of this sequence by a test-bench, electronic device that sends input signals to the PLC and observes its responses (Fig.1); comparison of these responses to those expected according to a conformance relation permits to emit a verdict. It must be underlined that this test is non-invasive. No piece of code or electronic component must be introduced in the PLC for test reasons; the PLC is tested in its real operation conditions.

[1] The singular spelling will be used for all acronyms.

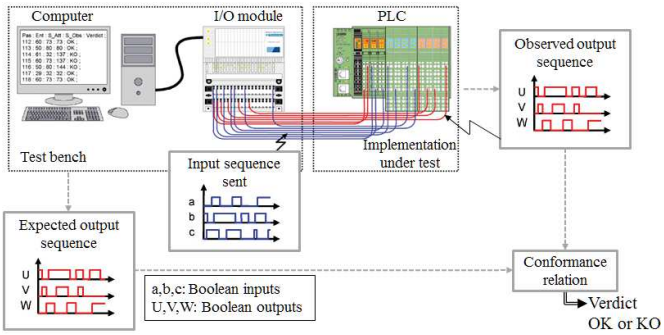

Fig. 1. Conformance test execution

When critical systems are considered, a usual objective to build the test sequence is to cross at least once every transition of the machine. To limit the length of this sequence, therefore the duration of the test execution, an optimization criterion is commonly introduced: the test sequence must be minimum-length; the algorithm proposed in Naito and Tsunoyama (1981) can be then selected to build the sequence from the specification model. However, the obtained sequence is a MIC (Multiple Input Change) sequence, i.e. several input values may be changed synchronously by the test-bench from one test step to the following one. This feature may lead to erroneous test verdicts as described in Provost et al. (2011a) because synchronous input changes may be detected as asynchronous by the PLC during test execution.

To solve this issue, a new algorithm to construct the test sequence has been presented in Provost et al. (2010); the aim of this algorithm is to build a SIC (Single Input Change) sequence to avoid asynchronous detections of synchronous input changes. Unfortunately, a SIC test

sequence does not generally cover at least once every transition of the machine; some transitions are not testable with such a sequence. To meet the test objective for critical systems, MIC test steps must be added at the end of the SIC sequence and the above-mentioned problem remains for these steps.

The objective of this paper is to propose a conformance relation which permits to provide a correct verdict even in case of asynchronous detection of synchronous input changes. This relation will be applicable when using a (fully or partially) MIC sequence. The paper is organized as follows. The section 2 presents a reminder on test sequence construction from a Mealy machine. A first conformance relation based on sequences of observations is defined in section 3. As it is shown in section 4 that this relation leads to an erroneous verdict when synchronous input changes are seen as asynchronous by the PLC, the appropriate conformance relation for testing of PLC is presented at section 5 while section 6 proposes different solutions to pursue the test after this happened. Concluding remarks and perspectives for further work are drawn up in the last section.

2. BACKGROUND

2.1 Programmable Logic Controllers

This work focuses on testing of mono-tasking PLC which are commonly integrated in automated systems. In this case, the control code is executed according to a cyclic I/O scanning that can be periodic or not and comprises 4 phases:

- Inputs reading,
- Internal and output variables computation,
- Outputs updating,
- Waiting time until the end of the period, if the cycle is periodic. [2]

It must be also reminded that a PLC does not receive (emit) input (output) events but input and output signals. Events can be built by sampling the signals and considering either the rising and falling edges of one input (output) signal or the changes of the whole set of inputs (outputs).

The cyclic I/O scanning may provoke two phenomena that can hinder the implementation of theoretical results on PLC:

- synchronisation of asynchronous input changes,
- desynchronization of synchronous input changes.

The first phenomenon occurs when several inputs are modified asynchronously during the same cycle; these changes will be detected synchronously at the beginning of the next cycle. This issue has been already studied by authors who focused on implementation of the SCT (supervisory control theory) (Fabian and Hellgren (1998)). The second phenomenon has been presented and quantified in (Provost et al. (2011a)). It occurs when synchronous changes of several inputs happen during the input reading phase; some changes may be detected during this phase and other ones only at the next cycle (Fig.2).

[2] This waiting time is equal to zero for a cyclic but not periodic I/O scanning.

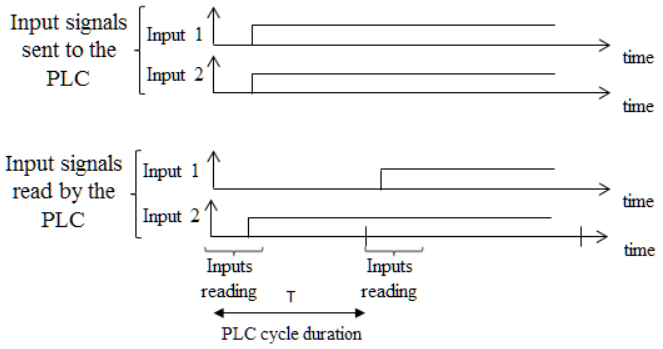

Fig. 2. Asynchronous detection of synchronous input changes

As during the execution of a conformance test, the inputs are changed synchronously by the test-bench, only the second phenomenon will be considered in what follows.

2.2 Test sequence construction

This work assumes that the specification is described by a Mealy machine; this formalism is widespread to represent the specifications of non-timed systems in conformance testing. Formally, a Mealy machine is a 6-tuple M = ($\mathcal{I}_M$, $\mathcal{O}_M$, S, s_{init}, δ, λ) where:

- $\mathcal{I}_M$ is the input alphabet
- $\mathcal{O}_M$ is the output alphabet
- S is the set of states $s \in S$
- $s_{init} \in S$ is the initial state
- $\delta : S \times \mathcal{I}_M \rightarrow S$ is the transition function
- $\lambda : S \times \mathcal{I}_M \rightarrow \mathcal{O}_M$ is the output function

Let V_I (resp. V_O) be the set of Boolean input (resp. output) variables of the PLC. The input alphabet $\mathcal{I}_M$ (output alphabet $\mathcal{O}_M$) of the machine is composed of all the combinations of input (output) variables. The dimension of $\mathcal{I}_M$ ($\mathcal{O}_M$) is $|\mathcal{I}_M| = 2^{|V_I|}$ ($|\mathcal{O}_M| = 2^{|V_O|}$) and each element of the input (output) alphabet is represented by a minterm [3] defined on V_I (V_O). Hence, every input (output) event of the model corresponds to a combination of the input (output) variables of the PLC

The following assumptions are to be made to build a conformance test sequence from this specification model:

- The transition function is complete and deterministic (i.e. every transition $\delta(s \times i_M)$ with $s \in S$ and $i_M \in I_M$ is defined once and only once).
- Each state is distinguishable from the other ones by observation of the output.
- There is no transient evolution, i.e. no input change causes successive changes of state or emitted output. This implies that for any transition leading from an upstream state to a downstream state and labeled by a given input event, there is a self-loop on the downstream state with the same input event.

The basic example presented at Fig.3 will be used to illustrate the results of this paper. This model represents the specified behavior of a PLC with two input variables V_I =

[3] A minterm defined on a set of n Boolean variables is the conjunction of all these variables in their positive or complemented form.

$\{a, b\}$ and one output variable $V_O = \{o\}$. The input alphabet of the specification is thus $\mathcal{I}_M = \{a.b, \overline{a}.b, a.\overline{b}, \overline{a}.\overline{b}\}$ [4] and the output alphabet $\mathcal{O}_M = \{o, \overline{o}\}$. This specification contains 2 states $S = \{s_1, s_2\}$ with $s_{init} = s_1$ and 8 transitions.

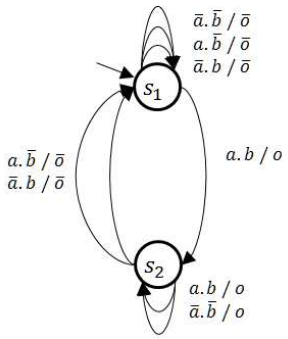

Fig. 3. A basic example of specification

The three assumptions introduced are verified on this Mealy machine:

- The transition function is complete and deterministic.
- When the state s_1 (s_2) is active, the output of the PLC is $False$ ($True$).
- It can be easily checked that there is no transient evolution. The state s_2, for instance, is reached from s_1 when the input event of the machine is $a.b$ (the two inputs of the PLC are $True$) and there is a self-loop on s_2 with this input event.

A test sequence can be then built from this specification. Formally, a test sequence is an ordered list of elementary test steps $et = (s_u,\ I_{exp},\ s_d,\ O_{exp})$ where:

- $s_u \times s_d \in S \times S$
- $I_{exp} \in \mathcal{I}_M$, $O_{exp} \in \mathcal{O}_M$
- $s_d = \delta(s_u, I_{exp})$
- $O_{exp} = \lambda(s_u, I_{exp})$

A test step represents a transition from an upstream state s_u to a downstream state s_d provoked by an input event I_{exp} and during which the output event O_{exp} is emitted.

It must be noted that, as there is no transient evolution, it is possible to test two successive transitions of the Mealy machine with only one input event, i.e. one change of the inputs of the PLC: the transition (s_u, I_{exp}, s_d, O_{exp}) and the transition (s_d, I_{exp}, s_d, O_{exp}), self-loop on the downstream state of the previous transition. In the case of the example of Fig.3, it is possible for instance to test the transitions (s_1, $a.b$, s_2, o) and (s_2, $a.b$, s_2, o) by using the input event $a.b$ from the state s_1. This strategy will be used for every self-loop on a state which is labeled by an input event that is also the label of an incoming transition of this state.

The test sequence must be *initializable*, i.e. the upstream state of the first test step must be the initial state, and *consistent*, i.e. the upstream state of any step, except the first one, must be identical to the downstream state of the previous state. An initializable and consistent test sequence can be built by using either the algorithm presented in Naito and Tsunoyama (1981) or that described in Provost et al. (2010). A minimum-length MIC sequence which crosses at least once every transition of the specification is obtained in the first case (the sequence is termed *complete*). A SIC sequence which does not compulsorily satisfy this objective is produced in the second case and MIC test steps are to be added to obtain a complete test sequence.

For the example of Fig.3 and assuming that the two input variables of the PLC (a and b) are initially $False$, the complete minimum-length MIC test sequence is:

$$\begin{aligned}\mathcal{TS}_{MIC} =&((s_1, \overline{a}.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.\overline{b}, s_2, o),\\ &(s_2, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}))\end{aligned} \tag{1}$$

This test sequence is complete whereas it contains only six test steps. This is due to the fact that, even if the same transition is tested on steps et_2 and et_5, the previously defined strategy is applied three times. For the test steps et_2, et_4 and et_6, two transitions are tested during one test step.

For the same example, the SIC test sequence is:

$$\begin{aligned}\mathcal{TS}_{SIC} =&((s_1, \overline{a}.\overline{b}, s_1, \overline{o}), (s_1, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o),\\ &(s_2, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}))\end{aligned} \tag{2}$$

In this case, the sequence has the same length but is not complete. Indeed, using the defined strategy, only 7 transitions are tested and the self loop on state s_2 labelled with $\overline{a}.\overline{b}$ is missing. There is indeed no possible test step $et_l = (s_l, i_l, s_2, o)$ that can be followed by the one which tests this transition $(s_2, \overline{a}.\overline{b}, s_2, o)$ without breaking the Single-Input-Change rule because the only two candidates $(s_1, a.b, s_2, o)$ and $(s_2, a.b, s_2, o)$ require two input value changes.

Hence, to ensure that all the transitions are tested at least once, MIC test steps must be added and the final complete sequence is given below:

$$\begin{aligned}\mathcal{TS}_{mixed} =&((s_1, \overline{a}.\overline{b}, s_1, \overline{o}), (s_1, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o),\\ &(s_2, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}),\\ &(s_1, a.b, s_2, o), (s_2, \overline{a}.\overline{b}, s_2, o))\end{aligned} \tag{3}$$

This sequence is obviously longer than $\mathcal{TS}_{MIC}$ and includes one test step (the last one) where asynchronous detection of synchronous input changes may happen; therefore this issue is not completely solved.

3. A BASIC CONFORMANCE RELATION

Once an initializable, consistent and complete test sequence built from the specification, it can be used to verify whether the implementation under test, a PLC that executes a control code in this work, behaves as specified. The aim of this section is to explain how this sequence is executed by the test-bench then to propose a relation that permits to emit a verdict about the conformance of the implementation. This presentation will be illustrated by means of the example of Figure 4, a generic

[4] (.) represents the operator of conjunction and (¯) represents the complement.

Mealy machine where only some states and transitions are represented; each transition is labeled with a couple of input event/output event (I^i/O^i) where $I^i \in \mathcal{I}_M$ and $O^i \in \mathcal{O}_M$. More precisely, the following test sequence will be considered:

$$\mathcal{TS} = ((s_0, I^i, s_1, O^i), (s_1, I^j, s_2, O^j), (s_2, I^t, s_3, O^t)) \quad (4)$$

and it will be assumed that this sequence is MIC, i.e. that several inputs of the PLC may be changed when the input event is changed from I^i to I^j or from I^j to I^t.

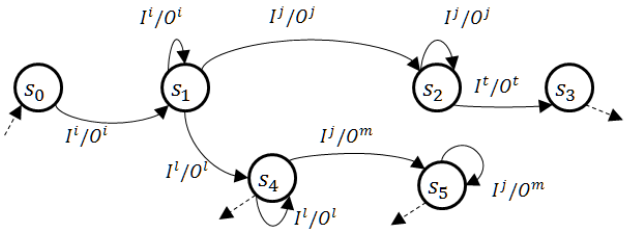

Fig. 4. Part of a generic Mealy machine

3.1 Execution of a test sequence

Each physical input of the PLC must be connected to one output of the test-bench and each physical output of the PLC must be connected to one input of the test-bench before test execution. Once the controller is connected to the test bench, each elementary test step $et = (s_u, I_{exp}, s_d, O_{exp})$ is executed as follows:

- The test bench applies the input combination I_{exp} to the logic inputs of the PLC under test.
- Then, the test bench observes the output combinations emitted by the PLC. This observation phase must be long enough to permit all output computations and updates to be finished. As the maximal value of the causality delay, delay between the date of an input combination change and the date of the corresponding change of the output combination is two PLC cycles [5], the duration of this phase must be 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. Moreover, the observed output combinations are obtained by sampling the output signals of the PLC; the sampling frequency must be high enough to guarantee that at least one combination is observed for every cycle so that every change of the output combination is observed.
- At the end of the observation phase, the observed sequence is compared to the expected one according to a given conformance relation.

It has been pointed out above that the duration of one experimental test step is at least equal to two PLC cycles. This minimal bound guarantees that each output combination update, consequence of an input combination change, can be 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 $et = (s_u, I_{exp}, s_d, O_{exp})$, where $s_u \neq s_d$, to test two successive transitions of the Mealy machine: the transition defined by this step and the transition $et = (s_d, I_{exp}, s_d, O_{exp})$, self-loop on the downstream state of the first transition, as explained at the sub-section 2.2.

The Fig.5 shows, for the second test step of (4) and for several PLC cycles, the input combinations applied to the PLC Fig.5a, as well as the input combinations read by the PLC Fig.5b and the observable output combinations Fig.5c) by assuming that the PLC behaves as specified, i.e. the implementation conforms to the specification. It is reminded that the inputs of a PLC are read at the beginning of each cycle and the outputs updated at the end of this cycle. It matters also to underline that the input combinations read by the PLC are not observable because the test is non-invasive; they are only represented in this figure to explain from which input combinations the observable combinations are computed.

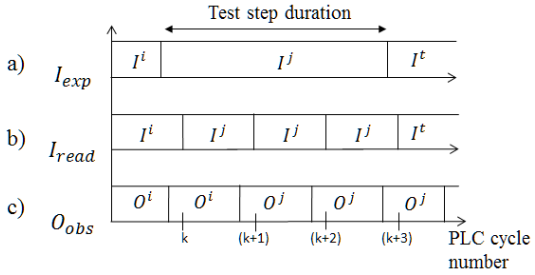

Fig. 5. Expected, read and observable combinations

As the test-bench is not synchronized with the PLC, the expected input combination is changed at any date during a PLC cycle. This input combination is read by the PLC at the beginning of the following cycle and the corresponding output combination is updated at the end of this cycle. The expected input combination remains unchanged during a given number of cycles; the minimal value of this number is equal to 3 according to the above discussion.

3.2 Conformance relation

A Mealy machine that represents an implementation conforms to another machine that represents the specification if and only if O_{obs} is equal to O_{exp} for every test step. This simple theoretical conformance relation cannot be directly applied for conformance testing of PLC because the observable output combination O_{obs} changes during one experimental test step as shown at Fig.5. A first solution would be to consider only the last output combination observed at the end of the observation phase but this solution does not permit to distinguish a correct evolution from s_u to s_d through the considered transition to a sequence of two transitions from s_u to s_d via a third state s'. This is why a solution which is based on the sequence of observed output combinations will be proposed; this sequence [6]:

$$\sigma_{Obs} = (O_{obs_1}, ..., O_{obs_n}) \text{ where } \forall i \in \mathbb{N}^*, O_{obs_i} \in \mathcal{O}_M \quad (5)$$

is composed of the n ($n \in \mathbb{N}^*, n \geq 3$) output combinations observed during this experimental test step.

[5] This maximal value is obtained when the input change occurs just after the input reading phase of a cycle k. This change is detected at the beginning of the cycle $k+1$ and the outputs are updated at the end of this cycle.

[6] Where $\mathbb{N}^*$ is the set of strictly positive integers.

A first conformance relation for a PLC can be then defined.

Definition 1.
Let $et_c = (s_b, I^j, s_c, O^j)$ *be the current test step*
Let $et_p = (s_a, I^i, s_b, O^i)$ *be the previous test step*
The implementation conforms to the specification if for every test step:
If $s_b \neq s_c$*: It exists* $k \in \mathbb{N}^*$ *such as* $k < n$ *and:*

- *If* $k > 1$: $\forall l \in \mathbb{N}^*$ *such as* $l < k$, $O_{obs_l} = O^i$,
- $O_{obs_k} = O^j$
- $\forall m \in \mathbb{N}^*$ *such as* $k < m \leq n$, $O_{obs_m} = O^j$

If $s_b = s_c$: $\forall k \in \mathbb{N}^*$ *such as* $k \leq n$, $O_{obs_k} = O^j$

This relation means that, to declare that an implementation conforms to a specification, the sequence of output combinations observed during every experimental test step must be composed of a sequence of identical combinations that correspond to the expected output combination for the current test step (O^j in Fig.5) that may be preceded (if $s_b \neq s_c$ and $k > 1$) by a subsequence of identical combinations that correspond to the expected output combination for the previous test step (O^i in Fig.5). This relation will be illustrated on the example of Fig.3 by using the test sequence (1); it will be assumed for this example that every experimental test step lasts three PLC cycles and that one output combination is observed for every cycle. Hence, n will be equal to 3.

- The first test step is $et_1 = (s_1, \overline{a}.\overline{b}, s_1, \overline{o})$ which corresponds to a self-loop on the initial state; a sequence of observations for a correct implementation is $\sigma_{Obs_{et1}} = (\overline{o}, \overline{o}, \overline{o})$ because the output combination is unchanged when the corresponding transition is fired.
- The second test step is $et_2 = (s_1, a.b, s_2, o)$. If the sequence of observations is $\sigma_{Obs_{et2}} = (\overline{o}, o, o)$, the implementation conforms to the specification; the state s_2 has been reached.
- The third test step is $et_3 = (s_2, \overline{a}.\overline{b}, s_2, o)$. If the following sequence is observed: (o, o, o), the implementation conforms to the specification. This is not the case according to Definition 1 when the observed sequence is $(o, \overline{o}, \overline{o})$. However this unexpected sequence may come either from a flawed implementation or an asynchronous detection of synchronous input changes. It must be noted indeed that the values of the two PLC input variables are modified synchronously from et_2 to et_3; if the change of b is detected before that of a, a correct implementation will fire the transition from s_2 to s_1 labeled with the input event $\overline{a}.b$ then will execute the self-loop on s_1 where both variables a and b are $False$.

Therefore, Definition 1, which is based only on analysis of a sequence of observations, is not sufficient to separate non-conformance cases from evolutions due to concurrent transitions in case of asynchronous detection of synchronous input changes; a more complete definition of a conformance relation that permits to distinguish these two cases will be given in section 5. It is necessary to study how a sequence of observations is modified when a transition which is concurrent to the considered transition is fired, however; this is the aim of the next section.

4. IMPACT OF AN ASYNCHRONOUS DETECTION OF SYNCHRONOUS INPUT CHANGES

An example of sequence of output combinations that can be observed when synchronous changes of the inputs of the PLC are detected as asynchronous, for the second test step of (4) and always assuming that the implementation behaves as specified (the PLC code is correct), is presented at Fig.6. While the input combination sent to the PLC is I^j, the combinations read by the PLC are successively I^l (at the beginning of the cycle $(k+1)$) then I^j (at the beginning of the cycle $(k+2)$). If several inputs are changed from I^i to I^j and these changes occur during the input reading phase, they may be indeed detected at two successive cycles and a combination I^l is used to compute the output combination during the cycle $(k+1)$. If p variables are changed from I^i to I^j, this unplanned input combination is obtained from I^i by changing only a subset of these variables; changing the remaining ones in I^l provides I^j.

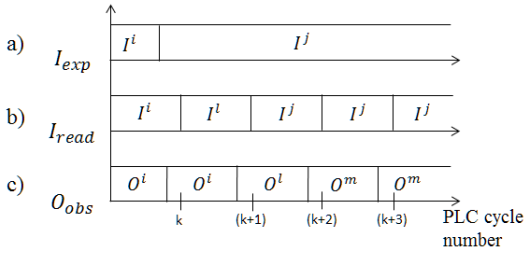

Fig. 6. Sequence of observable combinations when several synchronous input changes are detected as asynchronous

From this sequence of read input combinations, a correct implementation will emit the sequence (O^i, O^l, O^m) that corresponds to the successive firing of the transitions from s_1 to s_4 and from s_4 to s_5. As the expected output sequence is (O^i,O^j,O^j) the relation defined at the previous section will deliver a negative verdict: the implementation does not conform with the specification. This verdict is biased, i.e. a correct implementation is declared flawed (false negative).

A non-valid verdict (a flawed implementation is declared correct (false positive)) is also possible, as illustrated at the Fig.7. This machine represents a flawed implementation if the specification is the model depicted at Fig.4; however, the sequence observed will be (O^i,O^j,O^j) with the input sequence of Fig.6 and the verdict will be positive.

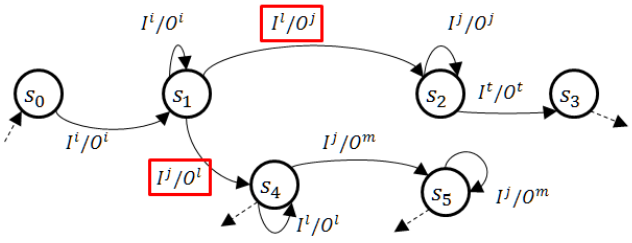

Fig. 7. A model of an erroneous implementation

To avoid these erroneous verdicts, the conformance relation must be modified as explained in the next section.

5. AN APPROPRIATE CONFORMANCE RELATION

The principle of the new conformance relation is not to limit the comparison of the sequence of observations with only one sequence, as in Definition 1, but, if this first comparison is negative, with other sequences which are possible in case of asynchronous detection of synchronous PLC input changes. Therefore, this new relation is an extension of Definition 1 by adding a supplementary comparison phase. The aim of this phase is to check whether the sequence of observations may be obtained from the current state by firing in the specification a sequence of two transitions such that:

- the first transition is labeled by an input event which can be obtained from the expected event of the previous elementary test step (I^i in the discussion of the previous section) by changing a subset of the variables which are changed when this event is replaced by that of the considered test step (I^j in the discussion of the previous section);
- the second transition is labeled by the expected input event of the considered test step.

5.1 Definition

From the non-formal definition above, a formal definition of the appropriate conformance relation for conformance test of PLC can be given. It is assumed that there is one observed output combination per PLC cycle in σ_{Obs}.

Definition 2.
Let $et_c = (s_b, I^j, s_c, O^j)$ be the current test step.
Let $et_p = (s_a, I^i, s_b, O^i)$ be the previous test step.
The implementation conforms to the specification if for every test step there is:

Either:
If $s_b \neq s_c$: It exists $k \in \mathbb{N}^*$ such as $k < n$ and:

- *If $k > 1$: $\forall l \in \mathbb{N}^*$ such as $l < k$, $O_{obs_l} = O^i$,*
- *$O_{obs_k} = O^j$*
- *$\forall m \in \mathbb{N}^*$ such as $k < m \leq n$, $O_{obs_m} = O^j$*

If $s_b = s_c$: $\forall k \in \mathbb{N}^*$ such as $k \leq n$, $O_{obs_k} = O^j$

Or:
It exists $k \in \mathbb{N}^*$ such as:

- *$k < n - 1$,*
- *If $k > 1$, $\forall l \in \mathbb{N}^*$ such as $l < k$, $O_{obs_l} = O^i$,*
- *It exists $I^x \in \mathcal{I}_M$ such as:*
  - *$\left(I^x \backslash I^i \cup I^i \backslash I^x\right) \subset \left(I^i \backslash I^j \cup I^j \backslash I^i\right)$,*
  - *$\lambda(s_b, I^x) = O_{obs_k}$*
- *Let $s = \delta(s_b, I^x)$ be the downstream state of the transition,*
- *It exists a transition such as $\lambda(s, I^j) = O_{obs_k+1}$*
- *and $\forall m \in \mathbb{N}^*$ such as $k + 1 < m \leq n$, $O_{obs_m} = O_{obs_k+1}$*

This conformance relation is split into two parts. The first one is identical to Definition 1. If the verdict is negative after this first comparison, a second analysis is performed. The transitions that start from the upstream state of the considered test step (s_b) and which are labeled with input events that are intermediate combinations (I^x) between I^i and I^j are first searched. If it is possible by firing one of these transitions to emit an output event that corresponds to the k^{th} observation and to reach a state s from which a transition labeled by the couple of events $I^j/O_{obs_{k+1}}$, where O_{obsk+1} is the $(k+1)^{th}$ observation, is possible and that leads to a state s' with a self-loop with this same label, the verdict is positive.

The intermediate input combinations are defined by the expression

$$\left(I^x \backslash I^i \cup I^i \backslash I^x\right) \subset \left(I^i \backslash I^j \cup I^j \backslash I^i\right) \tag{6}$$

where $I^1 \backslash I^2$ is the set of PLC input variables which are $True$ in I^1 and not in I^2. Hence, the term $I^x \backslash I^i$ ($I^i \backslash I^x$) represents the set of variables which are $True$ in I^x (I^i) and not in I^i (I^x). The disjunction of these two sets must be a subset of the set of variables changed from I^i to I^j.

These notations can be illustrated on the example of Fig.3. If $I^i = a.b$ and $I^j = \overline{a}.\overline{b}$, the intermediate combinations are $a.\overline{b}$ and $\overline{a}.b$. More generally, if n PLC input variables are changed from the input event of the previous test step to the input event of the considered test step, $2^n - 2$ intermediate input combinations can be defined.

At last, it must be noted that Definition 2 requires a longer observation phase than Definition 1 because it is based on a sequence of three firings: the first one caused by the intermediate input combination, the second one by the expected input combination and the latter one that corresponds to a self-loop on the final state of this sequence, while at most two firings were considered in Definition 1. As the causality delay is always the same, the observation phase must last at least four PLC cycles. This increased duration is the price to pay to avoid erroneous test verdicts, by distinguishing the unexpected observed sequences which come from errors in the implementation from those which are provoked by asynchronous detection of synchronous changes of the PLC inputs.

5.2 Illustration

This relation may be applied to the third step of the test sequence (1) already analyzed according to Definition 1 in subsection 3.2. As this was the case in this section, it is assumed that the observed sequence is $(o, \overline{o}, \overline{o}, \overline{o})$ while (o, o, o, o) is expected because the objective of this step is to test the self-loop on s_2 that corresponds to the label $(\overline{a}.\overline{b}/o)$. This observed sequence will be interpreted with Definition 2 as the successive firings of the transition from s_2 to s_1 then of the self-loop on s_1 that corresponds to the input combination $\overline{a}.\overline{b}$; the test verdict will be positive. This verdict is correct because the synchronous changes of a and b from the second to the third test step may be detected as asynchronous and the intermediate input combination will be $\overline{a}.b$ or $a.\overline{b}$ during one PLC cycle; an implementation that conforms to the specification (Fig.3) will then fire the transition from s_2 to s_1 then the self-loop on s_1.

In other words, the new definition provides a positive verdict if the observed sequence is accepted by the specification even if it is not the one expected because synchronous input changes have been detected by the PLC as asynchronous. If the sequence is not accepted, a negative

verdict must be obviously delivered. If for instance, for the third step of the test sequence (1), the observed sequence from state s_2 is $(o, \overline{o}, o, o)$, the verdict is negative because this sequence cannot be obtained in the model of Fig.3.

6. DISCUSSION

The definition proposed in subsection 5.1 permits that erroneous verdicts are avoided whatever the way that the PLC input changes are detected. However, when an accepted sequence of observations, which is not the one expected, is found, the test sequence built off-line cannot be pursued because the active state is no more the one that was planned; in the first example of subsection 5.2, this state is s_1 while s_2 is expected as the upstream state of the next test step.

Three solutions can be then considered:

- a test sequence that covers every transition which has not been tested is recomputed from the new active state (this state becomes the initial state for test sequence construction);
- the initial test sequence that has been built off-line prior to test execution is kept but the next test step will start from the new active state;
- a partial sequence that starts from the new active state and ends at the state where synchronous input changes have been detected as asynchronous is determined; the initial test sequence is executed again, once the latter state reached.

In the rest of this section the pros and cons of these solutions will be discussed and illustrated. This discussion will be presented assuming that a test step $et_j = (s_b, I^j, s_c, O^j)$ has been executed and that a desynchronization phenomenon lead to s_{new} as active state.

6.1 Recomputation of a new test sequence from s_{new}

In this case, a new sequence starting from s_{new} and covering at least once every transition which has not been previously tested is built. This solution is always possible if the set of transitions which have been already tested during the test steps et_1 to et_{j-1} has been stored.

On the considered example of the test sequence (1) executed as explained in subsection 5.2, the test steps et_1 and et_2 have been correctly performed; hence only 6 transitions remain to be tested. The new test sequence, from $s_{new} = s_1$, to be executed by the test-bench is:

$$\begin{aligned}\mathcal{TS}_\alpha =&((s_1, a.b, s_2, o), (s_2, \overline{a}.\overline{b}, s_2, o), (s_2, a.\overline{b},\\ &s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}))\end{aligned} \quad (7)$$

6.2 Repositioning in the initial test sequence

This solution does not require any computation of a test sequence during test execution but relies on the assumption that it is possible to find, in the initial test sequence, a previously executed test step that corresponds to the same situation than the current one (active state s_{new} and input values I^j):

$\exists k < j$ such as

$et_k = (s_x, I^j, s_{new}, O^x)$ with $(s_x \in S, O^x \in \mathcal{O}_M)$

If this assumption holds, the initial test sequence is executed from this step. If it does not hold, this solution cannot be selected because some transitions might remain not tested what is in conflict with the test objective.

Despite this solution induces no recomputation of the test sequence, it is not always applicable without a violation of the test objective.

On the example, the assumption holds. Indeed, the current situation $(s_1, \overline{a}.\overline{b})$ corresponds to the situation reached after the execution of the first test step $et_1 = (s_1, \overline{a}.\overline{b}, s_1, \overline{o})$. The initial test sequence is thus restarted since the second test step.

$$\begin{aligned}\mathcal{TS}_\beta =&((s_1, a.b, s_2, o), (s_2, \overline{a}.\overline{b}, s_2, o), (s_2, a.\overline{b},\\ &s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}))\end{aligned} \quad (8)$$

It must be noted that the test sequences (7) and (8) are identical ; as no recomputation of a test sequence is necessary to build (8), the second solution is preferable.

6.3 Coming back to the previous test step

This solution may be applied when the preceding assumption does not hold. It consists in recomputing a SIC test sequence that leads from s_{new} to s_b then to re-execute the initial test sequence from et_j. The SIC condition is introduced to avoid erroneous interpretations of synchronous events during the test steps from s_{new} to s_b. However it is not always possible to build a SIC sequence from two states (Provost et al. (2010)); hence this solution is not always possible.

For the considered example, the SIC sequence can be built and the whole test sequence from s_{new} is :

$$\begin{aligned}\mathcal{TS}_\delta =&((s_1, \overline{a}.b, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.\overline{b}, s_2, o),\\ &(s_2, a.\overline{b}, s_1, \overline{o}), (s_1, a.b, s_2, o), (s_2, \overline{a}.b, s_1, \overline{o}))\end{aligned} \quad (9)$$

It may be noted that (9) is longer than (7) and (8) and requires to compute SIC test steps; the third solution is not the most efficient.

7. CONCLUSION

This paper has shown that the theoretical results obtained for conformance testing of Mealy machines cannot be directly used when testing real PLC. The technological features of these devices (cyclic I/O scanning and non-instantaneous reading of the input signals) require to adapt the conformance relation defined when only models are considered, to obtain correct verdicts. A first relation based on a sequence of observations during several PLC cycles has been introduced. This relation is not sufficient when synchronous input changes are detected as asynchronous; biased and non-valid verdicts may be delivered. To solve this issue, a second conformance relation has been proposed; it is based on analysis of the possible concurrent transitions which start from the current active state and the sequences of observations which are accepted by the specification model.

On-going work is aiming at extending these results to validation of PLC by using HIL (hardware-in-the-loop) techniques. In this case, a real (hardware) PLC is connected to a software simulation of the plant to form a closed-loop system. The HIL approach differs from testing because the PLC is no more considered in isolation; hence, the state space to analyze is that of the PLC constrained by the plant. Up to now, only non-exhaustive simulation has been considered in this approach. Our objective is to investigate whether formal analysis techniques which have been developed for conformance testing of PLC can be used to automatically check the correctness of the PLC behavior from sequences of observations of the inputs and outputs of the simulated plant.

8. ACKNOWLEDGMENT

This work is funded by the French research agency (ANR) as part of the VACSIM project (ANR-11-INSE-004).

REFERENCES

Bochmann, G. and Jourdan, G.V. (2009). Testing k-safe petri nets. In M. Nunez, P. Baker, and M. Merayo (eds.), *Testing of Software and Communication Systems*, volume 5826 of *Lecture Notes in Computer Science*, 33–48. Springer Berlin Heidelberg.

Brinksma, E. and Tretmans, J. (2001). Testing transition systems: An annotated bibliography. In F. Cassez, C. Jard, B. Rozoy, and M. Ryan (eds.), *Modeling and Verification of Parallel Processes*, volume 2067 of *Lecture Notes in Computer Science*, 187–195. Springer Berlin Heidelberg.

Fabian, M. and Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete event systems. In *Decision and Control, 1998 Proceedings of the 37th IEEE Conference on*, volume 3, 3305–3310 vol.3.

Lee, D. and Yannakakis, M. (1996). Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*, 84(8), 1090–1123.

Li, M. and Kumar, R. (2012). Model-based automatic test generation for simulink/stateflow using extended finite automaton. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, 857–862.

Naito, S. and Tsunoyama, M. (1981). Fault detection for sequential machines by transitions tours. In *fault tolerant computer symposium, 1981 Proceedings of the IEEE*, 238–243.

Provost, J., Roussel, J.M., and Faure, J.M. (2010). SIC-testability of sequential logic controllers. In *10th International Workshop on Discrete Event Systems (WODES), 2010*, 203–208.

Provost, J., Roussel, J.M., and Faure, J.M. (2011a). Testing programmable logic controllers from finite state machines specification. In *Dependable Control of Discrete Systems (DCDS), 2011 3rd International Workshop on*, 1–6. IEEE.

Provost, J., Roussel, J.M., and Faure, J.M. (2011b). Translating Grafcet specifications into Mealy machines for conformance test purposes. *Control Engineering Practice*, 19(9), 947–957.

Tretmans, J. (1996). Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3), 103–120.