Conformance test of logic controllers of critical systems from industrial specifications

François Chériaux, Laurence Picci, Julien Provost, Jean-Marc Faure

▶ **To cite this version:**

François Chériaux, Laurence Picci, Julien Provost, Jean-Marc Faure. Conformance test of logic controllers of critical systems from industrial specifications. European Conference on Safety and Reliability - ESREL 2010, Sep 2010, Rhodes, Greece. paper 308. hal-00483215v1

HAL Id: hal-00483215

https://hal.science/hal-00483215v1

Submitted on 12 May 2010 (v1), last revised 16 Sep 2010 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Conformance test of logic controllers of critical systems from industrial specifications

François Chériaux & Laurence Picci
EDF R&D, 6 quai Watier, F-78401 Chatou
Julien Provost & Jean-Marc Faure
LURPA, ENS de Cachan, 61 avenue du Président Wilson, F-94235 Cachan

This paper presents a synthesis of works performed in the frame of an industry/academia cooperative research. The overall objective of this research is to automate the construction of test sequences for conformance test of industrial logic controllers when the expected behavior is described in industrial specification languages. The first contribution is aiming at preventing from combinatorial explosion by preliminary verifications on the implementation so as to check whether it satisfies some structural properties. The second contribution is a method to translate an industrial specification into a formal model to take benefit from theoretical results on conformance test of formal models for discrete event systems description. These two contributions are exemplified on case studies from the domain of energy production.

1 INTRODUCTION

Conformance test is aimed at checking whether an implementation, seen as a black-box with inputs/outputs, behaves correctly with respect to its specification. This kind of test is mandatory for the components of the control system of critical processes, like power production and distribution or transport, and particularly for the software-based logic controllers that are more and more often used in these systems, even to implement safety-related functions.

Conformance test of a logic controller is a non-invasive test that is performed (Figure 1) before operation and consists in sending to the controller an input sequence and comparing the observed output sequence, controller's response to the input sequence, to the expected output sequence so as to build a test verdict (the implemented controller conforms to its specification or not). The set of the input sequence and expected output sequence is termed test sequence.

Numerous theoretical results have been published in the domain of conformance test of logic systems, assuming that the specification is formally described, for instance in the form of a finite state machine (Lee and Yannakakis 1996), a transition system (Tretmans 2008) or, more recently, a particular class of Petri net (von Bochmann and Jourdan 2009). Generally speaking, these results provide a way to build automatically

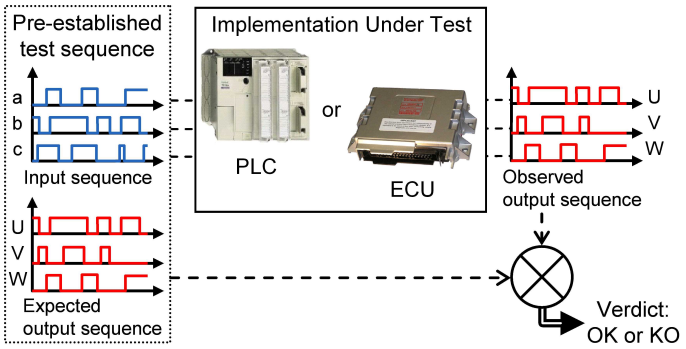

Figure 1: Conformance test of a logic controller

a test sequence from the formal specification model and to deliver a verdict from the observed output sequence.

Unfortunately, these results cannot be used directly for conformance test of industrial logic controllers, because, in industrial practice, the specifications of the behavior of the controllers are not given in formal languages but in tailor-made, often standardized, industrial specification languages, like Logic Functional Diagrams (LFD), Grafcet or state-charts. Test sequences are then built manually, what is a very tedious, time-consuming and error-prone task. Moreover, applying theoretical methods for conformance test sequences generation to industrial-sized examples may easily lead to combinatorial explosion.

This synthesis paper presents the results of a work performed in the frame of an industry/academia cooperative research[1], to contribute to solve these two issues. Section 2 describes shortly the main features of the two industrial specification languages that were selected in this study and underlines their complementary. Section 3 shows how the size of the test sequence can be reduced by verifying whether the implementation satisfies some structural properties, while section 4 presents a method to translate an industrial specification into a formal model, without semantics loss. These two contributions are exemplified on industrial case studies coming from the sector of energy production. Perspectives for further works are given in the last section.

2 INDUSTRIAL SPECIFICATION LANGUAGES FOR LOGIC CONTROLLERS

Specifications of the logic control of industrial systems are not expressed in formal languages but in languages that are tailored-made for automation and often (officially or de facto) standardized. If it is indeed possible to describe the behavior of small-sized control systems with languages issued from the theory of discrete event systems, like those above-mentioned, this is no more the case when industrial-sized systems are considered. Modeling the expected behavior of the logic control of a real industrial system with these languages leads easily to a huge, complex and not readable and understandable model that cannot be used as a specification. On the opposite, industrial specification languages allow to capture the know-how of engineers and to express clearly what they expect. In the case of energy production, two graphical specification languages: LFD and Grafcet, are widely used and have been selected for this study. It matters to note that these languages are also used in other industrial domains that deal with critical processes, for instance railway transport, chemistry, environment. Hence, the results of this study are not limited to a particular industrial domain.

These two languages are complementary because they are used to specify different kinds of functions, as pointed out in sub-section 2.1. Last, it must be underlined that LFD and Grafcet are functional diagrams that do not depend on implementation technology, i.e. a given diagram may be implemented in different ways, for instance in different programming languages or even using hardware solutions.

2.1 Control system architecture

The control system of a power plant is a hierarchical system that comprises four levels (Figure 2): planning and maintenance management (L3), monitoring and operation (L2), automation and control (L1), data acquisition and actuators (L0).

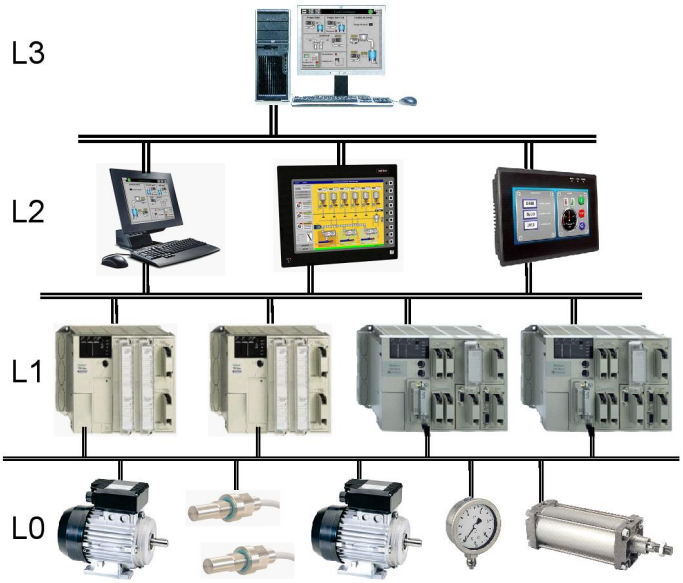

Figure 2: Architecture of the control system

Within each level, analogical and logic control functions are implemented. Experience has shown that the analogical functions are less critical than the logic ones. Hence, the parts of the control architecture that implement analogical functions are validated only by simulation, using a plant simulator, while conformance test of parts which perform logic control functions is compulsory. This explains why this work focuses only on logic controllers.

Moreover, only level L1 is considered in this study. It is closer to the plant, owns shorter response time; hence it impacts safety and dependability more strongly than the upper two levels. The expected behavior of the logic control functions at this level is specified either in LFD or in Grafcet, two languages that are briefly presented below, according to the following rules:

- basic protection functions are described in LFD,
- tasks control functions are described in Grafcet.

2.2 Brief description of the LFD language

LFD is a graphical specification language that is well suited to specify the behavior of elementary reactive functions, such as actuators control functions in accordance with sensors values. This explains why this language is used to specify basic protection functions.

A LFD (Figure 3) is a network of interconnected logic functions that may include combinatorial (NOT, AND, OR), sequential (set- and reset-dominant (SR and RS) memories) and timed functions (on-delay (TON) and off-delay (TOFF) timers). A LFD specifies the expected relations between its inputs (sensors

[1]This research was funded by the French Research Agency (TESTEC project).

values or commands from the upper levels) and outputs (commands to actuators).

The example of figure 3 will be used in the remainder of this paper to illustrate the contribution on test sequence size reduction.

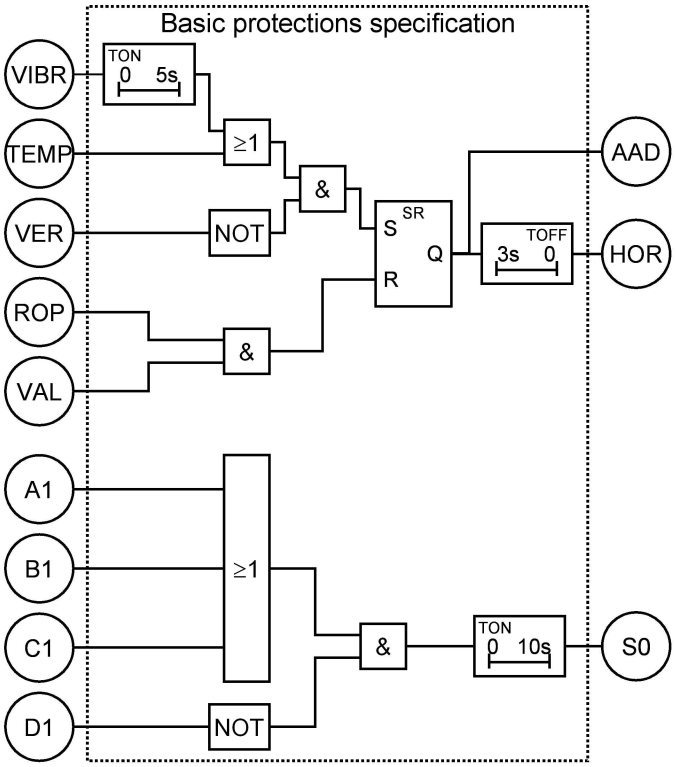

Figure 3: Example of LFD specification

2.3 Brief description of the Grafcet language

Grafcet is a standardized graphical specification language (IEC 60848 2002) to describe the behavior of logic sequential systems. Grafcet was developed from the results of the Petri Nets community, and particularly from those on Interpreted Petri Nets. This language was first standardized in France at the beginning of the 80s, and at the international level in 1988. Since this date, several extensions have been proposed to enhance the modeling possibilities; they are included in the last version of the standard (IEC 60848 2002). A good scientific presentation of the main features of the current version of the Grafcet standard can be found in (Guéguen and Bouteille 2001).

This language is widely used in several industrial domains because it allows engineers to describe easily tasks sequences, tasks selection, parallelism and synchronization. A Grafcet is composed of steps, represented by squares, and transitions, represented by horizontal lines; a step can be linked only to transitions and a transition linked only to steps. 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 steps activity variables.

The example of figure 4 will be used in the remainder of this paper to illustrate the contribution on translation of an industrial specification into a formal model.

3 BUILDING TEST SEQUENCES FROM LFD

The overall objective of the work described in this section is to avoid (or limit) combinatorial explosion during test sequence construction and to develop a simple, while efficient, method that allows conformance test of a logic controller with a size-reduced test sequence, while providing trustworthy test results. The principle of this method is to verify structural properties of the implementation before the test. This work focuses on conformance test of an implementation that performs highly critical, but simple, logic control functions, e.g. actuators start and stop, protection of actuators, alarms generation, and is specified in LFD.

3.1 Method overview

According to the standard on safety-related instrumentation and control systems for nuclear power plants (IEC 60880 2006), system validation is the confirmation by examination and provision of relevant evidences that the system fulfills its specification. Conformance test is clearly a validation technique for control systems and it could be thought that an exhaustive test sequence is to be constructed to provide relevant evidences, when critical systems are considered. Nevertheless, attempting to build an exhaustive test sequence may easily lead to combinatorial explosion; for a LFD with n_i logic inputs and n_m internal memories, this sequence can include up to 2^n test steps, with $n = n_i + n_m$, for instance. This explains why reduction of the test sequence size must be investigated.

The method that was developed to construct automatically a size-reduced conformance test sequence from a LFD comprises two main phases:

1. Verification of the following six structural properties:
   a) The specification and implementation own the same memories and timers.
   b) Each output of a memory is also an output of the system[2].
   c) There is no ‘backwards loop’, i.e. an output of a function must not be connected to an in-

[2]This restriction is not too strong because this configuration is very common for real LFDs; however, if this is not the case, an additional output must be added in a ‘design to test’ approach.

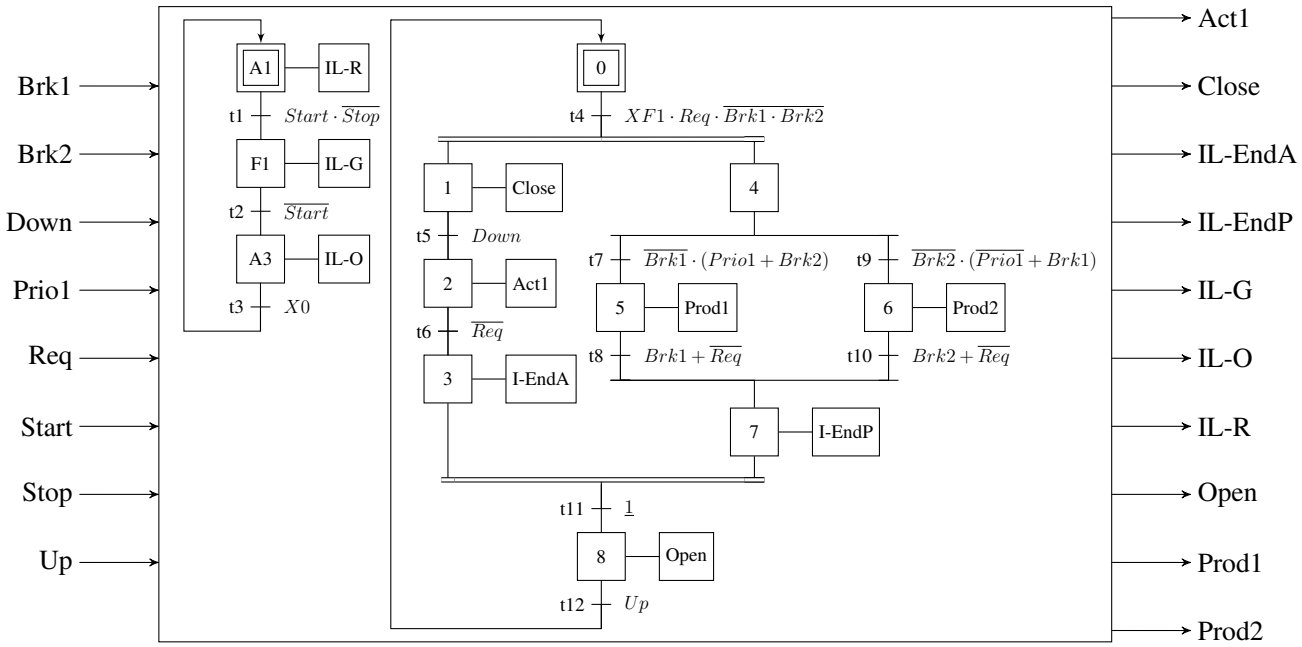

Figure 4: Example of Grafcet specification

put of an upstream function or be input of this same function. For instance, the output of the SR function in figure 3 cannot be connected to an input of the combinatorial functions that are located upstream of this sequential function.

d) The dependency relations that link the outputs of the system to its inputs and the stored variables (outputs of memories) in the implementation are identical to those of the specification.

e) The dependency relations that link the inputs of the timers and memories to the inputs of the system in the implementation are identical to those of the specification.

f) The dependency relations that link the outputs of the system to its inputs and the timers outputs in the implementation are identical to those of the specification.

2. Construction of the test sequence by using the results of the first phase. The core idea of this phase is to take benefit from the knowledge on the structure of the implementation to generate the only test steps that are necessary to test its behavior. The dependency relations that have been stated point out indeed how to control and observe the memories and timers states; among the whole set of test steps that can be obtained by simulation of the specification, the relevant test steps are those which satisfy these two conditions.

It must be noted that the first phase permits to avoid a conformance test be performed on an implementation whose structure does not satisfy some structural properties, what reduces implementation validation duration.

This method has been implemented in a prototype tool (Salaün, Chériaux, and Trognon 2007; Chériaux, Salaün, and Daumas 2009). The results obtained with this tool will be illustrated in the next sub-section on the basis of the example of figure 3. Although limited to logic systems, this approach corresponds to an innovative prospect for model-based conformance test, taking benefit from structural properties verification to reduce test complexity.

3.2 Verification of the structural properties of the implementation

The LFD specification of figure 3 includes one SR memory, 2 TON (set to 5 and 10 seconds) and one 3 seconds TOFF timers; the output of the memory is connected to output AAD and there is no backwards loop. The following relations between the three outputs and the nine inputs can be easily obtained by dependency analysis of the diagram:

- AAD = SR(AND(OR(TON_5S(VIBR),TEMP), NOT(VER)),AND(ROP,VAL))
- HOR = TOFF_3S(SR(AND(OR(TON_5S(VIBR),TEMP),NOT(VER)), AND(ROP,VAL)))
- S0 = TON_10S(AND(OR(A1,B1,C1),NOT(D1)))

From these global relations, class d, e and f properties can easily be obtained. For instance, the following properties can be stated from the first relation:

- AAD depends on SR_Q,
- TON_5S_I depends only on VIBR,
- AAD depends on TON_5S_O, TEMP, VER, ROP and VAL,

where SR_Q, TON_5S_I and TON_5S_O represent respectively the output of the memory and the input and output of the 5 s TON timer. Once the structural properties stated, they are checked on the implementation. Then, the prototype tool developed during this work includes a configurable parser to check structural properties for logic controllers developed in the following languages: two languages of the IEC 61131-3 standard for programming of programmable logic controllers (Ladder Diagram (LD) and Function Block Diagram (FBD)), C and VHDL). If verification shows that the implementation satisfies the structural properties that have been stated from the specification, conformance test can be performed with the test sequence whose construction is explained in the following sub-section.

This approach has been assessed successfully with several examples that describe real industrial LFD specifications. The main limitation is that the LFD must not include 'backwards loop'. However, this limitation is not really significant because industrial experience has shown that basic protection systems behavior can always be specified with LFD without backwards loop.

3.3 Test sequences construction

This construction takes benefit from the results of I/O dependency analysis and the properties that have been stated in the first phase. For room reasons, it is not possible to present all the details of the construction; only the two main principles will be given.

Dependency relations between outputs and inputs permit to pinpoint the inputs that are used to compute a given output; then, it is possible to construct a test sequence that contains only values of these inputs and expected values of the considered output. For the example, dependency relations show that the first two outputs only depend on the same five inputs and the third output on the remaining four inputs. Hence, the test sequence will comprise two parts; the first one will be composed of values of the only first five inputs (VIBR to VAL) and the corresponding expected values of the first two outputs, while the second part will contain values of the only last four inputs (A1 to D1) and the corresponding expected values of the last output. This splitting principle may shorten strongly test sequences.

Moreover timers behavior can be tested more easily when the relations between their inputs-outputs and the inputs-outputs of the system are known; this behavior is then controllable and observable and test patterns can be defined, as exemplified in table 1. The second test sequence must be performed after the first one because it assumes that the timer TON_5S, observed from the output AAD, behaves correctly.

Test of TON_5S by observing the output AAD						
TEMP	ROP	VAL	VER	VIBR	AAD	Remarks
0	1	1	0	0	0	SR, then AAD reset
0	0	0	0	0	0	TON_5S output remains at 0
0	0	0	0	1	0 during 5,000 ms, then 1	Timer behavior observation
Test of TOF_3S by observing the output HOR						
TEMP	ROP	VAL	VER	VIBR	HOR	Remarks
1	0	0	0	0	1	SR set
0	1	1	0	1	1 during 3,000 ms, then 0 during 2,000 ms, then 1	Timer behavior observation
0	0	0	0	1	1	SR remains set
0	1	1	0	0	1 during 3,000 ms, then 0	Timer behavior observation

Table 1: Testing timers

For this specification (nine inputs, one memory and three timers), a test sequence constructed without taking benefit from the structural properties analysis includes more than thousand test steps and is executed in about 5 hours. When the results of this analysis are introduced, less than hundred test steps are necessary and the test is performed in about 1 minute; thus, in this case, the test sequence length is reduced by one order of magnitude and the test duration by more than two orders.

3.4 Discussion

It has been shown in this section that a size-reduced test sequence can be used for conformance test from a LFD specification that does not contain any backwards loop, provided that some structural properties of the implementation have been checked previously; these properties are obtained by analysis of the specification model.

This approach combines verification techniques and conformance test. Moreover, when the verification results are negative, non-conformance of the implementation with regard to the specification can be stated without test, what is time-saving. These results have been implemented in a prototype software tool; experiments with several diagrams have shown that the overall time to build and verify the structural properties complies with the requirements of industrial processes of design and implementation of control systems.

It has been shown in this section that a size-reduced test sequence can be used for conformance test from a LFD specification that does not contain any back-

wards loop, provided that it has been checked previously that the I/O dependency relations of the implementation are identical to those of the specification. This approach combines verification of I/O dependency relations and conformance test. Moreover, when the result of this verification is negative, non-conformance of the implementation can be stated without test, what is time-saving.

4 BUILDING TEST SEQUENCES FROM GRAFCET SPECIFICATIONS

The aim of this section is to show how a test sequence can be automatically built from a Grafcet specification by translating this specification into a formal model, without semantics loss. This work considers only non timed models because the first concern of engineers during conformance test is functional correctness; conformance test for time correctness is a second concern, once functional correctness is ensured, and motivates further research. Mealy machines have been chosen as the target formalism because it is suited to non timed logic systems with inputs/outputs and that numerous results have been previously obtained in the field of conformance test of this kind of finite state machine. Hence, it will be possible to take benefit from these results, once the Mealy machine that is equivalent to the initial Grafcet obtained,

4.1 Translation of a Grafcet model into an equivalent Mealy machine

4.1.1 Method overview

The translation method of a Grafcet into an equivalent Mealy machine, without semantics loss, comprises two phases (Figure 5) that are detailed in the next two sub-sections:

1. Construction of the automaton, termed Stable Location Automaton (SLA), that represents formally all stable states of the logic system described by the Grafcet as well as all evolutions between these states;

2. Translation of this automaton into an equivalent Mealy machine.

4.1.2 Construction of the SLA

The reader is reminded that several steps may be simultaneously active in a Grafcet, for instance because they are located in two parallel sequences, like steps 1 and 4, or 2 and 7 in figure 4. Moreover, according to the evolution rule #4 of the Grafcet standard,

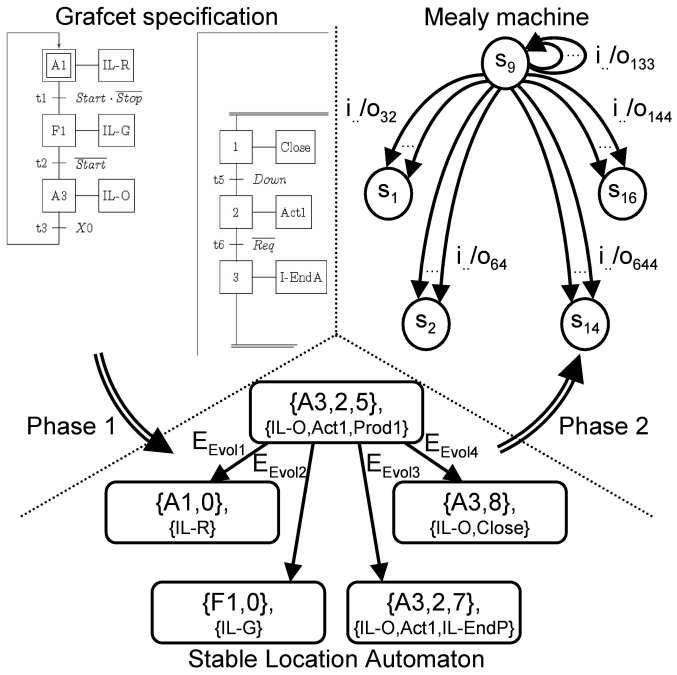

Figure 5: From Grafcet to Mealy machine

several transitions can be simultaneously fired if they are simultaneously fireable. Then, two new concepts, called location and evolution, have been to be introduced during this work, to define formally the state space of the logic system that underlies a Grafcet model.

A location is characterized by a set of simultaneously active steps, termed situation, and the set of outputs that are emitted when this situation is active. For the example of figure 4, location $(\{A3, 2, 5\}, \{IL - 0, Act1, Prod1\})$ means that steps $A3$, 2 and 5, and only these steps, are active and outputs $IL - 0$, $Act1$ and $Prod1$, and only these outputs, emitted. It might be argued that it is not necessary, for this example, to give the list of emitted outputs, once the list of active steps given. However, different sets of outputs may be emitted for the same situation, when the Grafcet contains stored actions or continuous actions that depend on inputs values (conditional actions). Moreover, only the outputs of the logic controller whose behaviour is specified by the Grafcet are observable during conformance test. These two reasons justify the location definition.

Regarding the evolutions in response to inputs events, the Grafcet standard introduces the concepts of transient and non-transient evolutions. A transient evolution is characterized by the firing of several successive sets of transitions on the occurrence of a single input event; on the opposite, a non-transient evolution is one and only one firing of a set of simultaneous transitions when an input event occurs. For the example of figure 4, the evolution from $(A3, 2, 5)$ to $(A3, 2, 7)$ is non-transient (only the transitions $t8$ is

fired) while the evolution from $(A3, 2, 5)$ to $(A1, 0)$ is transient (the two transitions $t6$ and $t8$ are simultaneously fired, then transition $t11$ is fired, then transition $t12$ is fired, and finally transition $t3$ is fired. During a transient evolution, several situations are crossed from the source to the target situation, $((A3, 3, 7), (A3, 8), (A3, 0))$ in the above example). The standard pinpoints that the continuous actions associated to these 'transient' situations are not executed and consequently the associated outputs are not emitted. As only the outputs of the implementation are observable during conformance test, the 'transient' situations are useless in this work and only the source and target situation of any evolution, named stable situations, must be kept.

Hence, the SLA obtained from a Grafcet is a finite state machine that contains:

- All stable locations, i.e. the locations defined from the stable situations;
- All evolutions from stable location to stable location. Each evolution is caused by a change of the inputs values and is labeled with an evolution condition, Boolean expression on the inputs values.

A software tool was developed during this work to construct automatically a test sequence from a Grafcet model; the first treatment performed by this tool is the automatic construction of the SLA. For the example of figure 4, this automaton comprises 16 stable locations and 102 evolutions.

4.1.3 Construction of the equivalent Mealy machine

A Mealy machine is a finite state machine that is defined on input and output events alphabets. Every transition of this machine is labeled with a couple of input and output event. Then, the issue to solve, to translate a SLA into a Mealy machine, is to translate a state machine whose evolution conditions are defined by Boolean expressions into an event-based state machine, without any semantics loss. This can be achieved by first defining the elements of the input and output alphabets as the different combinations of the logic inputs and outputs of the Grafcet[3]. For the example of figure 4, the input alphabet of the equivalent Mealy machine contains $2^8 = 256$ elements, because there are 8 logic inputs, and the output alphabet $2^{10} = 1024$ elements, because the Grafcet model owns 10 outputs.

As a SLA is already a state machine, each location of the SLA gives rise to a state of the Mealy machine. Transition and output functions, which define respectively the transitions between the states and the elements of the output alphabet associated to the transitions, can then be computed from the SLA evolution conditions. For room reasons, it is not possible to give a detailed definition of these two functions; the interested reader is referred to (Provost, Roussel, and Faure 2009). The size of the equivalent Mealy machine is easily computable; if n_{states} and $n_{transitions}$ represent respectively the number of states and transitions of this machine, and $n_{locations}$ and $n_{input-var}$ the number of locations and input variables of the SLA, it comes:

$$\begin{cases} n_{states} = n_{locations} \\ n_{transitions} = n_{locations} \cdot 2^{n_{input-var}} \end{cases} \quad (1)$$

It must be noted that the number of evolutions of the SLA has no influence on the size of the Mealy machine.

The Grafcet model of figure 4 can then be translated into an equivalent Mealy machine that comprises 16 states and $4,096 = 16 \cdot 2^8$ transitions.

4.2 Test sequence construction

Once the equivalent Mealy machine built, a test sequence for conformance test of a logic controller that is supposed to implement the initial Grafcet can then be obtained by using one of the methods surveyed in (Lee and Yannakakis 1996). This method will have to provide an exhaustive and minimum-length test sequence. When control of critical systems is considered, as this is the case in this work, the test sequence must be indeed exhaustive, i.e. all evolutions, for all combinations of the logic inputs, of the specification must be tested to avoid some expected behaviors be missing in the implementation. If the specification is formally represented with a Mealy machine, an exhaustive test sequence crosses every transition at least once. Last, a minimum-length test sequence is searched to lessen conformance test duration.

The method to obtain the minimum-length, while exhaustive, test sequence of a Mealy machine is called transition tour method (Naito and Tsunoyama 1981) and is a particular solution, for a graph that represents the structure of a Mealy machine (states and transitions between states), of a well-known problem in graph theory: the Chinese Postman Problem (Mei-Ko 1962). The general formulation of this problem is the following: "Find a minimum-length closed path that visits each edge in the graph at least once". As the graph which describes the structure of a Mealy machine is directed, but not weighted, the problem is simplified.

Using the software tool developed during this work and that integrates the transition-tour method, a $6,436$ steps test sequence was generated from the example of figure 4. If a step of this sequence is done every

[3]The inputs and outputs of the Grafcet and the corresponding SLA are identical.

20 ms during conformance test, the whole test execution will last approximately 2 minutes and 10 seconds, what is quite reasonable.

4.3 Discussion

This section has shown that an exhaustive conformance test sequence can be automatically generated from a non-timed Grafcet specification. The main contribution of this work is a method to translate automatically the industrial specification model into a formal model, without semantics loss. The principle of this method is to generate the whole state space of the specification and can be applied to other industrial languages for behavior specification.

All results were implemented in a prototype software tool. Experiments with several Grafcet models shown that the overall time to construct the test sequence from the initial Grafcet, including the translation from Grafcet to the equivalent Mealy machine is very short (some seconds at most), what guarantees control engineers acceptance.

5 CONCLUSION AND PERSPECTIVES

Safety and dependability of critical systems depend more and more on the correct behavior of logic controllers. This explains the strong demand from industry for conformance test techniques that can be used during the development of industrial control systems; to meet this objective, the test sequences must be built automatically from specifications in industrial specification languages. The two contributions that are presented in this paper are aiming to provide solutions to this issue. The first contribution focuses on specifications in LFD, a language for basic protection functions specification, and on reduction of the size of the test sequence by preliminary verifications of structural properties of the implementation. The second one considers specifications in Grafcet, a standardized language for more complex control functions for which the previous contribution cannot be applied, and on translation of the specification into a formal model so as to build an exhaustive and minimum-length test sequence. The test sequences that were obtained in both cases have been used successfully with real test-benches that integrated commercially available PLCs (Programmable Logic Controllers).

The perspectives of this research are both technical and scientific. First, integration of these results within an existing environment for controllers' specification, design and implementation is planned in a near future. In a longer term, the current limitations (no backwards loop in the LFD, only non-timed Grafcet) should be removed to increase the scope of the results and their interest for safety improvement.

REFERENCES

Chériaux, F., P. Salaün, and F. Daumas (2009). Functional test of control systems ensuring a high coverage rate. In *6th American Nuclear Society International Topical Meeting on NPIC&HMIT (NPIC&HMIT 2009)*.

Guéguen, H. and N. Bouteille (2001). Extensions of grafcet to structure behavioural specifications. *Control Engineering Practice 9*(7), 743 – 756.

IEC 60848 (2002). *GRAFCET specification language for sequential function charts*. Number 2. International Electrotechnical Commission.

IEC 60880 (2006). *Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions*. Number 2. International Electrotechnical Commission.

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

Mei-Ko, K. (1962). Graphic programming using odd or even points. *Chinese Mathematics 1*, 273–277.

Naito, S. and M. Tsunoyama (1981). Fault detection for sequential machines by transitions tours. In *Proceedings of the 11th IEEE Fault Tolerant Computer Symposium*, pp. 238–243.

Provost, J., J.-M. Roussel, and J.-M. Faure (2009). Test sequence construction from SFC specification. In *2nd IFAC Workshop on Dependable Control of Discrete Systems (DCDS'09)*.

Salaün, P., F. Chériaux, and D. Trognon (2007). Prospects for model-based testing of discrete safety systems. In *1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07)*.

Tretmans, J. (2008). Model based testing with labelled transition systems. *Lecture Notes in Computer Science 4949*, 1–38.

von Bochmann, G. and G.-V. Jourdan (2009). Testing k-safe petri nets. In *TestCom/FATES - Testing of Software and Communication Systems*, Volume 5826 of Lecture Notes in Computer Science, pp. 33–48.