Recent Innovations in Mechatronics (RIiM) Vol. 3. (2016). No. 1-2.
DOI: 10.17667/riim.2016.1-2/9.

Simulation and Formal Verification for Improving Safety of PLC Programs

Joel Galvão
MEtRICs Research Center, University of Minho
Campus of Azurém, 4800-058
Guimarães, Portugal

José Machado
Mechanical Engineering Department,
MEtRICs Research Center, University of Minho
Campus of Azurém, 4800-058
Guimarães, Portugal
jmachado@dem.uminho.pt

Abstract—**The use of analysis techniques for improving quality of software for industrial controllers is widely used. Mainly Simulation and Formal Verification can be used as complementary techniques improving dependability of mechatronic systems behavior. In this paper there are used Simulation and Formal Verification for guaranteeing safe software for Programmable Logic Controllers, mainly related with using Function blocks of IEC 61131-3 standard. For studying, simulating and verifying behavior of those blocks are used timed automata, as modeling formalism, and UPPAAL, as tool for simulation and Formal Verification purposes.**

Keywords—IEC 61131-3, Simulation, Formal Verification, Dependable Mechatronic Systems

I. INTRODUCTION

There are several techniques to analyses this type of systems, but Simulation by MiL (Model-in-the-Loop) and Formal Verification by Model Checking [1] two possible methods to achieve the aim secure command specification [2]. Same researchers believe that Simulation is considerably better because is possible to study if the developed code effectively perform the task needed, and if necessary realize same corrections according to the needs. Although was same disadvantages [3], such as, just a part of the domain of possible behaviours of the controller is tested.

One the other hand, the formal verification by Model Checking techniques make possible to test if the developed system respond to the project specifications in all the domain of possible behaviours of the controller, and the controller never reach a deadlock state, and is has been said that model checking is the only know method to ensure that the code is without any error [4]. Nevertheless, was the need to use logic that same consider difficult to utilize and understand [5].

The two techniques (Simulation by MiL and Formal Verification by Model Checking) the granularity of the models is very important. This fact leads to the objective of this research, that focuses on developing models for the behavior of the function blocks defined by the standard IEC 61 131-3 [6], taking into account a methodology that combines the advantages of Simulation by MiL and Formal Verification by Model Checking using the same models, to allow a more careful safety analysis of the command specification of PLC (Programmable Logic Controllers).

Considering simulation, one of the pioneers in this area is the work proposed by Baresi in 1997 [7]. There are same commercial software's like Arena and AutoMod, but this applications consider little about the logic behind the control therefor they cannot be effectively used to test command specifications [8]. The paper presented by reference [9], that demonstrates a technique to simulate and visual verify, that begins with the code written in Ladder Diagram, one of the IEC 61 131-3 languages, using finite state automata [10]. There are several works [7] [11] that use the formalism Discrete Event System Specification [12], this works try to reduce the time need to simulate a system. In [7] they present an inverse methodology that uses data from time-stamped signal history and a PLC input/output signal table extracted from the existing production system to create the simulation models. In the other hand the article [11] demonstrate the advantages of using templates to generate de models.

In the point of view of Formal Verification by Model Checking, this method was first applied to control systems by Moon in 1992 [13]. This technique was them utilized by a great deal of authors, but the formalisms used to specify the system behavior, the method the properties are written and the applications used are different [14][15] [16] [17] [18][19]

The work group of reference [20] present investigation using NuSMV [21]. The properties are specified in computation tree logic[22] or Linear Temporal Logic [23] , and the code is written in Structured Text , other of the IIEC 61 131-3 languages. The work focus on the modulation of time in a realistic manner, to accomplish that they developed models for the function block TON (Time ON delay) defined by the IEC 61 313-3 standard [6].

Other technique is proposed by [24], again using model checking based in models created in BIP (Behavior, Interactions, Priorities) [25]. In this article are proposed models for the Program Organization Unit define by the IEC 61 131-3 [6] standard, once more with especial attention to the function block TON. To verify properties is used the D-Finder [26], that allow detecting deadlock and other type of behaviours. This techniques do not consider time constraints, that turn the analyses limited [27].

The investigators in [28] propose a methodology that uses program code written in sequential function charts , another of the IEC 61 131-3 languages, that is than converted to Timed Automata [29] and from the project specifications are formulated affirmations to verify in the models utilizing TCTL (Timed Computation Tree Logic) [30] (Fig. 1). All the process for the modulation to the verification are realized on the application UPPAAL [31]. They propose a model for the function block TON, because they consider the modulation of the controller behaviour has to be as close as possible to what happens in the equipments. On the other hand reference [32] propose a technique to convert code written in Function Blocks Diagram to Timed Automata were they consider models for the Program Organization Units more particularly functions and function blocks.

In this paper is considered a methodology to make safety analyses of command specification of industrial controllers, more precisely PLC that tries to combine de advantages of simulation and formal verification, using to describe the behaviours of mechatronic system the formalism timed automata in the application UPPAAL, as displayed in Fig. 1.

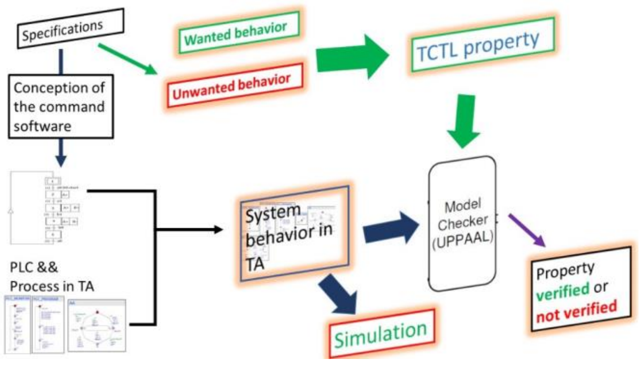

Fig. 1. Analyses methodology applied in this paper

For this approach, it is considered the specification developed in SFC (Sequential Function Chart). Also, the methodology for creating the global model of the system in Timed Automata, for simulation and formal verification purposes is proposed.

In order to achieve the goals proposed for this work, this paper is organised as follows: section 2 proposes a case study, illustrating the approach and, also presents the formal controller's specification taking into account the intended behaviour for the system; section 3 deals with some work hypothesis, mainly concerning the translation of the specification to timed automata, in order to achieve the task of simulation and formal verification, using the UPPAAL software; and, finally, there are presented some conclusions and future work, in section 4.

II. Case study

A. Specification of the controller

In this work, the automatic system used as case study is a car barrier, to be used in parking lot, schematically represented in Fig. 2. This automatic system is actuated by one motor with two directions of movement: one controls the movement with direction "up" (M_UP) and another controls the movement with direction "down" (M_D). Besides that, the system has a set of sensors: one detects the presence of one car at a time (s1) and two other sensors detect the barrier position, "s_up" on the up position and "s_d" on the down position

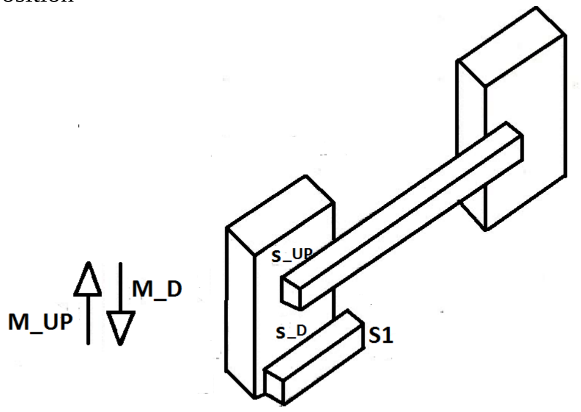

Fig. 2. Schematic representation of the car barrier, with respective sensors and actuators.

The input are: the sensor "si" responsible for detects presence of car; the sensor "s_up" that detects the barrier in the up position up; and the sensor s_down that has the task to detect the down position of the barrier.

In the other hand, the system was to output that are the orders to open the barrier (M_UP), and to close it (M_D).

The controller behavior are: when appears a car, the barrier must move up and when disappears the car, the barrier must move down. If, in some moment, a new car appears the barrier must go up and so on. This intended behavior is described on the Fig. 2 formalized by a SFC.

Consequently, when is no car detected in sensor the barrier must be closed (corresponding to down position) that corresponds to the initial position considered for the system. This way, all the Boolean conditions associated to all the transitions of this model correspond to rising or falling edges of the mentioned sensors.

It is intended that this specification be implemented in a PLC, which program will be written considering Ladder language and Functions blocks proposed in IEC 61131-3 [6] Because this work is devoted to the presentation and verification of the behavior correspondent to rising and falling edges, this subject will be treated with focused special attention.

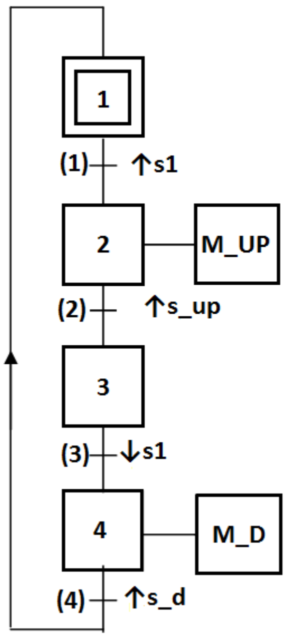

Fig. 3. Sequential Function Charts of the comportment described before.

B. *Translation of the controller specification to Ladder and Function blocks*

The translation of the presented specification, to PLC programming language defined in [6], considers two distinct parts: one concerns the translation of the dynamics of the model according methodology proposed in [2]. Concerning the behavior of the rising and falling edges there are considered the respective comportment defined on the standard. In this case, the behavior intended is as follows: there are two edge detection types, one used for transition of the logic value 0 to 1 (rising edge), and other that does the opposite recording (falling edge).

To model this, the rising edge behavior is described in Fig.4.

```
FUNCTION_BLOCK          R_TRIG
VAR_INPUT
CLK  :  BOOL
END_VAR
VAR_OUTPUT
Q  :  BOOL
END_VAR
VAR_RETAIN
MEM  :  BOOL:=0
END_VAR
Q  :=  CLK AND NOT MEM
MEM  :=  CLK
END_ FUNCTION_BLOCK
```

Fig. 4. Rising edge behavior [6] .

The code demonstrates that if input signal ("CLK"), that represents the variable that is intended to be recorded, the state changes, there is an internal variable ("MEM") that keeps the value of "CLK" in every scan cycle. That information is not wasted because is recycled in the output (Q) calculation in the next PLC scan cycle. When "Q" has the logic value 1 means that the "CLK" has made the rising edge changeover [6].

Nevertheless, sometimes the recording need is different. Some cases the need is to record the moment where a signal changes from Boolean value 1 to 0. This corresponds to the situation corresponding to the falling edge, which behavior is described and presented in Fig. 5.

```
FUNCTION_BLOCK          F_TRIG
VAR_INPUT
CLK  :  BOOL
END_VAR
VAR_OUTPUT
Q  :  BOOL
END_VAR
VAR_RETAIN
MEM  :  BOOL:=1
END_VAR
Q  :=  NOT CLK AND NOT MEM
MEM  :=  NOT CLK
END_ FUNCTION_BLOCK
```

Fig. 5. Falling edge behavior [6].

In this case the code is made for saving the moment when variable that we want to study changes from de logic value 1 to 0. As in the rising edge there is one input ("CLK"), one memory variable (MEM), but this time records the negation of CLK every PLC scan cycle. When CLK and MEM are zero, Q will be one. This has meaning that the analyzed variable changed from one to zero.

Concerning the specification presented in Fig. 3, there are considered both rising and falling edges. This way, those Boolean values will be calculated as demonstrated above.

III. Simulation and Formal verification of the specification

In order to perform the simulation and formal verification it was followed the approach proposed in for the translation of SFC to Timed Automata (TA) [33], for obtaining the timed automata model.

Also, it has been considered the modeling of the comportment of the controller. For this, a modular method has been followed for obtaining the global model to be simulated and formally verified in UPPAAL.

The simulation techniques can be classified by SiL (Software-in-the-Loop), MiL, HiL (Hardware-in-the-Loop), and LT (laboratory testing).

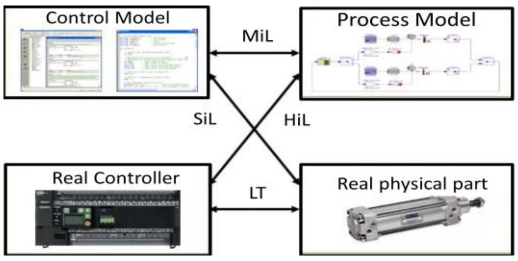

Fig. 6. Simulation tecniques

All these simulation techniques have the particularity of not test all the space behavior of the controller, making it impossible to assert its effectiveness to one hundred percent.

In this investigation, is considered a simulation technique using models (MIL), either the program or to the physical part of mechatronic closed-loop system. This technique is generally used in early phases of development of new process equipment. No need for special equipment, just are cheaper than before. When developed in an appropriate environment, and be able to simulate, it is also feasible to verification by model checking, which ensures analysis of all the controller behavior space.

The models attempt to interact with each other in the same way as mechatronic systems interact in reality. In order to achieve this purpose, a charge model is required to manage the order in which they are executed and how they interact. First, there are two major groups of models, representing the behavior of the PLC and a group which react as in the process.

The interaction between the two parts of the model is made through variables. the process variables every PLC cycle are assigned to the internal variables of the controller, and this data will run its internal code, which will calculate the outputs. This information is again transmitted to the process controller through the allocation of variables to their corresponding values of the process.

It must be highlighted that the main problem for performing simulation and formal verification is not the creation of the modules that compose the global model in TA, but the synchronization of the evolution of the modules. This is because it must be considered the internal PLC scan and the changing of the logical values of the variables must be guaranteed according the correct functioning of the PLC. For this purpose it was created a model for the management of all other modules, in order to guarantee the intended correct verification. The modelling of all system, in one only module, is not achievable and cannot be proposed as a methodology for solving problems of this kind.

In order to illustrate the model proposed for the rising and falling edges is presented, in figure 5 the TA model of the rising edge and falling edge of the sensor s1.

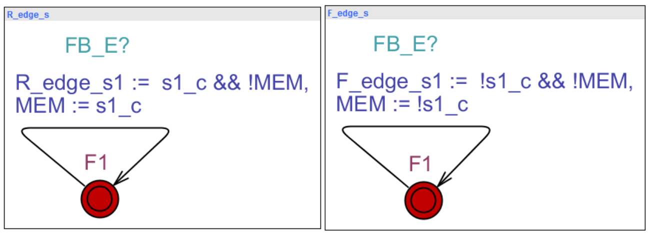

Fig. 1. Rising edge and falling edge models, of the sensor s1, developed in TA, to be formally verified with UPPAAL.

These modules (one for each edge) correspond to the behaviors presented in figures 3 and 4, respectively [25].The values that are assigned to the variables are directly obtained from what is described in those figures, but another variable (synchronization message “FB_E”) is considered in the model.

In fact, the synchronization, that is possible to see in the model of figure 5, is necessary due to the synchronization of the evolution of all models considered in the global model.

Figure 6 illustrates the existing relation between some modules considered for the global model of the system.

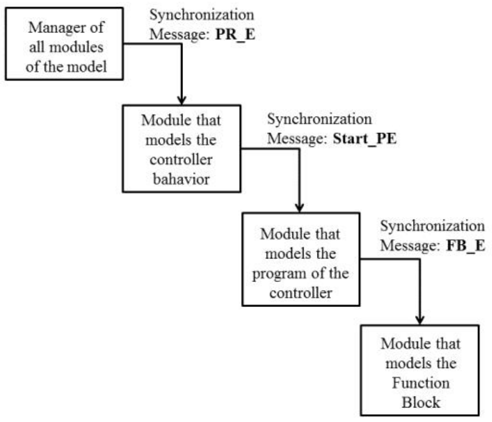
Manager of all modules of the model
Synchronization Message: PR_E
Module that models the controller bahavior
Synchronization Message: Start_PE
Module that models the program of the controller
Synchronization Message: FB_E
Module that models the Function Block

Fig. 2.Schematic synchronization between modules of the global TA model, used in UPPAAL, for formal verification purposes.

In fact, this relation between the modules makes possible that the values of variables are obtained in the equivalent moments that they correspond to the dynamics of the program execution in a PLC.

Figure 7 illustrates how it has been developed with parts of each module considered in figure 6.

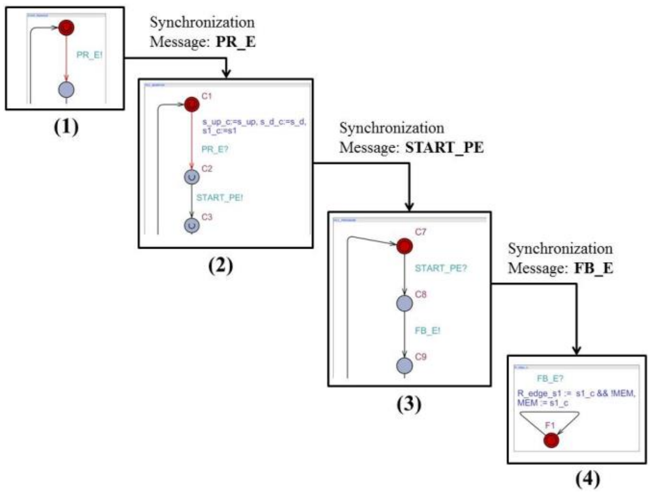

Fig. 3. Illustration of synchronization between modules of the global TA model, used in UPPAAL, for formal verification purposes.

Let's explain how the model has been developed in order to accomplish the desired behavior for the Function blocks considered (the rising and falling edges).

At beginning, when the model starts its evolution, the initial location of the module 1 (manager of all modules, figure 7) starts evolution and sends a synchronization message to module 2.

Module 2 models the behavior of the controller (figure 7) and the received message from manager module allows staring the respective evolution. It has been considered a monotask and sequential controller with, at least, three steps in the scan cycle: inputs reading, program execution and outputs updating.

After the step inputs reading, on the model 2, be performed this module sends a message (START_PE) to model 3 that will be responsible for the starting of the program evolution model.

The beginning of the evolution of the module corresponding to the program of the PLC has several steps, but the first one considered is the step concerning the calculation of the values corresponding to the modules of the rising and falling edges (module 4, figure 7). This evolution will occur in this precise moment and never more during the model evolution, unless that a new cycle of the PLC happens again.

When the evolution of the program ends, this is sent a message to the module corresponding to the PLC behavior, in order to be updated the outputs. After this, the evolution of the model is done by allowing evolution of the modules corresponding to the physical plant models.

IV. Conclusion

When developing a controller specification, the changing of logical value of discrete behavior variables is one of most common needs of modeling, namely the rising edge and faLling edge of a Boolean variable. The implementation of this behavior, in industrial controllers, more precisely in programmable logic controllers is by using IEC 61131-3 function blocks.

This means that the simulation and formal verification of the specification of the described behaviors is one of the most important tasks, in order to obtain safe and reliable controllers' software to be implemented in physical controlling devices, such as programmable logic controllers or others, commonly used in industry.

With this global modeling approach, it is possible to consider the behavior of controllers' variables in a very realistic way, obtaining a global model to be simulated and verified. This global model considers, also, the behavior of the plant, allowing to prove more behavior properties of the system. The use of UPPAAL an timed automata formalism, making possible to take the modeling of time into account, is crucial when models of the plant are considered because physical components behave in a non-deterministic way and always it is needed to consider their evolution in time.

Future works in this domain will consider controlled distributed systems and details on modelling those systems, mainly because of more or less complexity of the respective controllers.

Acknowledgment

The authors are grateful to MEtRICs Research Center Support for realization of this project.

References

[1] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving," Form. Methods Syst. Des., vol. 19, no. 1, pp. 7–34, 2001.

[2] J. M. R. Galvão, "Conversão sistemática do comportamento definido nos blocos funcionais da norma IEC 61 131-3 para autómatos finitos temporizados," University of Minho, 2015.

[3] J. J. T. Kleijn, M. A. Reniers, and J. E. Rooda, "Analysis of an industrial system," Form. Methods Syst. Des., vol. 22, no. 3, pp. 249–282, 2003.

[4] Y. Zhang, Y. Dong, H. Hong, and F. Zhang, "Code Formal Verification of Operation System," Int. J. …, vol. 2, no. December, pp. 10–18, 2010.

[5] J. Campos and J. Machado, "A Specification Patterns System for Discrete Event Systems Analysis," Int. J. Adv. Robot. Syst., vol. 10, p. 1, 2013.

[6] International Electrotechnical Commission, "IEC International Standard IEC 61131-3," Program. Control., vol. Part 3, 2003.

[7] S. C. Park, M. Ko, and M. Chang, "A reverse engineering approach to generate a virtual plant model for PLC simulation," Int. J. Adv. Manuf. Technol., vol. 69, no. 9–12, pp. 2459–2469, 2013.

[8] L. Baresi, S. Carmeli, A. Monti, and M. Pezzè, "PLC programming languages: A formal approach," Proc. Autom., vol. 98, 1998.

[9] C. M. Park, S. M. Bajimaya, S. C. Park, G. N. Wang, J. G. Kwak, K. H. Han, and M. Chang, "Development of virtual simulator for visual validation of PLC program," in Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet

Commerce, International Conference on, 2006, p. 32.

[10] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, “Finite state automata and simple recurrent networks,” Neural Comput., vol. 1, no. 3, pp. 372–381, 1989.

[11] M.-S. Ko, D. Chang, G.-N. Wang, and S. C. Park, “The Template Model Approach For PLC Simulation In An Automotive Industry.,” in ECMS, 2012, pp. 306–312.

[12] B. P. Zeigler, “DEVS representation of dynamical systems: Event-based intelligent control,” Proc. IEEE, vol. 77, no. 1, pp. 72–80, 1989.

[13] I. Moon, G. J. Powers, J. R. Burch, and E. M. Clarke, “Automatic verification of sequential control systems using temporal logic,” AIChE J., vol. 38, no. 1, pp. 67–75, 1992.

[14] H. Guéguen and J. Zaytoon, “On the formal verification of hybrid systems,” Control Eng. Pract., vol. 12, no. 10, pp. 1253–1267, 2004.

[15] J. M. Machado, “Influence de la prise en compte d’un modèle de processus en vérification formelle des Systèmes à Evénements Discrets,” Universidade do Minho, 2006.

[16] N. Sharygina, J. Browne, F. Xie, R. Kurshan, and V. Levin, “Lessons learned from model checking a NASA robot controller,” Form. Methods Syst. Des., vol. 25, no. 2–3, pp. 241–270, 2004.

[17] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen, “A compositional modelling and analysis framework for stochastic hybrid systems,” Form. Methods Syst. Des., vol. 43, no. 2, pp. 191–232, 2013.

[18] R. Passerone, J. R. Burch, and A. L. Sangiovanni-Vincentelli, “Refinement preserving approximations for the design and verification of heterogeneous systems,” Form. Methods Syst. Des., vol. 31, no. 1, pp. 1–33, 2007.

[19] S. Nadjm-Tehrani and J.-E. Strömberg, “Formal verification of dynamic properties in an aerospace application,” Form. Methods Syst. Des., vol. 14, no. 2, pp. 135–169, 1999.

[20] B. F. Adiego, D. Darvas, E. B. Vinuela, J.-C. Tournier, V. M. G. Suárez, and J. O. Blech, “Modelling and Formal Verification of Timing Aspects in Large PLC Programs,” in Proc. of IFAC World Congress, 2014.

[21] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, “NuSMV: A new symbolic model verifier,” in Computer Aided Verification, 1999, pp. 495–499.

[22] T. Hafer and W. Thomas, “Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree,” in Automata, Languages and Programming, Springer, 1987, pp. 269–279.

[23] P. Wolper, “Temporal logic can be more expressive,” Inf. Control, vol. 56, no. 1, pp. 72–99, 1983.

[24] R. Wang, Y. Guan, L. Liming, X. Li, and J. Zhang, “Component-based formal modeling of PLC systems,” J. Appl. Math., vol. 2013, 2013.

[25] A. Basu, M. Bozga, and J. Sifakis, “Modeling heterogeneous real-time components in BIP,” in Software Engineering and Formal Methods, 2006. SEFM 2006. Fourth IEEE International Conference on, 2006, pp. 3–12.

[26] S. Bensalem, M. Bozga, T.-H. Nguyen, and J. Sifakis, “D-finder: A tool for compositional deadlock detection and verification,” in Computer Aided Verification, 2009, pp. 614–619.

[27] M. Zhou, H. Wan, R. Wang, X. Song, C. Su, M. Gu, and J. Sun, “Formal component-based modeling and synthesis for PLC systems,” Comput. Ind., vol. 64, no. 8, pp. 1022–1034, 2013.

[28] M. Perin and J.-M. Faure, “Building meaningful timed models of closed-loop DES for verification purposes,” Control Eng. Pract., vol. 21, no. 11, pp. 1620–1639, 2013.

[29] R. ; Alur and D. Dill, “Automata for modeling real-time systems,” Proc. seventeenth Int. Colloq. Autom. Lang. Program., pp. 322–335, 1990.

[30] R. Alur, C. Courcoubetis, and D. Dill, “Model-checking in dense real-time,” Inf. Comput., vol. 104, no. 1, pp. 2–34, 1993.

[31] K. G. Larsen, P. Pettersson, and W. Yi, “Uppaal in a Nutshell,” Int. J. Softw. Tools Technol. Transf., vol. 1, no. 1, pp. 134–152, 1997.

[32] E. P. Enoiu, D. Sundmark, and P. Pettersson, “Model-based test suite generation for function block diagrams using the uppaal model checker,” in Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, 2013, pp. 158–167.

[33] M. Uzam, “A general technique for the PLC-Based implementation of RW supervisors with time delay functions,” Int. J. Adv. Manuf. Technol., vol. 62, no. 5–8, pp. 687–704, 2012