23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

FUNCTION BLOCKS AND FORMAL VERIFICATION: SYSTEMATIC MODELLING APPROACH

José Machado
Joel Galvão
Alexandre Fernandes
MEtRICs Research Center, Mechanical Engineering Department, University of Minho,
Campus of Azurém, 4800-058 Guimarães, Portugal
jmachado@dem.uminho.pt, a63363@alunos.uminho.pt, alexandre.fernandes93@gmail.com

Abstract. *Formal Verification of automation systems controllers' software is a complex task. This is, mainly, due to the fact that are necessary high skilled designers for modelling those automations systems, because the results obtained from formal verification task are highly dependent of the "quality" of the developed models for that purpose. In this paper, is explained and presented an approach for modelling some Function Blocks from IEC 61 131-3 standard and, in the same context, is proposed an approach for developing the respective Timed Automata model for formal verification purposes, by model-checking, using UPPAAL model-checker. Some interesting results are achieved and the proposed approach is simple to use and to obtain, following some systematic rules.*

Keywords: *Software Engineering, Formal verification, Automation, Software tools, Real-Time Systems*

1. INTRODUCTION

Formal Verification is an analysis technique widely used for increasing dependability of software for industrial controllers (Moon, et al., 1992). Several works have been developed in this domain (Boel and Stremersch, 2000) (Frey and Litz, 2000) (Valeriy Vyatkin and Hanisch, 2003) (Guéguen and Zaytoon, 2004) . Also several approaches have been considered on the application of this technique: using Theorem-Proving (Roussel and Denis, 2002) and Model-Checking (Rossi, 2004) approaches; considering (Machado, et al., 2006), or not (Rossi, 2004), a plant model for the behavior of the analysed systems. Some works consider the parameter time - such (Holz and Melnik, 2005), where the translation of the SFC (Sequential function Charts) to timed automata (Alur and Dill, 1990), from IEC 61131-3 standard, is performed – and the approach proposed by (Gaid, Bérard, and de Smet, 2005) where is proposed an approach for obtaining a controller model based on several hypothesis of temporal behaviour of this controller (Mader and Wupper, 1999).

Till now, there is a gap when it is intended to use formal verification in the context of PLC (Programmable Logic Controllers) programs, mainly concerning the function blocks of IC 61131-3 standard. Some works, like, try decrease this gap, proposing global solutions for function blocks formal verification, but is still needed to detail some specific aspects concerning this subject (Silva, et al., 2007) (Pavlovic and Ehrich, 2010) (Mazzolini, Brusaferri, and Carpanzano, 2010) (Patil, Bhadra, and Vyatkin, 2011) (Soliman, Thramboulidis, and Frey, 2012) (Perin and Faure, 2013)

In (Silva,et al., 2007) proposes a technique to create the timed automata model of the logic behaviour of the PLC program. This makes possible the FBD code to test against the specification behaviour without having any PLC around. There technique uses UPPAAL TRON (Larsen, et al., 2005) to test if the XLM file generated from the logic behaviour matches the XLM created from the FBD code.

Nevertheless (Pavlovic and Ehrich, 2010) is suggested a methodology for formal verifying PLC code written FBD (Function Block Diagrams), which is one of the IEC 611131-3 programming languages, using NuSMV model-checker (Cimatti, et al., 1999). This technique considers the program code in FBD and respective translation to IL (Instruction List). This last version of code (in IL programming language) is transformed into the NuSMV input language for being formally verified, using this model-checker.

On the other hand (Mazzolini, Brusaferri, and Carpanzano, 2010) proposes other method based on Model-Checking techniques and tools focused on the Verification of logic code in mechatronic systems. They applied graphical Stateflow Charts based model deployment, Bounded Model Checking techniques and Model Coverage properties have been utilized.

In reference (Patil, Bhadra, and Vyatkin, 2011) is proposed a technique for reducing the complexity of the models by infusing non determinism into certain parts of the plant model during formal verification process, to do that its modulated the close loop system in Net Condition/Event Systems (NCES) (Rausch and Hanisch, 1995) and then is verified by ViVe and SESA model-checkers (V. Vyatkin, Starke, and Hanisch, 2007) .

Reference (Soliman, Thramboulidis, and Frey, 2012) proposes a technique for transforming FBD to timed automata model. It is presented, also, a set of transformation rules to create automatically the model of the FBD safety database in models prepared for performing the respective formal verification tasks.

The authors of this paper (Perin and Faure, 2013), propose a methodology for doing formal verification based in the concept of urgent edges to prevent meaningless evolutions and states. In this work is considered models for the plant. They show that the reduction of meaningless states can be done by introducing variables that represent the input of the logic controller and the stability conditions.

In this work, it is intended to propose a methodology that starts on the specification of the Automation system controller behaviour till the formal verification of the specification, when rising and falling edges are considered on the description of the behaviour proposed for the controller of the system. After formal verification of the specification, it is assumed that – if systematic rules are adopted for translation of this specification to the PLC program - the program will behave accordingly with the designer's expectations. For this approach, it is considered the specification developed in SFC (Sequential Function Chart) formalism (EN, 2002) and the Timed Automata (Alur and Dill, 1990) as input formalism of the used model-checker UPPAAL (Bernardo and Corradini, 2004). Also, the methodology for creating the global model of the system in Timed Automata, for 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 be formally verified, using the UPPAAL software; and, finally, there are presented some conclusions and future work, in section 4.

2. Case study

2.1 Specification of the controller

The automatic system used as case study, in this work, is a car barrier to be used in parking lot, schematically represented in figure 1.

This automatic system is actuated by one motor with two directions of movement: one that controls the movement with direction "up" (M_UP) and another that controls the movement with direction "down" (M_D).

Besides that, the system has a set of sensors: one to detect the presence of car (s1) and other two sensors for detecting the barrier position, "s_up" on the up position and "s_d" on the down position.

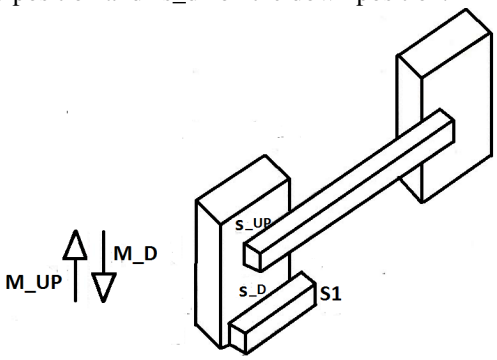

Figure 1.Schematic representation of the car barrier, with respective sensors and actuators.

The input and output signals of the controller are presented in table 1.

Table 1.Inputs and outputs of the controller

Inputs	Outputs	Physical meaning
s1		Sensor that detects presence of car
s_up		Sensor that detects barrier in position "up"
s_down		Sensor that detects barrier in position "down"
	M_UP	Order for moving barrier down
	M_D	Order for moving barrier down

The intended behavior for the system is very simple: 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 figure 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 (IEC, 2003)

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.

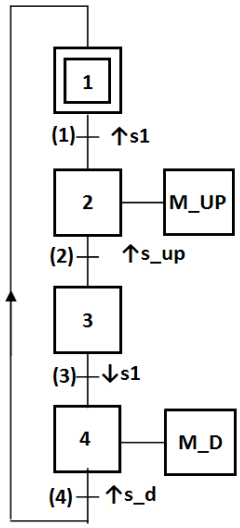

Figure 2. Sequential Function Charts of the comportment described before.

2.2 Translation of the controller specification to Ladder and Function blocks

The translation of the presented specification, to PLC programming language defined in (IEC, 2003), considers two distinct parts: one concerns the translation of the dynamics of the model according methodology proposed in (Machado, et al., 2006).

Concerning the behavior of the rising and falling edges there are considered the respective comportment defined on the standard (International Electrotechnical Commission, 2003). 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 figure 3.

```
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
```

Figure 3. Rising edge behavior (IEC, 2003).

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 (IEC, 2003).

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 figure 4.

```
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
```

Figure 4. Falling edge behaviour (IEC, 2003).

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 figure 2, there are considered both rising and falling edges. This way, those Boolean values will be calculated as demonstrated above.

3. Formal verification of the specification

In order to perform the formal verification it was followed the approach proposed in for the translation of SFC to Timed Automata (TA) (Uzam, 2012), for obtaining the TA model.

Also, it has been considered the modeling of the comportment of the controller (in this case, it has been considered a PLC). For this, a modular method has been followed for obtaining the global model to be formally verified in UPPAAL. It has been considered a module for the PLC program, a module for the PLC behavior, some modules for modeling the physical plant and one model for each rising or falling edge considered in the system.

It must be highlighted that the main problem for performing 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.

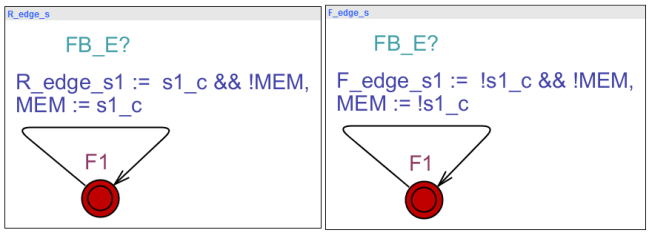

Figure 5. Rising edge and falling edge models, of the sensor s1, developed in TA, to be formally verified with UPPAAL.

These modules (one for each edge) correspond to the behaviors presented in figures 3 and 4, respectively (IEC, 2003).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.

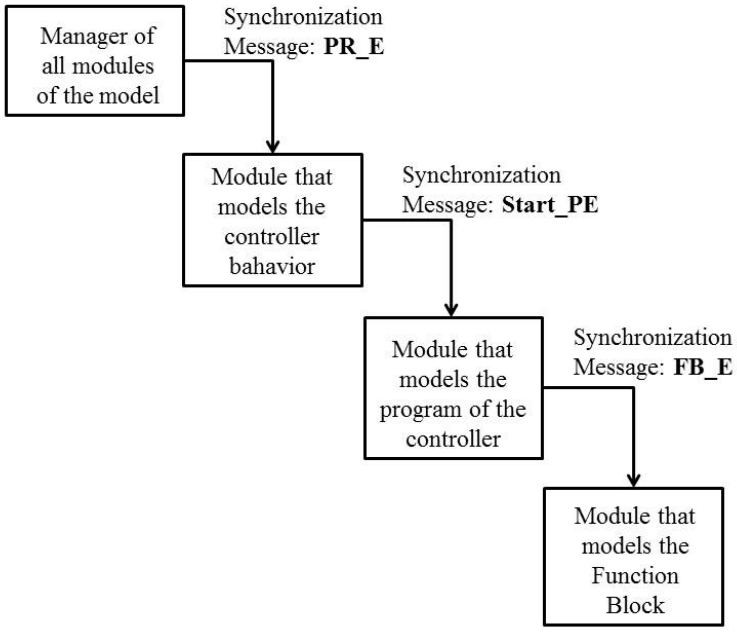

Figure 6. Schematic synchronization between modules of the global TA model, used in UPPAAL, for formal verification purposes.

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.

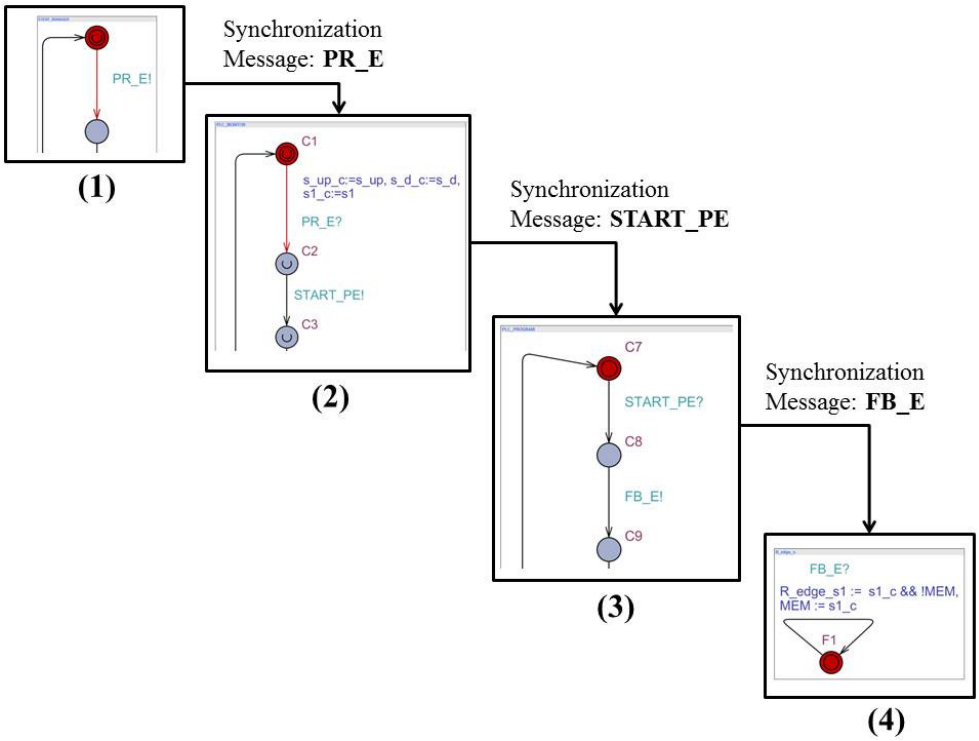

Figure 7. Illustration of synchronization between modules of the global TA model, used in UPPAAL, for formal verification purposes.

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.

4. CONCLUSION

This work presents a correct modeling approach for modeling Function Blocks of the IEC61131-3 standard. In this context has been illustrated, in detail, a simple way for developing a timed automata model, of an automation system, for being formally verified by model-checking, using model-checker UPPAAL.

The models of the rising and falling edges developed made possible to test a great new deal and as expected the models behavior is very similar to what happens in real mechatronic systems. For example if one sensor made the zero to one changeover two times in the same scan cycle the model only record one as takes place in real equipment. Consequently the results of the formal verification are considerably stronger.

This modeling approach has several advantages, mainly: it is a modular approach, so obtaining the global model for all the system (controller and plant) is easier; the synchronization of the models by message variables is relatively easy to do and allows the evolution of all modules models in a realistic way; the use of a module as manager of all modules evolution allows a correct evolution of all modules; the changing of one module does not changes the other ones (for instance instead of considering a monotask an sequential controller, like that one that has been considered) it is possible consider other controllers with different configuration and only module 2, figure 7, must be changed; it is possible to create a library of modules and reuse them when necessary (and to see different formal verification results with different configurations for the physical part of the system or different configurations of controllers); and it can be optimized in terms of the number of used variables (because of synchronizations), in order to increase the verification capacity of UPPAAL.

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.

5. ACKNOWLEDGEMENTS

The authors are grateful to the support of MEtRICs research center for the support in this project.

6. REFERENCES

Alur, R. and Dill, D., 1990. "Automata for modeling real-time systems". In *Proceedings of the seventeenth international colloquium on Automata languages and programming*. New York , USA.

Bernardo, M. and Corradini, F., 2004. Formal Methods for the Design of Real-Time Systems. Springer, Berlin, 1st edition.

Boel, R. and Stremersch, G., 2000. Discrete event systems: analysis and control. Springer Science and Business Media, Berlin .1st edition.

Cimatti, A. Clarke, E. Giunchiglia, F. and Roveri, M., 1999. NuSMV: A new symbolic model verifier. In *Computer Aided Verification*. Trento, Italy.

EN 2002., 2002. "European Standard 60848: SFC specification language for sequencial function Charts". *European Standard*.

Frey, G. and Litz, L., 2000. "Formal methods in PLC programming". In *Systems, Man, and Cybernetics, 2000 IEEE International Conference*. Nashville, USA.

Gaid, M. Bérard, B.and Smet, O., 2005. "Verification of an evaporator system with UPPAAL". *Journal Européen Des Systèmes Automatisés*, Vol 39, p. 1079–1098.

Guéguen, H. and Zaytoon, J., 2004. "On the formal verification of hybrid systems". *Control Engineering Practice*, Vol. 12, p. 1253–1267.

Holz, H. Melnik, G., 2005. Integration of Software Specification Techniques for Applications in Engineering. Springer, Berlin, 1st edition.

International Electrotechnical Commission., 2003. "IEC International Standard IEC 61131-3:Programmable Controllers, Part 3: Programing languages -

Larsen, K. G. Mikucionis, M. Nielsen, B. and Skou, A.,2005. "Testing Real-time Embedded Software Using UPPAAL-TRON: An Industrial Case Study". In *Proceedings of the 5th ACM International Conference on Embedded Software*. New York, USA.

Machado, J. Denis, B. Lesage, J. Faure, J.M. Fereira, J. and others., 2006. "Logic controllers dependability verification using a plant model". In *Proceedings of the 3rd IFAC Workshop on Discrete-Event System Design.* Rydzyna, Poland.

Mader, A. and Wupper, H., 1999. "Timed automaton models for simple programmable logic controllers". In *Real-Time Systems- 11th Euromicro Conference*. York , England.

Mazzolini, M. Brusaferri, A. and Carpanzano, E., 2010. Model-checking based verification approach for advanced industrial automation solutions. In *Emerging Technologies and Factory Automation-ETFA*. Bilbao, Spain

Moon, I. Powers, G. J. Burch, J. R. and Clarke, E. M., 1992. "Automatic verification of sequential control systems using temporal logic". *American Institute of Chemical Engineers-AIChE Journal*, Vol.38, p. 67–75.

Patil, S. Bhadra, S. and Vyatkin, V., 2011. "Closed-loop formal verification framework with non-determinism, configurable by meta-modelling". In *37th Annual Conference on IEEE Industrial Electronics Society - IECON*. Melbourne, Australia.

Pavlovic, O. and Ehrich, H.D., 2010 . "Model Checking PLC Software Written in Function Block Diagram". In *Third International Conference on Software Testing, Verification and Validation*, Paris, France.

Perin, M. and Faure, J.M., 2013. "Building meaningful timed models of closed-loop DES for verification purposes". In *Control Engineering Practice*, Vol. 21, p. 1620–1639.

Rausch, M. and Hanisch, H.M., 1995. "Net condition/event systems with multiple condition outputs" In *Emerging Technologies and Factory Automation- ETFA'95*. Paris, France

Rossi, O., 2004. *Validation formelle de programmes ladder pour automates programmables industriels*. PhD thesis, École Normale Supérieure de Cachan, France.

Roussel, J.M. and Denis, B. 2002. "Safety properties verification of ladder diagram programs".In *Journal Européen Des Systèmes Automatisés*, Vol.*36*, p. 905–917.

Silva, L. Barbosa, L. Gorgônio, K. Perkusich, A. and Lima, A., 2007."On the automatic generation of timed automata models from Function Block Diagrams for Safety Instrumented Systems". In *12th IEEE Conference on Emerging Technologies and Factory Automation - ETFA*. Patras, Greece.

Soliman, D. Thramboulidis, K. and Frey, G., 2012. "Transformation of Function Block Diagrams to UPPAAL timed automata for the verification of safety applications". in *Annual Reviews in Control*, Vol. 36, p. 338–345.

Uzam, M., 2012. "A general technique for the PLC-Based implementation of RW supervisors with time delay functions". *The International Journal of Advanced Manufacturing Technology*, Vol. 62, p. 687–704.

Vyatkin, V. and Hanisch, H.M., 2003. "Verification of distributed control systems in intelligent manufacturing". *Journal of Intelligent Manufacturing*, Vol.14, p. 123–136.

Vyatkin, V. Starke, P. and Hanisch, H.M., 2007. "ViVe and SESA Model Checkers". March 2015 < http://homepages.engineering.auckland.ac.nz/~vyatkin/tools/modelchekers.html >.

7. RESPONSIBILITY NOTICE

The author(s) is (are) the only responsible for the printed material included in this paper.