Formal Verification of Industrial Controllers: with or without a Plant model?

José J.B. Machado, Bruno Denis, Jean-Jacques Lesage

CONTROLO 2006
7th Portuguese Conference on Automatic Control
Instituto Superior Técnico, Lisboa, Portugal
September 11-13, 2006

CDROM paper n°80

FORMAL VERIFICATION OF INDUSTRIAL CONTROLLERS: WITH OR WITHOUT A PLANT MODEL?

José Machado[*], Bruno Denis[**], Jean-Jacques Lesage[**]

[*]Mechanical Engineering Department, University of Minho, Campus of Azurém, 4800-058 Guimarães, PORTUGAL; [**] University Research Laboratory in Automated Production, École Normale Supérieure de Cachan, 61 av. du Président Wilson, 94235 Cachan, FRANCE.

The use of a plant model on formal verification of industrial controllers makes the formal verification tasks more realistic, because any industrial system is always composed by a controller and a plant. Therefore, if the plant model is not used, there is a part of the system that is not considered. However, if there are some cases where the use of a plant model becomes the formal verification results more realistic and robust there are another cases where it nor always happens. In this paper there are indicated which are the circumstances where it is useful to use, or not, a plant model on formal verification tasks, using model-checking techniques.

Safety Analysis, Discrete Event Systems, Dynamic Behaviour, Industrial Production Systems.

1. INTRODUCTION

The work presented herein lies within the framework of a cooperative research program between the Mechanical Engineering Department of the University of Minho in Portugal (DEM) and the LURPA (University Research Laboratory in Automated Production) of ENS de Cachan (École Normale Supérieure) in France. This joint program focuses on the topic of "Dependable Control of Manufacturing Systems", more precisely, on the formal verification of Industrial Controllers.

Formal verification techniques stem from the field of computer science. Only recently they have been adapted and applied to Discrete Event Systems (DES) verification (Moon I., 1994). The most common techniques used on this field are the theorem proving (Roussel and Denis, 2002) and the model-checking (Bérard, *et al.*, 1999). The general principle behind model-checking may be expressed as follows (see Fig. 1). Let's start with a system that has been designed to verify an entire array of properties (logical correctness, dependability, liveness, etc.). The first task consists of formalizing system behaviour in the form of a finite state automaton: S, plus the properties to be verified within a temporal algebra such as CTL (Emerson and Halpern, 1986): φ.

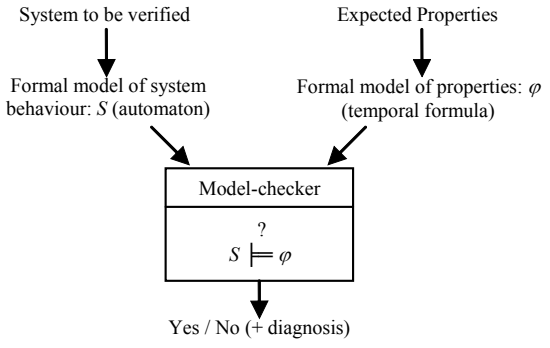

Fig. 1. Model-checking scheme.

The model-checker then conducts a thorough analysis of the state space reachable by *S*, which serves either to prove that $S \models \varphi$ (this algebraic statement denotes that "the system model satisfies the set of properties φ") or, when such is not the case, to propose a counterexample that revokes those properties not verified by S.

A DES may be represented in a generic manner, as shown in Fig. 2: a discrete controller acting in a closed loop on a plant.

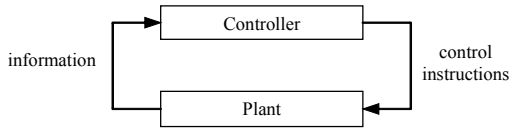

Fig. 2. A generic closed-loop DES.

As part of a dependable controller design approach, the system being targeted for verification can thus be (according to Frey. and Litz, 2000) either the controller on its own, presumed to be operating within an open loop on the plant (a non "model-based" verification), or the {controller + plant} assembly set interacting within a closed loop ("model-based" verification).

Fig. 3, extracted from (Merkte and Menzel, 2000), provides an accurate description of the DES formal verification process using model-based model-checking selected by most authors. The model to be verified results from the composition of three automata (cf. Zone A in Fig. 3): the user program model, the control unit execution model, and the plant model (also named “environmental model” in Fig. 3). The plant model is often derived from a library of generic automata instanced and composed.

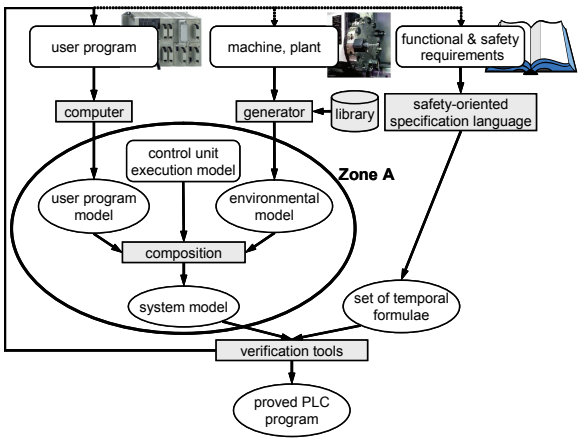

Fig. 3. Formal verification of PLC programs using model-based model-checking (Merkte and Menzel, 2000).

The work presented herein is intended to highlight the advantages and drawbacks of taking into account a plant model for DES model-checking.

The paper is organized as follows. In section 1, it is presented the challenge proposed to this work. Section 2 is devoted to the general presentation of a case study involving a "pick-and-place" workstation. Next, it is provided (section 3) a set of system behaviour properties to prove. Section 4 discusses the selected automaton and the models used on formal verification tasks. In section 5, the system behaviour properties are formalised. On section 6 are presented the formal verification experiments and discussion of the obtained results. Finally, in section 7, are present some conclusions and future work.

2. CASE STUDY

The chosen system for this case study lies in the well-known category of "pick-and-place" systems (Fig. 4); its function is to take parts, fed by gravity into three feed chutes, for placement in a single unloading chute. Sensors pp1, pp2 and pp3 indicate the presence of a part in one of the feed chutes, while sensor pp0 signals the presence of a part in the unloading chute. The device that enables picking and placing a part is composed of a group of three pneumatic cylinders plus a vacuum suction cup system. The vertical cylinder (VC) places the suction cup in contact with a part. Longitudinal cylinders L1C and L2C are arranged in series to allow positioning the vertical cylinder VC in front of the four chutes (L2C stroke is twice as long as than L1C stroke). The four reached positions are thereby detected by position sensors s0, s1, s2 and s3. The depression in the suction cup is obtained by virtue of a venturi and detected by a vacuum sensor.

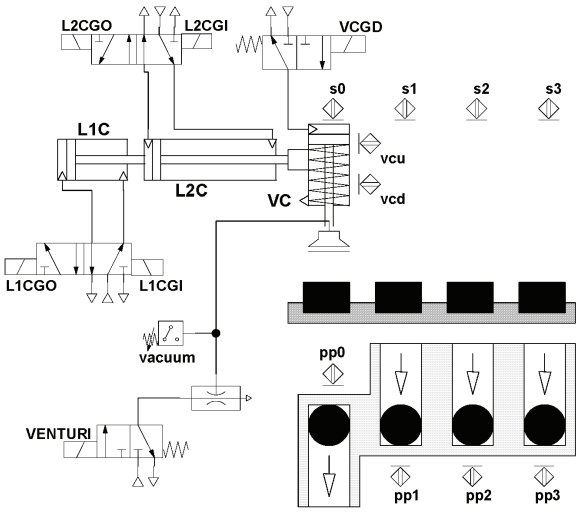

Fig. 4. Schematic view of the studied plant.

3. SYSTEM BEHAVIOUR PROPERTIES

Formal verification calls for two categories of properties to prove: safety properties and liveness properties; (Alpern and Schneider, 1985), (Manna and Pnueli, 1995).

In other hand, each property is expressed using a subset of three kinds of variables: the control variables (outputs and states of the controller model), observation variables (inputs of the controller model) and/or plant variables (state variables of the plant model which are not directly observable nor controllable).

The properties that it is intended to verify are:

- PR_1i: The controller never commands a horizontal cylinder i in two directions at the same time;
- PR_2: If the controller commands the vertical cylinder to go down, then it must not command any movement to the horizontal cylinders;
- PR_3: The controller commands horizontal cylinders only while sensor "vcu" is "on";
- PR_4: The horizontal cylinders move only while the sensor "vcu" is "on";
- PR_5i: When a part is detected by the sensor "ppi", then in the future, the corresponding horizontal cylinder(s) will deploying;
- PR_6i: When a part is detected by the sensor "ppi", then in the future, it will be picked;
- PR_7: While the vertical cylinder is moving down, all the other cylinders stay in deployed or retracted position.

Table 1 presents the types of variables chosen by the engineer to translate each functional property into formal expression. It also points out the diversity of the properties we consider in this case study. SF stands for safety property; LV stands for liveness property and C, O and P refers to properties expressed with Control variables, Observation variables and Plant variables, respectively.

Table 1 Characteristics of the properties to prove

Property	Type of Property		Used Variables		
	SF	LV	C	O	P
PR_1i	X		X		
PR_2	X		X		
PR_3	X		X	X	
PR_4	X			X	X
PR_5i		X		X	X
PR_6i		X		X	
PR_7	X				X

4. SELECTED AUTOMATON AND MODELS FOR VERIFICATION

4.1 Selected automaton class

The selected automaton class stems from the one described in (Bengtsson and Yi, 2004) for the UPPAAL model-checker and have elected to retain the same notations.

Automaton $\mathcal{A}$ is thus a triplet $\langle N, n_0, E\rangle$, where:

- N is a finite set of states;
- $n_0 \in N$ is the initial state;
- $E \subseteq N \times \tau \times \Sigma \times N$ is the set of transitions, with τ being the set of Boolean expressions defined on the set of logic variables $\mathcal{V}$, and Σ a partition of the set of assignments on $\mathcal{V}$.

An automaton network $\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}$ is a set of automata whose Boolean expressions τ, associated with the transitions, are defined on the same set of logic variables $\mathcal{V}$. Within an automaton network, the evolutions are asynchronous and based on variable sharing. This implies that at each point in time, a single transition of a single automaton may be fired. Throughout the remainder of this paper, the syntax of Boolean expressions τ will replicate the syntax from C programming language ("&&" for the AND operator, "||" for the OR operator and "!" for the NOT operator). In the discussion of the ensuing example, clocks and hand-shake synchronizations will not be employed.

4.2 Models to be verified

For plant modelling we retain a modular-based technique. The behaviour of each plant component (sensor, actuator and pre-actuator …) is translated into an automaton. All these automata are coordinated using a "Plant Sequencer" automaton ($\mathcal{A}_{PS}$) to guarantee the correct interaction between the plant components. The set of all these component automata plus $\mathcal{A}_{PS}$ consists of the plant automaton network.

In a same way the controller behaviour model is the automaton network consisting of the "Control Program" automaton ($\mathcal{A}_{CP}$) and the "Control Sequencer" automaton ($\mathcal{A}_{CP}$) that translate the control unit execution algorithm.

The whole automaton network of the case study is:

$$\{\mathcal{A}_{GS},\ \underbrace{\mathcal{A}_{CS}, \mathcal{A}_{CP}}_{\text{Controller model}},\ \underbrace{\mathcal{A}_{PS}, \mathcal{A}_{P1},\ldots, \mathcal{A}_{P12}, \mathcal{A}_{T1},\ldots, \mathcal{A}_{T13}}_{\text{Plant model}}\}$$

$\mathcal{A}_{GS}$: General sequencer

All these automata are given in Fig. 5. Model-based verification will consider this whole network while non model-based verification will only consider the sub network $\{\mathcal{A}_{GS}, \mathcal{A}_{CS}, \mathcal{A}_{CP}\}$. Reader could get detailed information about the automaton modular construction technique in (Machado *et al.*, 2006).

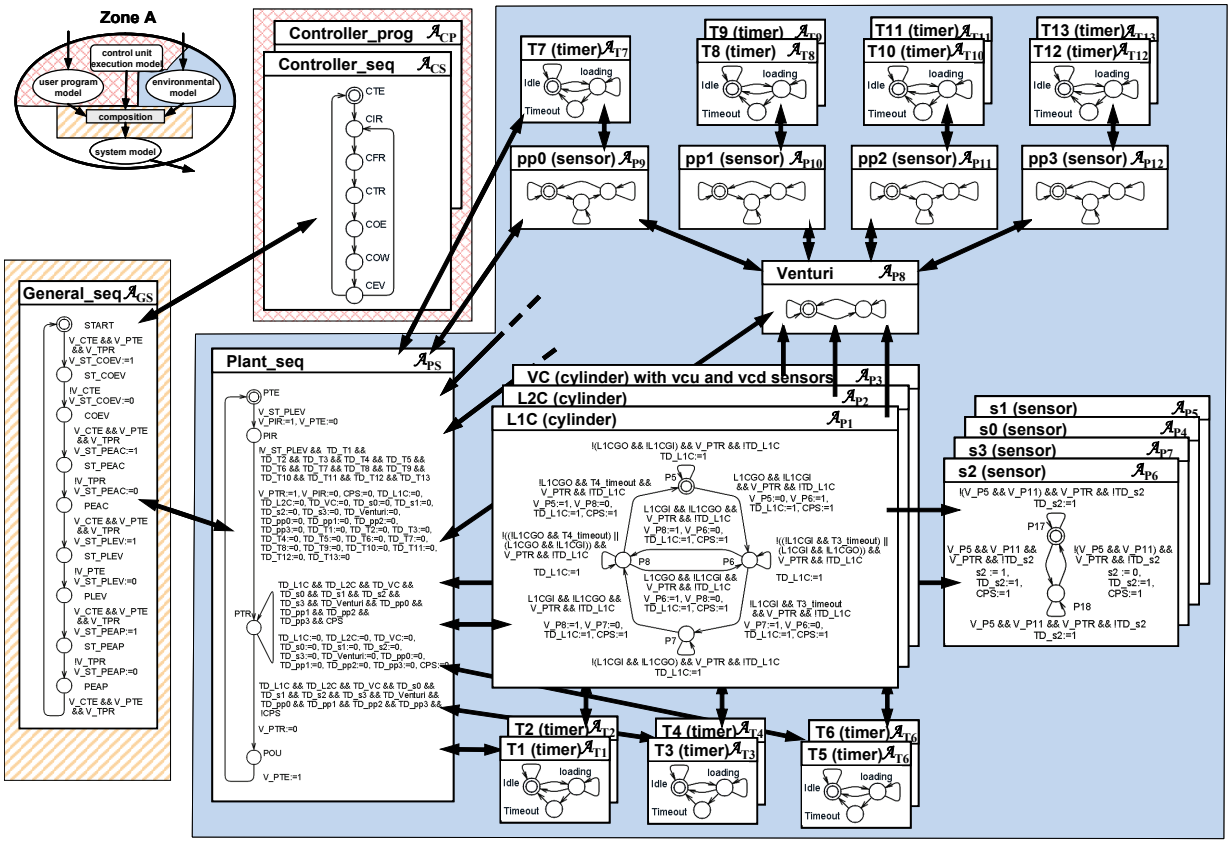

Fig. 5. The automata model of the case study.

5. PROPERTIES FORMALIZATION

A property may be formalized using temporal logic expressions or using, jointly, a temporal logic expression and a observer automata (Cheung and Kramer, 1999). To formalize the chosen properties we have decided to use the Computation Tree Logic (CTL), well defined on (Bérard *et al.* 1999) and the selected automata (used on all system modelling) to describe sequential behaviours. Here, the automata are used as observer automata.

Almost all the variables used on properties formalisation (presented on Table 2) can be deduced analysing the Fig. 4. However, we present the meaning of the used variables (variables representing the plant model states) that the reader can't deduce analysing the Fig. 4:

- V_P2 : VC is in the deployment movement;
- V_P5 : L1C is in the retracted position;
- V_P6 : L1C is in the deployment movement;
- V_P7: L1C is in the deployed position;
- V_P8: L1C is in the retraction movement;
- V_P9: L2C is in the retracted position;
- V_P10: L2C is in the deployment movement;
- V_P11: L2C is in the deployed position;
- V_P12: L2C is in the retraction movement.

Table 2 Properties formalisation on CTL

Property	CTL formalisation
PR_1.1	AG !(L1CGO && L1CGI)
PR_1.2	AG !(L2CGO && L2CGI)
PR_2	AG (VCGD ⇒ !(L1CGI || L1CGO || L2CGI || L2CGO))
PR_3	AG ((L1CGI || L1CGO || L2CGI || L2CGO ⇒ vcu)
PR_4	AG ((V_P6 || V_P8 || V_P10 || V_P12) ⇒ vcu)
PR_5.1	AG (pp1 ⇒ EF (V_P6))
PR_5.2	AG (pp2 ⇒ EF (V_P10))
PR_5.3	AG (pp3 ⇒ EF (V_P6 && V_P10))
PR_6.1	AG (pp1 EF (s1 && vcd && vacuum))
PR_6.2	AG (pp2 EF (s2 && vcd && vacuum))
PR_6.3	AG (pp3 EF (s3 && vcd && vacuum))
PR_7	AG (V_P2 ⇒ ((V_P5 && V_P9) || (V_P5 && V_P11) || (V_P7 && V_P9) || (V_P7 && V_P11)))

6. FORMAL VERIFICATION

NuSMV has been chosen as model-checker, due to its performance for timeless verification. The computer which performs the verification hosts a Pentium III processor, with 1 GHz and 1 GB of RAM, and support the 2.1.2 NuSMV version.

Model-checking consists in an analysis of the reachable state space of the verified automaton to search if it satisfies a property (Fig. 3). A major computational difficulty is then to handle the huge number of states to check due to combinatory explosion. Experimental results illustrate this phenomenon. The number of reachable states is equal to:

- $6{,}1 \times 10^6$ for verification of the $\{\mathcal{A}_{GS}, \mathcal{A}_{CS}, \mathcal{A}_{CP}\}$ network (non model-based (NMB) approach);
- $5{,}0 \times 10^8$ for verification of the $\{\mathcal{A}_{GS}, \mathcal{A}_{CS}, \mathcal{A}_{CP}, \mathcal{A}_{PS}, \mathcal{A}_{P1},\ldots, \mathcal{A}_{P12}, \mathcal{A}_{T1},\ldots, \mathcal{A}_{T13}\}$ network (model-based (MB) approach).

Table 3 First results obtained on formal verification: NMB and MB approaches

Property		NMB approach		MB approach	
Name	Type	Result	Time	Result	Time
PR_1.1	SF	true	7 s	true	134 min
PR_1.2	SF	true	7 s	true	134 min
PR_2	**SF**	**false**	**13 s**	**true**	**136 min**
PR_3	SF	false	9 s	true	135 min
PR_4	SF	**	**	true	135 min
PR_7	SF	**	**	true	137 min
PR_5.1	LV	**	**	false	532 min
PR_5.2	LV	**	**	true	237 min
PR_5.3	LV	**	**	false	510 min
PR_6.1	**LV**	**true**	**8 s**	**false**	**486 min**
PR_6.2	**LV**	**true**	**8 s**	**false**	**453 min**
PR_6.3	**LV**	**true**	**8 s**	**false**	**534 min**

** Properties 4, 7 and 5.i can not be verified using a NMB approach because their formalization (cf Table 2) needs the use of Plant variables that are not defined into the verified model.

The combinatory explosion we mention above can made the computation time too long to obtain a result. In this case study, all the verifications have been done in an acceptable duration. Moreover, as expected the NMB approach needs a shorter computational effort than the MB approach, due to the shorter size of the verified model.

Some properties have been checked as false with NMB or with MB approach, but never with both (in bold characters in Table 3). Properties PR_6.i are checked as false using MB. Consequently deadlocks could appear when the controller will operate in closed loop with the plant. The controller model includes probably design errors that need to be corrected by the engineer. For that, the counterexample trace given by the model-checker will be a precious help.

Indeed, after analysis of the controller and the plant models we have detected a little incoherence in the controller model. The elimination of this incoherence gives the new following results:

Table 4 Formal verification results of the corrected controller model

Property		NMB approach		MB approach	
Name	Type	Result	Time	Result	Time
PR_1.1	SF	true	7 s	true	108 min
PR_1.2	SF	true	7 s	true	108 min
PR_2	SF	true	8 s	true	108 min
PR_3	SF	false	9 s	true	108 min
PR_4	SF	**	**	true	109 min
PR_7	SF	**	**	true	109 min
PR_5.1	LV	**	**	true	142 min
PR_5.2	LV	**	**	true	146 min
PR_5.3	LV	**	**	true	160 min
PR_6.1	LV	true	8 s	true	150 min
PR_6.2	LV	true	8 s	true	160 min
PR_6.3	LV	true	8 s	true	156 min

** Properties 4, 7 and 5.i can not be verified using a NMB approach because their formalization (cf Table 2) needs the use of Plant variables that are not defined into the verified model.

Let us examine these new results for the different classes of properties.

6.1 Safety properties

Concerning the safety properties expressed with only control variables (PR_1.1, PR_1.2, PR_2), there are now true with both approaches. However the significance of these results is not the same for the two approaches:

- for NMB verification, model-checker proves that whatever the plant behaviour the controller behaviour is safe ;
- for MB verification, model-checker proves that the controller behaviour is safe if the plant behaves as expected.

Concerning the safety property expressed with both control and observation variables (PR_3), checking results are different in accordance with approaches. The property is false in NMB because values of observation variables are not constrained by a plant behaviour. So, only MB verification is meaningful for this class of properties.

Concerning the safety properties expressed with at least one plant variable (PR_4, PR_7), MB approach results are significant while NMB approach can not conclude.

6.2 Liveness properties

Concerning the liveness properties expressed with at least one plant variable (PR_5.i) conclusions are identical to the PR_4 and PR_7 ones: only MB approach gives significant results.

Concerning the liveness properties expressed with at least one observation variable (PR_7.1, PR_7.2, PR_7.3), there are now true with both approaches. However the significance of these results is not the same for the two approaches:

- for NMB verification, model-checker proves that the controller behavior is deadlock free without constraints coming from the plant behaviour ;
- for MB verification, model-checker proves that the controller behavior is deadlock free even with plant behavior constraints.

6.3 Synthesis and discussion

In a NMB approach, the reachable state space of the controller model is built in the most permissive manner possible, i.e. such that the evolution of its inputs are in no way constrained by plant behavior (cf. Fig. 6a). In this case, the safety properties capable of being demonstrated provide stronger proofs than those demonstrated taking into account a plant model constraint. On the other hand, some safety properties may be declared non-verified, in light of the counterexamples generated with unrealistic evolution in controller inputs that cannot in reality be produced by the plant. In this case, a MB verification imposes realistic constraints to the controller inputs evolution (cf. Fig. 6b). At the opposite, liveness properties capable of being demonstrated via a MB approach provide stronger proofs than those demonstrated without taking into account a plant model. Indeed, if a controller model is deadlock free for a subset of input evolutions then it will stay deadlock free for the entire set of input evolutions.

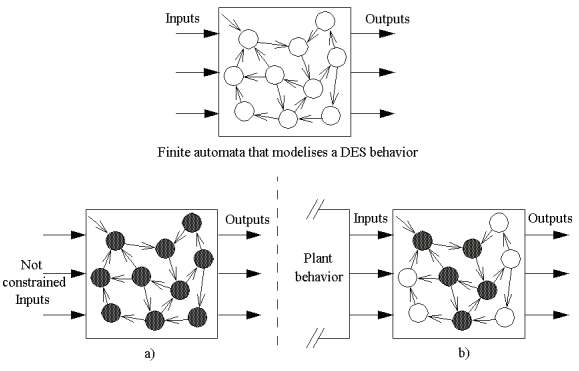

Fig. 6. Controller behavior model and respective reachable states if: (a) it is not considered a plant model and (b) it is considered a plant model.

7. CONCLUSIONS AND FUTURE WORK

In this paper it is shown that the use of a plant model has a great impact on formal verification of discrete event systems. In scientific literature, model-based and non model-based approaches are often compared or brought into conflict. It is demonstrated through a study case that in fact those two approaches complement each other. Furthermore, it is also indicated which approach must be preferred depending on class of properties.

Nevertheless, model-based verification asks the problem of the construction of the plant model. Current and future works aim to produce generic and modular method for the design of such plant model: including time or not, deterministic or not, including faulty behaviour or not.

REFERENCES

Alpern B. and F. B. Schneider (1985). Defining liveness. *Inf. Process*. Lett. 21, 4 (Oct.), 181–185.

Bengtsson J. and W. Yi (2004). Timed Automata: Semantics, Algorithms and Tools. *LNCS 3098, Springer-Verlag*.

Bérard B., M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen (1999). Systems and software verification: model-Checking techniques and tools. *Springer, 1999, 190 pages.*

Cheung and Kramer 1999. S. C. Cheung, J. Kramer : *Checking Safety Properties Using Compositional Reachability Analysis*, ACM Transactions on Soft Eng and Meth, Vol.8, No. 1, January 1999.

Emerson E.A. and Halpern J.Y. (1986). Sometimes and Not Never revisited : on branching versus linear time temporal logic. *Journal of the ACM*, 33, 1, p. 151-178.

Frey G. and L. Litz (2000). Formal methods in PLC programming. *Proceedings of the IEEE Conference on Systems, Man and Cybernetics, SMC 2000*. Nashville, USA.

Manna Z. and A. Pnueli (1995). Temporal Verification of Reactive Systems : Safety. *Springer-Verlag, NewYork, NY*.

Machado J., B. Denis, J.-J. Lesage (2006). A generic approach to build plant models for DES verification purposes. *Wodes'2006*. Ann Arbor, Michigan, USA, 2006.

Merkte T. and T. Menzel (2000). Methods and tools to the verification of safety-related control software. *Proc. IEEE int. conf. on Systems Man and Cybernetics*, Nashville, Tennessee, USA.

Moon I. (1994). Modeling programmable logic controllers for logic verification. *IEEE Control Systems*, **14,** 2, 1994, p. 53-59.

Roussel and Denis. J-M. Roussel, B. Denis : Safety properties verification of ladder diagram programs: Journal Européen des Systèmes Automatisés, Vol. 36, pp. 905-917, 2002.