

Formal methods in PLC programming


chethana h j

Cite this paper

Downloaded from [Academia.edu](#) 

[Get the citation in MLA, APA, or Chicago styles](#)

Related papers

[Download a PDF Pack](#) of the best related papers 



[Safe controllers design for industrial automation systems](#)

Eurico Seabra, Filomena Soares

[III-Phase Verification and Validation of IEC Standard Programmable Logic Controller](#)

Suraj Dangol

[Palabras clave](#)

Rosa Melano

Formal methods in PLC programming

Georg Frey and Lothar Litz

Abstract — A detailed generic model of the control design process is introduced and discussed. It is used for surveying different formal approaches in the context of PLC-programming. The survey focuses on formal methods for verification and validation (V&V). The varying works in this area are categorized using three criteria: the general Approach to the task (model based, constrained based or without a model), the Formalism (Petri net, automata,...) used to state the formal description, and the Method (Model-Checking, Reachability Analysis, ...) used to analyze the properties. Based on these three criteria (A-F-M) a three letter code for V&V approaches is introduced. Some works from the multitude of V&V research are presented and categorized using this new system.

Index terms—PLC, logic control, verification, validation, formal methods.

I. INTRODUCTION

Since the 1970s PLC has been the primary workhorse of industrial automation. For a long time it has provided a distinct field of research, development and application, mainly for Control Engineering. This area has produced its own design methods and programming languages. Due to its importance for industrial application a lot of these methods have been standardized internationally. Figure 1 (adapted from [1]) shows an overview of the standardization. Currently the most influential standards are IEC 1131 [2], [3] and IEC 1499 [4], [5], [6].

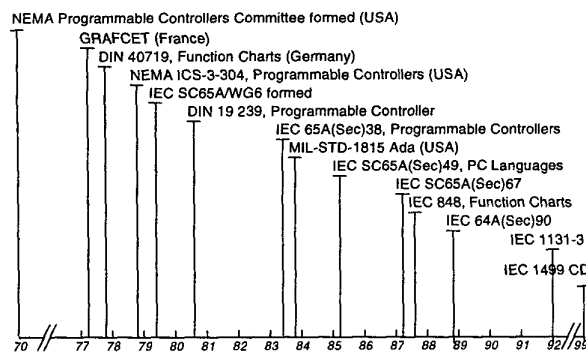


Figure 1: Standardization in PLC programming

Contact: Georg Frey, University of Kaiserslautern, Institute of Process Automation, PO Box 3049, 67653 Kaiserslautern, Germany, tel.: +49 631 205 3652, fax: +49 631 205 4462, e-mail: frey@eit.uni-kl.de.

Although being a rather intuitive discipline for a long time, industrial PLC programming will be more and more supported by formal methods. There are several reasons for the application of formal methods with PLC programming:

- The growing complexity of the control problems, the demand for reduced development time, and the possible reuse of existing software modules result in the need for a formal approach in PLC programming. The papers by Baresi et al. [7] and by Antoniadis and Leopoulos [8] primarily aim in this direction.
- The demand for high quality solutions and especially the application of PLC in safety-critical processes need verification and validation procedures, i.e. formal methods to prove specific static and dynamic properties of the programs, as for example liveness, unambiguity or response times. The papers by Canet et al. [9] and by Mertke and Menzel [10] deal with this aspect of formal method application to PLC programming.

In Figure 2 a generic model of the logic control design process is given [11]. The presentation form is a Channel Agency Net, see e.g. Reisig [12]. Without the use of formal methods the controller design process only consists of the outer ring: The realization of the controller is derived from the informal specification by direct implementation and afterwards it is informally validated against the informal specification.

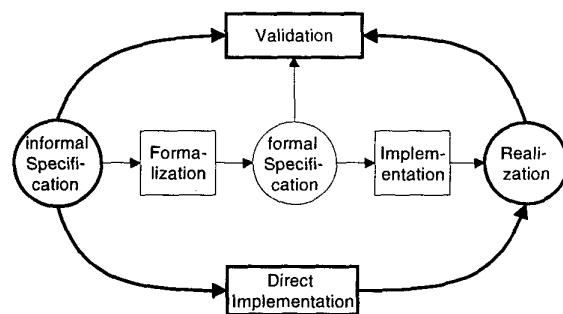


Figure 2: Design process for logic control systems [7].

In almost all cases, the designer of a logic control system starts with a given **informal specification** of the control problem. The term “informal” refers to everything that is not based on a strictly composed, syntactically and semantically well-defined form. In addition to an ‘easy to under-

The informal method of **validation** is the test of the implemented controller against the informal specification. Today this often used approach involves a team of control designers and users. The problem with informal validation is that it is never complete and it takes quite a lot of time and person-power.

The different parts of the **formal specification** with its new abilities in controller design are discussed in this paper: In Section II a more detailed generic model of the logic control design process is introduced and the formal methods associated with it are presented. Section III focuses on different approaches for verification and validation—the main aim of formal methods in PLC programming today.

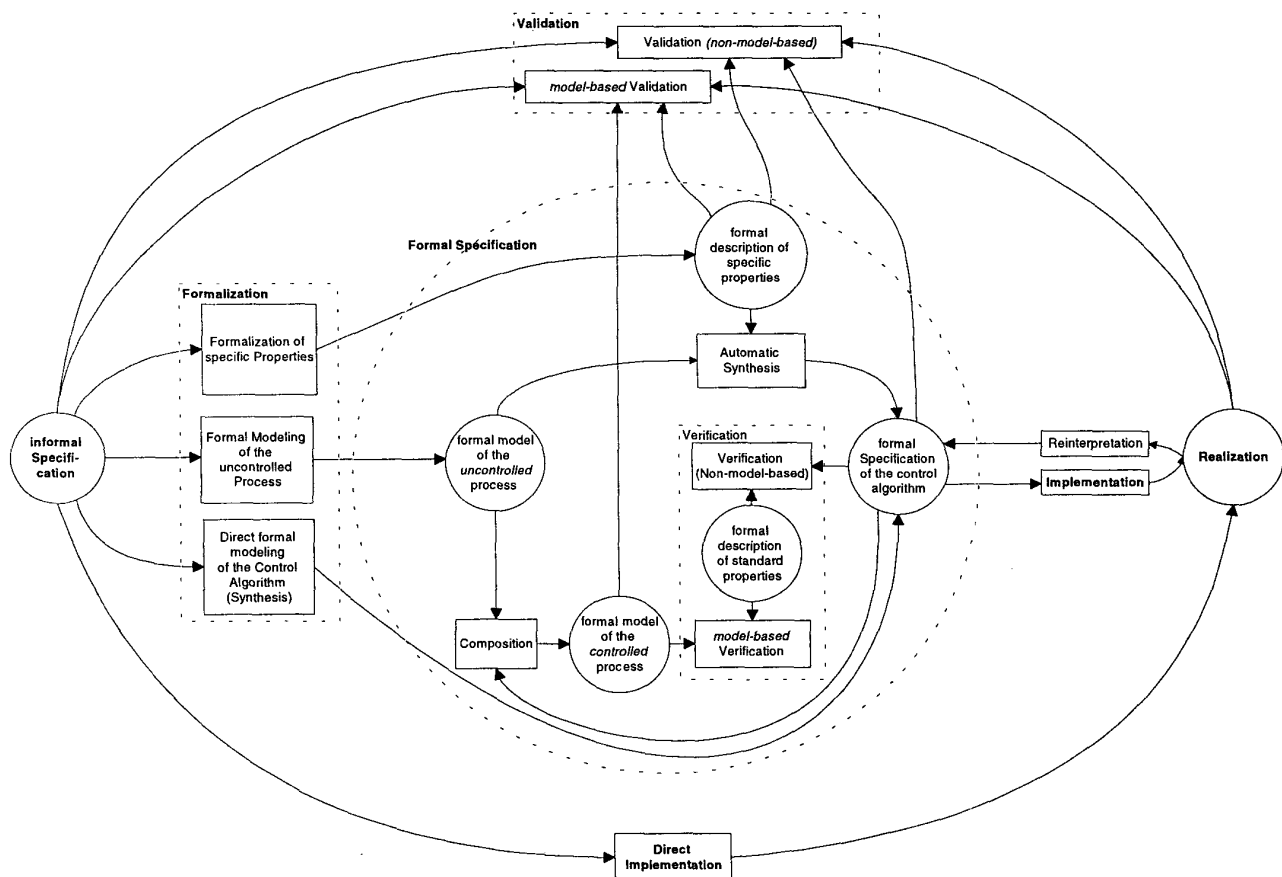


Figure 3: Detailed Design Process with Formal Specification and the Methods.

II. THE CONTROL DESIGN PROCESS

A. Formalization and Reinterpretation

A more detailed model of the logic control design process is given in Figure 3. The formalization of the informal specification consists of three different tasks:

- **Formalization of specific properties**, resulting in a set of properties to be fulfilled by the controller or the controlled process. These control objectives are formalized using temporal logic [15], algebraic conditions [16] or automata [17].
- **Formal modeling of the uncontrolled process** resulting in a process model that is needed in model based approaches. This model may be discrete or hybrid, depending on the properties to check.
- **Direct formal modeling of the control algorithm** can be done if the control problem given by the informal specification is clearly structured and not too voluminous. It is some kind of manual synthesis. Semi-formal synthesis approaches include the step-wise refinement following strict rules that guarantee given properties in the design process.

As Figure 3 shows, depending on the formal methods applied, not all of these tasks have to be done. Furthermore, there are specification methods that combine several parts of the formalization in one step, resulting in a combined model. For instance the Process IPN (PIPN) presented in [18] contains the model of the plant and the properties to be fulfilled.

The **formal specification of the control algorithm** can be derived in different ways. Either it is the result of the synthesis (manual or automatic). Or it is build via a **reinterpretation** procedure from the already implemented PLC code.

There are two reasons for the **reinterpretation** (also called translation, e.g. in [19]) of existing PLC code into a formal description:

- Most PLC programmers have no formal background and hence they stay with their programming techniques.
- There are millions of already existing PLC programs that can not be formally treated in any other way.

Approaches for the reinterpretation of PLC programs written in IEC 1131 Languages (IL = Instruction List, SFC = Sequential Function Chart, LD = Ladder Diagram, FBD = Function Block Diagram, ST = Structured text) can be found in the following papers:

- Mertke and Menzel [10] translate IL to Petri nets.
- Canet et al. [9] use a transition system to reinterpret IL.
- Hassapis et al. [20] translate SFC to hybrid automata.

- Kowalewski and Preußig [21] reinterpret SFC by C/E (Condition/Event)-systems.
- Völker and Krämer [22] represent ST, SFC and FBD by higher order logic, the latter two by representing them in ST first.
- Jiménez-Fraustro and Rutten [23] use the synchronous language SIGNAL to reinterpret ST.

B. Synthesis

The **automatic synthesis** of the control algorithm uses the formal description of the process and the formal properties. Methods for an automatic controller synthesis based on Petri Nets are described e.g. by Holloway and Krogh [24]. Hanisch et al. use Condition/Event systems [25] whereas Moor et al. use automata [15]. Dierks [26] developed a synthesis method based on Duration Calculus. For formal approaches, see also the works of Ramadge [27] and Li [28] both with Wonham.

C. Implementation

Using one of the standardized PLC languages the formal description of the control algorithm is implemented directly (using a compiler) or indirectly (using an interpreter implemented in the PLC). In the following papers direct implementation methods are described:

- Frey works out a direct implementation of Petri nets using SFC [29] or IL [30].
- Dierks shows in [31] a method to directly implement automata using ST.
- Cutts and Rattigan [32], Stanton et al. [33], and Uzam and Jones [34], present different approaches for the implementation of Petri nets in LD.

III. V&V VERIFICATION AND VALIDATION

Verification and Validation are the main areas for applying formal methods in PLC programming. Nevertheless, the notions are often confused. They answer, in fact, different questions. This is pointed out by Boehm [35] as follows:

- ‘Validation: Are we building the right product’
- ‘Verification: Are we building the product right’

Roussel and Lesage state more precisely [36]: ‘The verification is the proof that the internal semantics of a model is correct, independently from the modeled system. The searched properties of the models are stability, deadlock existence, ... The validation determines if the model agrees with the designer's purpose.’

Verification and Validation may use the same formal methods but the properties investigated in verification are stan-

dard and hence can be assumed as already formalized. Therefore in principle, verification can be fully automated. In validation specific properties of the controller have to be formalized. Therefore the investigation of the informal specification is necessary. Hence validation can not be fully formal and not be fully automated.

The generic model shows different approaches for verification and validation. These are discussed in detail in subsection A. The varying approaches often use the same modeling or description mechanisms. Hence, the formalisms are presented separately in subsection B. Finally the methods used to check properties are presented in subsection C. Examples of verification and validation are presented in subsection D. Each of the examples consists of a combination of approach, formalism, and method. Using the results of sub-sections A to C a three-letter-code is assigned to them.

A. Approach

Validation as well as Verification can be model based or non model based.

- M Model based:** In model based approaches a model of the process under control is included in the analysis. The properties checked are statements on the controlled system.
- N Non Model based:** non model based approaches analyze the formal description of the control algorithm without taking the process into account. Connections of the controller to its environment are treated either as if they were not present or as if anything could happen.
- C Constrained based:** Constrained based approaches are typically non-model-based with the inclusion of some very restricted knowledge about the process, for instance that two binary input signals are always disjoint.

B. Formalism

The presented approaches and methods are based on formal models. The following six formalisms are (among others) used for the formal description of PLC programs:

- P Petri nets:** For an introduction of different Petri net models see David and Alla [37].
- C Condition/Event (C/E) Systems:** C/E-Systems are introduced by Sreenivas and Krogh [38]
- A Automata:** Especially hybrid automata are used in V&V of logic controllers, see Henzinger [39] for an introduction to this formalism.

- L (Higher order) Logic:** For an introduction to Higher order Logic see [40].
- S Synchronous Languages:** the synchronous approach is presented in [41]. A synchronous language used in control applications is "Signal"[42].
- T General Transition Systems:** See Ostroff [43] and Canet et al. [9] for examples.
- E (Algebraic) Equations:** Gunnarson [16] presents an approach using algebraic equations over finite fields. (Max,+) algebra [44] approaches also fall in this category.

C. Method

- S Simulation** is a widely used method for verification and validation. Especially if there is a huge number of input and output signals, simulation is very time-consuming since every possible situation has to be checked. Hence, in most cases simulation is restricted to the direct application of input signals and comparing the resulting output signals to the specification. Hereby, the behavior of the process, i.e. its reaction to the input signals, is neglected and – more critical – only parts of the controller are tested. Simulation is not considered in this survey.
- R Reachability Analysis:** Methods based on reachability analysis build the complete state-space of the modeled system and check properties by investigating the structure and the components of this state-space. The problem with reachability analysis is the state-explosion in discrete systems: The number of states in the system grows exponentially with the number of discrete variables.
- M Model checking:** In model-checking, specifications of the system behavior are checked automatically on a finite model of the system. The specifications are formulated in a temporal logic (see [45] and [46] for an overview on temporal logic). The model is formalized using automata or Petri nets e.g.. Model-checking does not avoid the problem of state-explosion.
- T Theorem Proving:** In theorem proving methods the system and its expected properties are formalized using some mathematical logic. Then the property formulas have to be proofed from the axioms of the system description using some inference rules. A Theorem Prover assists the user in formulating the proof. Intelligent approaches using machine-reasoning may avoid this drawback of needing a highly qualified user. A great advantage of theorem proving is the avoidance of the state-explosion problem.

D. Examples

In the following some examples of verification and validation approaches are given. For further approaches including V&V for Grafcet see [19]. The presented methods are assigned a three-letter-code A-F-M indicating the used Approach, the Formalism to build the formal specification and the Method for analysis.

M-P-R: Frey and Litz [11] use a special Petri net as process model and another one as model of the controller. The verification is done using reachability analysis of the combined model.

M-P-M: Weng and Litz [47] present a model based verification approach using model checking with LTL (linear time temporal logic) as method and Petri nets as formal description.

M-P-M: Mertke and Menzel [10] present a model based validation approach. Their process model is build as Petri net and the PLC code is translated into another Petri net. The aggregation of both nets is used as the basis for model checking with LTL or CTL (computational tree logic). They also propose the specification of properties in semi-formal natural language with an automatic generation of the formal description.

M-A-M: Hassapis et al. [20] translate an SFC to an hybrid automaton. The process is also modeled with a hybrid automaton. With the aggregated model of the controlled process, model checking is performed using CTL and the HyTech tool.

M-C-R: Kowalewski and Preußig [21] translate SFC programs into C/E systems. Another C/E system is used to model the uncontrolled plant. The composition of these C/E systems results in a model of the controlled plant. Reachability analysis shows if the specifications (formalized in terms of forbidden states) are fulfilled.

N-P-R: Frey and Litz [48] use a special Petri net model of the controller. The verification is done using reachability analysis of the Petri net.

N-T-M: The Carnegie Mellon research group around G.J. Powers developed a method for the verification of given LD programs [49], [50], [51]. The LD is reinterpreted using a Transition system and the properties to check are formalized using CTL. The model-checker Symbolic Model Verifier (SMV) takes the model and the properties and implicitly builds the state-automaton of the system and checks if the properties hold. If this is not true a state-sequence leading to the contradiction is produced.

N/C-T-M: Canet et al. [9] present an approach for the validation of existing PLC programs written in Instruction List. The PLC code is translated into a transition system. For this system specific properties are investigated using model-checking with LTL. The example presented in [9] shows

that the results are of more practical value with the inclusion of additional process knowledge (constraints).

N-L-T: The group of B.J. Krämer [22] use higher order logic to represent ST programs. The requirements are specified in LTL. The model and the requirements are used in a theorem prover.

IV. CONCLUSION

The paper gives an overview of the current state of the art of formal methods in PLC design. It rather aims to present examples then to be complete.

The presented generic model of the control design process and the definition of related terms allows the categorization of different approaches in the fast growing area of research and application.

A three-letter-code for verification and validation methods based on the describing triple Approach-Formalism-Method is introduced and explained by some examples.

V. REFERENCES

- [1] J.H. Christensen (Figure: International Language Standardization) in *PLCopen Standard Presentation V1.0*, 1998.
- [2] International Electrotechnical Commission (IEC), *International Standard 61131: Programmable Logic Controllers, Part 3: Languages*, 1993.
- [3] R.W. Lewis, *Programming industrial control systems using IEC 1131-3*, IEE Publishing, London, United Kingdom, 1998.
- [4] IEC 65/240/CD, *Function blocks for industrial-process measurement and control systems – Part 1: Architecture*, June 1999.
- [5] IEC 61499-2 (2nd Committee Draft, Ed. 1.0), *Function blocks for industrial-process measurement and control systems – Part 2: Engineering Task Support*, April 2000.
- [6] J.H. Christensen, 'Basic Concepts of IEC 61499', *Proceedings of Fachtagung Verteilte Automatisierung*, Magdeburg, Germany, pp. 55-62, 2000.
- [7] L. Baresi, M. Mauri, A. Monti and M. Pezzè, 'PLCTOOLS: Design, formal validation, and code generation for programmable controllers', *Proc. of the IEEE SMC*, 2000.
- [8] I.A. Antoniadis and V.I.N. Leopoulos: 'A concept for the integrated process description, PLC programming and simulation using Petri nets: Application in a production process' *Proc. of the IEEE SMC*, 2000.
- [9] G. Canet, S. Couffin, J.-J. Lesage, A. Petit and P. Schnoebelen, 'Towards the automatic verification of PLC programs written in instruction list', *Proc. of the IEEE SMC*, 2000.
- [10] Th. Mertke and Th. Menzel, 'Methods and tools to the verification of safety-related control software', *Proc. of the IEEE SMC*, 2000.
- [11] G. Frey and L. Litz 'Verification and Validation of Control Algorithms by Coupling of Interpreted Petri Nets', *Proc. of the IEEE SMC'98*, San Diego, Vol. 1, pp. 7-12, 1998.
- [12] W. Reisig, *A Primer in Petri Net Design*, Berlin, Heidelberg, New York, Springer, 1992.
- [13] P.W. Murrill, *Fundamentals of Process Control Theory*, ISA Press, 3rd ed., 2000.

- [14] Instrument Society of America (ISA), *ANSI/ISA-Standard S5.1: Instrumentation Symbols and Identification*, 1984, Re-affirmed 1992.
- [15] J.G. Thistle and W.M. Wonham, 'Control Problems in a Temporal Logic Framework', *International Journal of Control*, Vol. 44 (4), pp. 943-976, 1986.
- [16] J. Gunnarsson, 'Algebraic Methods for Discrete Event Systems - A Tutorial', *Proc. of the IEE WODES'96*, Edinburgh (GB), pp. 18-30, 1996.
- [17] T. Moor, J. Raisch, and S.D. O'Young 'Supervisory Control of Hybrid Systems via l-Complete Approximations'. *Proc. IEE WODES'98*, Cagliari, Italy, pp. 426-431, 1998.
- [18] C. Jörns, 'Transparent Representation of Information Flow in Automatic Control Systems for Verification Purposes' *Proc. of the IEE WODES'96*, Edinburgh (GB), pp. 368-373, 1996.
- [19] S. Lampérière-Couffin, O. Rossi, J.-M. Roussel and J.-J. Lesage, 'Formal validation of PLC programs: A survey', *Proceedings of the ECC'99*, 1999.
- [20] G. Hassapis, I. Kotini and Z. Doulgeri, 'Validation of a SFC software specification by using hybrid automata', *INCOM'98*, Volume II, pp. 65-70, 1998.
- [21] S. Kowalewski and J. Preußig, 'Verification of sequential controllers with timing functions for chemical processes', *Proc. of the 13th IFAC World Congress*, San Francisco, Vol. J, pp. 419-424, 1996.
- [22] N. Völker and B.J. Krämer, 'Modular Verification of Function Block Based Industrial Control Systems', *Proc. 24th IFAC/IFIP Workshop on Real-Time Programming*, May 1999.
- [23] F. Jiménez-Fraustro and E. Rutten, 'A synchronous model of the PLC programming language ST' *Proceedings of the Work In Progress session, 1st Euromicro Conference on Real-Time Systems, ERTS'99*, York (GB), June 9-11, pp. 21-24, 1999.
- [24] L.E. Holloway and B.H. Krogh, 'Synthesis of feedback control logic for a class of controlled Petri nets', *IEEE Trans. on Automatic Control*, Vol. 35, No. 5, pp. 514-523, 1989.
- [25] H.-M. Hanisch, A. Lüder and J. Thieme 'A Modular Plant Modeling Technique and Related Controller Synthesis Problems' *Proceedings of the IEEE SMC'98*, San Diego, pp. 686-691, 1998.
- [26] H. Dierks, 'Synthesizing Controllers from Real-Time Specifications' *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(1), pp. 33-43, 1999.
- [27] P.J.G. Ramadge and W.M. Wonham, 'The Control of Discrete Event Systems', *Proc. of the IEEE*, Vol. 77, pp. 81-97, 1989.
- [28] Y. Li and W.M. Wonham, 'Control of Vector Discrete-Event Systems I – The Base Model', *IEEE Transactions on Automatic Control*, Vol. 38, No. 8, Aug. 1993, pp. 1214-1227.
- [29] G. Frey, 'PLC Programming for Hybrid Systems via Signal Interpreted Petri Nets', *Proceedings of the 4th International Conference on Automation of Mixed Processes ADPM, Dortmund, Germany*, September 2000.
- [30] G. Frey, 'Automatic Implementation of Petri net based Control Algorithms on PLC', *Proceedings of the American Control Conference ACC2000, Chicago*, June 2000.
- [31] H. Dierks, 'PLC-Automata: A New Class of Implementable Real-Time Automata', *Transformation-Based Reactive Systems Development (ARTS'97)*, M. Bertran and T. Rus, editors, *Springer LNCS 1231*, pp. 111-125, 1997.
- [32] G. Cutts and S. Rattigan, 'Using Petri Nets to Develop Programs for PLC Systems.' *Proc. of Application and Theory of Petri Nets 1992, Springer LNCS 616*, pp. 368-372, 1992.
- [33] M.J. Stanton, W.F. Arnold and A.A. Buck, 'Modelling and Control of Manufacturing Systems using Petri Nets', *Proceedings of the 13th IFAC World Congress*, pp. 329-334, 1996.
- [34] M. Uzam and A. H. Jones, 'Discrete Event Control System Design using Automation Petri Nets and their Ladder Diagram Implementation', *Int. Journal of Advanced Manufacturing Systems, special issue on Petri Nets Applications in Manufacturing Systems*, Vol. 14, No. 10, pp. 716-728, 1998.
- [35] B. W. Boehm, 'Software Engineering: R&D trends and defense needs', *Research Directions in Software Technology (P. Wegner, Ed.)*, MIT Press, Cambridge, 1979.
- [36] J.-M. Roussel and J.-J. Lesage, 'Validation and Verification of Grafset using state machine', *Proceedings of IMACS-IEEE CESA'96, Lille (F)*, pp. 758-764, July 1996.
- [37] R. David and H. Alla, *Petri Nets and Grafset - Tools for Modeling Discrete Event Systems*, Prentice Hall, 1992.
- [38] R.S. Sreenivas and B.H. Krogh, 'On Condition/Event Systems with Discrete State Realizations', *Discrete Event Dynamic Systems: Theory and Applications*, Kluwer Academic Publishers, Boston, USA, Vol. 1, pp. 209-236, 1991.
- [39] T.A. Henzinger, 'The Theory of Hybrid Automata', *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 278-292, July 1996.
- [40] M.J.C. Gordon. and T.F. Melham, *Introduction to HOL*. Cambridge University Press, 1993.
- [41] A. Benveniste and G. Berry, 'The Synchronous Approach to Reactive and Real-Time Systems', *Proceedings of the IEEE*, Vol. 79, No. 9, pp. 1270-1282, 1991.
- [42] A. Benveniste and P. Le Guernic, 'Hybrid Dynamical Systems Theory and the SIGNAL Language', *IEEE Transactions on Automatic Control*, Vol. 35, No. 5, pp. 525-546, 1990.
- [43] J.S. Ostroff, 'Automated verification of timed transition models', *Int. Workshop on Automatic Verification Methods for Finite State Systems (Springer LNCS 407)*, pp. 247-256, 1989.
- [44] F. Bacelli, G. Cohen, G.J. Olsder and J.P. Quadrat, *Synchronization and Linearity (An algebra for discrete event systems)*, John Wiley & Sons, 1992.
- [45] R. Alur and T.A. Henzinger, 'Logics and models of real time: a survey', *Real Time: Theory in Practice, Springer LNCS 600*, pp. 74-106, 1992.
- [46] T.A. Henzinger, 'It's about time: real-time logics reviewed', *Proceedings of the Ninth International Conference on Concurrency Theory (CONCUR 1998)*, *Springer LNCS 1466*, pp. 439-454, 1998.
- [47] X. Weng and L. Litz, 'Verification of logic control design using SIPN and model checking—methods and case study' *Proceedings of the American Control Conference ACC2000, Chicago*, 2000.
- [48] G. Frey and L. Litz, 'Correctness Analysis of Petri Net Based Logic Controllers', *Proceedings of the American Control Conference ACC2000, Chicago*, 2000.
- [49] I. Moon, 'Modelling Programmable Logic Controllers for Logic Verification', *IEEE Control Systems Magazine*, pp. 53-59, 1994.
- [50] S.T. Probst, 'Chemical Process Safety and Operability Analysis Using Symbolic Model Checking', *Ph.D. Thesis, Department of Chemical Engineering, Carnegie Mellon University*, 1996.
- [51] I. Moon, G. Powers, J.R. Burch, and E.M. Clarke, 'Automatic Verification of Sequential Control Systems Using Temporal Logic', *AIChE Journal*, Vol. 38 (1), pp.67-75, 1992.