Annual Reviews in Control 000 (2017) 1–17

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Synthesis and implementation of logic controllers – A review

J. Zaytoon[*], B. Riera

CReSTIC, Université de Reims Champagne-Ardenne, Reims, France

ARTICLE INFO

Article history:
Received 5 October 2016
Revised 21 February 2017
Accepted 8 March 2017
Available online xxx

Keywords:
Control synthesis
Implementation
Supervisory control theory
Algebraic synthesis
Logic filter
Programmable logic controllers

ABSTRACT

This paper provides an overview of some well-known formal approaches for the synthesis and implementation of logic controllers. Most of these approaches are based on the use and the adaptation/extension of the supervisory control theory of discrete-event systems. Recent contributions, based on algebraic synthesis and logic constraints are also be highlighted.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

Since the 1970 s Programmable Logic Controllers (PLCs) are used in a very large number of systems such as embedded systems, transport systems, power plants or production systems. Several components of our daily lives and of the economy at large thereby rely upon the successful operations of these controllers.

Control engineers today mainly handle the development of industrial automation applications by direct implementation of the control task based on the interpretation of informal specification and text documents. This effort is assisted by standardized engineering tools for the programming of PLCs. However, the informal specification of the control software have to be manually and intuitively transferred into the control program as they are not formally defined in practice due to a lack of time and expertize. This practice (Johnson, 2007) most often leads to a deficient documentation of sequential interdependencies within the control program and additional costs caused by the erroneous interpretation of the textual requirements. This renders the error detection processes as well as the reconfiguration and the maintenance of the control logic extremely difficult. Several ad hoc design approaches have therefore been proposed, while human ingenuity is still an essential component of the design procedure.

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 logic control design and PLC programming. In particular, the application of PLC in safety-critical processes need formal verification and design procedures to prove or to reinforce specific static and dynamic properties of the programs, and to ensure that no flaw due to misinterpretation of the informal specification and no errors leading to non-functional or hazardous behaviors have been introduced during design.

To reach this goal, many formal approaches have been proposed for the design of logic controllers, and some of them go back to the 80′s (Krogh & Beek, 1986; Martinez, Munro, & Silva, 1987; Zhou, Dicesare, & Rudolph, 1992). They aim at detecting flaws once the controller has been designed or avoiding flaws during design.

A first class of formal approaches consists in letting the control system designer develop control laws based on the requirements contained in the set of specifications, and then in automatically analyzing a formal representation of these control laws. The aim here is to check whether the specifications and/or their implementation, are conform to what is expected. Such analysis may be carried out by using formal verification techniques (Bel Mokadem, Bérard, Gourcuff, De Smet, & Roussel, 2010; Boulanger, 2012; Frey & Litz, 2000; Hanisch, Lobov, Lastra, Tuokko, & Vyatkin, 2006; Johnson, 2007; Perin & Faure, 2013; Soliman & Frey, 2011; Zaytoon, 2000) or formal test methods (Provost, Roussel, & Faure, 2010; Broy, Jonsson, Katoen, Leucker, & Pretschner, 2005; Jamro, 2015; Rösch & Vogel-Heuser, 2017; Rösch, Ulewicz, Provost, & Vogel-Heuser, 2015).

Another class of formal approaches, qualified as synthesis (Ramadge & Wonham, 1987; Ramadge & Wonham, 1989; Roussel & Lesage, 2014; Queiroz & Cury, 2002; Zaytoon & Carré-Ménétrier, 2001), aim at generating the control laws that satisfy the required properties by construction, without involvement of the

[*] Corresponding author.
E-mail address: janan.zaytoon@univ-reims.fr (J. Zaytoon).

http://dx.doi.org/10.1016/j.arcontrol.2017.03.004
1367-5788/© 2017 Elsevier Ltd. All rights reserved.

designer (or at least by limiting his/her involvement as much as possible).

Modeling languages and appropriate methods from the software engineering domain have also been proposed to be adapted to the automation engineering domain to generate control code from a specification model of the controller. Though having great opportunities and a wide acceptance among researchers (Delaval, Rutten, & Marchand, 2013; Hajjar, Dumitrescu, Pietrac, & Niel, 2015; Lukman, Godena, Gray, Hericko, & Strmcnik, 2013; Thramboulidis & Frey, 2011; Witsch & Vogel-Heuser, 2009), most of those methods and tools still lack acceptance in industrial practice. The reason is twofold: on the one hand formal methods proposed in the literature are based on modeling languages which are usually not familiar to control practitioners. On the other hand, most model-driven approaches are not easy to apply in practice since they only allow performing modifications and revisions within the (formal) models. However, practical experience of the PLC engineering shows that the major modifications are directly implemented within the PLC code and thus need to be re-documented into the corresponding specification.

The aim of this paper is to provide an overview of formal approaches for the synthesis and implementation of logic controllers. A sketch and a brief description of the models and activities underlying the approaches for synthesis and implementation of logic controllers are presented in Section 2. A majority of these approaches are based on the Supervisory Control Theory (SCT) (Ramadge & Wonham, 1987, 1989) that will be presented in Section 3 as well as a discussion concerning its advantages and limits regarding control implementation. Section 4 overviews some representative contributions aiming at improving the applicability of the SCT to the control of industrial processes. Some recent control synthesis and implementation approaches, based on algebraic synthesis and logic constraints, are discussed in Section 5. An extended abstract of this paper was presented at Zaytoon and Riera (2016).

2. Models and activities for control synthesis and implementation

Control engineers handle the development of an industrial automation starting from informal specification of the process and the control tasks as well as requirements for the controlled system. However, the different parts of the specification are not always clearly separated. These informal specifications are based on informal text documents that can include timing diagrams, sketches, and/or a set of equations (Frey & Litz, 2000).

The standard industrial approach to get the realization from the informal specification is the direct implementation of the controller using a PLC programming language. With standard hardware and well-defined PLC-functionality, the realization consists of the programmed controller.

Many approaches for the synthesis and implementation of control realization have been proposed. These approaches rely on different models and formalisms. Fig. 1 depicts an abstract view of the different routes that can be followed by the synthesis and implementation approaches to obtain the target control realization starting from the informal control specification. The overall design process involves three types of generic activities: Formalization, Synthesis and Implementation. These activities are briefly described in the following subsections. They involve a variety of validation and verification procedures underlying the different models and routes shown in Fig. 1. In the remainder of this paper, each class of approaches that will be reviewed will be positioned with respect to Fig. 1 by highlighting the activities and models that are involved, and using dotted lines for the activities and model that are not used.

2.1. Formalization

The formalization of the informal specification is a human core capability that consists of three tasks:

- Formal specification of the controller, which is some kind of manual synthesis that usually include step-wise refinements to guarantee given properties in the design process.
- Formal modeling of the process to be controlled, resulting in a plant model that is needed in model based approaches. This model may be discrete or hybrid, depending on the properties to check or to reinforce.
- Formalization of the desired properties, resulting in a set of safety (what to avoid) and liveness (what to accomplish) properties to be fulfilled by the controller or the controlled process.

Depending on the formal methods applied, not all of these 3 tasks have to be performed. Furthermore, some methods can combine two or the three above-mentioned formalization tasks in one step, resulting in a combined-model. Nevertheless, irrespective of the synthesis and implementation method used, the formalization activity by the designer remains one of the weak links of the automatic generation of the controller.

Industrial practice to develop a control realization is primarily concerned with control-based specifications rather than plant-based specifications. These control specifications are commonly given by means of high-level specification formalism, such as Grafcet (David, 1995; David & Alla, 1992), that provides a straightforward means to describe the required control tasks and capture the concurrency (of actions and transitions), synchronization and the possibility of using events, conditions and complex logic operators. In this traditional practice, the control designer tends to implicitly predict the reactions and the behavior of the plant to be controlled within the control specifications. However, the parallel and interleaving reactions of the plant are not easy to identify in the case of complex systems. The behavior of the control realization resulting from such a specification model may therefore be different from both the intended and the specified behavior. For example, Zaytoon and Carré-Ménétrier (2001) have shown that control execution may lead to deadlocks in situations that have been proved to be deadlock-free in the control specification model. Conversely, some identified deadlocks in the specification model cannot occur throughout the execution of the correspondingly implemented controller. In the same way, the safety and liveness constraints that are proved to be satisfied with respect to the control specifications may be invalidated by the real execution of the controller and vice-versa.

The use of an explicit model of the plant may therefore be necessary to identify the correct behavior of the control realization, and to perform model analysis and control synthesis accurately. However, obtaining a precise model of the plant is not a trivial task, and the difficulty lies in the choice of the degree of granularity and the required abstraction level for this model (Carré-Ménétrier & Zaytoon, 2002). A higher-level description often leads to a synthesis problem that is easy to solve but may not be the right level for converting this solution into a control code. Thus the solution obtained at a higher-level may be useless for control implementation (Roussel & Giua, 2005). On the other hand, a more detailed model may lead the synthesis algorithms to generate unrealistic-size controllers, due to a possible explosion of the state space. A well-known answer to these needs consists in decreasing the complexity of the modeling phases by promoting a decomposition strategy that leads to introduce modularity and hierarchy within the modeling/synthesis framework.

Another common problem consists in the fact that there is no clear separation between the model of the plant and the model of controller: a plant model may implicitly contain significant