Automatica 56 (2015) 93–104

Contents lists available at ScienceDirect

Automatica

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

Supervisory control of discrete event systems with distinguishers[☆]

José E.R. Cury[a], Max Hering de Queiroz[a], Gustavo Bouzon[b], Marcelo Teixeira[a]

[a] Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, Brazil

[b] Optimale Engenharia e Soluções Tecnológicas, Campo Grande, Brazil

ARTICLE INFO

Article history:
Received 4 September 2012
Received in revised form
25 November 2014
Accepted 19 March 2015
Available online 16 April 2015

Keywords:
Discrete event systems
Supervisory control
Distinguishers
Approximations

ABSTRACT

This paper deals with two relevant aspects of the Supervisory Control Problem (SCP) of Discrete Event Systems (DES): the degree of difficulty faced when modeling specifications to be fulfilled by the system under control, and the computational complexity of the synthesis procedure. The presented approach consists in refining the set of events of a DES model into a new set. Each refinement is properly chosen to identify a particular instance of the original event in the system, which may simplify the modeling of specifications. A map named *Distinguisher* is then proposed to establish the relationship between strings of the original and refined alphabets. It is initially shown that using a refined set of events to solve a SCP directly leads to the optimal control solution, yet without providing computational advantages in synthesis with respect to the nonrefined method. Then, we propose the use of outer-approximations for the refined DES model as a way to reduce the cost of synthesis, while preserving controllability, least restrictiveness and nonblocking of the control solution. Two examples of manufacturing systems illustrate our results.

1. Introduction

Information is a key issue to be considered in the Supervisory Control Problem (SCP) for Discrete Event Systems (DES) (Ramadge & Wonham, 1987), since it affects several aspects of the problem resolution. Three of those aspects are: the conditions for the existence of solutions to the problem; the computational complexity of the synthesis procedure; and the degree of difficulty faced when modeling specifications to be fulfilled by the system under control.

On one hand, restrictions imposed to the information channels connecting a plant to controller agents may induce partial observability in the system. This may happen due to the lack of specific sensors (Cieslak, Desclaux, Fawaz, & Varaiya, 1988; Huang, Rudie, & Lin, 2008; Lin & Wonham, 1988, 1990) or by the use of decentralized control structures where local supervisors have access only to local subsets of the whole set of events generated by the plant (Lin & Wonham, 1990; Rudie & Wonham, 1992; Yoo & Lafortune, 2002). Partial observability of events leads to changes in the conditions under which a solution can exist for the SCP, compared to the case of total observability. In this sense, some authors have considered the problem of computing sets of events that should be made observable in order to assure the existence of solutions to the SCP (Haji-Valizadeh & Loparo, 1996; Khuller, Kortsarz, & Rohloff, 2004; Yoo & Lafortune, 2002). Moreover, whenever a solution exists, partial observability implies modifications in the supervisor synthesis and associated computational complexity.

Information can also be examined from another perspective: the abstraction level used to represent a plant model can be suitable to express certain specifications, yet inadequate for others. Consider a particular occurrence of an observable event in the system evolution. Depending on the abstraction level assumed in the DES model, a control designer may have to keep track of previous event traces leading to the occurrence of a specific event, in order to completely assess the state information brought by that event. For example, modeling the underflow requirement for workpieces in a buffer with events "pick-up" and "insert" requires keeping track of the whole trace of such events in order to distinguish the occurrences of events that make the buffer empty.

☆ This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grants 307655/2014-8 and 554056/2009-7. The material in this paper was partially presented at the 9th International Workshop on Discrete Event Systems, May 28–30, 2008, Göteborg, Sweden and at the 7th IEEE International Conference on Control and Automation, December 9–11, 2009, Christchurch, New Zealand. This paper was recommended for publication in revised form by Associate Editor Jan Komenda under the direction of Editor Ian R. Petersen.

E-mail addresses: cury@das.ufsc.br (J.E.R. Cury), max@das.ufsc.br (M.H. de Queiroz), gustavobouzon@gmail.com (G. Bouzon), mt@das.ufsc.br (M. Teixeira).

http://dx.doi.org/10.1016/j.automatica.2015.03.025
0005-1098/© 2015 Elsevier Ltd. All rights reserved.

Extensions of the hierarchical control abstract the original system model into a higher-level model in such a way that previously complex specifications can be expressed using high-level symbols. Two of these extensions are in general seen in the literature: one that builds abstractions by erasing some events of the original low-level model through *natural projections* (da Cunha & Cury, 2007; Feng & Wonham, 2008; Schmidt, Moor, & Perk, 2008); and another, that assigns to the abstracted model its own set of events, and the high-level model is obtained by a map that reports the occurrence of particular high-level events, whenever the original model reaches some specific states (Wong & Wonham, 1996; Zhong & Wonham, 1990). In this sense, events of the abstracted model represent actually strings in the original low-level model.

In opposition to the bottom-up approach considered in hierarchical control, in many situations it may happen that the abstraction level of a DES model, available for the resolution of a particular SCP, is too high. Then, representing a specification may become a too hard engineering task. In fact, while the ability to manage a modeling task is usually limited to a few dozens of states, an automaton model for a single specification may involve hundreds of states. In these cases, refining the original model by enriching information on occurrences of some events in the system can facilitate or make modeling tasks feasible.

This paper proposes an approach to modify a given model in order to simplify the design of originally complex specifications. It subsumes preliminary results in Bouzon, de Queiroz, and Cury (2008, 2009). The presented method is based on the refinement of the original set of events, along with the concept of a *Distinguisher*, a map of each string of original system events to strings of refined events, representing occurrences of the original events with particular semantics. By suitably refining a model using a *Distinguisher*, a complex specification modeling task can be turned into a much easier one, by an appropriate use of the distinguished set of events. We show, then, that the optimal solution of the SCP can be obtained by incorporating the distinguisher in the plant model. We also show that refining a model does not imply increasing the computational complexity of synthesis.

Although this approach allows the designer to more easily express specifications, the synthesis effort may remain an obstacle to solve the SCP. Then, a further contribution of this paper is to show how distinguishers can be used to construct outer-approximations of a refined model, such that a solution to the SCP is synthesized with computational gains. Conditions to guarantee controllability, least restrictiveness and nonblocking are also derived.

The idea of events distinction, as presented in this paper, has already been adopted in the literature for specific purposes. For example, inspired on the results in Bouzon et al. (2008, 2009), a distinguisher has been used as a step of the compositional synthesis to avoid nondeterminism of automata (Mohajerani, Malik, & Fabian, 2014). This improves the construction of abstractions to be used in synthesis, which may gratefully reduce its computational cost. In contrast, modeling tasks are addressed in Oliveira, Cury, and Kaestner (2004), Chen and Lin (2001), Kumar and Garg (2005), Ouedraogo, Kumar, Malik, and Akesson (2011), by extending a standard automaton with the addition of variables. In Chen and Lin (2001); Kumar and Garg (2005); Oliveira et al. (2004), infinite states systems are handled, with forbidden state specifications being expressed as predicates defined on the variables, while non-blocking issues are addressed only by Ouedraogo et al. (2011). None of these works, however, deals with the idea of using approximations as a way to alleviate the computational cost of synthesis.

The results of this paper are illustrated in the context of two examples. The first example describes a manufacturing system with material feedback, which contains a particular (complex) specification. The modeling is addressed with the help of a distinguisher and the synthesis complexity is reduced by using an approximation for the plant. This approach leads, firstly, to a sub-optimal solution. Then an intermediate approximation is proposed, which finally leads to the optimal solution, while keeping important computational gains. The second example exploits the overflow and underflow control of a buffering system. The modeling is again handled using distinguishers and the synthesis with approximations (coarsest), in this case, directly leads to the optimal solution.

The paper is structured as follows. Notation, preliminary concepts and the SCP are introduced in Section 2. Section 3 presents the SCP with distinguishers and introduces a manufacturing system example. Key results regarding the efficient solution to this problem are presented in Section 4 and illustrated using the same example. An additional example is presented in Section 4.4 and some conclusions are discussed in Section 5.

2. Preliminaries

Some notations related to the Supervisory Control Theory (Ramadge & Wonham, 1987) are presented. Let Σ be a finite set of symbols, named *alphabet*. Σ^* denotes the set of finite strings formed with elements of Σ, including the empty string ϵ. The length of a string s is denoted by $|s|$. Given two strings $s, t \in \Sigma^*$, s is a *prefix* of t ($s \le t$) if there is $u \in \Sigma^*$ such that the concatenation $su = t$. Any subset of Σ^* is a *language* on Σ. A language L is *prefix-closed* if $L = \overline{L}$, where $\overline{L} = \{u \in \Sigma^* : u \le s,\ s \in L\}$ is the *prefix closure* of L. We say that an event $\sigma \in \Sigma$ is *eligible* after a string s in a language L if $s\sigma \in \overline{L}$. Two languages L and M are *nonconflicting* when $\overline{L \cap M} = \overline{L} \cap \overline{M}$.

The *open-loop behavior* of a DES (plant) is modeled by an automaton $G = (\Sigma, X, g, x_0, X_m)$, where Σ is the alphabet of events, X is the set of states, $x_0 \in X$ is the initial state, $X_m \subseteq X$ is the subset of marked states and $g : X \times \Sigma \to X$, the transition function, is a partial function defined in each state of X for a subset of Σ. The behavior of a plant G is represented by the *generated language* $L(G) \subseteq \Sigma^*$, a prefix-closed language containing all the sequences of events that can be generated by the plant, and by the *marked language* $L_m(G) \subseteq L(G)$, containing strings that represent complete tasks. The parallel composition (Cassandras & Lafortune, 2008) of two automata, G_1 and G_2, is denoted by $G_1||G_2$ and the same notation is used for the parallel composition of languages.

The control structure of G is defined by a partition $\Sigma = \Sigma_c \cup \Sigma_u$, where Σ_c is the set of controllable events, whose occurrence can be disabled, and Σ_u is the set of uncontrollable events, that cannot be disabled. A *(marker) supervisor* S is a map $S : L(G) \to 2^{\Sigma}$ together with a language $L_S \subseteq L_m(G)$ such that, for each string s generated by G, the control action of S over G (denoted by S/G) enables events of $S(s) \subseteq \Sigma$, with $\Sigma_u \subseteq S(s)$, and marks strings $s \in L_S$. The *generated closed-loop behavior*, given by the language $L(S/G)$, corresponds to the set of strings of $L(G)$ that survive under control; it can be recursively defined as (i) $\epsilon \in L(S/G)$, and (ii) $s\sigma \in L(S/G) \Leftrightarrow s \in L(S/G) \wedge s\sigma \in L(G) \wedge \sigma \in S(s)$. On the other hand, the *marked closed-loop behavior* is given by $L_m(S/G) = L(S/G) \cap L_S$. S is said to be nonblocking when $L(S/G) = \overline{L_m(S/G)}$. Now, the supervisory control problem can be presented as follows.

Problem 1 (*[SCP] Ramadge & Wonham, 1987*)**.** Given a plant G defined on Σ and a specification $E \subseteq \Sigma^*$ for G, defining a desired behavior $K = E \cap L_m(G)$, find a nonblocking supervisor S such that $L_m(S/G) \subseteq K$.

A language $K \subseteq \Sigma^*$ is said to be *controllable* with respect to (wrt) L when $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$. Controllability of a nonempty language $K \subseteq L_m(G)$ is a necessary and sufficient condition for the existence of a nonblocking supervisor S such that $L_m(S/G) = K$ (Ramadge & Wonham, 1987). In this case, S such that $L_m(S/G) = K$ may be implemented by any automaton V such that $K = L_m(V) \cap L_m(G)$ and $\overline{K} = L(V) \cap L(G)$. The control action of S is implemented by disabling eligible events in $L(G)$ that are not eligible in $L(V)$, after a