Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-7 (2015) 161–166

Fault-Tolerant Control of Discrete-Event Systems with Lower-Bound Specifications

Thomas Moor * **Klaus Werner Schmidt** **

* *Lehrstuhl für Regelungstechnik*
Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
(e-mail: lrt@fau.de)
** *Mechatronics Engineering Department*
Çankaya University, Ankara, Turkey
(e-mail: schmidt@cankaya.edu.tr)

Abstract: Fault-tolerant control addresses the control of dynamical systems such that they remain functional after the occurrence of a fault. To allow the controller to compensate for a fault, the system must exhibit certain redundancies. Alternatively, one may relax performance requirements for the closed-loop behaviour after the occurrence of a fault. To achieve fault tolerance for a hierarchical control architecture, a combination of both options appears to be advisable: on each individual level of the hierarchy, the controller may compensate the fault as far as possible, and then pass on responsibility to the next upper level. This approach, when further elaborated for discrete-event systems represented by formal languages, turns out to impose a hard lower-bound inclusion specification on the closed-loop behaviour. The present paper discusses the corresponding synthesis problem and presents a solution.

Keywords: Discrete-event systems, supervisory control, fault-tolerant control, hierarchical control.

1. INTRODUCTION

The aim of fault-tolerant control is the continuation of the system operation in case of a fault occurrence. Hereby, it is possible to make use of system redundancies or to allow a degradation of the system performance after encountering a fault. In this paper, the stated options are interpreted in the context of a hierarchical control architecture. We argue that, for each level of the hierarchy, a controller should use redundancies to compensate the effects of the fault whenever possible. When compensation becomes impossible, the fault should be made explicit for compensation by the next higher level, and so forth.

For discrete-event systems represented by formal languages, the described strategy can be formalised as a supervisory control problem with a hard lower-bound inclusion specification (such as the fault-free behaviour) and an upper-bound inclusion specification, that can be relaxed if otherwise no solution exists (in case of uncontrollable behaviour after a fault). This contrasts the well studied problem with a hard upper bound and the option to relax the lower bound. We note that the idea of relaxing the upper bound is solved by Lafortune and Chen (1990) for the prefix-closed case using the *infimal closed controllable superlanguage*. The authors also remark that for the general case of non-closed languages the infimum fails to be controllable. However, in the present paper, we are interested in non-closed languages for the specific situation of fault-tolerent control and propose a non-infimal but sensible solution for the problem at hand. Our approach retains regularity and we indicate how it can be implemented for finite automata representations.

Several approaches for fault-tolerance are proposed in the existing discrete-event systems literature, and we give selected references relevant to the present paper. Paoli and Lafortune (2005); Paoli et al. (2011) consider fault detection by a diagnoser and switching to a different supervisor for each fault before the desired system behaviour is violated. In (Wen et al., 2008) it is proposed that the system behaviour shall converge to the nominal system behaviour after a fault occurrence, whereas Kumar and Takai (2012) determine necessary and sufficient conditions for controller reconfiguration in the case of faults. Fault-accommodating models are used in Wittmann et al. (2013) to represent the fault and to develop a fault-hiding control architecture similar to the well established approach for continuous systems. The computation of supremal fault-tolerant sublanguages is suggested by Sülek and Schmidt (2013) for systems where certain events can not longer occur. Sülek and Schmidt (2014) propose a method for converging to a desired behaviour under fault. It is common to all of the above approaches that one needs to explicitly provide a formal representation of some desirable (but potentially degraded) behaviour that is to be achieved after a fault occurrence. In the present contribution, we propose a systematic way to derive this design parameter from the nominal specification and a fault-accommodating plant model. Motivated by hierarchical control architectures, our controller indicates any effects of the fault that it cannot compensate by a distinguished event.

The paper is organised as follows. In Section 2, we provide the relevant notation. Section 3 discusses the classical supervisory control method. Section 4 illustrates fault-accommodating models by example and points out limitations of this approach. The main idea and method of relaxing the upper-bound specification language for fault tolerance is developed in Section 5 and applied to an example in Section 6.

2. PRELIMINARIES AND NOTATION

Let Σ be a *finite alphabet*, i.e., a finite set of symbols $\sigma \in \Sigma$. The *Kleene-closure* Σ^* is the set of finite strings $s = \sigma_1\sigma_2\cdots\sigma_n$,

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.06.488

$n \in \mathbb{N}$, $\sigma_i \in \Sigma$, and the *empty string* $\epsilon \in \Sigma^*$, $\epsilon \notin \Sigma$. The length of a string $s \in \Sigma^*$ is denoted $|s| \in \mathbb{N}_0$, with $|\epsilon| = 0$. If, for two strings $s, r \in \Sigma^*$, there exists $t \in \Sigma^*$ such that $s = rt$, we say r is a *prefix* of s, and write $r \leq s$; if in addition $r \neq s$, we say r is a *strict prefix* of s and write $r < s$. The prefix of $s \in \Sigma^*$ with length $n \in \mathbb{N}_0$, $n \leq |s|$, is denoted $\mathrm{pre}_n\, s$. In particular, $\mathrm{pre}_0\, s = \epsilon$ and $\mathrm{pre}_{|s|}\, s = s$. If, for two strings $s, t \in \Sigma^*$, there exists $r \in \Sigma^*$ such that $s = rt$, we say t is a *suffix* of s. The suffix of a string $s \in \Sigma^*$ obtained by deleting the prefix of length n, $n \leq |s|$, is denoted $\mathrm{suf}_n\, s$; i.e., $s = (\mathrm{pre}_n\, s)(\mathrm{suf}_n\, s)$.

A *formal language* (or short a *language*) over Σ is a subset $L \subseteq \Sigma^*$. The *prefix* of a language $L \subseteq \Sigma^*$ is defined by $\mathrm{pre}\, L := \{r \in \Sigma^* \mid \exists\, s \in L : r \leq s\}$. A language L is *closed* if $L = \mathrm{pre}\, L$. A language K is *relatively closed w.r.t. L* if $K = (\mathrm{pre}\, K) \cap L$. The prefix operator distributes over arbitrary unions of languages. However, for the intersection of two languages L and H, we have $\mathrm{pre}\,(L \cap H) \subseteq (\mathrm{pre}\, L) \cap (\mathrm{pre}\, H)$. If equality holds, L and H are said to be *non-conflicting*.

For two languages $K,\ M \subseteq \Sigma^*$, K is said to *converge* to M, denoted by $M \Leftarrow K$, if there is a non-negative integer n such that for each $s \in K$, there exists an $i \leq n$ such that $\mathrm{suf}_i\, s \in M$. For two languages $N,\ K \subseteq \Sigma^*$, let $K/N := \{\, t \mid \exists\, s \in N : st \in K \,\}$. Then, K is said to *converge to M after N* if $M \Leftarrow K/N$ (also referred to as *conditional convergence*).

For the *observable events* $\Sigma_\mathrm{o} \subseteq \Sigma$, the *natural projection* $\mathrm{p_o}\colon \Sigma^* \to \Sigma_\mathrm{o}^*$ is defined iteratively: (1) let $\mathrm{p_o}\epsilon := \epsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $\mathrm{p_o}(s\sigma) := (\mathrm{p_o}s)\sigma$ if $\sigma \in \Sigma_\mathrm{o}$, or, if $\sigma \notin \Sigma_\mathrm{o}$, let $\mathrm{p_o}(s\sigma) := \mathrm{p_o}s$. The set-valued inverse $\mathrm{p_o^{-1}}$ of $\mathrm{p_o}$ is defined by $\mathrm{p_o^{-1}}(r) := \{s \in \Sigma^* \mid \mathrm{p_o}(s) = r\}$ for $r \in \Sigma_\mathrm{o}^*$. When applied to languages, the projection distributes over unions, and the inverse projection distributes over unions and intersections. The prefix commutes with projection and inverse projection.

Given two languages $L,\ K \subseteq \Sigma^*$, and a set of *uncontrollable events* $\Sigma_\mathrm{uc} \subseteq \Sigma$, we say K is *controllable w.r.t. L*, if $(\mathrm{pre}\, K)\Sigma_\mathrm{uc} \cap (\mathrm{pre}\, L) \subseteq \mathrm{pre}\, K$. Controllability is retained under union.

An *automaton* is a tuple $G = (Q, \Sigma, \delta, q_\mathrm{o}, Q_\mathrm{m})$, with *state set* Q, *initial state* $q_\mathrm{o} \in Q$, *marked states* $Q_\mathrm{m} \subseteq Q$, and the partial *transition function* $\delta\colon Q \times \Sigma \to Q$ with its common extension to the domain $Q \times \Sigma^*$. We denote the *generated language* $\mathrm{L}(G) := \{s \in \Sigma^* \mid \delta(q_\mathrm{o}, s)!\}$ and the *marked language* $\mathrm{L_m}(G) := \{s \in \Sigma^* \mid \delta(q_\mathrm{o}, s) \in Q_\mathrm{m}\}$.

Given a language, define the equivalence relation $[\equiv_L]$ on Σ^* by $s'\,[\equiv_L]\,s''$ if and only if $(\forall\, t \in \Sigma^*)[\, s't \in L \leftrightarrow s''t \in L\,]$. Then there exists a state minimal automata representation of L with the equivalence classs of $[\equiv_L]$ as state set. If the latter is finite, L is called *regular*.

3. SUPERVISORY CONTROL

We give a concise review of the basic control problem introduced by Ramadge and Wonham (1987, 1989), in a variation that turns out convenient for the present paper.

Given an alphabet Σ, consider a language $L \subseteq \Sigma^*$ to represent the *plant* with *local behaviour* $\mathrm{pre}\, L$ (set of all event sequences that can be generated as time passes) and *accepted behaviour L* (to indicate task completion).

For the purpose of control, the alphabet is partitioned in *controllable events* and *uncontrollable events*, i.e., $\Sigma = \Sigma_\mathrm{c} \,\dot\cup\, \Sigma_\mathrm{uc}$. In the original literature, the controller is represented as a *causal feedback map* $f\colon \mathrm{pre}\, L \to \Gamma$ with the set of *control-patterns* $\Gamma := \{\gamma \mid \Sigma_\mathrm{uc} \subseteq \gamma \subseteq \Sigma\}$. In this paper, we represent f as a language H and impose three requirements to obtain a setting equivalent to *non-blocking supervisory control* from the cited literature [1].

Definition 1. Given $\Sigma = \Sigma_\mathrm{c} \,\dot\cup\, \Sigma_\mathrm{uc}$, a language $H \subseteq \Sigma^*$ is an *admissible controller* for the plant $L \subseteq \Sigma^*$, if

(H0) H is closed,
(H1) H is controllable w.r.t. L, and
(H2) L and H are non-conflicting. □

For the commonly studied control problem, we consider a plant $L \subseteq \Sigma^*$ and lower- and upper *language inclusion specifications* $A \subseteq \Sigma^*$ and $E \subseteq \Sigma^*$, in order to ask for an admissible controller H, such that the *accepted closed-loop behaviour* $K = L \cap H$ satisfies the closed-loop requirement $A \subseteq K \subseteq E$.

Definition 2. Given $\Sigma = \Sigma_\mathrm{c} \,\dot\cup\, \Sigma_\mathrm{uc}$, the *control problem* under consideration is parametrised by languages $\emptyset \neq A \subseteq E \subseteq L \subseteq \Sigma^*$. A controller $H \subseteq \Sigma^*$ *solves* the problem, if it is admissible to the plant L and if in addition

(H3) $A \subseteq L \cap H \subseteq E$. □

Except for differences in notation, a constructive solution has been presented by Ramadge and Wonham (1987, 1989). It is based on two technical results. First, it is observed that a language K can be obtained as the closed-loop behaviour $K = L \cap H$ with some admissible controller H if and only if K is itself controllable and relatively closed w.r.t. L. Second, controllability and relative closedness are retained under arbitrary union. Thus, given the upper bound E, there exists a unique supremal subset that is controllable and relatively closed:

$$K^\uparrow := \sup\{K \subseteq E \mid K \text{ is cntrl. and rel. closed w.r.t. } L\}. \quad (1)$$

In particular, $H := \mathrm{pre}\, K^\uparrow$ is admissible and we have $L \cap H = K^\uparrow \subseteq E$. If $K^\uparrow$ happens to also satisfy the lower bound specification $A \subseteq K^\uparrow$, then H solves the control problem. If, on the other hand, $K^\uparrow$ does not satisfy the inclusion $A \subseteq K^\uparrow$, then neither does any other achievable closed-loop behaviour and, thus, the control problem has no solution. If the parameters $A \subseteq E \subseteq L$ are regular, then so is $K^\uparrow$, and an automaton representation can be computed in polynomial time [2]. Thus, for practical problems, one can first compute a representation of $K^\uparrow$ and then test for $A \subseteq K^\uparrow$.

4. EXAMPLE

We provide a simple example to illustrate the well-known results presented so far and we utilise the example to outline how supervisory control is applied to hierarchical control architectures and how one may address fault tolerance in this context.

Consider a processing machine that interacts with its environment by receiving a workpiece, processing the workpiece, returning the workpiece, and so forth. At a suitable level of abstraction, the machine is represented by the automaton defined in Fig. 1, referring to the events g (get a workpiece), a (use tool A for processing), b (use tool B for processing), p (progress increment), d (processing complete), and x (workpiece exits

[1] In the original literature, the plant is represented by an automaton, and, thus, can itself be blocking. When using a single language to represent the plant, blocking can be modelled by a distinguished event.

[2] More precisely, the complexity is $O(n^2m^2)$, where n and m denote the state counts of automata representations of L and E, respectively.