ARTICLE IN PRESS

JID: JARAP [m5G;May 12, 2016;7:44]

Annual Reviews in Control 000 (2016) 1–11

Contents lists available at ScienceDirect

Annual Reviews in Control

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

A discussion of fault-tolerant supervisory control in terms of formal languages

Thomas Moor

Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

ARTICLE INFO

Article history:
Received 29 October 2015
Revised 20 January 2016
Accepted 4 February 2016
Available online xxx

Keywords:
Discrete-event systems
Supervisory control
Fault-tolerant control
Passive fault-tolerant control
Active fault-tolerant control
Post-fault recovery
Fault-hiding approach

ABSTRACT

A system is *fault tolerant* if it remains functional after the occurrence of a fault. Given a plant subject to a fault, *fault-tolerant control* requires the controller to form a fault-tolerant closed-loop system. For the systematic design of a fault-tolerant controller, typical input data consists of the plant dynamics including the effect of the faults under consideration and a formal performance requirement with a possible allowance for degraded performance after the fault. For its obvious practical relevance, the synthesis of fault-tolerant controllers has received extensive attention in the literature, however, with a particular focus on continuous-variable systems. The present paper addresses discrete-event systems and provides an overview on fault-tolerant supervisory control. The discussion is held in terms of formal languages to uniformly present approaches to passive fault-tolerance, active fault-tolerance, post-fault recovery and fault hiding.

© 2016 The Authors. Published by Elsevier Ltd on behalf of International Federation of Automatic Control.
This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

A fault is considered a sudden change in the behaviour of a system with potentially undesired consequences. In technical applications, the overall effect of a single faulty component can range from degraded performance up to total breakdown, including environmental damage or human operator injury. By a *fault-tolerant design* one seeks to avoid such negative consequences in a systematic manner and up to a prescribed degree. Here, a common approach is to relate the reliability of individual components and the dependencies among different components with the overall functionality regarding safety and performance; see e.g. Dubrova (2013) for an introduction to fault-tolerant design from this perspective.

When it comes to control, the system consists of a plant and a controller where the latter is interpreted as a degree of freedom in the design of the overall closed-loop behaviour. Assuming that the plant is subject to a fault, one requires the controller to compensate the fault to some degree in order to maintain an operational closed loop with a well defined overall performance that complies to relevant safety requirements. Such a controller is termed *fault tolerant*, with *fault-tolerant control* as a particular approach to fault-tolerant design. For its obvious practical relevance, fault-tolerant control has received extensive attention in the literature; see e.g. Blanke, Kinnaert, Lunze, Staroswiecki, and Schröder (2006) for a comprehensive study.

Given the *nominal plant behaviour* in the absence of the fault and the *degraded plant behaviour* after the occurrence of the fault, the literature proposes two alternative strategies to achieve a fault-tolerant closed-loop system. First, one may design a single controller that can handle both plant models satisfactory. This is referred to as *passive fault-tolerant control* and is closely related to robust control. In contrast to plain robust control, however, attention needs to be paid to the transient behaviour when the fault occurs. Moreover, depending on the system classes under consideration, passive fault-tolerant control may impose unacceptable limitations on the nominal closed-loop behaviour. As a second strategy, one may refer to methods related to adaptive control and design one controller for the nominal plant, one controller for the degraded plant and a diagnoser to detect the fault. The latter is used to activate the appropriate controller. This strategy is referred to as *active fault-tolerant control*. In contrast to the common setting in adaptive control, the particular challenge again is the switching, now with three modes of operation: (a) no fault, (b) fault has occurred but is not yet diagnosed, and (c) fault present and diagnosed. In particular during (b) the degraded plant is under nominal control and may fail to satisfy even elementary requirements like stability.

E-mail address: lrt@fau.de

http://dx.doi.org/10.1016/j.arcontrol.2016.04.001
1367-5788/© 2016 The Authors. Published by Elsevier Ltd on behalf of International Federation of Automatic Control. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

A comparative study of active and passive fault-tolerant control is given by Jiang and Yu (2012).

As a further variation of active fault-tolerant control, a so-called *reconfiguration block* can be used to adapt the nominal controller to the faulty plant by hiding the effect of the fault. To further illustrate the *fault-hiding approach*, consider the situation of a sensor failure. Here, one can implement an observer with the reconfiguration block that, together with the faulty plant, mimics the nominal plant behaviour. With this approach, the nominal controller remains active even when the fault occurs and, hence, the closed-loop performance may benefit from advanced tuning strategies used for the nominal case. Virtual sensors and likewise virtual actuators are discussed by Blanke et al. (2006), for more general forms of fault-hiding see Steffen (2005) and Richter (2011).

The references provided so far focus attention on continuous-signal plant models represented by ordinary differential equations. In contrast, the present paper discusses the synthesis of fault-tolerant control for discrete-event systems that are adequately representable by regular languages. As a general framework for the control of this system class we refer to supervisory control as proposed by Ramadge and Wonham (1987,1989) and demonstrate in a concise and homogeneous notation how the above strategies for fault tolerance can be applied.

A preliminary observation for the system class under consideration is that passive fault-tolerant control plays a special role: since discrete-event systems model sudden changes of behaviour seamlessly, any switching scheme introduced to achieve fault-tolerance can be alternatively interpreted as passive fault-tolerant control. Moreover, an ordinary event can be used to represent the occurrence of the fault. Thus, a *fault-accommodating model* that summarises the nominal plant behaviour and the degraded plant behaviour still belongs to the same system class as the nominal plant and solutions to the synthesis problem for passive fault-tolerant control can be obtained by the very same established procedures known from the nominal synthesis problem. We refer to this perspective as *naive fault-tolerant control*. In general, we expect that alternative control architectures motivated by additional control objectives or specific design strategies also comply with the naive setting.

In this paper, we discuss approaches to the synthesis of fault-tolerant supervisory control provided by the literature. We make use of a homogeneous notation in order to demonstrate how the approaches relate to the naive approach as a common technical base. Observing applicable constraints and conducting the discussion up to a relevant level of detail, we focus attention to active fault-tolerant control (Paoli, Sartini, & Lafortune, 2008,2011) and fault hiding (Wittmann, 2014; Wittmann, Richter, & Moor, 2013) for specific control architectures, as well as variants of post-fault recovery (Sülek & Schmidt, 2014; Wen, Kumar, & Huang, 2008a, 2014; Wen, Kumar, Huang, & Liu, 2008b) for additional control objectives.

To complement the references further discussed in the body of this paper, we account for related work, that does not fit the language based framework. Park and Lim (1998) propose a notion of fault tolerance in terms of reachability of marked states. The discussion includes a characterisation of the existence of a fault tolerant controller that in addition exhibits a robustness property. Rohloff (2005) addresses the specific situation of faulty sensors and proposes to represent the effect of a fault by an according variation of the projection operator chosen for observations. The cited reference gives detailed account on modelling and provides a procedure to test for fault-tolerance, as well as an outline of possible synthesis procedures. Girault and Rutten (2009) adapt methods from supervisory control for the synthesis of fault-tolerant programs. A particular focus here is on the systematic generation of models for certain classes of faults and for certain classes of components subject to a fault. The cited work uses labelled transitions systems with guards and actions as a modelling framework. Nke and Lunze (2011a,2011b) discuss fault-tolerant control for automata with inputs and outputs. The contributions include a systematic approach to model sensor and actuator faults as well as a synthesis procedure for reconfiguration to achieve fault tolerance w.r.t. prescribed performance objectives. Sülek and Schmidt (2013) consider faults with the effect that certain events can no longer occur. The discussion includes a synthesis procedure to achieve fault tolerance in the closed-loop configuration. Moor and Schmidt (2015) address fault-tolerance in a hierarchical control architecture and discuss the option to pass on undesired post-fault behaviour for compensation further up in the hierarchy.

The paper is organised as follows. A language based framework for the control of discrete-event systems is introduced in Sections 2 and 3, as a variation of *supervisory control under partial observation* originally proposed by Lin and Wonham (1988) and referring to Ramadge and Wonham (1987). As a further development of Wittmann, Richter, and Moor (2012), Section 4 elaborates the naive approach to fault-tolerant control to motivate closed-loop requirements relevant for fault tolerance. The subsequent discussion addresses active fault-tolerant control in Section 5, post-fault recovery in Section 6 and the fault-hiding approach in Section 7. We conclude with a summary. The paper is an extended transcript of a plenary talk held at the *5th International Workshop on Dependable Control of Discrete-Event Systems (5th IFAC DCDS 2015), Mexico*; see Moor (2015) for the corresponding conference contribution.

2. Preliminaries and notation

This section provides notation and elementary facts on formal languages as relevant for the present paper. For a general introduction see Hopcroft and Ullman (1979), and, for a discrete-event systems perspective, Cassandras and Lafortune (2008).

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 $s = \sigma_1\sigma_2\ldots\sigma_n$, $n \in \mathbb{N}$, $\sigma_i \in \Sigma$, 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 $|\epsilon| = 0$. Given two strings $s = \sigma_1\sigma_2\ldots\sigma_n \in \Sigma^*$ and $t = \tau_1\tau_2\ldots\tau_m \in \Sigma^*$, the *concatenation* is defined $st := \sigma_1\sigma_2\ldots\sigma_n\tau_1\tau_2\ldots\tau_m \in \Sigma^*$ with $s\epsilon = s = \epsilon s$. 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 $r \le s$; 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 $n \in \mathbb{N}_0$, $n \le |s|$, 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, $n \le |s|$, is denoted $\mathrm{suf}_n\, s$; i.e., $s = (\mathrm{pre}_n\, s)(\mathrm{suf}_n\, s)$.

A $*$*-language* (or short a *language*) over Σ is a subset $L \subseteq \Sigma^*$. Given a language $L \subseteq \Sigma^*$, the equivalence relation $[\equiv_L]$ on Σ^* is defined by $s' [\equiv_L] s''$ if and only if $(\forall t \in \Sigma^*)[\, s't \in L \leftrightarrow s''t \in L\,]$. The language L is *regular* if $[\equiv_L]$ has only finitely many equivalence classes, and, thus is accepted by a finite automaton.

The *prefix* of a language $L \subseteq \Sigma^*$ is defined by $\mathrm{pre}\, L := \{r \in \Sigma^* \,|\, \exists s \in L : r \le s\}$. The prefix operator distributes over arbitrary unions of languages. However, for the intersection of two languages L and K, we have $\mathrm{pre}\,(L \cap K) \subseteq (\mathrm{pre}\, L) \cap (\mathrm{pre}\, K)$. If equality holds, L and K are said to be *non-conflicting*. This is trivially the case for $K \subseteq L$. The prefix operator is also referred to as the *prefix-closure*, and, a language L is *prefix-closed* (or short *closed*) if $L = \mathrm{pre}\, L$. A language K is *relatively prefix-closed w.r.t. L* (or short *relatively closed w.r.t. L*), if $K = (\mathrm{pre}\, K) \cap L$. The intersection $(\mathrm{pre}\, K) \cap L$ is always relatively closed w.r.t. L. If a language K is relatively closed w.r.t. a closed language, then K itself is closed.

For two languages $L, M \subseteq \Sigma^*$, the *concatenation* is defined $LM := \{st \,|\, s \in L,\ t \in M\}$. The concatenation of closed languages is

Download English Version:

https://daneshyari.com/en/article/7107909

Download Persian Version:

https://daneshyari.com/article/7107909

Daneshyari.com