Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-17 (2016) 200–205

Robust Supervisory Control of a Spacecraft Propulsion System

Farid Yari, Shahin Hashtrudi-Zad, Siamak Tafazoli

Department of Electrical and Computer Engineering, Concordia University, Montréal, QC, Canada H3G 1M8
f_yari@encs.concordia.ca, shz@ece.concordia.ca, stafazoli@videotron.ca

Abstract: In this paper the theory of supervisory control of discrete-event systems is used to develop command sequences for turning on and off a spacecraft propulsion subsystem. The subsystem considered is a simplified version of the Propulsion Module Subsystem of the Cassini spacecraft. The supervisor controls the system in such a way that the design specifications are satisfied in both normal and faulty modes of operation. The study shows that to meet the specifications of both modes, the supervisor has to be a "robust" supervisor, and that a conventional (non-robust) supervisor could lead to engine getting stuck in shutdown state.

Keywords: Supervisory control, Spacecraft propulsion, Fault recovery, Discrete-event systems, Robust control.

1. INTRODUCTION

This paper is concerned with the design of supervisory control system for spacecraft propulsion system. Specifically we seek to design a controller (supervisor) that issues appropriate command sequences to turn on and off the engines of a simplified spacecraft propulsion system. The objective is to enhance spacecraft autonomy. The subject of spacecraft autonomy has been studied by many researchers. Notably, the diagnosis and recovery system, Livingstone, has been designed based on probabilistic discrete models [Muscettola et al. (1998)]. Livingstone was tested on Deep Space 1 (1998) and its successor, Livingstone 2, was used on Earth Observing 1 (2000). In another approach, the methods of formal verification have been used to verify if a given control logic satisfies the design specifications (see, e.g. [Pekala et al. (2008)] and [Bensalem et al. (2010)]).

In this paper we apply the Ramadge-Wonham theory of supervisory control ([Ramadge and Wonham (1987); Wonham (2014)]) to systematically design controller (supervisor) from the design specifications based on deterministic discrete-event models. The resulting supervisor (which will be in the form of an automaton) is guaranteed to satisfy the specifications. We study a simplified version of the Propulsion Module Subsystem (PMS) of the Cassini spacecraft. Our study shows that the application of conventional (non-robust) supervisory control [Ramadge and Wonham (1987)] does not result in the appropriate command sequences. However, an extension of this approach to the Robust Supervisory Control ([Saboori and Hashtrudi Zad (2006)]) produces the desired solution.

The concept of robust control arises in control theory in dealing with modeling uncertainties or model changes. Several approaches have been explored for robust supervisory control of discrete-event systems (DES). In [Lin (1993)], the plant model belongs to a finite family of DES models G_1,...,G_n, and the objective is to design a supervisor such that all plant models under supervision satisfy a common design specification. [Bourdon et al. (2005)] extends the results of [Lin (1993)] to the case involving separate design specifications for each plant model and considers the nonblocking property. The nonblocking property is guaranteed through the (sufficient condition of) *nonconflicting* property. [Saboori and Hashtrudi Zad (2006)] extends the results of [Bourdon et al. (2005)] to the case of control under partial observation. Furthermore, [Saboori and Hashtrudi Zad (2006)] replaces the nonconflicting property with the $\boldsymbol{G}_i$*-nonblocking* property to obtain a set of necessary and sufficient conditions for the solvability of the robust control problem.

Robust control is used to deal with model changes. For instance, in some fault recovery problems, the plant starts in normal mode and may enter one of several faulty modes. Each mode can have its own design specification and set of marked states. This control and fault recovery problem can be solved as a robust control problem [Saboori and Hashtrudi Zad (2005)].

[Yari and Hashtrudi Zad (2016)] develops automaton-based computational procedures for the design of robust supervisors of [Saboori and Hashtrudi Zad (2006)]. The procedures have been implemented in MATLAB environment using Discrete Event Control Kit (DECK) ([DECK (2013)]). In this paper, these procedures are used to solve the control and fault recovery problem for the simplified spacecraft propulsion system. Our study shows that to arrive at an appropriate control logic satisfying all design specifications, the supervisor has to be a robust supervisor. Furthermore, the features of the system studied here that necessitate the use of robust supervisor seem to be present in many other aerospace systems as well.

⋆ This work was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) under grant RGPIN-227688-2011.

10.1016/j.ifacol.2016.09.035

This paper is organized as follows. Sec. 2 reviews some preliminaries of supervisory control. Sec. 3 formulates the supervisory control and fault recovery problem. Sec. 4 discusses the simplified propulsion system and the design of supervisor. Sec. 5 further analyzes the results and provides comments on the methodology and its relation to other approaches. The paper is concluded in Sec. 6.

2. PRELIMINARIES

In this section, we briefly review the supervisory control of DES [Wonham (2014); Cassandras and Lafortune (2008)].

Languages and Automata. A finite set of symbols, Σ, is called an *alphabet*. A sequence of symbols $\sigma_1\sigma_2...\sigma_n$ (with $\sigma_i \in \Sigma$) is called a *word* or *sequence*. A sequence with no symbols is called the *empty sequence* and denoted by ϵ. Σ^* denotes the set of all finite sequences. Any subset of Σ^* is called a *language*. A deterministic automaton $\boldsymbol{G} = (X, \Sigma, \eta, x_0, X_m)$ is a model for a discrete-event system with X the state set, Σ the finite set of events, $\eta : X \times \Sigma \longrightarrow X$ the partial transition function, x_0 the initial state and $X_m \subseteq X$ the set of *marked states*. The set of sequences that can be generated by automaton $\boldsymbol{G}$ (from initial state x_0) is called the *closed behavior* of $\boldsymbol{G}$: $L(\boldsymbol{G}) = \{s \in \Sigma^* \mid \eta(x_0, s) \text{ is defined}\}$. The subset of $L(\boldsymbol{G})$ consisting of sequences that end in a marked state is referred to as the *marked behavior* of $\boldsymbol{G}$: $L_m(\boldsymbol{G}) = \{s \in L(\boldsymbol{G}) \mid \eta(x_0, s) \in X_m\}$.

A state $x \in X$ is called *reachable* if there exists a sequence $s \in L(\boldsymbol{G})$ from x_0 to x. Also, a state x is called *coreachable* if there is a sequence, say s, from x to some marked state. A DES $\boldsymbol{G}$ is *nonblocking* if every reachable state of $\boldsymbol{G}$ is coreachable. A nonblocking automaton is free of deadlocks and livelocks. An automaton is *trim* if all of its states are reachable and coreachable. The product of automata $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ is denoted by $\boldsymbol{G}_1 \times \boldsymbol{G}_2$ and generates $L(\boldsymbol{G}_1) \cap L(\boldsymbol{G}_2)$ and marks $L_m(\boldsymbol{G}_1) \cap L_m(\boldsymbol{G}_2)$. The synchronous product (parallel composition) of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ is denoted by $sync(\boldsymbol{G}_1, \boldsymbol{G}_2)$ and can be used to model the joint operation of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$.

Supervisory Control. Consider a plant modeled by a DES $\boldsymbol{G}$. Suppose a language $E \subseteq L_m(\boldsymbol{G})$ represents the *legal marked behavior*; i.e., the marked sequences that satisfy the design specifications. In the supervisory control theory, a supervisor S is to be designed to restrict the behavior of the plant $\boldsymbol{G}$ to the legal behavior E and to ensure that the resulting system is nonblocking (Fig. 1). It is assumed that the plant event set Σ can be partitioned

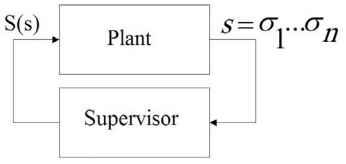

Fig. 1. Block Diagram of Supervisory control.

into two disjoint subsets, Σ_c the set of *controllable events*, and Σ_{uc} the set of *uncontrollable events*. The supervisor is allowed to disable the controllable events only. A supervisory control for DES $\boldsymbol{G}$ can be designed as a map $S : \Sigma^* \to \Gamma_\Sigma$ where $\Gamma_\Sigma = \{\gamma \in Pwr(\Sigma) \mid \Sigma_{uc} \subseteq \gamma\}$ denotes the set of all control patterns in Σ. In this context, for $s \in \Sigma^*$ (generated in $\boldsymbol{G}$), $S(s)$ is the set of events enabled by the supervisor. Let $S/\boldsymbol{G}$ represent the plant $\boldsymbol{G}$ under the supervision of S (or the closed-loop system). Thus the **Supervisory Control Problem (SCP)** is to find a supervisor S such that (1) $L_m(S/\boldsymbol{G}) \subseteq E$, and (2) $S/\boldsymbol{G}$ is nonblocking. ((1) and (2) imply that all sequences in $L(S/\boldsymbol{G})$ are legal.)

Next we consider the robust supervisory control problem which can be regarded as an extension of SCP.

Problem 1. **Robust Nonblocking Supervisory Control Problem (RNSCP)** [Bourdon et al. (2005); Saboori and Hashtrudi Zad (2006)]. Consider the set of plant models $\mathcal{G} = \{\boldsymbol{G}_1, ..., \boldsymbol{G}_n\}$ with $\boldsymbol{G}_i = (X_i, \Sigma_i, \eta_i, x_{0,i}, X_{m,i})$ where $i \in I = \{1, 2, ..., n\}$. Let the controllable and uncontrollable sets of events in $\boldsymbol{G}_i$ be denoted by $\Sigma_{c,i}$ and $\Sigma_{uc,i}$ respectively. It is assumed that all plant models agree on the controllability of events, that is $\Sigma_{c,i} \cap \Sigma_{uc,j} = \emptyset$ for all $i, j \in I$. Furthermore, suppose language K_i denotes the design specification for plant model $\boldsymbol{G}_i$ and hence $E_i = K_i \cap L_m(\boldsymbol{G}_i)$ denotes the corresponding legal marked behavior. Design a supervisor S such that (1) $L_m(S/\boldsymbol{G}_i) \subseteq E_i$ and (2) $S/\boldsymbol{G}_i$ is nonblocking, for all $i \in I$.

Thus in RNSCP we seek to find a supervisor S such that (for every i) $S/\boldsymbol{G}_i$ is nonblocking and satisfies its design specification.

Let $\Sigma = \bigcup_{i \in I} \Sigma_i$ and define E as

$$E = \bigcap_{i \in I} (E_i \cup (\Sigma^* - L_m(\boldsymbol{G}_i))) \cap (\bigcup_{i \in I} L_m(\boldsymbol{G}_i)) \quad (1)$$

The solutions of RNSCP can be characterized in terms of relative-closed, controllable and $\boldsymbol{G}$-nonblocking sublanguages of E [Saboori and Hashtrudi Zad (2006)]. Let $RCNb(E, \mathcal{G})$ denote the set of relative-closed, controllable and $\boldsymbol{G}_i$-nonblocking sublanguages of E (for all $\boldsymbol{G}_i \in \mathcal{G}$). This set is nonempty and has a supremal element denoted by $E^\uparrow = SupRCNb(E, \mathcal{G})$. Then $E^\uparrow$ is the maximally permissive solution of RNSCP. A trim automaton $\mathbf{S}$ that marks $E^\uparrow$ can be regarded as an implementation of a maximally permissive supervisor with $\mathbf{S} \times \boldsymbol{G}_i$ modeling the system under supervision $\mathbf{S}/\boldsymbol{G}_i$. [Yari and Hashtrudi Zad (2016)] provides a computational procedure to find an automaton to mark $E^\uparrow$. This algorithm has been implemented in MATLAB environment using DECK [DECK (2013)].

3. SUPERVISORY CONTROL AND FAULT RECOVERY PROBLEM

In this section we explain how a problem of control and fault recovery can be cast as a robust control problem. Consider a DES plant $\boldsymbol{G}$ that starts in normal mode N. The plant may enter a set of permanent failure modes. For simplicity we assume a single failure mode F (Fig. 2). Let

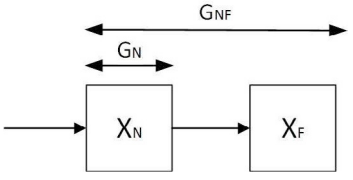

Fig. 2. DES $\boldsymbol{G}$ with a single failure mode.