Information Sciences 235 (2013) 259–279

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Robustness of deadlock control for a class of Petri nets with unreliable resources

G.Y. Liu [a], Z.W. Li [a,*], Kamel Barkaoui [b], Abdulrahman M. Al-Ahmari [c]

[a] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, PR China
[b] Cedric Lab and Computer Science Department, Conservatoire National des Arts et Métiers, Paris 75141, France
[c] FARCAMT, Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia

ARTICLE INFO

Article history:
Received 5 January 2012
Received in revised form 18 August 2012
Accepted 13 January 2013
Available online 29 January 2013

Keywords:
Petri net
Deadlock
Automated manufacturing system
Supervisor
Siphon

ABSTRACT

A variety of deadlock control policies based on Petri nets have been proposed for automated manufacturing systems (AMSs). Most of them prevent deadlocks by adding monitors for emptiable siphons that, without an appropriate control policy, can cause deadlocks, where the resources in a system under consideration are assumed to be reliable. When resources are unreliable, it is infeasible or impossible to apply the existing control strategies. For systems of simple sequential processes with resources (S^3PR), this paper bridges the gap between a divide-and-conquer deadlock control strategy and its application to real-world systems with unreliable resources. Recovery subnets and monitors are designed for unreliable resources and strict minimal siphons that may be emptied, respectively. Normal and inhibitor arcs are used to connect monitors with recovery subnets in case of necessity. Then reanalysis of the original Petri net is avoided and a robust liveness-enforcing supervisor is derived. Examples are presented to illustrate the proposed methodology.

1. Introduction

An automated manufacturing system (AMS) is composed of limited resources and can process different kinds of parts based on resource sharing and a specified sequence of operations. On the one hand, resource sharing may lead to deadlocks in which the global or local system is crippled [11,28,30,32]. On the other hand, an AMS often suffers from unreliable resource failures that may also cause processes to halt [45,13]. Thus, it is a necessary requirement to develop an effective and robust deadlock control policy to ensure that deadlocks cannot occur even if some resources in a system break down.

As a graphical and mathematical tool, Petri nets provide a uniform paradigm for modeling and formal analysis of AMS. They are well suitable to describe AMS' behavior and characteristics such as concurrency, conflict, and causal dependency. They can be used to reveal behavioral properties such as liveness and boundedness [9].

Over the past three decades, a variety of deadlock control policies based on Petri nets have been proposed for AMS [38,47,10,2,7,35,21,22,37,36,40–43,24,25,29,31,23,44,4,6]. From a technical perspective, most of the control policies resolving deadlocks are developed via state space analysis or structural analysis of Petri nets. Deadlock control policies based on the former can approach the maximally permissive behavior, but may encounter highly computational complexity and a state explosion problem. Recently, more effective methods based on reachability graph analysis have been proposed

* Corresponding author.
E-mail addresses: lg_2005@163.com (G.Y. Liu), zhwli@xidian.edu.cn (Z.W. Li), kamel.barkaoui@cnam.fr (K. Barkaoui), alahmari@ksu.edu.sa (A.M. Al-Ahmari).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.01.003

[37,39,36,4,5,1]. Deadlock control policies based on structural analysis can avoid the state explosion problem successfully, but always forbid some legal states. The studies in [10,24,35,3,46,33] deal with deadlocks based on siphons. However, all of them prevent deadlocks on condition that the resources in the considered systems are assumed to be reliable.

There is a lack of research in Petri nets regarding the impacts of unreliable resources on AMS under the supervisory control of deadlocks. In fact, resource failures are a common problem in real-world systems, which pose challenges in supervisory control of discrete event systems including AMS. In case of resource failures, the existing deadlock control policies are always no longer in force and deadlocks in the disturbed system may be caused. Therefore, reanalysis of the disturbed system is usually necessary. Robustness analysis provides an alternative way to determine whether the operation of a disturbed system or a part of it can still be maintained in case of resource failures. As far as the authors know, no much work is found on robust supervision of AMS based on Petri nets.

Hsieh develops a variety of methods to determine the feasibility of production with a set of resource failures modeled as the extraction of tokens from a Petri net [13–19]. This researcher has studied the robustness of several subclasses of Petri nets, including controlled production Petri nets (CPPNs) [13], controlled assembly Petri nets (CAPNs) [14], controlled assembly/disassembly Petri nets (CADPNs) [15], controlled assembly Petri nets with alternative routes (CAPN-ARs) [16], collaborative Petri nets (CPNs) [17], and non-ordinary controlled flexible assembly Petri nets with uncertainties (NCFAPNUs) [19]. In these papers, liveness conditions and robustness analysis of the nets are based on the concepts of token flow paths and minimal resource requirements (MRRs). His work reports fault tolerant conditions and proposes a structural decomposition method to test the feasibility of production routes. However, all these methods are not intuitive to the Petri net models. In this paper, we try to enforce liveness and robustness via a supervisor by adding monitors and recovery subnets. This implies that both a plant and its supervisor are unified in a Petri net formalism.

An interesting issue is how to make the existing deadlock control policies possess a desirable robust property to cope with resource failures. Specifically, the desirable robustness is a system property to keep a controlled system live as some resources break down. For an uncontrolled system of simple sequential processes with resources (S^3PR), monitors and recovery subnets are designed for strict minimal siphons that may be emptied and unreliable resources, respectively. Monitors, complementary places of monitors, and recovery subnets are connected by normal arcs in case of necessity. By adding monitors for siphons, deadlocks in original Petri nets can be controlled and by adding recovery subnets, complementary places of monitors, and necessary arcs for unreliable parts, deadlocks in the disturbed systems can be controlled. We also find that complementary places of monitors and related arcs can be replaced by inhibitor arcs. Note that monitors are added by the divide-and-conquer strategy [26]. In order to analyze the robustness of a supervisor, we propose a new pause state called a waiting-for-repair state, which is different from a deadlock state.

Based on the divide-and-conquer strategy [26], the supervisor designed for S^3PR by the proposed method has the following characteristics: (1) it can prevent deadlocks for a plant model when all resources work normally; (2) deadlocks are prevented even if some resources fail to work and are removed to repair at any time; and (3) waiting-for-repair states disappear after the repaired resources are returned. Compared with the existing deadlock control policies [10,26], this work can deal with more complex cases in which the reliability of resources is fully assumed. Hence, the proposed method is more practical to real-world manufacturing systems. If all the unreliable resources work normally, the proposed method works as the method in [26].

The rest of this paper is organized as follows. Section 2 briefly reviews the basics of Petri nets as well as S^3PR, with which the paper deals. Section 3 motivates this study via an example. A design method of a robust liveness-enforcing supervisor for an S^3PR net with unreliable resources is proposed in Section 4. Examples are given in Section 5 to demonstrate the proposed method. Section 6 discusses a number of open problems. Finally, Section 7 concludes this work and suggests directions for future research.

2. Preliminaries

2.1. Basics of Petri nets

A Petri net [34] is a four-tuple $N = (P,T,F,W)$ where P and T are finite and nonempty sets. P is a set of places and T is a set of transitions with $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is called a flow relation of the net, represented by arcs with arrows from places to transitions or from transitions to places. $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is a mapping that assigns a weight to an arc: $W(f) > 0$ if $f \in F$ and $W(f) = 0$ otherwise, where $\mathbb{N} = \{0, 1, 2, \ldots\}$. $N = (P,T,F,W)$ is said to be ordinary, denoted as $N = (P,T,F)$, if $\forall f \in F$, $W(f) = 1$. A marking M of a Petri net N is a mapping $M : P \rightarrow \mathbb{N}$. The number of tokens in place p is denoted by $M(p)$. We use $\sum_{p \in P} M(p)p$ to denote vector M for economy of space. Place p is said to be marked at M if $M(p) > 0$. A subset $S \subseteq P$ is marked at M if $\exists p \in S$, $M(p) > 0$. $M(S) = \sum_{p \in S} M(p)$ denotes the sum of tokens in all places in S. S is said to be emptied at M if $M(S) = 0$. (N,M_0) is called a marked net and M_0 is called an initial marking. A pair of a place p and a transition t is called a self-loop if p is both an input and output place of t. A net $N = (P,T,F,W)$ is pure (self-loop free) if $\forall x,y \in P \cup T$, $W(x,y) > 0$ implies $W(y,x) = 0$. A pure net structure can be fully described by its incidence matrix denoted by $[N]$. It is a $|P| \times |T|$ integer matrix with $[N](p,t) = W(t,p) - W(p,t)$.

Given a node $x \in P \cup T$, ${}^\bullet x = \{y \in P \cup T \mid (y,x) \in F\}$ is called the preset of x, while $x^\bullet = \{y \in P \cup T \mid (x,y) \in F\}$ is called its postset. We can extend this notation to a set of nodes as follows: given $S \subseteq P \cup T$, ${}^\bullet S = \cup_{x \in S} {}^\bullet x$ and $S^\bullet = \cup_{x \in S} x^\bullet$. A net $N = (P,T,F)$ is called