Computers & Industrial Engineering 56 (2009) 53–62

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

On systematic methods to remove redundant monitors from liveness-enforcing net supervisors ☆

ZhiWu Li *, HeSuan Hu

School of Electro-mechanical Engineering, Xidian University, No. 2 South Taibai Road, Xi'an, Shaanxi 710071, China

ARTICLE INFO

Article history:
Received 14 June 2006
Received in revised form 16 January 2008
Accepted 26 March 2008
Available online 4 April 2008

Keywords:
Petri net
Flexible manufacturing system
Monitor
Implicit place
Liveness-restrictive place
Liveness-enforcing supervisor

ABSTRACT

Petri nets based deadlock prevention for flexible manufacturing systems has received much attention over the past decade, primarily due to the seminal work of Ezpeleta et al. in 1995. A Petri net based deadlock prevention mechanism is usually implemented by adding monitors or control places to a plant Petri net model such that liveness can be enforced. The significance of this methodology lies in that both a plant model and its supervisor are in a same formalism-Petri nets. Due to the inherent complexity of Petri nets, in theory, the number of additional monitors that have to been added to achieve liveness-enforcement purpose for an uncontrolled plant model is exponential with respect to the size of the model. This paper first proposes a systematic method to minimize the number of additional monitors in a liveness-enforcing Petri net supervisor such that the resultant net system has the same permissive behavior while liveness can still be preserved. Furthermore, for the liveness-enforcing Petri net supervisors of flexible manufacturing systems, which have some particular property, an algorithm is developed such that more permissive liveness-enforcing Petri net supervisors can be obtained after liveness-restrictive monitor removal. Compared with the existing techniques of eliminating redundant monitors in the literature, the complete state enumeration of a supervisor is avoided, which implies the high computational efficiency of the methods in this paper. Flexible manufacturing examples are used to demonstrate the proposed approaches.

1. Introduction

In a flexible manufacturing system (FMS), different types of raw parts enter the system at discrete points of time and are processed concurrently, sharing a limited number of resources such as machine tools, AGVs, robots, buffers, and fixtures. In such a system, every raw part follows a preestablished production sequence through the set of system resources. These production sequences are executed concurrently and therefore they have to compete for the set of shared resources. This competition for shared resources can cause deadlocks that are a highly undesirable situation, where each of a set of two or more jobs keeps waiting indefinitely for the other jobs in the set to release resources (Viswanadham, Narahari, & Johnson, 1990).

Deadlocks and related blocking phenomena often cause unnecessary costs. Therefore, it is a necessary requirement to develop a way to make sure that deadlocks will never occur in a system. Modeling an FMS with Petri nets (Murata, 1989;Zhou & Venkatesh, 1998) is such a way of dealing with deadlock problems. Three major strategies using Petri net techniques to cope with deadlocks in FMS are deadlock detection and recovery (Kumaran, Chang, Chao, & Wysk, 1994;Wysk, Yang, & Joshi, 1994), deadlock avoidance (Abdallah & ElMaraghy, 1998;Banaszak & Krogh, 1990;Barkaoui & Abdallah, 1994;Ezpeleta, Tricas, García-Vallés, & Colom, 2002;Ezpeleta & Recalde, 2004;Hsien & Chang, 1994;Park & Reveliotis, 2001;Viswanadham et al., 1990), and deadlock prevention that is a well-defined problem in discrete event systems (DES). Petri nets have been used widely to describe, analyze, and control FMS (Zhou & Venkatesh, 1998), as they are well suited to represent FMS characteristics such as precedence relations, concurrency, conflict and synchronization.

Deadlock prevention is usually achieved either by effective system design (Jeng & Xie, 1999;Jeng, Xie, & Peng, 2002;Xie & Jeng, 1999;Zhou & DiCesare, 1991;Zhou & Venkatesh, 1998) via limiting the number of raw products into a system or by using an off-line mechanism to control the requests for resources to ensure that deadlocks never occur. The former, however, often degrades the performance of a system since more permissive behavior is restricted. The latter, in general, largely due to the seminal work of Ezpeleta, Colom, and Martinez (1995), uses monitors (control places) and related arcs to achieve deadlock prevention purposes (Abdallah & ElMaraghy, 1998;Barkaoui & Abdallah, 1995;Barkaoui et al., 1997;Ezpeleta et al., 1995;Ghaffari, Nidhal, & Xie,;Huang,

☆ This manuscript was processed by Area Editor Gunsel A. Suer.
* Corresponding author.
E-mail address: zhwli@xidian.edu.cn (Z. Li).

0360-8352/$ - see front matter © 2008 Published by Elsevier Ltd.
doi:10.1016/j.cie.2008.03.013

Jeng, Xie, & Chung, 2001;Huang, Jeng, Xie, & Chung, 2006;Iordache, Moody, & Antsaklis, 2002;Li & Zhou, 2004;Li & Zhou, 2006;Li & Zhou, 2008;Tricas, Valles, Colom, & Ezpeleta, 1998;Uzam, 2002;Xing, Hu, & Chen, 1996). The work of Ezpeleta et al. (1995) is usually considered to be the first using structure theory of Petri nets to design monitor-based liveness-enforcing Petri net supervisors for FMS. Ezpeleta et al. (1995) defined a subclass of ordinary and conservative Petri nets called Systems of Simple Sequential Processes with Resources (S^3PR) and required the target Petri net to be in that subclass. A monitor is added for every strict minimal siphon such that liveness can be enforced. However, too many monitors and arcs have to be added, leading to a much more complex Petri net supervisor than the originally built Petri net model. This is not surprising since the number of siphons that need to be controlled grows quickly and in the worse case is exponential with respect to the size of a plant net model (Ezpeleta, Couvreur, & Silva, 1993;Lautenbach, 1987). Furthermore, the behavior of the system is much restrictive. This particular research is motivated by the urgent need to explore an effective and computationally efficient way that systematically eliminates redundant monitors such that the structural complexity of the liveness-enforcing Petri net supervisors computed by the existing deadlock prevention policies in the literature can be reduced while liveness is preserved.

The importance of designing structurally simple Petri net supervisors for FMS is well recognized in recent years (Li & Zhou, 2004;Li & Zhou, 2008;Li, Hu, & Wang, 2007). Uzam, Li, and Zhou (2007) proposed a methodology to identify and eliminate redundant control places in a Petri net supervisor of an FMS. Generally speaking, it can minimize the number of additional monitors in a supervisor. However, it suffers from much computational cost since generating the reachability set of markings of a system is the first step of the methodology. Traditional "explicit" algorithms that explore the reachability graph of a Petri net require memory and time at least proportional to the number of reachable markings, thus they are applicable only to fairly small systems. In practice, we are limited to hoping that a plant net model is not only small-sized, but also with small initial markings such that its reachability set is small enough to fit in the memory of our computers. Such a brute-force approach makes nevertheless sense, but, from both practical and theoretical points of view, research efforts should be directed at finding more efficient redundant monitor removal algorithms that avoid to generate and store the reachability set.

Fortunately, there is an established tool inside Petri net theory, which can be used to remove redundant monitors from a liveness-enforcing Petri net supervisor. It is *implicit places* (Garcia-Vallés & Colom, 1999;Recalde, Teruel, & Silva, 1997). Implicit places are a kind of places with the property that their addition to or removal from a net system does not change its behavior, i.e., an implicit place is an redundancy. This paper first utilizes this tool to develop a method to remove implicit monitors in a liveness-enforcing Petri net supervisor. Then, motivated by the observation that some monitors' removal can lead to more permissive behavior as well as liveness preservation,[1] we also propose an algorithm to eliminate them from liveness-enforcing net supervisors. This algorithm can also eliminate implicit places that cannot be removed by the first proposed method. It is done by using a mixed integer programming based approach that is well established by Chu and Xie (1997).

[1] A monitor with this property is called a liveness-restrictive place that is defined in Section 4.

The remainder of the paper is organized as follows. Basics of Petri nets and the concept of implicit places are briefly reviewed in Section 2. Section 3 develops a method to systematically remove the redundant (implicit) monitors from a liveness-enforcing Petri net supervisor. In Section 4, we propose an algorithm that can eliminate implicit and liveness-restrictive monitors whose removal may lead to a more permissive supervisor with liveness preservation. FMS examples are used to demonstrate the proposed methods throughout Section 4. Some interesting problems are discussed in Section 5 that is mainly concerned with the size of structurally optimal liveness-enforcing supervisors and the computational complexity of the proposed algorithms. Finally, Section 6 concludes this paper.

2. Preliminaries

2.1. Basics of Petri nets

A (general) Petri net (Murata, 1989;Zhou & Venkatesh, 1998) N is a 4-tuple (P, T, F, W) where P and T are finite, nonempty, and disjoint sets. P is the set of places and T is the set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is called flow relation of the net, represented by arcs with arrows from places to transitions or from transitions to places. $W : F \to \mathbf{N}^+$ is a mapping that assigns a weight to an arc, where $\mathbf{N}^+ = \{1, 2, \cdots\}$. $N = (P, T, F, W)$ is called an ordinary net, denoted as $N = (P, T, F)$, if $\forall f \in F$, $W(f) = 1$. A marking M of N is a mapping from P to $\mathbf{IN}$, where $\mathbf{IN} = \{0, 1, 2, \cdots\}$. A net is self-loop free iff $\nexists x, y \in P \cup T, f(x, y) \in F \wedge f(y, x) \in F$. The pre-incidence matrix *Pre*: $P \times T \to \mathbf{IN}$ of N is $Pre(p, t) = W(p, t)$. The post-incidence matrix *Post*: $P \times T \to \mathbf{IN}$ of N is $Post(p, t) = W(t, p)$. A self-loop free Petri net $N = (P, T, F, W)$ can be alternatively represented by its flow matrix or incidence matrix $[N]$, where $[N]$ is a $|P| \times |T|$ integer matrix with $[N](p, t) = Post(p, t) - Pre(p, t)$. Let A be any one of $[N]$, *Pre*, and *Post*. $A(\alpha, \beta)$ is used to denote the sub-matrix of A corresponding to rows of places in $\alpha \subseteq P$ and columns of transitions in $\beta \subseteq T$.

Let $N = (P, T, F, W)$ be a net. The preset of a node $x \in P \cup T$ is defined as ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$. While the postset of a node $x \in P \cup T$ is defined as $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$. This notation can be extended to a set of nodes as follows: Given $X \subseteq P \cup T$, ${}^\bullet X = \cup_{x \in X}({}^\bullet x)$, and $X^\bullet = \cup_{x \in X}(x^\bullet)$. $M(p)$ indicates the number of tokens contained in place p. p is marked by M iff $M(p) > 0$. A subset $D \subseteq P$ is marked by M iff at least one place in D is marked by M. The sum of tokens in all places in D is denoted by $M(D)$ where $M(D) = \sum_{p \in D} M(p)$.

A transition $t \in T$ is enabled at a marking M iff $\forall p \in {}^\bullet t$, $M(p) \geqslant W(p, t)$; this fact is denoted as $M[t\rangle$; when fired in a usual way, this gives a new marking M' such that $\forall p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$; it is denoted as $M[t\rangle M'$. Marking M' is said to be reachable from M if there exists a sequence of transitions $\sigma = t_0 t_1 \cdots t_n$ and markings $M_1, M_2, \cdots$, and M_n such that $M[t_0\rangle M_1[t_1\rangle M_2 \cdots M_n[t_n\rangle M'$ holds. The set of markings reachable from M in N is denoted as $R(N, M)$. The set of all the occurrence sequences, or language, from M is denoted by $L(N, M)$.

A transition $t \in T$ is live under M_0 iff $\forall M \in R(N, M_0)$, $\exists M' \in R(N, M)$, $M'[t\rangle$. N is dead under M_0 iff $\nexists t \in T$, $M_0[t\rangle$ holds. (N, M_0) is deadlock-free iff $\forall M \in R(N, M_0)$, $\exists t \in T$, $M[t\rangle$ holds. (N, M_0) is live iff $\forall t \in T$, t is live under M_0. (N, M_0) is bounded iff $\exists k \in \mathbf{IN}$, $\forall M \in R(N, M_0)$, $\forall p \in P$, $M(p) \leqslant k$ holds.

A P-vector is a column vector $I : P \to Z$ indexed by P and a T-vector is a column vector $J : T \to Z$ indexed by T, where Z is the set of integers. I^T and $[N]^T$ are the transposed versions of a vector I and matrix $[N]$, respectively. P-vector I is a P-invariant (place invariant) iff $I \neq \mathbf{0}$ and $I^T[N] = \mathbf{0}^T$.