MELİKŞAH ÜNİVERSİTESİ
KAYSERİ

The Graduate Institute of Science and Engineering

M.Sc. Thesis in Electrical and Computer Engineering

# A PETRI NET BASED DIVIDE AND CONQUER METHOD FOR THE SYNTHESIS OF LIVENESS ENFORCING SUPERVISORS IN FMS

by

Rabiu Saleh ZAKARIYYA

June 2014
Kayseri, Turkey

# A PETRI NET BASED DIVIDE AND CONQUER METHOD FOR THE SYNTHESIS OF LIVENESS ENFORCING SUPERVISORS IN FMS

**by**

**Rabiu Saleh ZAKARIYYA**

**A Thesis Submitted to**

**the Graduate Institute of Science and Engineering**

**of**

**Melikşah University**

**in partial fulfillment of the requirement for the degree of**
**Master of Science**

**in**

**Electrical and Computer Engineering**

**June 2014**
**Kayseri, Turkey**

This is to certify that I have read the thesis titled "A Petri Net Based Divide and Conquer Method for the Synthesis of Liveness Enforcing Supervisors in FMS" by Rabiu Saleh Zakariyya and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electrical and Computer Engineering, the Graduate Institute of Science and Engineering, Melikşah University.

_______________________

June 6, 2014 Prof. Dr. Murat UZAM
Supervisor

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_______________________

June 6, 2014 Prof. Dr. Murat UZAM
Head of Department

<u>Examining Committee Members</u>

| | | |
|---|---|---|
| Prof. Dr. Murat UZAM | June 18, 2014 | _____________________ |
| Prof. Dr.M. Halidun KELEŞTEMUR | June 18, 2014 | _____________________ |
| Yrd. Doç. Dr. M. Mete ÖZBILEN | June 18, 2014 | _____________________ |

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Science and Engineering.

___________________________

Prof. Dr. M. Halidun KELEŞTEMUR
Director

June 2014

# A PETRI NET BASED DIVIDE AND CONQUER METHOD FOR THE SYNTHESIS OF LIVENESS ENFORCING SUPERVISORS IN FMS

Rabiu Saleh ZAKARIYYA

M.Sc. Thesis – Electrical and Computer Engineering

June 2014

Supervisor: Prof Dr. Murat UZAM

## ABSTRACT

In flexible manufacturing systems (FMS), deadlocks occur due to shared resources and lead to catastrophic results. Petri nets have been widely used as a modeling and design tool for the study of deadlock problems in FMSs. In this thesis, a Petri net based divide and conquer method is proposed for the synthesis of liveness enforcing supervisors (LES) in FMSs. To deal with deadlocks in a complex Petri net model (PNM), the use of reachability graph (RG) is unmanageable. To overcome this problem, in the method proposed in this thesis, the Petri net model is divided into small connected sub-nets. Then, a LES is computed for the original PNM by using these sub-nets. The proposed method is generally applicable, very effective and straight forward although its off-line computation is of exponential complexity in theory. Examples are provided to show the applicability of the proposed method.

**Keywords:** Flexible Manufacturing Systems, Deadlock Prevention, Petri nets.

# ESNEK ÜRETİM SİSTEMLERİNDE CANLILIK SAĞLAYICI GÖZETİCİLERİN SENTEZLENMESİ İÇİN PETRİ AĞI TEMELLİ BÖLVE KAZAN METODU

Rabiu Saleh ZAKARIYYA

Yüksek Lisans Tezi – Elektrik ve Bilgisayar Mühendisliği

Haziran 2014

Tez Danışman: Prof. Dr. Murat UZAM

## ÖZ

Esnek üretim sistemlerinde (Flexible Manufacturing Systems – FMS) kördüğümler paylaşılan kaynaklar nedeniyle ortaya çıkar ve feci sonuçlara yol açarlar. Petri ağları FMS' lerde kördüğüm problemlerini çalışmak için yaygın olarak kullanılan bir modelleme ve tasarım aracıdır. Bu yüksek lisans tezinde, FMS' lerde canlılık sağlayıcı denetçilerin sentezi (LES) için Petri ağı temelli bir böl ve kazan yöntemi önerilmiştir. Karmaşık bir Petri ağı modelinde (Petri net model – PNM) kördüğüm problemleriyle başa çıkmak için, ulaşılabilirlik grafı (RG) kullanımı mümkün değildir. Bu sorunun üstesinden gelmek için, bu tezde önerilen yöntemde, Petri ağı modeli, küçük bağlı alt ağları bölünür. Sonra, LES bu alt-ağları kullanarak orijinal Petri ağı modeliiçin hesaplanır. Her ne kadar önerilen yöntemin çevrim dışı (off-line) hesaplanması teorisi olarak üstel karmaşıklığa sahip olsa daönerilen yöntem uygulanabilir, çok etkili ve kolay anlaşılırdır. Önerilen yöntemin uygulanabilirliğini göstermek için bazı örnekler sunulmuştur.

**Anahtar Kelimeler:** Esnek Üretim Sistemleri, Kördüğüm Önleme, Petri ağları.

## DEDICATION

I dedicate this thesis to the governor of Kano state Engr. Dr Rabiu Musa Kwankwaso for his contribution to my carrier education.

# ACKNOWLEDGEMENTS

This work was supported by the research grant of the Scientific and Technological Research Council of Turkey (Türkiye Bilimselve Teknolojik Araştırma Kurumu - TÜBİTAK) under the project number TÜBİTAK-112M229.

I would like acknowledge my supervisor Prof. Dr. Murat UZAM for his wonderful effort and endless contribution given to me throughout this research.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ABREVIATIONS

**SYMBOL/ABBREVIATIONES**

| | |
|---|---|
| DES | Discrete Event System |
| FMS | Flexible Manufacturing System |
| SA | Structural Analysis |
| FBM | First met Bad Marking |
| LES | Liveness Enforcing Supervisor |
| DAC | Divide-and-Conquer |
| PNM | Petri Net Model |
| pCPNM | Partially Controlled Petri Net Model |
| pCRPNM | Partially Controlled Reduced Petri Net Model |
| $S^3PR$ | Systems of simple sequential processes with resources |
| $S^4PR$ | System of sequential systems with shared resources |
| W | Weight of an arc |
| $\ni$ | Belongs to |
| $\exists$ | Exist |
| $\forall$ | For all |
| iff | If and only if |

# CHAPTER 1

## INTRODUCTION

A flexible manufacturing system (FMS) is an automated manufacturing system that produces parts in many sequences of operations using limited number of shared resources which includes machines, robots, fixtures and buffers.  The flexibility of an FMS depends on an automated programmable transportation system linking the workstations and on a complex computer control supervision processes activities, transfer of operations, information channel, and so on. As a result of shared resources, the system may enter into a deadlock. Deadlocks are undesirable situation in a resource allocation system. Their occurrence implies the stoppage of the whole or partial system operation. In a production system, deadlocks and related blocking phenomena often cause unnecessary costs such as long downtime and low utilization of some critical and expensive resources, and may lead to catastrophic results in highly automated systems, e.g., semiconductor manufacturing systems. Therefore, it is necessary to develop an effective control policy to make sure that deadlocks never occur in these systems [1]. There are four necessary conditions for a deadlock to occur [2], known as the Coffman conditions:

1. ***Mutual exclusion condition*****:** a resource can only get involved in one process at a time.
2. ***Hold and wait condition*****:** processes (operation) already holding resources may request new resources.
3. ***No preemption condition:*** no resource can be forcibly released from a process holding it, when processing is under progress and resources can be released only by the explicit action of the process.

4. ***Circular wait condition:*** two or more processes form a circular chain where each process waits for a resource that the next process in the chain holds.

A deadlock will never occur if one of these conditions is not satisfied. The physical characteristics and technical background of an FMS which is a discrete event system, show that the first three deadlock conditions always hold and any method which guarantees that the fourth condition (the circular wait) does not hold, is a valid solution to tackle problem of deadlock in FMS [3].

## 1.2 DEADLOCK HANDLING STRATEGIES

To deal with a deadlock problems in FMS, there are mainly three approaches [4, 5]: deadlock detection and recovery [2, 6], deadlock avoidance [7, 8] and deadlock prevention [5, 10 and 22].

### 1.2.1 Deadlock detection and recovery approach

A deadlock detection and recovery approach permits the occurrence of deadlocks. When a deadlock occurs, it is detected and then the system is put back to a deadlock-free state, by simply reallocating the resources. The efficiency of this approach depends upon the response time of the implemented algorithms for deadlock detection and recovery. In general, these algorithms require a large amount of data and may become complex when several types of shared resources are considered [1].

### 1.2.2 Deadlock avoidance approach

In deadlock avoidance, at each system state an on-line control policy is used to make a correct decision to proceed among the feasible evolutions. The main purpose of this approach is to keep the system away from deadlock states. Aggressive methods usually lead to higher resource utilization and throughput, but do not totally eliminate all deadlocks for some cases. In such cases if a deadlock arises, suitable recovery strategies are still required [1].

### 1.2.3 Deadlock prevention approach

Deadlock prevention is considered to be a well-defined problem in DES literature. It is usually achieved by using an off-line computational mechanism to control the request for resources to ensure that deadlocks never occur. The goal of a deadlock prevention approach is to impose constraints on a system to prevent it from reaching deadlock states. In this case, the computation is carried out offline in a static way and once the control policy is established, the system can no longer reach undesirable deadlock states. A major advantage of deadlock prevention algorithms is that they require no run-time cost since problems are solved in system design and planning stages. The major criticism is that they tend to be too conservative, thereby reducing the resource utilization and system productivity [1].

## 1.3 TOOLS USED IN HANDLING DEADLOCK IN FMS

Digraphs, Automata and Petri nets are major mathematical tools to investigate deadlock problems in FMSs. Recent decades have seen that Petri nets are increasingly becoming an important, popular and fully-fledged mathematical model to provide solutions to the issues. There are three criteria to evaluate and design a liveness-enforcing Petri nets based supervisor for an FMS to be controlled, which takes the form of monitors sometimes called control places that can be regarded as the intervention from human being or other external agencies. The criteria include behavioral permissiveness, computational complexity and structural complexity [9].

A maximally permissive supervisor implies that all legal states in the sense of deadlock control in a plant to be controlled are reachable in the controlled system, which, from the productivity point of view, usually leads to high utilization of system resources. A deadlock control algorithm with low computational complexity usually means that the calculation of its corresponding supervisor is tractable and that it can potentially be applied to the real-world systems. Structural complexity of liveness-enforcing supervisor is referred to as the number of monitors as well as related arcs in the supervisor. A supervisor with a small number of monitors can always decrease the hardware and software costs in the stage of model checking and verification, and control validation and implementation [9].

In general, it is difficult or even impossible, given a real world system, to find a maximally permissive, yet computational efficient, supervisor with a minimal number of monitors. A trade-off among behavioral permissiveness, structural complexity, and computational tractability is usually adopted. For example, siphon-based deadlock prevention approaches that do not depend on a partial or complete state enumeration cannot in general lead to a maximally permissive supervisor. On the other hand, most deadlock prevention approaches, existing in the literature that can derive maximally permissive liveness-enforcing supervisors expressed by a set of monitors depend on a complete marking enumeration except for some net subclasses at special initial markings [9].

In this thesis, we focus on deadlock prevention approach. In dealing with a deadlock prevention policy in FMS, Petri nets are widely used due to their properties such as liveness, boundedness, deadlock freeness and conservativeness. The analysis of Petri nets are mostly categorized into two techniques: structural (SA) analysis and reachibility graph (RG) analysis. The works found in [5, 10] make use of the structural analysis of Petri nets for enforcing liveness on FMS by adding control places (monitor) and related arcs to the plant net model. Some methods based on RG analysis are proposed in [11-12]. The problem with these methods is that when dealing with large Petri net models, the RG analysis becomes difficult and time consuming due to “state explosion problem”. To overcome this problem, an efficient method based on the first met bad marking (FBM) is proposed in Chen *et al.* [13]. It is an expensive method that requires high computational cost in obtaining the optimal solution to the deadlock prevention problem. Therefore, to provide a solution to the deadlock prevention problem for a complex FMS as in [14], the net model is divided into smaller sub-nets and then control places are computed easily.

## 1.4 THESIS OBJECTIVE

The objective of this thesis is to propose a Petri net based divide and conquer method for the synthesis of liveness enforcing supervisors (LES) in FMSs. To obtain the LESs from a very big PNM is not an easy task. In this thesis, to ease this problem the PNM of a system is divided into small connected subnets. Each connected subnet prone to deadlock is then used to compute the LES for the original PNM. The proposed method makes it possible to compute monitors for PNMs with very big RGs, for which traditional RG based deadlock prevention methods [14, 15] cannot be used due to very hard computation load.

The remainder of this thesis is organized as follows. Chapter 2 is the introduction and basics of Petri nets followed by the definition of some terms and properties of Petri net with examples. Chapter 3 provides the implementation of the proposed method with an illustrative FMS example prone to deadlocks. Chapter 4 considers two application examples. Chapter 5 gives the conclusions.

# CHAPTER 2

# BASICS OF PETRI NETS

This chapter focuses on the basics, structural and behavioral properties of Petri nets [1, 6, and 16] such as liveness, boundedness, safeness, conservativeness and reversibility. The reachability graph analysis of Petri nets, Petri net reduction rule, redundancy test and computation of monitors are considered. Finally, modeling of an FMS with Petri net structures such as sequence, concurrency, conflict and synchronization are explained in this chapter.

## 2.1 INTRODUCTION

A Petri net is a directed bipartite graph. It consists of two components: a net structure and an initial marking. A net (structure) contains two sorts of nodes: places and transitions. There are directed arcs from places to transitions and directed arcs from transitions to places in a net. Places are graphically represented by circles and transitions by boxes or bars. A place can hold tokens denoted by black dots, or a positive integer representing their number. The distribution of tokens over the places of a net is called a marking that corresponds to a state of the modeled system. The initial token distribution is hence called the initial marking [1].

## 2.2 BASIC STRUCTURAL PRORERTIES OF PETRI NETS

**Definition 2.1.** A Petri net [1, 16] in terms of structure, is a 4-tuple $\boldsymbol{N} = (P, T, F, W)$, where

$P = \{p1, p2, \ldots\ldots, pa\}$ is a finite set of places represented by circles, where $a > 0$, and
$T = \{t1, t2, \ldots\ldots, tb\}$ is a finite set of transitions represented by bars or boxes, where $b > 0$.
$F \subseteq (P \times T) \cup (T \times P)$ is a flow relation of the net with directed arcs represented by arrows connecting places to transitions or transitions to places.
$W$: $(P \times T) \cup (T \times P)$ is a mapping of $N$ called the weight of an arc, i.e. $W(x, y) > 0$ *iff* $(x, y) \in F$, and $W(x, y) = 0$ otherwise, where $x, y \in P \cup T$.

Example 2.1. Fig. 2.1 shows a Petri net with $P = \{p1, p2, p3, p4\}$, $T = \{t1, t2, t3, t4\}$, $F = \{(p1, t1), (t3, p1), (p2, t2), (t1, p2), (p3, t3), (t2, p3), (p4, t2), (t3, p4), (p2, t4), (t4, p4)\}$, $W(p1, t1) = W(t3, p1) = W(p2, t2) = W(t1, p2) = W(p3, t3) = W(t2, p3) = W(t3, p4) = W(p2, t4) = W(t4, p4) = 1$, and $W(p4, t2) = 2$. It is clear that the net is not ordinary because of the multiplicity of arcs $(p4, t2)$.

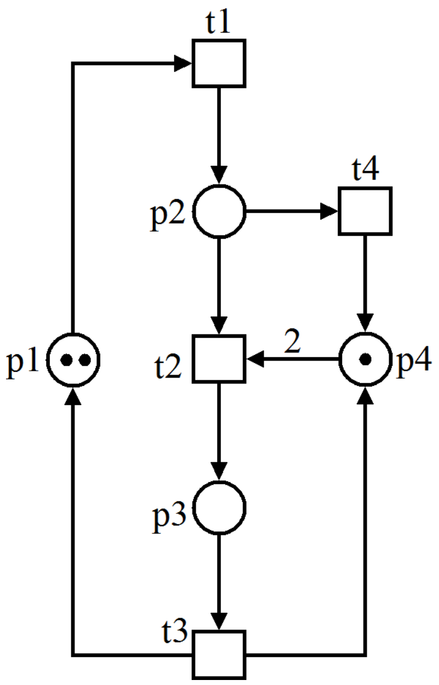


Fig. 2.1. A simple Petri net.

**Definition 2.2.** A marking $M$ of a Petri net $\boldsymbol{N}$ is a mapping from $P$ to $\mathbf{N}$. $M(p)$ denotes the number of tokens in place $p$. A place $p$ is marked by a marking $M$ iff $M(p) > 0$. A subset $S \subseteq P$ is marked by $M$ iff at least one place in $S$ is marked by $M$. The sum of tokens of all places in $S$ is denoted by $M(S)$, i.e., $M(S) = \sum_{p \in s} M(P)$, $S$ is said to be empty at $M$ iff $M(S) = 0$. $(N, M_0)$ is called a net system or marked net and $M_0$ is called an initial marking of $N$. Example 2.2. In Fig. 2.1, the initial marking $M_0 = (2\ 0\ 0\ 1)^T$ which shows that only $p1$ and $p4$ are marked.

**Definition 2.3.** For $t \in T$, $p \in {}^{\bullet}t$ is defined as input place of t and $p \in t^{\bullet}$ is defined as output place of $t$. For $p \in P$, $t \in {}^{\bullet}p$ is defined as input transition of p and $t \in p^{\bullet}$ is defined as output transition of $p$.

**Definition 2.4.** A net $\boldsymbol{N} = (P, T, F, W)$ with a node by $x \in P \cup T$. preset of $x$ is defined as ${}^{\bullet}x = \{y \in P \cup T | (y, x) \in F\}$, while the postset of $x$ is defined as $x^{\bullet} = \{y \in P \cup T | (x, y) \in F\}$. For an extended node, the set of nodes are defined as follows: given $X \subseteq P \cup T$, ${}^{\bullet}X = \cup_{x \in X} {}^{\bullet x}$ and $X^{\bullet} = \cup_{x \in X} {}^{x\bullet}$. Also ${}^{\bullet\bullet}X$ is defined as the preset of ${}^{\bullet}X$ and $X^{\bullet\bullet}$ defined as the postset of $X^{\bullet}$.

Example 2.3. In Fig. 2.1, we have ${}^{\bullet}t1 = \{p1\}$, ${}^{\bullet}t2 = \{p2, p4\}$, $t2^{\bullet} = \{p3\}$, $t3^{\bullet} = \{p1, p4\}$, ${}^{\bullet}p3 = \{t2\}$, $p3^{\bullet} = \{t3\}$, ${}^{\bullet}p4 = \{t3, t4\}$ and $p4^{\bullet} = \{t2\}$. let $S = \{p3, p4\}$. Then ${}^{\bullet}S = {}^{\bullet}p3 \cup {}^{\bullet}p4 = \{t2, t3, t4\}$ and $S^{\bullet} = p3^{\bullet} \cup p4^{\bullet} = \{t2, t3\}$. It is easily seen that $max_{p4^{\bullet}} = 2$ and $\forall p \in P | \{p4\}$, $max_{p^{\bullet}} = 1$. $p1$ is an input place of transition $t1$ and output place of $t3$.

**Definition 2.5.** A transition $t \in T$ is enabled at a marking $M$ iff $\forall p \in {}^{\bullet}t$, $M(p) \geq W(p, t)$. This fact is denoted by $M[t >$. Firing it yields a new marking $M'$ such that $\forall p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$, as denoted by $M[t > M'$. $M'$ is called an immediately reachable marking from $M$. Marking $M''$ is said to be reachable from $M$ if there exists a sequence of transitions $\sigma = t_0t_1t_2...tn$ and markings $M_1$, $M_2$, $\cdots$, and $M_n$ such that $M[t_0 > M_1[t1 > M_2 \cdots M_n[tn > M''$ holds. The set of markings reachable from $M$ in $N$ is called the reachability set of Petri net $(N, M)$ and denoted by $R(N, M)$.

Example 2.4. In Fig. 2.2, $t1$ is enabled at initial marking $M_0 = 2p1 + p4$ since $\bullet t1 = \{p1\}$ and $M_0(p1) = 2 > W(p1, t1) = 1$. Firing $t1$ leads to $M_1$ with $M_1(p1) = M_0(p1) - W(p1, t1) + W(t1,$

$p1) = 1$, $M_1(p2) = M_0(p2) - W(p2, t1) + W(t1, p2) = 1$, $M_1(p3) = M_0(p3) - W(p3, t1) + W(t1, p3) = 0$, and $M_1(p4) = M_0(p4) - W(p4, t1) + W(t1, p4) = 1$.

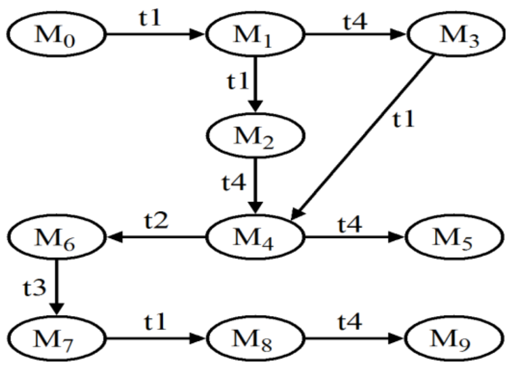


Fig. 2.2. The reachability graph of net $(N, M_0)$ shown in Fig. 2.1.

At marking $M_1$, both $t1$ and $t4$ are enabled. Firing $t1$ at $M_1$ leads to $M_2$. Firing $t4$ at $M_1$ leads to $M_3$. Only $t1$ is enabled at $M_3$. After $t1$ fires at $M_3$, it leads to $M_4$. At $M_2$, only $t4$ is enabled. Firing it leads also to $M_4$. After $t4$ fires at $M_4$, it leads to $M_5$, after $t2$ fires at $M_4$, it leads to $M_6$, after $t3$ fires at $M_6$, it leads to $M_7$, after $t1$ fires at $M_7$, it leads to $M_8$, after $t4$ fires at $M_8$, and it leads to $M_9$. The reachability set of the net in Fig. 2.2 is $R(N, M_0) = \{M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9\}$, where $M_0 = 2p1 + p4$, $M_1 = p1+ p2+ p4$, $M_2 = 2p2+ p4$, $M_3 = p1+2p4$, $M_4 = p2+2p4$, $M_5 = 3p4$, $M_6 = p3$, $M_7 = p1 + p4$, $M_8 = p2 + p4$, and $M_9 = 2p4$. Note that at markings $M_5$ and $M_9$, no transition is enabled. Therefore these states are deadlock states.

**Definition 2.6.** A pure net $N = (P, T, F, W)$ can be represented by its incidence matrix *[N]*, where $[N]$ is a $|P| \times |T|$ integer matrix with $[N](p, t) = W(t, p) - W(p, t)$. For a place $p$ (transition $t$), its incidence vector, a row (column) in $[N]$, is denoted by $[N](p, \cdot)$ ($[N](\cdot, t)$), or $[N] = [N]^+ - [N]^-$. Where, $[N]^+ = W(t, p)$ and $[N]^- = W(p, t)$ is called input and output incidence matrix.

According to the definition, it is easy to see the physical meanings of an element in an incidence matrix of a Petri net N. Specifically, [N](p, t) indicates that p receives (loses)

$|[N](p, t)|$ tokens if $[N](p, t) > 0$ ($[N](p, t) < 0$) after $t$ fires. The number of tokens in $p$ does not change if $[N](p, t) = 0$ after $t$ fires. Vector $[N](p, \cdot)$ shows the token variation in $p$ with respect to the firing of each transition once in the net $N$.

Example 2.5. In Fig. 2.1, the input and output incidence matrices are;

$$[N]^+ = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \text{ and } [N]^- = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

Therefore, the incidence matrix $[N]$ is

$$[N] = [N]^+ - [N]^- = \begin{array}{c} \begin{array}{cccc} \text{t1} & \text{t2} & \text{t3} & \text{t4} \end{array} \\ \begin{bmatrix} -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -2 & 1 & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ \text{p1} \\ \text{p2} \\ \text{p3} \\ \text{p4} \end{array}$$

## 2.3 BEHAVIORAL PROPERTIES OF PETRI NETS

1. **Boundedness:** A net $(N, M_0)$ is said to be *k-bounded* if the number of tokens in each place does not exceed 'k' for every $R(N, M_0)$ from $M_0$.
2. **Safeness:** A net $(N, M_0)$ is said to be *safe* in any $R(N, M_0)$ if all its places are safe. A place 'p' is safe if it contains no more than one token. In other words, a Petri net is called *safe* if it is 1-bounded.
3. **liveness:** A net $(N, M_0)$ is said to be *live* if all its transitions are firable in all $R(N, M_0)$. A *live* Petri net is free from *deadlocks*.
4. **Conservativeness:** A net $(N, M_0)$ is said to be *conservative,* if the total number of tokens in all of places in all $R(N, M_0)$ is constant.
5. **Reversibility:** A Petri net $(N, M_0)$ is said to be *reversible*, if for each marking $M \in R(N, M_0)$, $M_0$ is reachable from $M$. It is possible for a reversible net to go back to initial marking $M_0$. This shows that a *deadlocked* Petri net is not *reversible*.

## 2.4 PETRI NET REDUCTION APPROACH

Petri net reduction approach is a well-known method to derive the properties of a complex Petri net model, while preserving the concerned properties, such as boundedness, liveness and reversibility [4]. It is possible to analyse and derive the properties of a complex Petri net model, by simplifying the subnet or structure. In this section, some simple reduction rules are considered [15]. A set of easy-to-use reduction rules is given in Fig. 2.3, including the following.

Rule 1: Fusion of series places as shown in Fig. 2.3 (a).

Rule 2: Fusion of series transitions as shown in Fig. 2.3 (b).

Rule 3: Fusion of parallel places as shown in Fig. 2.3 (c).

Rule 4: Fusion of parallel transitions as shown in Fig. 2.3 (d).

Rule 5: Elimination of self-loop places as shown in Fig. 2.3 (e).

Rule 6: Elimination of self-loop transitions as shown in Fig. 2.3 (f).

It can be proven that these six operations preserve the properties of liveness, safeness and boundedness, when they are applied to reduce a Petri net. That is, let ($N$, $M_0$) and ($N'$, $M'_0$) be the Petri nets before and after one of the above mentioned operations. Then ($N'$, $M'_0$) is live, safe, or bounded *iff* ($N$, $M_0$) is live, safe, or bounded, respectively. These rules are from [15].

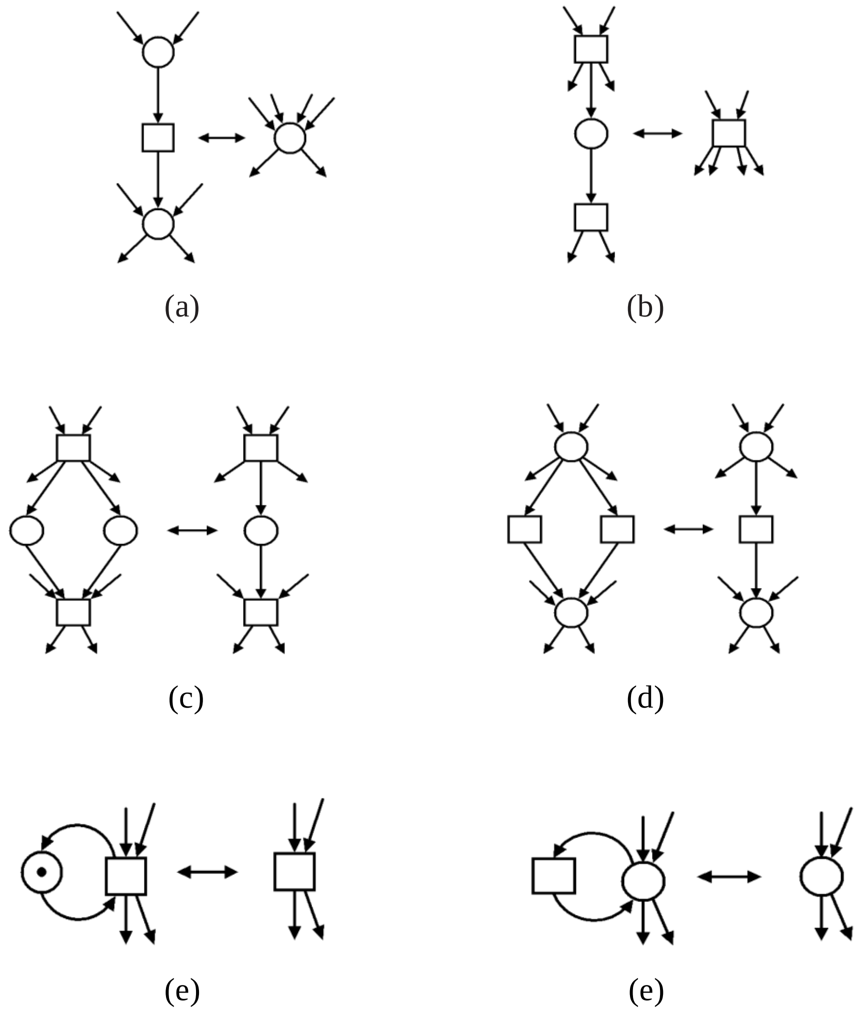


Fig. 2.3. Petri net reduction rules from [15].

## 2.5 SIMPLIFIED CONTROLLER COMPUTATION FOR A PLACE INVARIANT

In this section, we briefly recall the controller computation method mentioned in [15] to be used in the computation of a Petri net controller when there is only one place invariant to be enforced on a plant Petri net. In [17], a computationally efficient method was presented for constructing Petri net controller for a discrete event system modeled by a Petri net. The controller consists of places and input-output arcs, and is computed based on the concept of Petri net place invariants and is able to enforce logical and algebraic constraints containing elements of the marking and firing vectors. The system (also known as the *plant* or the *process net*) to be controlled is modeled by a Petri net with *n* places and *m* transitions. The incidence matrix of the plant net is $D_p$. The controller net is a Petri net with incidence matrix $D_c$ made up of the transitions of the plant net and a separate set of places. The controlled plant is the Petri net with incidence matrix *D* made up of both the original plant net and the added controller. The control goal is to force the plant to obey constraints of the form

$$\sum_{i=1}^{n} l_i \mu_i \leq \beta \qquad (1)$$

Where $\mu_\mathrm{i}$ is the initial marking of place $p_i$, and the $l_i$ and $\beta$are integer constants. By introducing a nonnegative slack variable $\mu_\mathrm{c}$ this inequality constraint can be transformed into equality as follows:

$$\sum_{i=1}^{n} l_i \mu_i + \mu_c = \beta \qquad (2)$$

In this case, the slack variable denotes a new place $p_c$, generally called a *control place* or a *monitor*, which holds the extra tokens required to meet the equality. The control place ensures that the weighted sum of tokens in the places of the plant net is always less than or equal to$\beta$. The controller net, composed of the control places and their input and output arcs, maintains the inequality constraint. Place invariants are sets of places whose token count remain constant for all possible markings. All constraints of the type (1) can be grouped in matrix form as follows:

$$L\ \mu_p \leq b \qquad (3)$$

Where $\mu_p$ is the marking vector of the plant Petri net model, $L$ is an $n_c \times n$ integer matrix, $b$ is an $n_c \times 1$ integer vector and $n_c$ is the number of the constraints of type (1). All place invariants of type (2) can be grouped in the matrix form as follows:

$$L\ \mu_p + \mu_c = b \qquad (4)$$

Where $\mu_c$ is an $n_c{\times}1$ integer vector, representing the marking of the control places. Finally, given a plant Petri net model $D_p$ and the constraints the plant must satisfy, namely $L$ and $b$, the Petri net controller $D_c$ is defined as follows:

$$D_c = -LD_p \qquad (5)$$

The initial marking of the controller Petri net $\mu_{c0}$, is calculated in such a way that the place invariant equation (4) is initially satisfied. Therefore the initial marking vector is as follows:

$$\mu_{c0} = b - L\ \mu_{p0}. \qquad (6)$$

In [17], it is assumed that all place invariants to be enforced on the plant net are given and therefore in the controller computation the incidence matrix $D_p$ of the plant net is used. As a result the controller net can be computed with one matrix multiplication as shown in (5). In iterative deadlock prevention approach, one control place with its input-output arcs at each iteration is computed. In [12], the method of [17] was used, namely (5) and (6) as it is, for the computation of the controller at each iteration, the incidence matrix $D_p$ of the plant plus the controller net obtained in the previous iterations are used. This means that for the computation of the controller the matrix multiplication must be performed with a very big incidence matrix at each iteration. However, it can be observed that the control net, i.e. the control place and its input-output arcs, to be computed at each iteration within our deadlock prevention approach consists of only the input-output transitions of those places which appear within the place invariant. This means that for computing one control net for a single place invariant there is no need to use the incidence matrix $D_p$ of the plant net. Rather, this computation can be carried out by using the incidence matrix $D_{PI}$ of the place invariant related net. Of course at each iteration, for each controller computation we must use a different incidence matrix $D_{PI}$. However, when we are dealing with very complex Petri net models this simplification is justified because otherwise we must use a very big incidence matrix $D_p$ of the plant net at each iteration. Given a place invariant related net, i.e., a set of places with their input-output arcs and the constraint, the place invariant related net must satisfy, i.e. $L_{PI}$ and $b$, we simplify (5) and as a result the Petri net controller $D_c$ is defined as follows:

$$D_c = -L_{PI}D_{PI} \qquad (7)$$

Where $D_{PI}$ is the incidence matrix of the place invariant related net with $j$ places and $k$ transitions, $L_{PI}$ is a $j \times 1$ integer row vector representing the invariant related places and $D_c$ is a $k \times 1$ integer row vector representing the incidence matrix of the computed controller net.

The initial marking of the computed controller net $\mu_{c0}$, can be found by

$$\mu_{c0} = b - L_{PI0PI0} \qquad (8)$$

Where $\mu_{PI0}$ is the initial markings of the place invariant related places. Note that by using the (7) and (8) it is possible to simplify the computation of a Petri net controller when there is only one place invariant to be enforced on a plant Petri net. In the special case of (8), within our deadlock prevention approach a place invariant consists of only activity places. By definition, there is no token within the activity places initially. This means that $L_{PI0PI0} = 0$. As a result in our deadlock prevention approach, (8) becomes

$$\mu_{c0} = b \qquad (9)$$

This shows that when we obtain a place invariant to be enforced on a plant net in our deadlock prevention approach, the initial marking of the controller is equal to $b$. This also further simplifies the controller computation. Let us consider an example to see how useful the simplification of the controller computation. Assume that we have a place invariant ($PI$) $\mu_2 + \mu_5 \leq 1$ to be enforced on the Petri net model shown in Fig. 2.4. The incidence matrix $D_p$ of the Petri net model, shown in Fig. 2.4, is as follows:

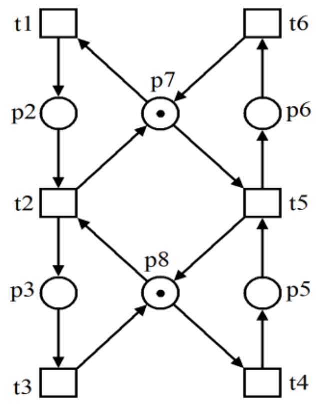


Fig. 2.4. Petri net model

$$D_p = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{cccccc|c} t1 & t2 & t3 & t4 & t5 & t6 & \\ 1 & -1 & 0 & 0 & 0 & 0 & p2 \\ 0 & 1 & -1 & 0 & 0 & 0 & p3 \\ 0 & 0 & 0 & 1 & -1 & 0 & p5 \\ 0 & 0 & 0 & 0 & 1 & -1 & p6 \\ -1 & 1 & 0 & 0 & -1 & 1 & p7 \\ 0 & -1 & 1 & -1 & 1 & 0 & p8 \end{array}$$

while its initial marking is

$$\begin{array}{rcccccc} & p2 & p3 & p5 & p6 & p7 & p8 \\ \mu_{p0} = & [0 & 0 & 0 & 0 & 1 & 1]^T \end{array}$$

Here $b$ = 2 and we have $L$ as follows:

$$\begin{array}{rcccccc} & p2 & p3 & p5 & p6 & p7 & p8 \\ L = & [\,1 & 0 & 1 & 0 & 0 & 0] \end{array}$$

We compute the Petri net controller $D_c$ by using (5) as follows:

$$\begin{array}{rcccccc} & t1 & t2 & t3 & t4 & t5 & t6 \\ D_c = & [\,-1 & 1 & 0 & -1 & 1 & 0] \end{array}$$

The initial marking of the Petri net controller is computed by using (6) $\mu_{c0} = b - L\ \mu_{p0} = 1$.

The above computations require $D_p$ to find out the Petri net controller for the given place invariant. If we use (7) and (9) we can obtain the same Petri net controller easily. Here we have a place invariant (*PI*) $\mu_2 + \mu_5 \leq 1$ to be enforced on the Petri net model shown in Fig. 2.4. It is obvious from (9) that the initial marking of the Petri net controller $\mu_{c0} = 1$ and that the *PI* related places are p2 and p5. This means that the *PI* related net consists of p2 and p5 together with their input-output arcs as shown in Fig. 2.5 (a). Therefore we have:

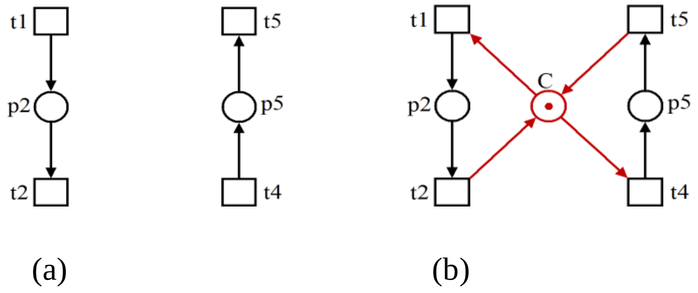


Fig. 2.5. (a) Place invariant related net for PI = $\mu_2 + \mu_5 \leq 1$, (b) Controller *C* computed for PI = $\mu_2 + \mu_5 \leq 1$.

$$D_{PI} = \begin{array}{c} \\ p2 \\ p5 \end{array} \begin{array}{c} \begin{array}{cccc} t1 & t2 & t4 & t5 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \end{array}$$

and

$$L_{PI} = \begin{array}{cc} p2 & p5 \\ [1 & 1] \end{array}$$

We compute the Petri net controller $D_c$ by using (7) as follows

$$D_c = \begin{array}{cccc} t1 & t2 & t4 & t5 \\ [-1 & 1 & -1 & 1] \end{array}$$

Graphically the computed Petri net controller *C* is shown in Fig. 2.5 (b).

## 2.6 REDUNDANCY TEST FOR LIVENESS ENFORCING SUPERVISORS OF FMS

In Petri-net-based deadlock-prevention/liveness enforcing approaches, an FMS is modeled as a Petri net, and then the liveness enforcing supervisor (*LES*), consisting of a number of control places (*CP*s), together with their related arcs and initial markings, is computed as a Petri net. There may exist redundant *CP*s in a live Petri net (*LPN*) model, denoted by a net system ($N_0$, $M_0$), controlled by *nCP*s: $CP = \{C_1, C_2,\ldots,C_n\}$. In this paper, a *CP* is called redundant if removing it still keeps the net live. It should be noted that this definition is different from that of a redundant place in literature. Removing the latter does not change the net's reachability graph. Also, redundant *CP*s are not necessarily unique given a set of *CP*s used to make a deadlock-prone net live [15].

**Redundancy Test Algorithm: Redundancy test for *LES* of FMS.**

**Input**: A live Petri net (*LPN*) model, denoted by a net system ($N_0$, $M_0$), of an FMS, controlled by *nCP*s; $CP= \{C_1, C_2, \ldots, C_n\}$;

1) [Define] $\beta_0$: the number of reachable markings or states of reachability graph ($R_0$) of ($N_0$, $M_0$)

   [Defined for Algorithm A] $\beta_A$: the number of reachable markings or states of $R_A$ *of* ($N_A$, $M_A$); $n = j + k$, where $n$: the number of *CP*s of *LPN; j*: the number of redundant *CP*s; $k$: the number of necessary *CP*s*;*

   [Defined for Algorithm B] $\beta_B$: the number of reachable markings or states of $R_B$ of ($N_B$, $M_B$); $n = l + m$, where $n$: the number of *CP*s of *LPN; l*: the number of redundant *CP*s; $m$: the number of necessary *CP*s*;*

2) Apply Algorithm A to ($N_0$, $M_0$) and the resultant net system is denoted as ($N_A$, $M_A$).
3) Apply Algorithm B to ($N_0$, $M_0$) and the resultant net system is denoted as ($N_B$, $M_B$).

**Output**: *If* ($j$>0) [for Algorithm A] *then* Output A = an *LPN*, denoted by a net system ($N_A$, $M_A$),controlled by *k* necessary *CP*s; there are *j* redundant *CP*s;

*if* $\beta_A = \beta_0$ *then* the controlled behaviour of ($N_A$, $M_A$) is the same as ($N_0$, $M_0$)

*if* $\beta_A > \beta_0$ *then* the controlled behaviour of ($N_A$, $M_A$) is more permissive than ($N_0$, $M_0$)

*else* there is no redundant *CP*s obtained due to Algorithm A and therefore for Algorithm A: Output = Input;

*If* ($l > 0$) [for Algorithm B]

*then* Output B = an *LPN*, denoted by a net system ($N_B$, $M_B$),controlled by *m* necessary *CP*s; there are *l* redundant *CP*s;

*if* $\beta_B = \beta_0$ *then* the controlled behaviour of ($N_B$, $M_B$) is the same as ($N_0$, $M_0$)

*if* $\beta_B > \beta_0$ *then* the controlled behaviour of ($N_B$, $M_B$) is more permissive than ($N_0$, $M_0$)

*else* there is no redundant *CP*s obtained due to Algorithm B and therefore for Algorithm B: Output = Input;

**end Redundancy Test Algorithm**

**Algorithm A: Front-to-Back (*FTB*) redundancy test for *LES* of FMS.**

**Input**: A live Petri net (*LPN*) model, denoted by a net system ($N_0$, $M_0$), of an FMS, controlled by *nCP*s; *CP*= {$C_1$, $C_2$,…,$C_n$};

1) [Initialize]$N_A := N_0$; $M_A := M_0$; $i = 1$; $j = 0$;$k = 0$;
2) Remove $C_i$ from ($N_A$, $M_A$). Denote the resultant net system by ($N_i$, $M_i$).
3) Check the liveness property of ($N_i$, $M_i$), compute the reachability graph ($R_i$) of ($N_i$, $M_i$) and define $\beta_{Ai}$, i.e., the number of reachable markings of $R_i$;
   *If* ($N_i$, $M_i$) is NOT LIVE

*then* put $C_i$ back into $(N_i, M_i)$; $k = k + 1$; which means that $C_i$ is necessary to keep the *PN* model live,

*else* [i.e., *If* $(N_i, M_i)$ is LIVE], $j = j + 1$; which means that $C_i$ is redundant,

*if* $\beta_{Ai} = \beta_0$ *then* the controlled behaviour of $(N_i, M_i)$ is the same as $(N_0, M_0)$

*if* $\beta_{Ai} > \beta_0$ *then* the controlled behaviour of $(N_i, M_i)$ is more permissive than $(N_0, M_0)$

*end if*

4) $N_A{:} = N_i$ ; $M_A{:} = M_i$
5) $i = i + 1$.
6) *If* $i \leq n$ *then* go to step 2.

**Output**: *If* $(j > 0)$

*then* Output = an *LPN*, denoted by a net system $(N_A, M_A)$, controlled by $k$ necessary

*CP*s; there are $j$ redundant *CP*s;

*if* $\beta_A = \beta_0$ *then* the controlled behaviour of $(N_A, M_A)$ is the same as $(N_0, M_0)$

*if* $\beta_A > \beta_0$ *then* the controlled behaviour of $(N_A, M_A)$ is more permissive than $(N_0, M_0)$

*else* there is no redundant *CP*s and therefore Output = Input;

**end Algorithm A**

**Algorithm B: Back-to-Front (*BTF*) redundancy test for *LES* of FMS.**

**Input**: A live Petri net model (*LPN*), denoted by a net system $(N_0, M_0)$, of an FMS, controlled by *nCP*s; $CP = \{C_1, C_2, \ldots, C_n\}$;

1) [Initialize] $N_B{:} = N_0$; $M_B{:} = M_0$; $i = n$; $l = 0$; $m = 0$;
2) Remove $C_i$ from $(N_B, M_B)$. Denote the resultant net system by $(N_i, M_i)$.

```
3) Check the liveness property of $(N_i, M_i)$, compute the reachability graph $(R_i)$ of $(N_i, M_i)$
   and define $\beta_{Bi}$, i.e., the number of reachable markings of $R_i$;
If $(N_i, M_i)$ is NOT LIVE
then put $C_i$ back into $(N_i, M_i)$; $m = m + 1$; which means that $C_i$ is necessary to keep the PN
model live,
else [i.e., If$(N_i, M_i)$ is LIVE], $l = l + 1$; which means that $C_i$ is redundant,
if $\beta_{Bi} = \beta_0$ then the controlled behaviour of $(N_i, M_i)$ is the same as $(N_0, M_0)$
if $\beta_{Bi} > \beta_0$ then the controlled behaviour of $(N_i, M_i)$ is more permissive than $(N_0, M_0)$
end if
4) $N_B := N_i$ ; $M_B := M_i$
5) $i = i - 1$.
6) If $i \neq 0$ then go to step 2.
Output: If $(l > 0)$
then Output = an LPN, denoted by a net system $(N_B, M_B)$,controlled by $m$ necessary CPs;
there are $l$ redundant CPs;
if $\beta_B = \beta_0$ then the controlled behaviour of $(N_B, M_B)$ is the same as $(N_0, M_0)$
if $\beta_B > \beta_0$ then the controlled behaviour of $(N_B, M_B)$ is more permissive than $(N_0, M_0)$
else there is no redundant CPs and therefore Output = Input;
```

**end Algorithm B**

The Redundancy Test Algorithm makes use of both Algorithms A and B. The former tests each *CP* starting from number 1 to the end, i.e., to *n*, while the latter tests each *CP* starting from number n to 1. Both tests may produce the same result or it may be possible to obtain different outcomes. It depends on the controlled live net system $(N_0, M_0)$ considered. Of course if there is no redundant *CP* in an *LPN*, then the Algorithm

Redundancy Test finds no redundant *CP*. In the existence of one or more redundant *CP* in an *LPN*, we may obtain the following results:

1. We may obtain the same set of redundant*CP*s and necessary *CP*s. In this case, the live behaviour of the Petri net model, controlled by the set of necessary*CP*s, may be the same as or more permissive than the original controlled net system, obtained with a smaller number of *CP*s.
2. We may obtain two different sets of redundant*CP*s and necessary *CP*s.The live behaviour of the Petri net model obtained with each set of necessary *CP*s, may be the same as or more permissive than the original controlled net system, obtained with a smaller number of *CP*s.

The Redundancy Test Algorithm is easy to use, very effective and straight forward. Its complexity is, however, exponential with respect to the net size since it requires generating the reachability graph. At the worst cases, Algorithm A and Algorithm B, i.e. BTF and FTB redundancy tests respectively, also exhibit the same exponential complexity. When dealing with a particular case, their performance may vary significantly. The Redundancy Test Algorithm is applicable to any *LPN* consisting of a *PNM*, prone to deadlocks, of an FMS, controlled by means of a set of*CP*s. It has been applied to a number of *LPN* currently available within the Petri net based deadlock prevention/liveness enforcing literature with success. The liveness property can be checked and the reachability analysis can be carried out by currently available Petri net analysis tools. In this work, INA [14] is used.

## 2.7 PETRI NETS MODELS OF MANUFACTURING SYSTEMS

Petri nets are characterized by their ability to represent operation sequences, concurrency, conflict, mutual exclusion, and synchronization in a system. These features make them a promising tool for describing and analyzing DES including automated manufacturing systems, as seen in Fig. 2.6 [18].

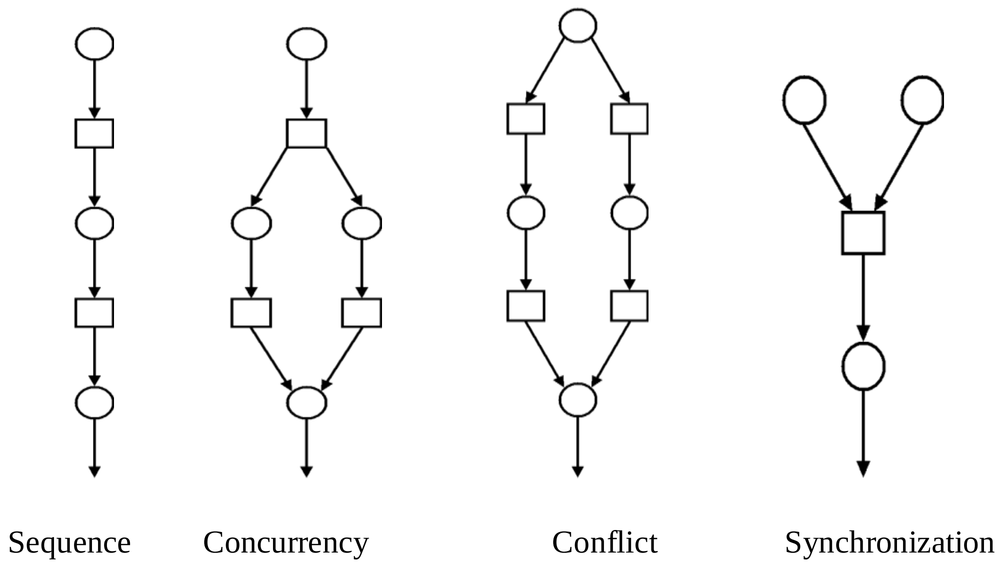


Fig. 2.6. Petri net representations of system features from [18].

1. **Sequence:** The operation of DESs or manufacturing systems occurs in a sequential manner. For example, in Fig. 2.5 the following events occur in sequential order before the raw part P2 to become finished product. “robot upload M2”, “M2 processing P2”, “robot download M2 and upload M1”, “M1 processing P2”, and “Robot download M1” occur in a sequential way.
2. **Concurrency:** In manufacturing systems, many operations occur simultaneously in order. For example, in Fig. 2.5, “P1 is machined in M3” and “P2 is machined in M1” can occur concurrently.
3. **Conflict:** In manufacturing systems, conflict may occur when many events share same resources at the same time. For example, in Fig. 2.5, initially, the robot may upload M1 or M3. A simple way to resolve the conflict is to assign a priority level to each of conflicted events.
4. **Synchronization:** In manufacturing systems, the occurrence of events is aperiodic. In the case of the parts assembly, there is a time variation from one machine to another so that parts are produced asynchronously. For example, in Fig. 2.5, the events “M1 is processing a part” and “the robot is downloading M3” are asynchronous.

5. **Event Driven:** In manufacturing systems, the behavior of a system can be described by a discrete set. That is to say the effect of an event may propagate through the system. For example, in Fig. 2.5, uploading M1 with a raw part of P1 makes the robot from being available to be occupied, implying the change of system states.

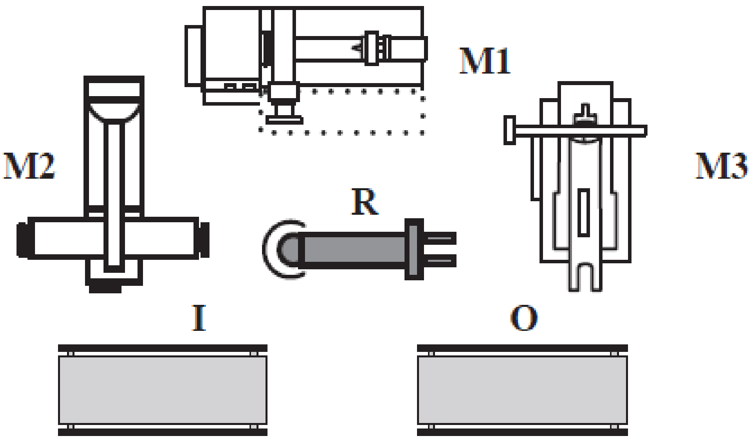


Fig. 2.7. A manufacturing system from [18].

# CHAPTER 3

# A DIVIDE AND CONQUER SYNTHESIS APPROACH FOR DEADLOCK PREVENTION IN FMS

In this chapter, a divide and conquer synthesis approach is proposed for deadlock prevention in FMS. An illustrative example from the literature is considered to show the applicability of the proposed method.

## 3.1 A DIVIDE AND CONQUER (DAC) SYNTHESIS APPROACH FOR DEADLOCK PREVENTION IN FMS

It is assumed that a Petri net model (PNM) of an FMS suffering from deadlocks is given. The proposed method requires the use of reachability graph of the given PNM. The RG of the PNM consists of a dead zone (DZ) and live zone (LZ). The DZ consists of deadlock states and bad states of the RG while LZ consists of good states representing the optimal solution. Therefore, the control policy in this case depends on the removal of the DZ from the RG.

In PNM of an FMS, places are categorized into three groups: sink/source places ($P_{S/S}$), resource places ($P_R$) and activity places ($P_A$). Resource places correspond to the shared/non shared resources. Activity places correspond to the part of production by a resource while sink/source places correspond to the maximum number simultaneous activities that can take place in production sequences. In order to simplify the computation, Petri net reduction approach is used as in [14]. Therefore, to obtain the LES of a complex PNM, it is divided into small connected sub-nets. Each connected sub-net suffering from deadlocks is then used to compute the LES for the PNM. The RG of each sub-net suffering from deadlock is divided into DZ and LZ. All states in the DZ are considered as bad markings (BM) and they are prevented from being reached using an invariant based control method [15].

Next, the computation of monitors is followed for bigger (reduced) sub-nets. Previously computed monitors are included within the bigger (reduced) sub-nets based on proposed method. This process keeps the DZ of the bigger (reduced) sub-nets smaller compared with the original uncontrolled sub-nets. When all (reduced) sub-nets are live we obtain a set of monitors (control places), which are included within the PNM to obtain a partially controlled PNM (pCPNM). A new set of monitors is also computed for the pCPNM. Finally, a live controlled Petri net system is obtained. The redundancy check [19] is used to remove redundant monitors.

### 3.2.1 The DAC Algorithm

Definition 3.0 $[PI]_{map}$ is the set of marked activity places of a bad state within the DZ. For example, assuming that a deadlock is reached when the activity places $p5$ and $p9$ are both marked with one token each i.e. $\mu5 = 1$ and $\mu9 = 1$. Then the place invariant (PI) is defined as $\mu5 + \mu9 \leq 1$. Therefore, the marked activity places of this PI are $[PI]_{map}$= {p5, $p9$}.

**Algorithm: Divide-and-conquer Based liveness Enforcing supervisor Design**

---

**Input:** The *PNM* of an FMS prone to deadlocks;

**Output:** A live controlled *PNM* for the FMS;

1) Divide the given PNM into *connected* sub-nets $SN_{ij}$, $i = 1, 2 \ldots, I$; $j = 1, 2 \ldots, J$.

for ($i = 1$; $i \leq I$; $i$ ++)

{

for ($j = 1; j \leq J; j$ ++)

*Extract* the connected sub-net $SN_{ij}$ from *PNM*;

}

/*$I$ is the number of total shared resources; $i$ is the index value showing the number shared resources in a connected sub-net; $J$ is the total number of connected sub-nets for the considered $i$; $j$ is the index value showing the current connected sub-net for the considered $i$. Note that $I$ is known beforehand but $J$ is defined from the considered *PNM*.*/

2) *Compute* the $RG_{ij}$ of all connected sub-nets $SN_{ij}$, and then divide these sub-nets into two sets $SN = SN_L \, U \, SN_D$ (*SN:* the set of all connected sub-nets; $SN_L$*:* the set of connected live sub-nets; $SN_D$ : the set of connected sub-nets prone to deadlock). Let $K$ be the number of all connected sub-nets in $SN_D$ and then rename the set of connected sub-nets prone to deadlock from 1 to $K$, i.e., $SN_D = SN_1, SN_2, \ldots, SN_K$

3) *Define* the set of activity places exists within $SN_D$

4) *For* ($k = 1; k \leq K; k$ ++)

{

4.1 *If* there are previously computed monitors and *if* the set of marked activity places of a sub-set of $SN_k$, i.e., $[\mathrm{PI}]_{\mathrm{map}} \subseteq [SN_k]_{\mathrm{ap}}$,

*Then Obtain* the partially controlled $SN_k$, called $pCSN_k$, by including all previously computed monitors whose $[\mathrm{PI}]_{\mathrm{map}} \subseteq [SN_k]_{\mathrm{ap}}$, within $SN_k$;

Compute the $RG_k$ of $pCSN_k$,

*If* $pCSN_k$ is live,

*Then Consider* the next *SN*, i.e. $SN_{k+1}$: goto Step 4.1

*Else Define* the $LZ_k$ and the $DZ_k$ of the $RG_k$

*Else Make use of* the $RG_k$ together with its $LZ_k$ and $DZ_k$ computed in Step 2.

4.2 *Define* a *PI* for each *BM* within the $DZ_k$, from the sub-set of the marked activity places of the *BM*;

4.3 *Compute* a monitor (control place) *C*, for each *PI* using the invariant-based control method [12]

4.4 *Obtain* the live controlled sub-net $CSN_k$ by including all computed monitors for $SN_k$, within $pCSN_k$ (resp. $SN_k$)

4.5 *If* the live controlled sub-net $CSN_k$ includes more than 1 monitor *C*, then the redundancy check using the method proposed [25] is used to identify the set of necessary monitors for $CSN_k$

}

5) *Obtain* the partially controlled *PNM*, called *pCPNM*, by including all computed monitors in Step 4 within the *PNM*

6) *Compute* the *RG* of the *pCPNM*,

*If pCPNM* is live, *then* the *PNM* is live, therefore go to Step 10

*Else* Define the LZ and the DZ of the *pCPNM*

7) *Define* a *PI* for each *BM* within the *DZ*, from the sub-set of the marked activity places of the *BM*;

8) *Compute* a monitor *C*, for each *PI* using the simplified invariant-based control method [12]

9) *Obtain* the live controlled *PNM* by including all computed monitors within the *PNM*

10) *Carry out* the redundancy check using the method of [13] to find out the set of necessary monitors for *PNM*

11) Exit

**End of the Algorithm**

First of all it is checked to see if the *RG* of the given PNM is the same as the *RG* of the PNM with no *sink/source places*. If this is true, then the connected sub-nets to be constructed need not to have *sink/source places*. Otherwise all sub-nets will include them. Connected sub-nets $SN_{ij}$, $i = 1, 2 \ldots, I$; $j = 1, 2, \ldots, J$, include SR (shared resource) place(s) with their input/output transitions and input and output arcs, to and from these input/output transitions. They also include activity places with their input/output arcs present between the input/output transitions of SR place(s). *INA* [14] is used for the *RG* analysis of PNMs. *DZ* is then considered as the collection of all bad markings $BM_i$, i= 1, 2, … for each connected sub-net. Now let us apply the proposed algorithm to the Petri net model of an FMS prone to deadlocks.

## 3.3 AN FMS ILLUSTRATIVE EXAMPLE

To show the applicability of the proposed method here is illustrative example from [8] is considered. Fig. 3.1 shows a System of Simple Sequential Process with Resources ($S^3$PR) model of an FMS with deadlocks. In this PNM there are six activity places $P_A$ = {p2-p4 and p6-p8}, three shared resource places $P_R$ = {p9-p11}, and two sink/source places $P_{S/S}$ = {p1, p5}. The *RG* of this PNM is computed by using INA [14]. It contains 20 states, 5 of which are bad states. Then, the optimal solution should provide a live system behavior with 15 good states.

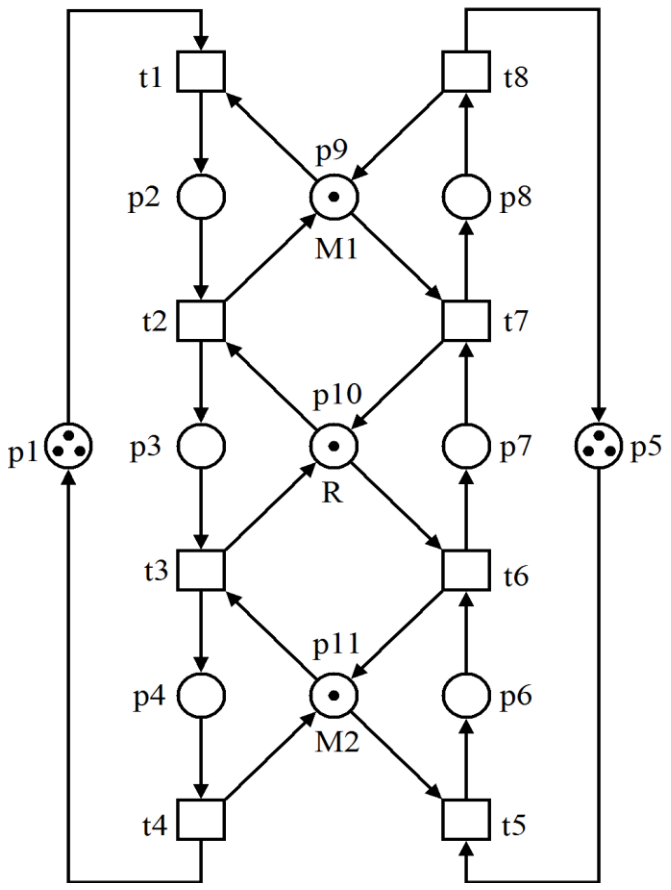


Fig. 3.1. $S^3PR$ model of an FMS with deadlocks from [21].

As the *RG* of the given *PNM* computed, it can be observed it is the same as the *RG* of the *PNM* without *sink/source places*. Therefore, the connected sub-nets constructed here have no *sink/source places*.

**Step 1.** The *PNM* is divided into five *connected* sub-nets $SN_{ij}$, There are three connected sub-nets with one shared resource place (i.e. $SN_{1,1}$, $SN_{1,2}$, $SN_{1,3}$ ) as shown in Fig. 3.2 and two connected sub-nets with two shared resource places (i.e. $SN_{2,1}$, $SN_{2,2}$) as shown in Fig. 3.3.

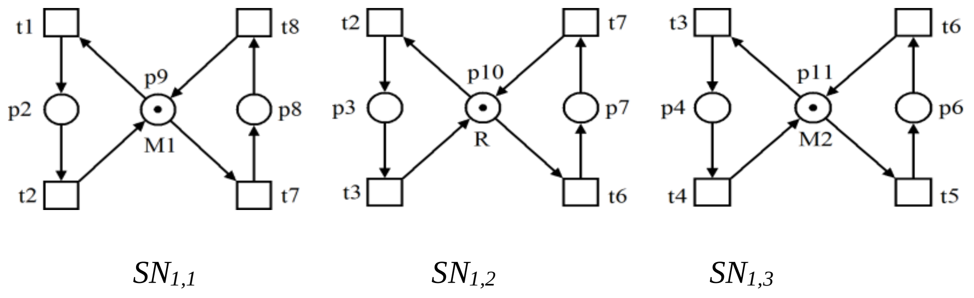


Fig. 3.2. Three connected subnets with one shared resource place.

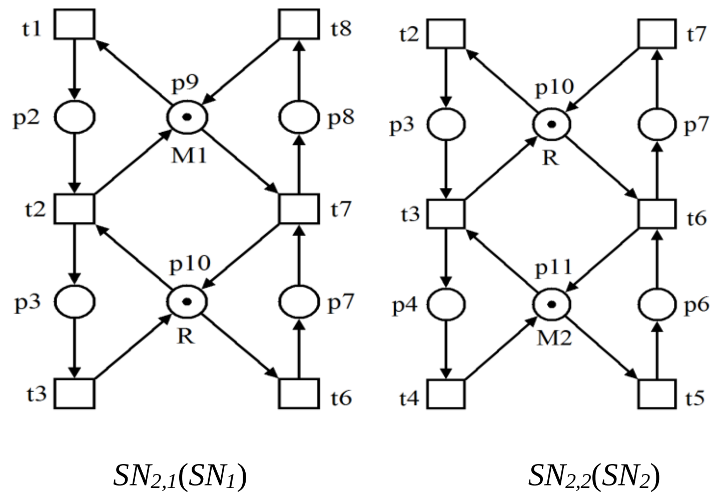


Fig. 3.3. Two connected subnets with two shared resources places.

**<u>Step 2</u>.** As the $RG_{\mathrm{ij}}$ of all connected sub-nets $SN_{ij}$ computed, the following results are obtained: $SN_L$ = {$SN_{1,1}$, $SN_{1,2}$, $SN_{1,3}$} and $SN_D$ = {$SN_{2,1}$, $SN_{2,2}$} which are renamed as $SN_1$ and $SN_2$. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected sub-nets $SN_1$ and $SN_2$ and the *PNM* are all provided in Table 3.1.

Table 3.1. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected sub-nets $SN_1$, $SN_2$ and the *PNM*.

| #S | $SN_1$ | $SN_2$ | *PNM* |
|---|---|---|---|
| *RG* | 8 | 8 | 20 |
| *DZ* | 1 | 1 | 5 |
| *LZ* | 7 | 7 | 15 |

**Step 3.** The set of activity places exist within the connected sub-nets $SN_1$, $SN_2$ and the *PNM* are all shown in Table 3.2. For example the set of activity places in $SN_1$ is $[SN_1]_{ap}$ = {p2, p3, p7, p8}.

Table 3.2. The set of activity places exist within the connected sub-nets $SN_1$, $SN_2$ and the *PNM*.

| NET | p2 | p3 | p4 | p6 | p7 | p8 |
|---|---|---|---|---|---|---|
| $SN_1$ | 1 | 1 | | | 1 | 1 |
| $SN_2$ | | 1 | 1 | 1 | 1 | |
| *PNM* | 1 | 1 | 1 | 1 | 1 | 1 |

**Step 4**. ($k$ = 1); The *RG* of the $SN_1$ consists of 8 states, with $BM_1$ constituting the *DZ* and 7 good states (*GS*s) representing the *LZ*. The marking of the activity places of the $BM_1$ is shown below:

| State nr. | p2 | p3 | p7 | p8 |
|---|---|---|---|---|
| 5 | 1 | 0 | 1 | 0 |

In order not to reach $BM_1$, the following place invariant is established: $PI_1 = \mu_2 + \mu_7 \leq 1$. $[PI_1]_{map}$ = {p2, p7}. The monitor *C1* to enforce $PI_1$ is computed as follows:

$$D_{C\,1} = -L_{PI1} \,.\, D_{PI1}$$

$$\mu_{c1(0)} = 1$$

where $L_{PI1}$ is the place invariant related place and is given by:

$$L_{PI1} = \begin{array}{cc} p2 & p7 \\ \left[1\right. & \left.1\right] \end{array}$$

and $D_{PI1}$ is the incidence matrix of the *PI* related places.

$$D_{PI1} = \begin{array}{c|cccc} & t1 & t2 & t6 & t7 \\ \hline p2 & 1 & -1 & 0 & 0 \\ p7 & 0 & 0 & 1 & -1 \end{array}$$

Therefore $D_{C1} = -L_{PI1} \,.\, D_{PI1}$

$$D_{C1} = -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = -\begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = \begin{array}{cccc} t1 & t2 & t6 & t7 \\ -1 & 1 & -1 & 1 \end{array}$$

The computed monitor C1 is shown below.

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C1 | t2, t7 | t1, t6 | 1 |

It is verified that the controlled $SN_1$, obtained by adding the monitor *C1* to the uncontrolled $SN_1$ is live with 7 good states. This is the optimal live behavior for the controlled $SN_1$.

(*k* = 2); The set of marked activity places of the $PI_1$ of the previously computed *C1* is not a sub-set of $[SN_2]_{ap}$. The *RG* of the $SN_2$ consists of 8 states, with $BM_2$ constituting the *DZ* and 7 *GS*s representing the *LZ*. The marking of the activity places of the $BM_2$ is shown below:

| State nr. | p3 | p4 | p6 | p7 |
|---|---|---|---|---|
| 5 | 1 | 0 | 1 | 0 |

Therefore, in order not to reach $BM_2$, the following place invariant is established: $PI_2 = \mu_3 + \mu_6 \leq 1$. $[PI_2]_{map}$ = {p3, p6}. The monitor *C2* to enforce $PI_2$ is computed as follows:

$D_{C\,2} = -L_{PI2}\,.\,D_{PI2}$

$\mu_{c2(0)} = 1$

$$L_{PI2} = \begin{array}{cc} p3 & p6 \\ \left[1\right. & \left.1\right] \end{array}$$

$$D_{PI2} = \begin{array}{c} \\ p3 \\ p6 \end{array} \begin{array}{cccc} t2 & t3 & t5 & t6 \\ \end{array}$$
$$D_{PI2} = \begin{array}{c|cccc} & t2 & t3 & t5 & t6 \\ \hline p3 & 1 & -1 & 0 & 0 \\ p6 & 0 & 0 & 1 & -1 \end{array}$$

Therefore $D_{C2} = -L_{PI2}\,.\,D_{PI2}$

$$D_{C2} = -[1 \quad 1]\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C2} = -[1 \quad -1 \quad 1 \quad -1]$$

$$D_{C2} = \begin{array}{cccc} t2 & t3 & t5 & t6 \\ \left[-1\right. & 1 & -1 & \left.1\right] \end{array}$$

The computed monitor C2 is shown below.

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C2 | t3, t6 | t2, t5 | 1 |

It is verified that the controlled *SN*$_2$, obtained by adding the monitor *C2* to the uncontrolled *SN*$_2$ is live with 7 good states. This is the optimal live behavior for the controlled *SN*$_2$.

**<u>Step 5</u>**. The *pCPNM* is obtained by including monitors *C1* and *C2* as computed in Step 4, within the *PNM*.

**<u>Step 6</u>**. The *RG* of the *pCPNM* consists of 16 states, with *BM*$_3$ constituting the *DZ* and 15 *GS*s representing the *LZ*. The marking of the activity places of *BM*$_3$ are shown below:

| State nr. | p2 | p3 | p4 | p6 | p7 | p8 |
|---|---|---|---|---|---|---|
| 9 | 1 | 0 | 0 | 1 | 0 | 0 |

**Step 7.** In order not to reach $BM_5$, the following place invariant is established: $PI_3 = \mu_2 + \mu_7 \leq 1$.

**Step 8**. The monitor *C3* to enforce $PI_3$ is computed as follows:

$$D_{C\,3} = -L_{PI3} \,.\, D_{PI3}$$

$$\mu_{c3(0)} = 1$$

$$L_{PI3} = \begin{array}{cc} p2 & p6 \\ \left[1\right. & \left.1\right] \end{array}$$

$$D_{PI3} = \begin{array}{c|cccc} & t1 & t2 & t5 & t6 \\ p2 & 1 & -1 & 0 & 0 \\ p6 & 0 & 0 & 1 & -1 \end{array}$$

Therefore $D_{C3} = -L_{PI3} \,.\, D_{PI3}$

$$D_{C3} = -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C3} = -\begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}$$

$$D_{C3} = \begin{array}{cccc} t1 & t2 & t5 & t6 \\ \left[-1\right. & 1 & -1 & \left.1\right] \end{array}$$

The computed monitor C3 is shown below:

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C3 | t2, t6 | t1, t5 | 1 |

**Step 9**. It is verified that the controlled *PNM*, obtained by adding monitor *C3* to the *pCPNM* is live with 15 good states. This is the optimal live behavior for the controlled *PNM*. The optimally controlled *PNM* is shown in Fig. 3.3. Table 3 briefly shows the liveness enforcing procedure applied for the $S^3$PR Petri net model. In this table the last column shows the number of unreachable states. As the elements of this column are all zero this indicates that there are no good states lost due to included monitors.

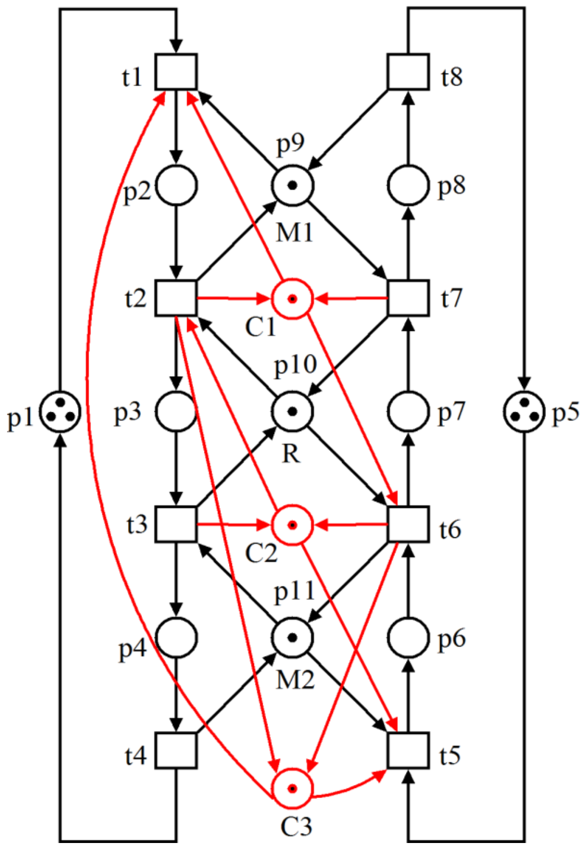


Fig. 3.4. The optimally controlled $S^3PR$.

Table 3.3. The Divide-and-conquer based liveness enforcing procedure applied for *$S^3PR$ PNM*.

| *steps* | *net* | *Included C* | *# states within controlled net* | | | *Computed C* | *# states within controlled net* | |
|---|---|---|---|---|---|---|---|---|
| | | | *RG* | *DZ* | *LZ* | | *RG* | *UR* |
| *4.1* | *$SN_1$* | - | *8* | *1* | *7* | *C1* | *7* | *0* |
| *4.2* | *$SN_2$* | - | *8* | *1* | *7* | *C2* | *7* | *0* |
| *5-9* | *pCPNM* | *C1, C2* | *16* | *1* | *15* | *C3* | *15* | *0* |
| *10-11* | *PNM* | *C1, C2, C3* | *15* | - | *15* | - | *15* | *0* |

**Step 10**. The redundancy check carried out show that there is no redundant monitor, i.e., the computed monitors *C1*, *C2* and *C3* are all necessary.

**Step 11**. Exit.

# CHAPTER 4

# APPLICATION EXAMPLES

## 4.1 INTRODUCTION

In this chapter, three example Petri net models suffering from deadlock problems from the literature are considered in order to show the applicability of the proposed divide and conquer method. The performance comparison of the proposed method and the previous methods in terms of behavioral permissiveness are also included.

## 4.2 FMS APPLICATION EXAMPLE 1

Fig. 4.1 shows an $S^3PGPR^2$ Petri net model of an FMS taken from [15]. In this *PNM*, there are nine activity places $P_A$ = {p11-p15, p21-p24}, three shared resource places $P_R$ = {p31-p33}, and two sink/source places $P_{S/S}$ = {p10, p20}. The *RG* of this *PNM* is computed by using INA [14]. It contains 363 states, 40 of which are bad states. Then, the optimal solution should provide a live system behavior with 323 good states.

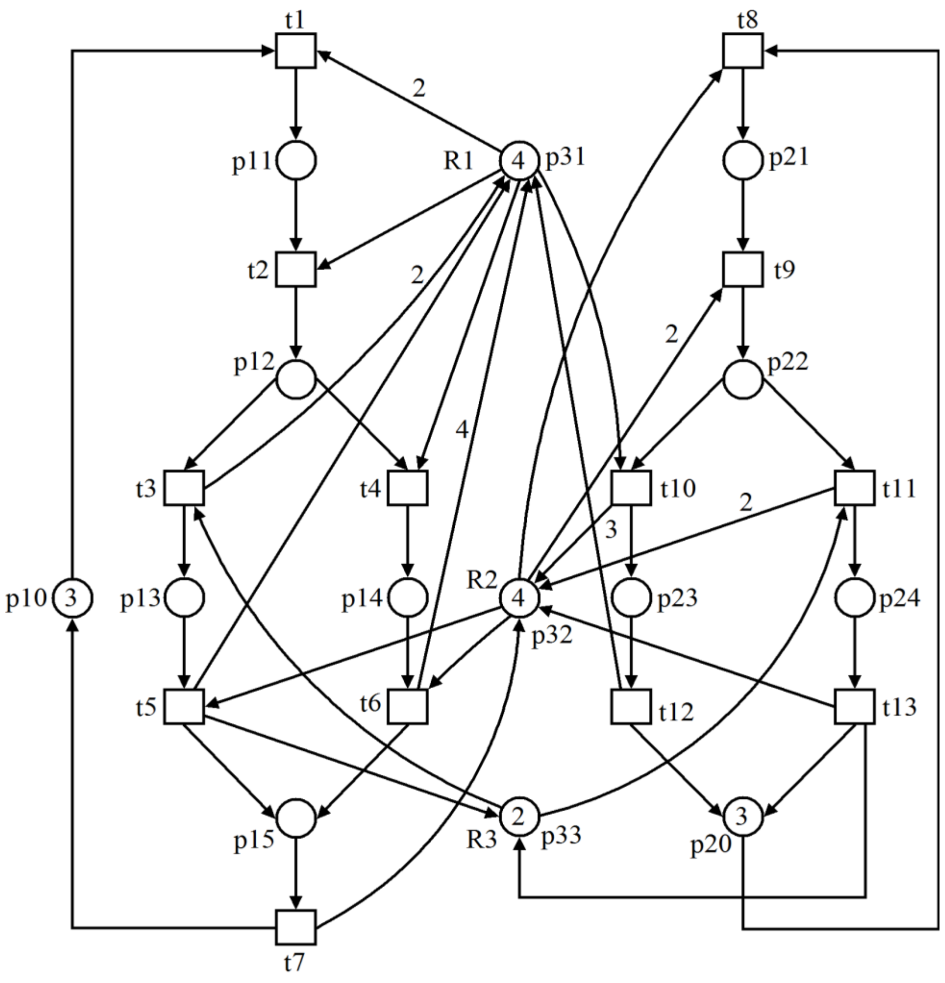
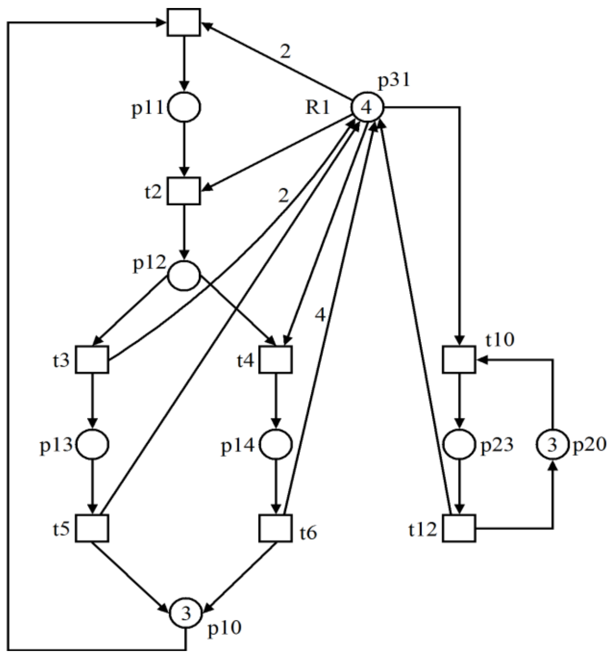


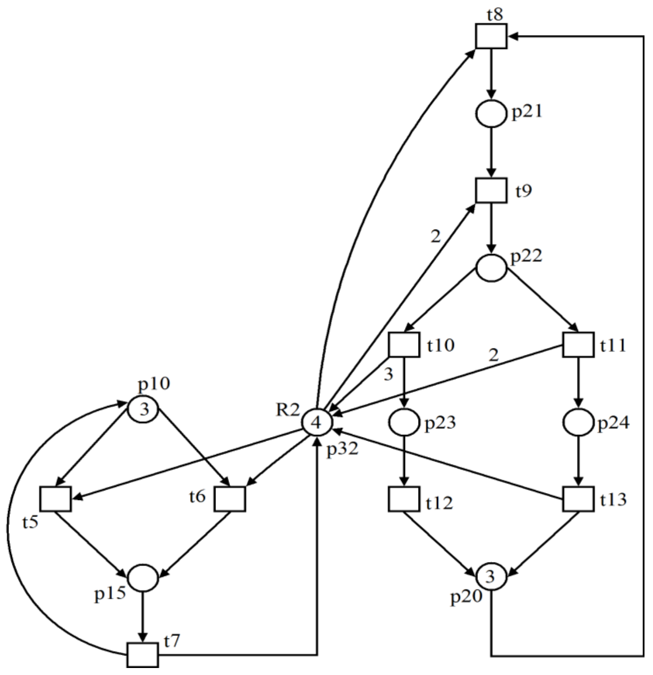
Fig. 4.1. $S^3PGPR^2$ Petri net model of an FMS taken from [15].

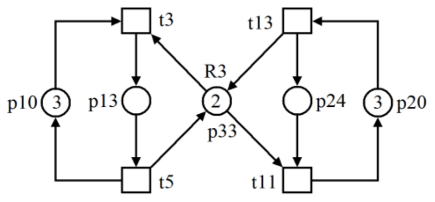
It is observed that the *RG* of the given *PNM* is not the same as the *RG* of the *PNM* with no *sink/source places*. Therefore, the *sink/source places* will not be removed.

**Step 1**. The *PNM* is divided into six *connected* sub-nets $SN_{ij}$. There are three connected sub-nets with one shared resource place (i.e. $SN_{1,1}$, $SN_{1,2}$, $SN_{1,3}$), as shown in Fig. 4.2 and three connected sub-nets with two shared resource places (i.e. $SN_{2,1}$, $SN_{2,2}$, $SN_{2,3}$), as shown in Fig. 4.3.



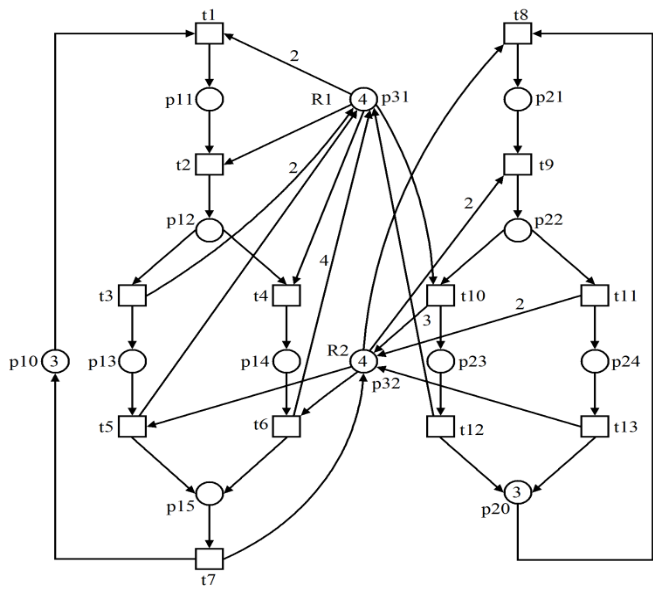
*SN*$_{1,1}$(*SN*$_{1}$)



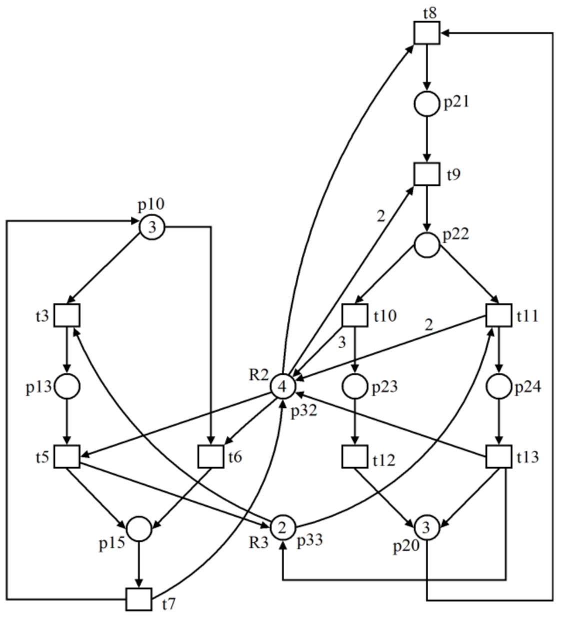


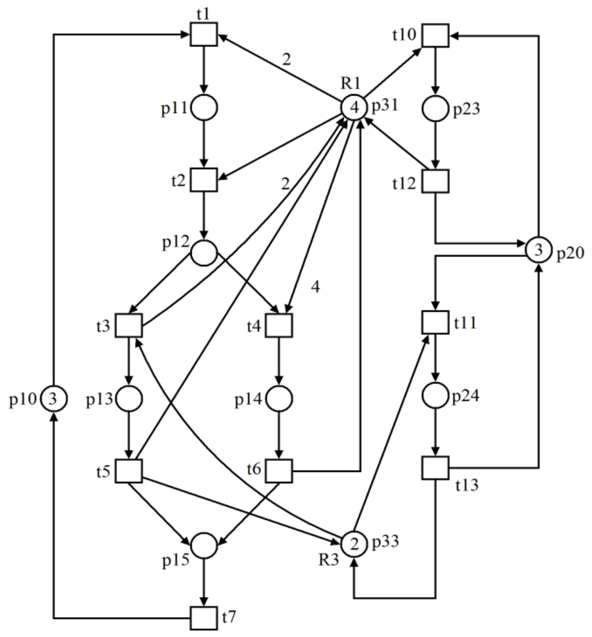
*SN*$_{1,2}$(*SN*$_{2}$) *SN*$_{1,3}$

Fig. 4.2. Three connected sub-nets with one shared resource place.



*SN$_{2,1}$(SN$_3$)*





*SN$_{2,2}$(SN$_4$)* *SN$_{2,3}$(SN$_5$)*

Fig. 4.3. Three connected sub-nets with two shared resource places.

**<u>Step 2</u>.** After computing the $RG_{ij}$ of all connected sub-nets $SN_{ij}$ computed, the following results are obtained: $SN_L = \{SN_{1,3}\}$, $SN_D = \{SN_{1,1}, SN_{1,2}, SN_{2,1}, SN_{2,2}, SN_{2,3}\}$ renamed as $SN_1$, $SN_2$, $SN_3$, $SN_4$, $SN_5$. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected sub-nets $SN_1$, ..., $SN_5$ and the *PNM* are all provided in Table 4.1.

Table 4.1 The number of states exists within the *RG*, the *DZ* and the *LZ* of the $SN_1$, ..., $SN_5$ and the *PNM*.

| #S | $SN_1$ | $SN_2$ | $SN_3$ | $SN_4$ | $SN_5$ | *PNM* |
|---|---|---|---|---|---|---|
| *RG* | 22 | 74 | 414 | 158 | 133 | 363 |
| *DZ* | 1 | 2 | 39 | 6 | 6 | 40 |
| *LZ* | 21 | 72 | 375 | 152 | 127 | 323 |

**<u>Step 3</u>.** The set of activity places exist within the $SN_1$, ..., $SN_5$ and the *PNM* are shown in Table 4.2.

Table 4.2 The set of activity places exist within the $SN_1$, ..., $SN_5$ and the *PNM*.

| NET | p11 | p12 | p13 | p14 | p15 | p21 | p22 | p23 | p24 |
|---|---|---|---|---|---|---|---|---|---|
| $SN_1$ | 1 | 1 | 1 | 1 | | | | 1 | |
| $SN_2$ | | | | | 1 | 1 | 1 | 1 | 1 |
| $SN_3$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $SN_4$ | | | 1 | | 1 | 1 | 1 | 1 | 1 |
| $SN_5$ | 1 | 1 | 1 | 1 | | | | | 1 |
| *PNM* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**<u>Step 4</u>**. ($k$ = 1); The $RG_1$ of the $SN_1$ consists of 22 states, with $BM_1$ constituting the *DZ* and 21 good states (*GS*s) representing the *LZ*. The marking of the activity places of the $BM_1$ is shown below:

| State nr. | p11 | p12 | p13 | p14 | p23 |
|---|---|---|---|---|---|
| 23 | 2 | 0 | 0 | 0 | 0 |

In order not to reach $BM_1$, the following place invariant is established: $PI_1 = \mu_{11} \leq 1$. $[PI_1]_{\text{map}} = \{p11\}$. The monitor *C1* to enforce $PI_1$ is computed as follows:

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C1 | t2 | t1 | 1 |

It is verified that the controlled $SN_1$ is obtained by adding the monitor *C1* to the uncontrolled $SN_1$ is live with 21 good states. This is the optimal live behavior for the controlled $SN_1$.

($k$ = 2); the set of marked activity places of the $PI_1$ of the previously computed *C1* is not a subset of $[SN_2]_{\text{ap}}$. The *RG* of the $SN_2$ consists of 74 states, with $BM_2$ and $BM_3$ constituting the *DZ* and 72 *GS*s representing the *LZ*. The markings of the activity places of the $BM_2$ and $BM_3$ are shown below:

| State nr. | p15 | p21 | p22 | p23 | p24 |
|---|---|---|---|---|---|
| 12 | 0 | 3 | 0 | 0 | 0 |
| 19 | 0 | 1 | 0 | 0 | 1 |

In order not to reach $BM_2$ and $BM_3$, the following place invariants are established: $PI_2 = \mu_{21} \leq 2$. $[PI_2]_{\text{map}} = \{p21\}$ and $PI_3 = \mu_{21} + \mu_{24} \leq 1$. $[PI_3]_{\text{map}} = \{p21, p24\}$. The monitors *C2* and *C3* to enforce $PI_2$ and $PI_3$ are computed as follows:

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C2 | t9 | t8 | 2 |
| C3 | t13 | t8 | 1 |

It is verified that the controlled $SN_2$ is obtained by adding the monitor *C2* to the uncontrolled $SN_2$ is live with 72 good states, this is the optimal live behavior for the controlled $SN_2$. *C3* is redundant.

($k$ = 3); It is verified that $[PI_1]_{\text{map}} \subseteq [SN_3]_{\text{ap}}$ and $[PI_2]_{\text{map}} \subseteq [SN_3]_{\text{ap}}$. Therefore, the $pCSN_3$ is obtained by including *C1* and *C2* within $SN_3$. The controlled $SN_3$ is equal to the $pCSN_3$ is live with 375 good states. This is the optimal live behavior for the controlled $SN_3$.

($k$ = 4); It is verified that $[PI_2]_{\text{map}} \subseteq [SN_4]_{\text{ap}}$. Therefore, the *pCSN₄* is obtained by including *C2* within $SN_4$. The controlled $SN_4$ is equal to the $pCSN_4$ is live with 152 good states. This is the optimal live behavior for the controlled $SN_4$.

($k$ = 5); It is verified that $[PI_1]_{\text{map}} \subseteq [SN_5]_{\text{ap}}$. Therefore, the $pCSN_5$ is obtained by including *C1* within $SN_5$. The controlled $SN_5$ is equal to the $pCSN_5$ is live with 127 good states. This is the optimal live behavior for the controlled $SN_5$.

**Step 5**. The *pCPNM* is obtained by including the necessary monitors *C1* and *C2* computed in Step 4, within the *PNM*.

**Step 6**. The *RG* of the *pCPNM* consist of 324 states, with $BM_3$ constituting the *DZ* and 323 *GSs* representing the *LZ*. The marking of the activity place of $BM_3$ is shown below:

| State nr. | p11 | p12 | p13 | p14 | p15 | p21 | p22 | p23 | p24 |
|---|---|---|---|---|---|---|---|---|---|
| 66 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 0 | 0 |

**Step 7**. In order not to reach $BM_3$, the following place invariant is established: $PI_3 = \mu_{11} + \mu_{13} + \mu_{21} + \mu_{22} \leq 4$. $[PI_3]_{\text{map}} = \{p11, p13, p21, p22\}$.

**Step 8**. The monitor *C3* to enforce $PI_3$ is computed as follows:

| $C_i$ | $^{\bullet}C_i$ | $C_i^{\bullet}$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C3 | t2, t5, t10,t11 | t1, t3, t8 | 4 |

**Step 9**. It is verified that the controlled *PNM*, obtained by adding monitors *C1*, *C2*, and *C3* to the *pCPNM* is live with 322 good states. This is the optimal live behavior for the controlled $S^3PGPR^2$ model. The optimally controlled PNM is shown in Fig. 4.4. Table 4.3 briefly shows the liveness enforcing procedure applied for the $S^4PR$ *PNM* model. In this table the last column shows the number of unreachable states. As the elements of this

column are all zero this indicates that there are no good states lost due to included monitors.

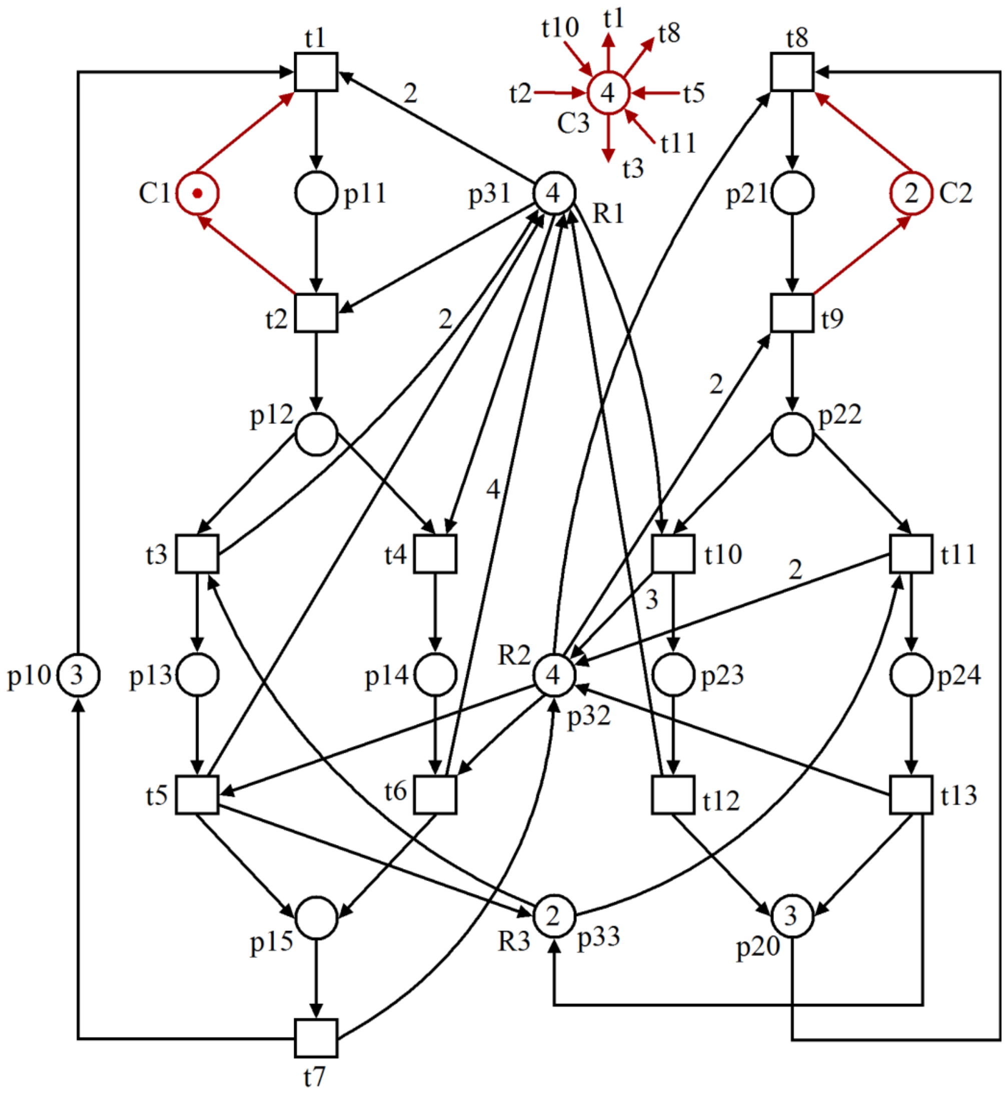


Fig. 4.4. The optimally controlled $S^3PGPR^2$.

Table 4.3 The divide-and-conquer based liveness enforcing procedure applied for $S^3PGPR^2$

| *steps* | net | included *C* | #states within controlled net *RG* | *DZ* | *LZ* | computed *C* | # states within controlled net *RG* | *UR* |
|---|---|---|---|---|---|---|---|---|
| *4.1* | $SN_1$ | - | *23* | *1* | *22* | *C1* | *22* | *0* |
| *4.2* | $SN_2$ | - | *78* | *2* | *76* | *C2* *Rd:1* | *76* | *0* |
| *4.3* | $SN_3$ | *C1, C2* | *375* | - | *375* | - | *375* | *0* |
| *4.4* | $SN_4$ | *C2* | *190* | - | *190* | - | *190* | *0* |
| *4.5* | $SN_5$ | *C1* | *157* | - | *157* | - | *157* | *0* |
| *5-9* | *pCPNM* | *C1, C2* | *324* | *1* | *322* | *C3* | *322* | *1* |
| *9-10* | *PNM* | *C1, C2, C3* | *322* | - | *322* | - | *322* | *0* |

*Rd.: the number of redundant monitors.*

**Step 10.** The redundancy check carried out shows that the computed monitors *C1*, *C2* and *C3* are necessary.

**Step 11.** Exit.

## 4.3 FMS APPLICATION EXAMPLE 2

Fig. 4.5 shows an $S^4PR$ Petri net model of an FMS taken from [22]. In this *PNM*, there are eleven activity places $P_A$ = {p1-p11}, six resource places $P_R$ = {p21-p26}, and two sink/source places $P_{S/S}$ = {p31, p32}. p21, p22 and p24 are non-shared resource places while p22, p25 and p26 are shared resource places. The *RG* of this *PNM* is computed by using INA [20]. It contains 282 states, 77 of which are bad states. Then, the optimal solution should provide a live system behavior with 205 good states.

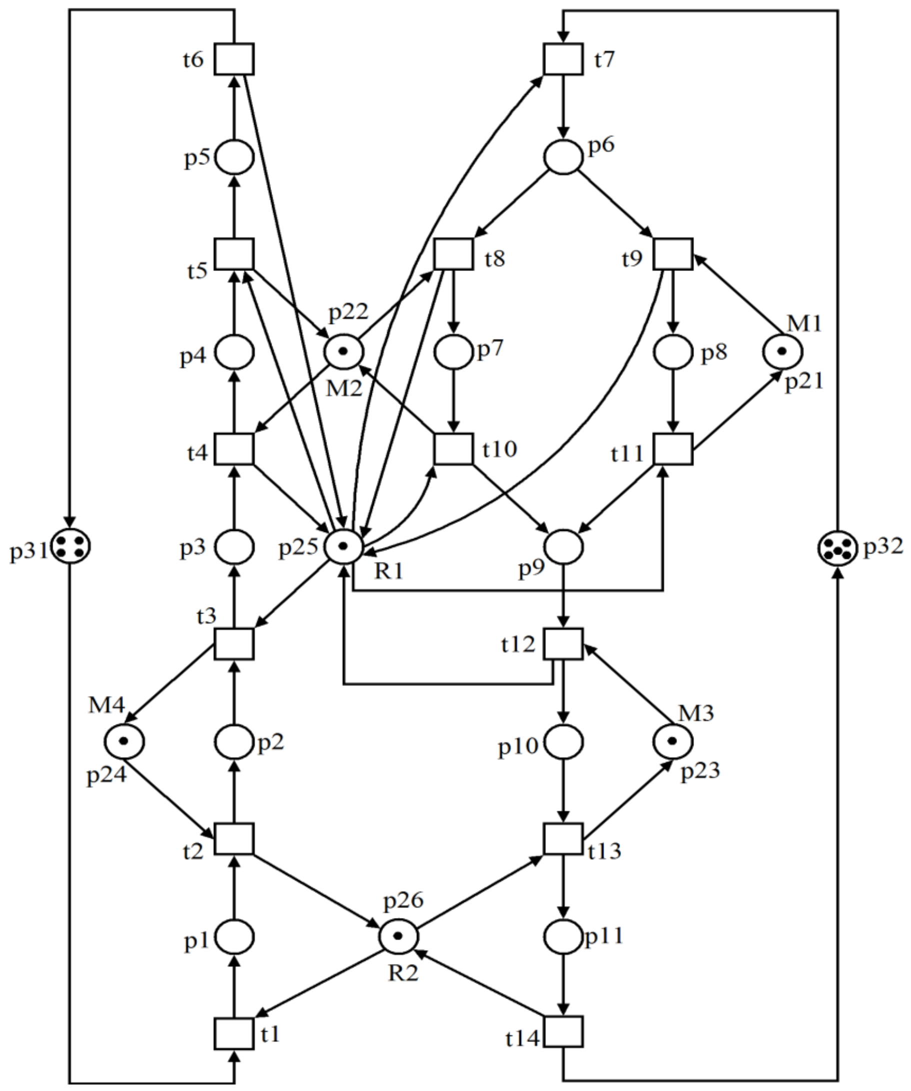
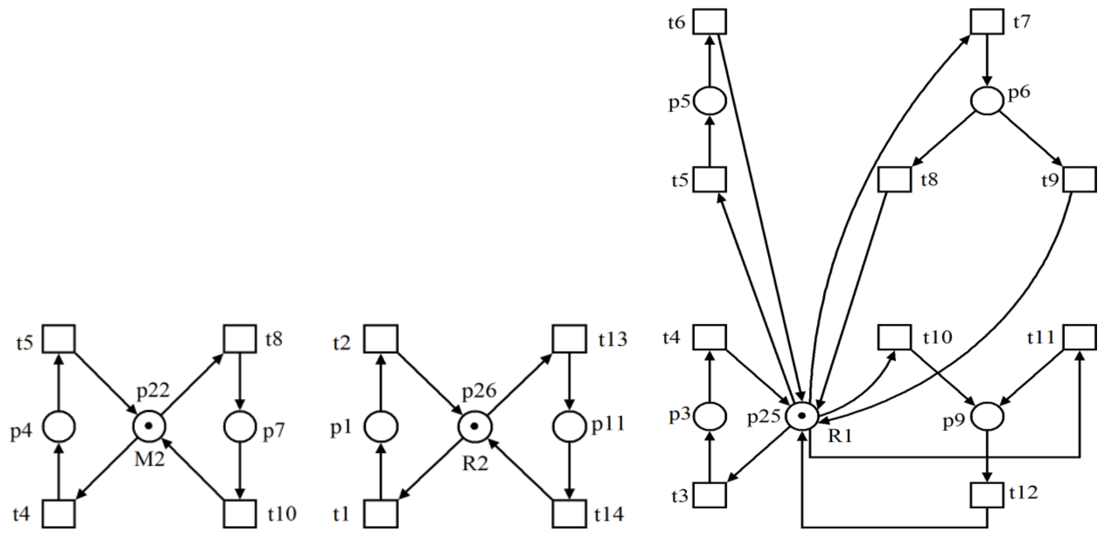


Fig. 4.5. $S^4PR$ Petri net model of an FMS taken from [22].

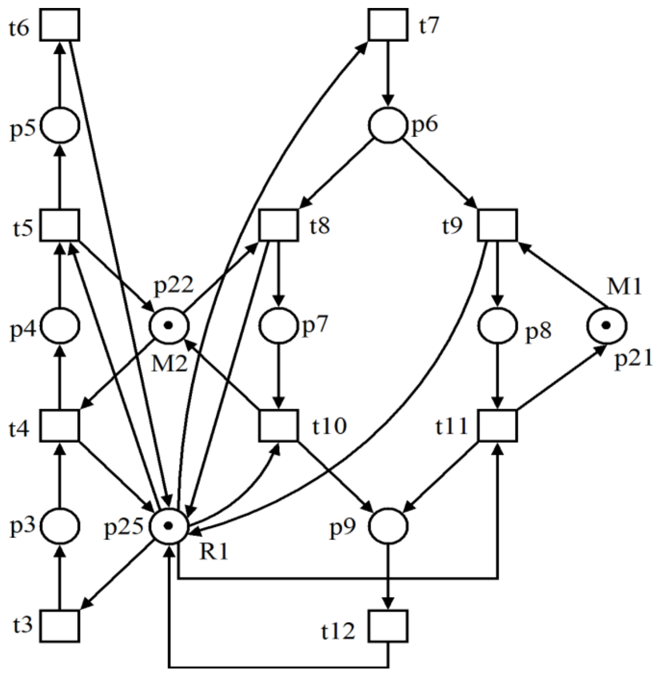
It is observed that the *RG* of the given *PNM* is the same as the *RG* of the *PNM* with no *sink/source places*. Therefore, the connected subnets constructed here have no *sink/source places*.

**<u>Step 1</u>.** The *PNM* is divided into four *connected* sub-nets $SN_{ij}$. There are three connected sub-nets with one shared resource place (i.e. $SN_{1,1}$, $SN_{1,2}$, $SN_{1,3}$) as shown in Fig. 4.6 and one connected sub-net with two shared resource places (i.e. $SN_{2,1}$ ) as shown in Fig. 4.7.



*SN*$_{1,1}$ *SN*$_{1,2}$ *SN*$_{1,3}$

Fig. 4.6. Three connected sub-nets with one shared resource place.



*SN*$_{2,1}$(*SN*$_{1}$ )

Fig. 4.7. One connected sub-net with two shared resource places.

**Step 2.** After computing the $RG_{\mathrm{ij}}$ of all connected sub-nets $SN_{ij}$ computed, the following results are obtained: $SN_L = \{SN_{1,1}, SN_{1,2}, SN_{1,3}\}$, $SN_D = \{SN_{2,1}\}$ renamed as $SN_1$. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected sub-net $SN_1$ and the *PNM* are all provided in Table 4.4.

Table 4.4 The number of states exists within the *RG*, the *DZ* and the *LZ* of the $SN_1$ and the *PNM*.

| #S | $SN_1$ | *PNM* |
|---|---|---|
| *RG* | 24 | 282 |
| *DZ* | 6 | 77 |
| *LZ* | 18 | 205 |

**Step 3.** The set of activity places exist within the $SN_1$ and the *PNM* are shown in Table 4.5.

Table 4.5 The set of activity places exist within the $SN_1$ and the *PNM*.

| NET | p1 | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p9 | p10 | p11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $SN_1$ | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| *PNM* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Step 4**. ($k$ = 1); The $RG_1$ of the $SN_1$ consists of 24 states, with $BM_1$, …, $BM_6$ constituting the *DZ* and 18 good states (*GS*s) representing the *LZ*. The markings of the activity places of the $BM_1$, $BM_2$, …, $BM_6$ are shown in Table 4.6.

Table 4.6 The markings of the activity places of the $BM_1$, …, $BM_6$.

| State nr. | p3 | p4 | p5 | p6 | p7 | p8 | p9 |
|---|---|---|---|---|---|---|---|
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 14 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 15 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 19 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 22 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

In order not to reach bad markings $BM_1$, …, $BM_6$ the following place invariants are established as shown in Table 4.7.

Table 4.7 The place invariants of the $BM_1$, …, $BM_6$.

| | |
|---|---|
| $PI_1$ | $\mu_3 + \mu_4 \leq 1$ |
| $PI_2$ | $\mu_3 + \mu_4 + \mu_8 \leq 2$ |
| $PI_3$ | $\mu_3 + \mu_7 \leq 1$ |
| $PI_4$ | $\mu_6 + \mu_7 + \mu_8 \leq 2$ |
| $PI_5$ | $\mu_3 + \mu_4 + \mu_7 \leq 2$ |
| $PI_6$ | $\mu_4 + \mu_6 + \mu_8 \leq 2$ |

Monitors *C1*, ..., *C6* to enforce place invariants $PI_1$, ..., $PI_6$ for the $SN_1$ are computed as shown in Table 4.8.

Table 4.8 Computed monitors for the $SN_1$.

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C1 | t5 | t3 | 1 |
| C2 | t10, t11 | t7 | 2 |
| C3 | t4, t10 | t3, t8 | 1 |
| C4 | t5, t8, t11 | t4, t7 | 2 |
| C5 | t5, t11 | t3, t9 | 2 |
| C6 | t4, t10, t11 | t3, t8, t9 | 2 |

It is verified that controlled $SN_1$, obtained by adding monitors *C1*, *C2*, *C3* and *C4* to the uncontrolled $SN_1$, is live with 18 good states. This is the optimal live behavior for the controlled $SN_1$. The monitors *C5* and *C6* are redundant.

**Step 5**. The *pCPNM* is obtained by including the necessary monitors *C1*, *C2*, *C3* and *C4* computed in Step 4, within the *PNM*.

**Steps 6-8**. The *RG* of the *pCPNM* consists of 209 states, with $BM_5$, …, $BM_8$ constituting the *DZ* and 205 *GS*s representing the *LZ*. The markings of the activity places of the $BM_5$, …, $BM_8$ are shown in Table 4.9.

Table 4.9 The markings of the activity places of the *$BM_5$*, …, *$BM_8$* of the *pCPNM*.

| State nr. | p1 | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p9 | p10 | p11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 97 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 102 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 105 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 115 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

In order not to reach bad markings *$BM_5$*, …, *$BM_8$* the following place invariants are established as shown in Table 4.10.

Table 4.10 The place invariants of the *$BM_5$*, …, *$BM_8$*.

| | |
|---|---|
| $PI_5$ | $\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$ |
| $PI_6$ | $\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 4$ |
| $PI_7$ | $\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 3$ |
| $PI_8$ | $\mu_1 + \mu_2 + \mu_8 + \mu_{10} \leq 3$ |

The computed monitors for the place invariants *$PI_5$, ..., $PI_8$* are shown in Table 4.11.

Table 4.11 The Computed monitors of the *PNM*.

| Ci | $^\bullet$Ci | Ci$^\bullet$ | $\mu$0(Ci) |
|---|---|---|---|
| C5 | t3, t13 | t1, t10, t11 | 3 |
| C6 | t3, t8, t11, t13 | t1, t7, t12 | 4 |
| C7 | t3, t12 | t1, t8, t11 | 3 |
| C8 | t3, t10, t13 | t1, t8 t12 | 3 |

**Step 9**. It is verified that the controlled *PNM*, obtained by adding monitors *C5*, ..., *C8* to the *pCPNM* is live with 205 good states. This is the optimal live behavior for the controlled *$S^4PR$* model. The optimally controlled PNM is shown in Fig. 4.8. Table 4.9 briefly shows the liveness enforcing procedure applied for the *$S^4PR$ PNM* model. In this table the last column shows the number of unreachable states. As the elements of this column are all zero this indicates that there are no good states lost due to included monitors.

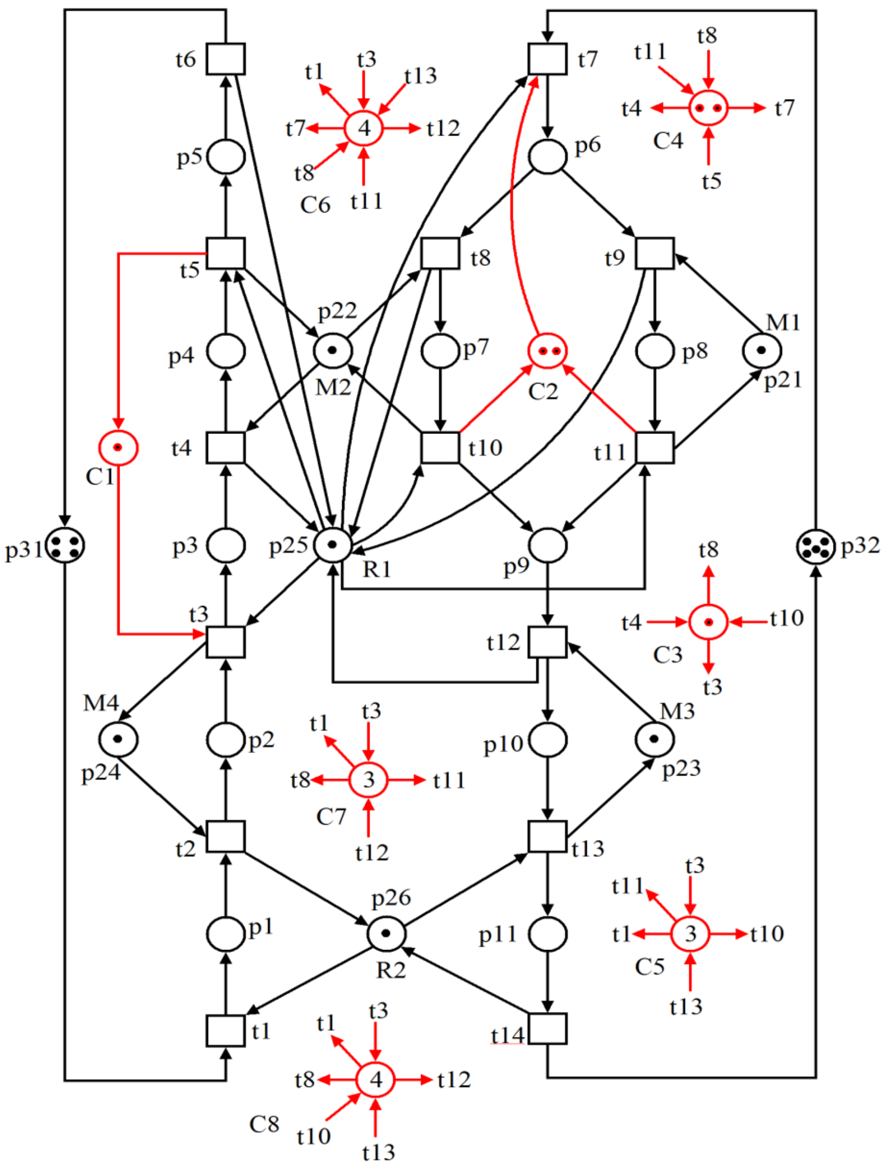


Fig. 4.8. The optimally controlled *$S^4PR$ PNM*.

Table 4.12 The divide-and-conquer based liveness enforcing procedure for *$S^4PR$ PNM*.

| *steps* | *net* | *Included C* | *# states within controlled net* | | | *Computed C* | *# states within controlled net* | |
|---|---|---|---|---|---|---|---|---|
| | | | *RG* | *DZ* | *LZ* | | *RG* | *UR* |
| *4.1* | *SN1* | - | *24* | *6* | *18* | *C1-C4 Rd:2* | *18* | *0* |
| *5-9* | *pCPNM* | *C1-C4* | *209* | *4* | *205* | *C5-C8* | *205* | *0* |
| *10-11* | *PNM* | *C1-C8* | *205* | - | *205* | - | *205* | *0* |

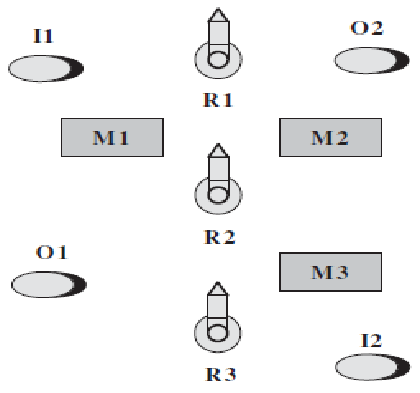
*Rd.: the number of redundant monitors.*

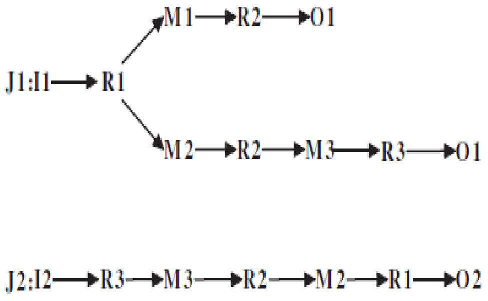
**<u>Step 10</u>**. The redundancy checks carried out and show that the computed monitors *C1*, *C2*, ..., *C8* are all necessary.

**<u>Step 11</u>**. Exit.

## 4.4 FMS APPLICATION EXAMPLE 3

A hypothetical FMS with its layout shown in Fig. 4.9 (a) and production routings in Fig. 4.9 (b) from [23] are considered as another application example. It consists of three robots R1-R3, each of which can hold two, one and three products respectively, and three machines M1-M3, each of which can process one, two and two products respectively. There are two loading buffers I1 and I2, and two unloading buffers O1 and O2 to load and unload the FMS. There are two raw product types, namely J1 and J2, to be processed. For these raw product types the production cycles are as shown in Fig. 4.9 (b). According to the production cycles, a raw product J1 is taken from I1 by R1 and put in M1 or M2. After being processed by M1 it is then moved to O1 by R2 or after being processed by M2, it is moved to M3 by R2 and then moved to O1 by R3. A raw product J2 is taken from I2 to M3 by R3. After being processed by M3 it is then moved from M3 to M2 by R2. Finally, after being processed by M2 it is then moved from M2 to O2 by R1.





(a) (b)

Fig. 4.9. (a) Layout of an FMS (b) production routings.

The $S^3PR$ *PNM* of the FMS is shown in Fig. 4.10. In this *PNM* there are twelve activity places $P_A$ = {p2-p8, p10-p14}, five shared resource places $P_{SR}$ = {p15-p16, p19-p20}, one non-shared resource place p17, and two sink/source places $P_{S/S}$ = {p1, p9}. The *RG* of this *PNM* is computed by using INA [20]. It contains 14473 states, 1510 of which are bad states. Then, the optimal solution should provide a live system behavior with 12963 good states.

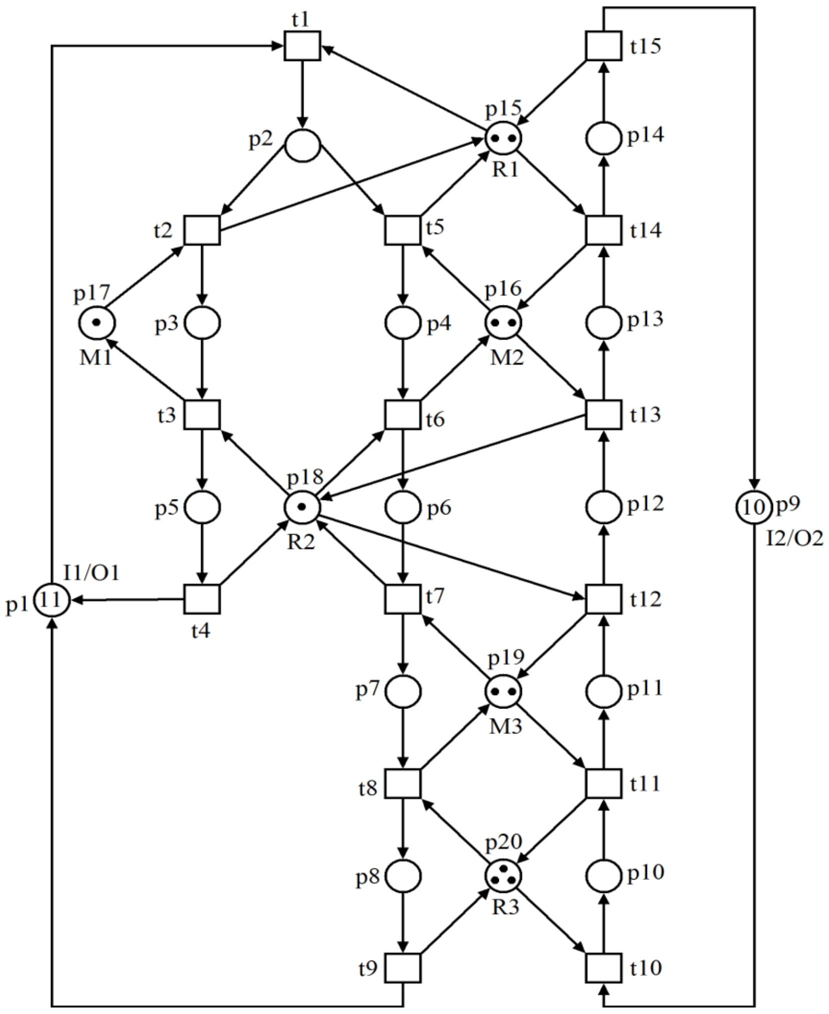


Fig. 4.10. $S^3PR$ ($N$, $M_0$) taken from [23].

It is observed that the *RG* of the given *PNM* is the same as the *RG* of the *PNM* with no *sink/source places*. Therefore, the connected subnets constructed here have no *sink/source places*.

**<u>Step 1</u>.** The *PNM* is divided into fourteen connected sub-nets $SN_{ij}$. There are five connected sub-nets with one shared resource place (i.e. $SN_{1,1}$-$SN_{1,5}$,), as shown in Fig. 4.11, four connected sub-nets with two shared resource places (i.e. $SN_{2,1}$-$SN_{2,4}$), as shown in Fig. 4.12, three connected sub-nets with three shared resource places (i.e. $SN_{3,1}$-$SN_{3,3}$), as shown in Fig. 4.13 and two connected sub-nets with four shared resource places (i.e. $SN_{4,1}$-$SN_{4,2}$), as shown in Fig. 4.14.

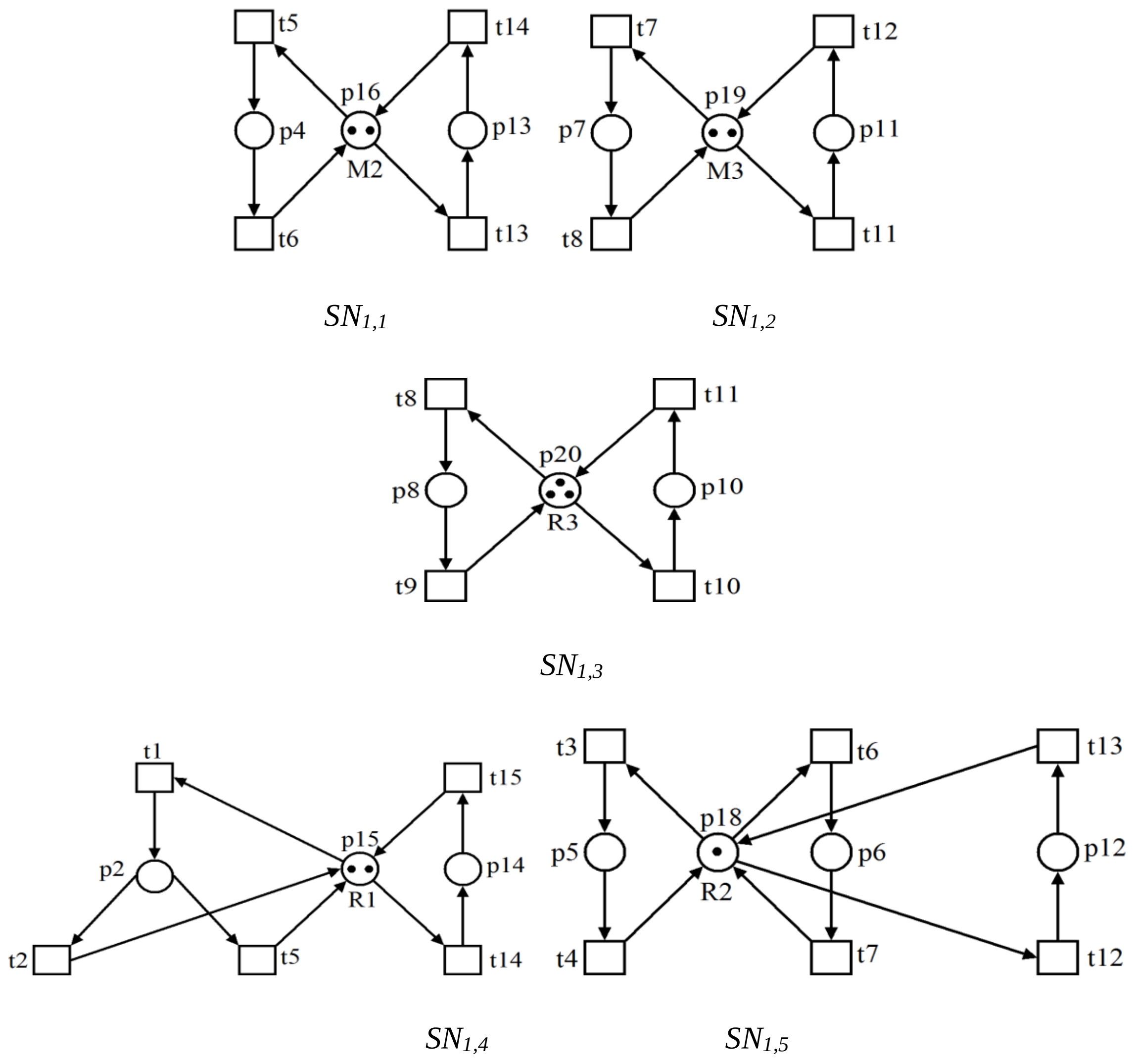
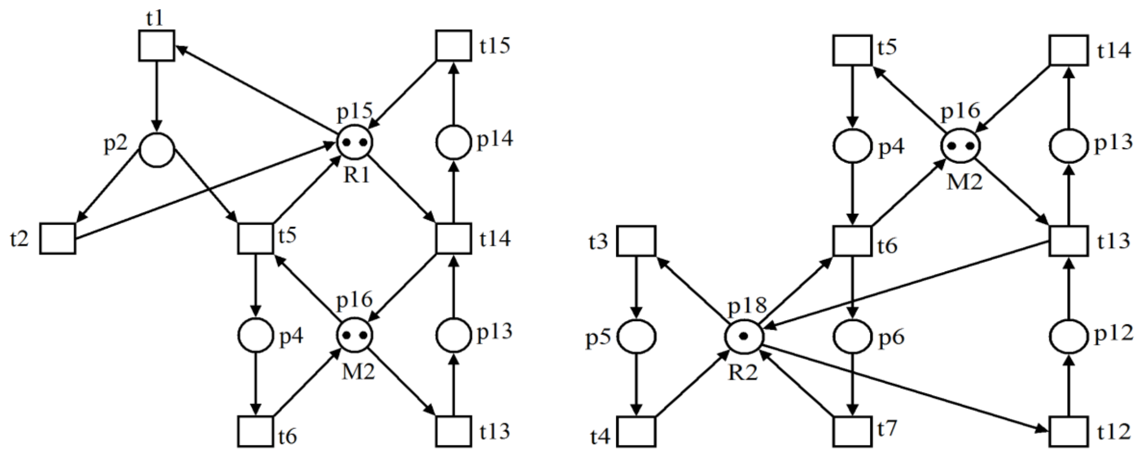
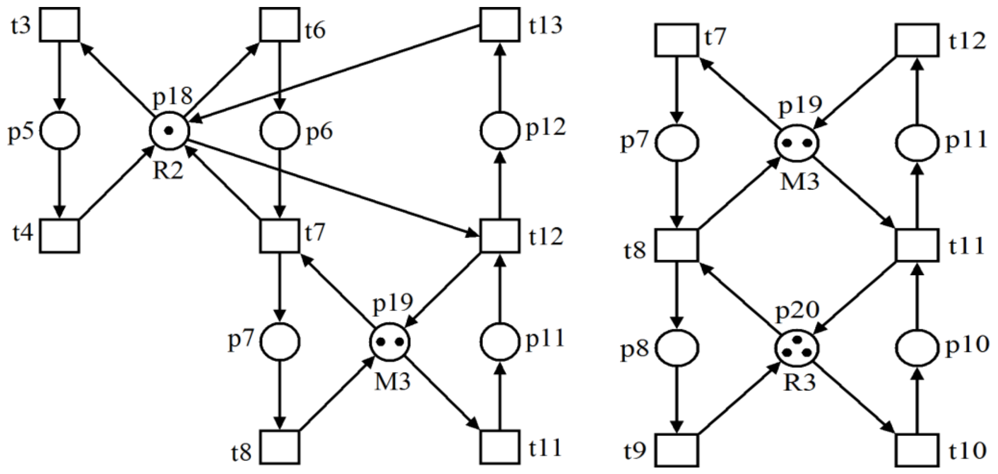


Fig. 4.11. Five connected sub-nets with one shared resource place.



*SN2,1* *SN2,2(SN1)*



*SN2,3(SN2)* *SN2,4(SN3)*

Fig. 4.12. Four connected sub-nets with two shared resource places.

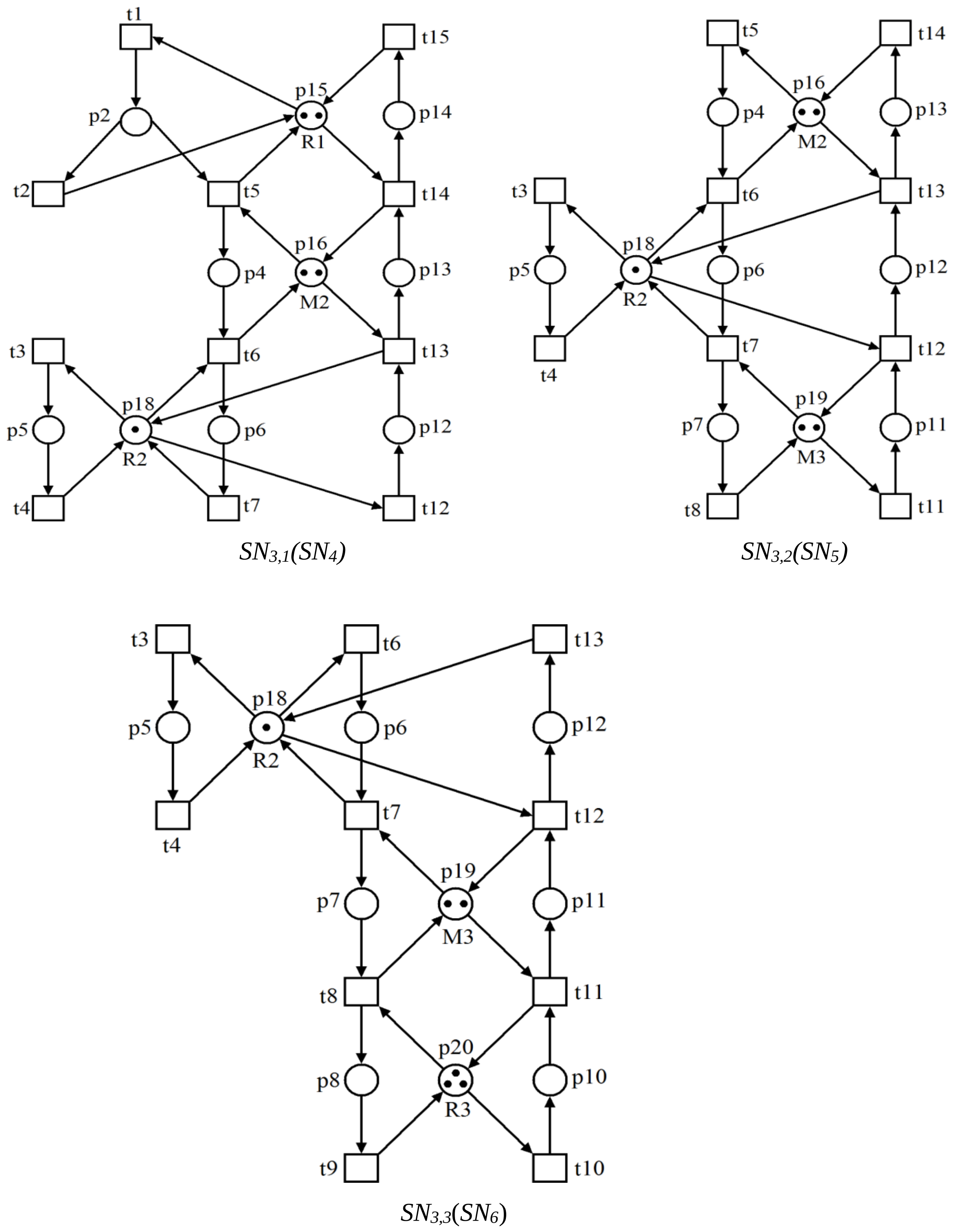


Fig. 4.13. Three connected sub-nets with three shared resource places.

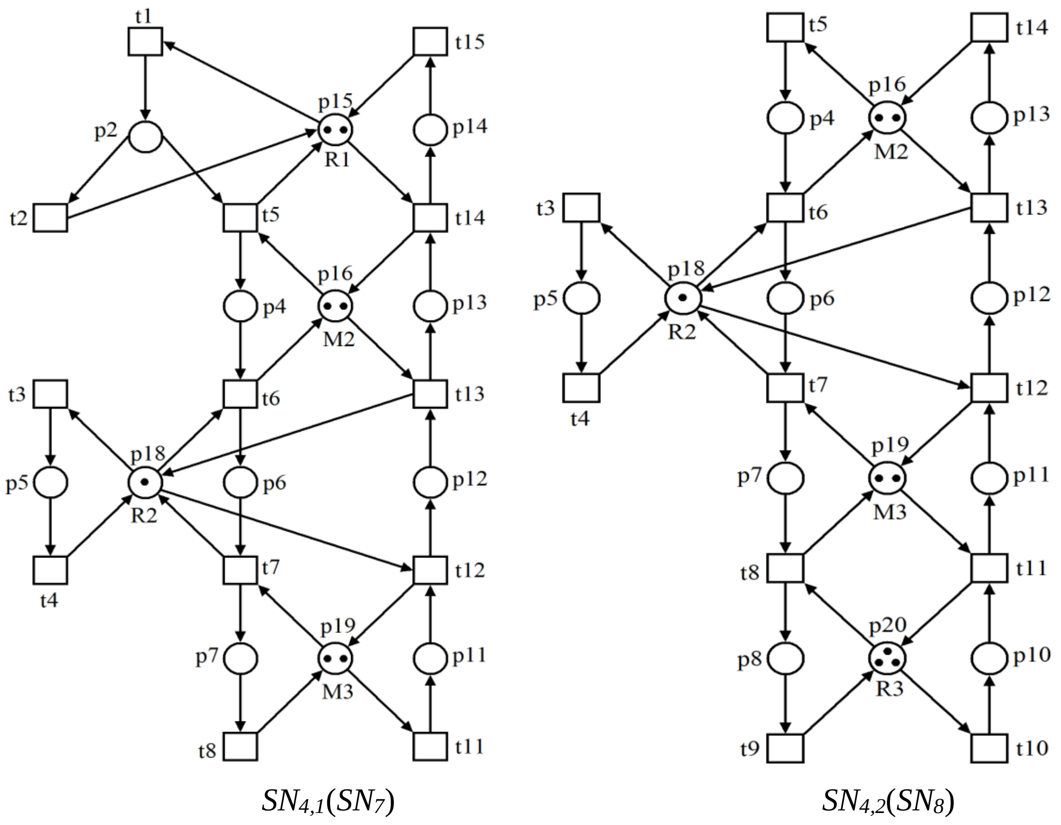


Fig. 4.14. Two connected sub-nets with four shared resource places.

**<u>Step 2</u>.** After computing the $RG_{\text{ij}}$ of all connected sub-nets $SN_{ij}$, the following results are obtained: $SN_L$ = {$SN_{1,1}$, $SN_{1,2}$, $SN_{1,3}$, $SN_{1,4}$, $SN_{1,5}$ and $SN_{2,1}$}, $SN_D$ = {$SN_{2,2}$, $SN_{2,3}$, $SN_{2,4}$, $SN_{3,1}$, $SN_{3,2}$, $SN_{3,3}$, $SN_{4,1}$, $SN_{4,2}$} that are renamed as $SN_1$, …, $SN_8$ respectively. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected subnets $SN_1$, $SN_2$ …, $SN_8$ and the *PNM* are all shown in Table 4.13.

Table 4.13. The number of states exists within the *RG*, the *DZ* and the *LZ* of the connected subnets $SN_1$, …, $SN_8$ and the *PNM*.

| *#S* | $SN_1$ | $SN_2$ | $SN_3$ | $SN_4$ | $SN_5$ | $SN_6$ | $SN_7$ | $SN_8$ | *PNM* |
|---|---|---|---|---|---|---|---|---|---|
| *RG* | *23* | *23* | *59* | *133* | *130* | *225* | *748* | *1268* | *14473* |
| *DZ* | *1* | *1* | *1* | *5* | *12* | *13* | *63* | *134* | *1510* |
| *LZ* | *22* | *22* | *58* | *128* | *118* | *212* | *685* | *1134* | *12963* |

**Step 3.** The set of activity places exist within the connected subnets $SN_1$, …, $SN_8$ and the *PNM* are all shown in Table 4.14. For example the set of activity places in $SN_1$ is $[SN_1]_{ap}$ = {p4, p5, p6, p12, p13}.

Table 4.14. The set of activity places exist within the subnets $SN_1$, …, $SN_8$ and the *PNM*.

| net | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p10 | p11 | p12 | p13 | p14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $SN_1$ | | | 1 | 1 | 1 | | | | | 1 | 1 | |
| $SN_2$ | | | | 1 | 1 | 1 | | | 1 | 1 | | |
| $SN_3$ | | | | | | 1 | 1 | 1 | 1 | | | |
| $SN_4$ | 1 | | 1 | 1 | 1 | | | | | 1 | 1 | 1 |
| $SN_5$ | | | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | |
| $SN_6$ | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| $SN_7$ | 1 | | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 |
| $SN_8$ | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| *PNM* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Step 4.** ($k$ = 1); The *RG* of the $SN_1$ consists of 23 states, with $BM_1$ constituting the *DZ* and 34 good states (*GS*s) representing the *LZ*. The marking of the activity places of the $BM_1$ is shown below:

| State nr. | p4 | p5 | p6 | p12 | p13 |
|---|---|---|---|---|---|
| 9 | 2 | 0 | 0 | 1 | 0 |

In order not to reach $BM_1$, the following place invariant is established: $PI_1 = \mu_4 + \mu_{12} \leq 2$. $[PI_1]_{map}$ = {p4, p12}. The monitor *C1* to enforce $PI_1$ is computed as follows:

| $C_i$ | $^{\bullet}C_i$ | $C_i^{\bullet}$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C1 | t6, t13 | t5, t12 | 2 |

It is verified that the controlled $SN_1$ is obtained by adding the monitor *C1* to the uncontrolled $SN_1$ is live with 22 good states. This is the optimal live behavior for the controlled $SN_1$.

($k$ = 2); the set of marked activity places of the $PI_1$ of the previously computed *C1* is not a subset of $[SN_2]_{ap}$. The *RG* of the $SN_2$ consists of 23 states, with $BM_2$ constituting the *DZ* and

22 *GS*s representing the *LZ*. The marking of the activity places of the $BM_2$ and $BM_3$ are shown below:

| State nr. | p5 | p6 | p7 | p11 | p12 |
|---|---|---|---|---|---|
| 15 | 0 | 1 | 0 | 2 | 0 |

In order not to reach $BM_{2,}$ the following place invariant is established: $PI_2 = \mu_{6} + \mu_{11} \leq 2$. $[PI_2]_{\text{map}} = \{p6, p11\}$. The monitor *C2* to enforce $PI_2$ is computed as follows:

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C2 | t7, t12 | t6, t11 | 2 |

It is verified that the controlled $SN_2$ is obtained by adding the monitor *C2* to the uncontrolled $SN_2$ is live with 22 good states, this is the optimal live behavior for the controlled $SN_2$.

($k = 3$); The set of marked activity places of the $PI_1$ and $PI_2$ of the previously computed *C1* and *C2* are not a subset of $[SN_3]_{\text{ap}}$. The *RG* of the $SN_3$ consists of 59 states, with $BM_3$ constituting the *DZ* and 58 *GS*s representing the *LZ*. The marking of the activity place of the $BM_3$ are shown below:

| State nr. | p7 | p8 | p10 | p11 |
|---|---|---|---|---|
| 27 | 2 | 0 | 3 | 0 |

In order not to reach $BM_3$, the following place invariant is established: $PI_3 = \mu_7 + \mu_{10} \leq 4$. $[PI_3]_{\text{map}} = \{p7, p10\}$. The monitor *C3* to enforce $PI_3$ is computed as follows:

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C3 | t8, t11 | t7, t10 | 4 |

The controlled $SN_3$ is obtained by adding the monitor *C3* to the uncontrolled $SN_3$ is live with 58 good states. This is the optimal live behavior for the controlled $SN_3$.

($k = 4$); It is verified that $[PI_1]_{\text{map}} \subseteq [SN_4]_{\text{ap}}$. Therefore, the $pCSN_4$ is obtained by including *C1* within $SN_4$. The controlled $SN_4$ is equal to the $pCSN_4$ is live with 128 good states. This is the optimal live behavior for the controlled $SN_4$.

($k$ = 5); It is verified that $[PI_1]_{map} \subseteq [SN_5]_{ap}$ and $[PI_2]_{map} \subseteq [SN_5]_{ap}$. Therefore, the $pCSN_5$, is obtained by including *C1* and *C2* within $SN_5$. The *RG* of the $pCSN_5$ consists of 119 states, with $BM_4$ constituting the *DZ* and 118 *GS*s representing the *LZ*. The marking of the activity places of the $BM_4$ is shown below:

| State nr. | p4 | p5 | p6 | p7 | p11 | p12 | p13 |
|---|---|---|---|---|---|---|---|
| 15 | 2 | 0 | 0 | 0 | 2 | 0 | 0 |

In order not to reach $BM_4$, the following place invariant is established: $PI_4 = \mu_4 + \mu_{11} \leq 3$. $[PI_4]_{map}$ = {p4, p11}. The monitor *C4* to enforce $PI_4$ is computed as follows:

| $C_i$ | $^{\bullet}C_i$ | $C_i^{\bullet}$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C4 | t6, t12 | t5, t11 | 3 |

It is verified that the controlled $SN_5$ is obtained by adding the monitor *C4* to the $pCSN_5$ is live with 118 good states. This is the optimal live behavior for the controlled $SN_5$.

($k$ = 6); It is verified that $[PI_2]_{map} \subseteq [SN_6]_{ap}$ and $[PI_3]_{map} \subseteq [SN_6]_{ap}$. Therefore, the $pCSN_6$, is obtained by including *C2* and *C3* within $SN_6$. The *RG* of the $pCSN_6$ consists of 213 states, with $BM_5$ constituting the *DZ* and 212 *GS*s representing the *LZ*. The marking of the activity places of the $BM_5$ is shown below:

| State nr. | p5 | p6 | p7 | p8 | p10 | p11 | p12 |
|---|---|---|---|---|---|---|---|
| 47 | 0 | 1 | 1 | 0 | 3 | 1 | 0 |

In order not to reach $BM_5$, the following place invariant is established: $PI_5 = \mu_6 + \mu_7 + \mu_{10} + \mu_{11} \leq 5$. $[PI_5]_{map}$ = {p6, p7, p10, p11}. The monitor *C5* to enforce $PI_5$ is computed as follows:

| $C_i$ | $^{\bullet}C_i$ | $C_i^{\bullet}$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C5 | t8, t12 | t6, t10 | 5 |

It is verified that the controlled $SN_6$ is obtained by adding the monitor *C5* to the $pCSN_6$ is live with 212 good states. This is the optimal live behavior for the controlled $SN_6$.

($k$ = 7); It is verified that $[PI_4]_{\text{map}} \subseteq [SN_7]_{\text{ap}}$ and $[PI_5]_{\text{map}} \subseteq [SN_7]_{\text{ap}}$. Therefore, the $pCSN_7$ is obtained by including *C1*, *C2* and *C5* within $SN_7$. The controlled $SN_7$ is equal to the $pCSN_7$ is live with 685 good states. This is the optimal live behavior for the controlled $SN_7$.

($k$ = 8); It is verified that $[PI_5]_{\text{map}} \subseteq [SN_8]_{\text{ap}}$ and $[PI_6]_{\text{map}} \subseteq [SN_8]_{\text{ap}}$. Therefore, the $pCSN_8$ is obtained by including *C1*, *C2*, *C3*, *C4* and *C5* within $SN_8$. The *RG* of the $pCSN_8$ consists of 1137 states, with $BM_6$, $BM_7$ and $BM_8$ constituting the *DZ* and 1134 *GS*s representing the *LZ*. The marking of the activity places of the $BM_6$, *BM7* and $BM_8$ are shown in Table 4.15.

Table 4.15. The markings of the activity places of the $BM_6$, $BM_7$ and $BM_8$.

| State nr. | p4 | p5 | p6 | p7 | p8 | p10 | p11 | p12 | p13 |
|---|---|---|---|---|---|---|---|---|---|
| 277 | 2 | 0 | 1 | 0 | 0 | 3 | 1 | 0 | 0 |
| 357 | 2 | 0 | 0 | 1 | 0 | 3 | 1 | 0 | 0 |
| 415 | 2 | 1 | 0 | 1 | 0 | 3 | 1 | 0 | 0 |

In order not to reach $BM_6$, $BM_7$ and $BM_8$, the following places invariant are established:

Table 4.16. Place invariants for $BM_6$, $BM_7$ and $BM_8$.

| | |
|---|---|
| $PI_6$ | $\mu_4 + \mu_6 + \mu_{10} + \mu_{11} \leq 6$ |
| $PI_7$ | $\mu_4 + \mu_7 + \mu_{10} + \mu_{11} \leq 6$ |
| $PI_8$ | $\mu_4 + \mu_5 + \mu_7 + \mu_{10} + \mu_{11} \leq 7$ |

The monitors *C6*, *C7* and *C8* to enforce $PI_6$, $PI_7$ and $PI_8$ are computed as shown in Table 4.17.

Table 4.17. Computed monitors for $PI_6$, $PI_7$ and $PI_8$.

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C6 | t7, t12 | t5, t10 | 6 |
| C7 | t6, t8, t12 | t5, t7, t10 | 6 |
| C8 | t4, t7, t12 | t3, t5, t10 | 7 |

It is verified that the controlled $SN_8$ is obtained by adding the monitor *C6* and *C7* to the uncontrolled $SN_8$ is live with 1134 good states. This is the optimal live behavior for the controlled $SN_8$. The monitor *C8* is redundant.

**Step 5.** The *pCPNM* is obtained by including all monitors, namely *C1*, …, *C7*, computed in step 4, within the *PNM*.

The RG of the *pCPNM* consist of 13055 states, with $BM_8$, ..., $BM_{102}$ (92 bad markings) constituting the *DZ* and 12963 *GS*s representing the *LZ*. In order to reduce the bad markings, we apply the Petri net reduction approach to the *pCPNM* to obtain the partially controlled reduced *PNM* (*pCRPNM*) model as shown in Fig. 4.15.

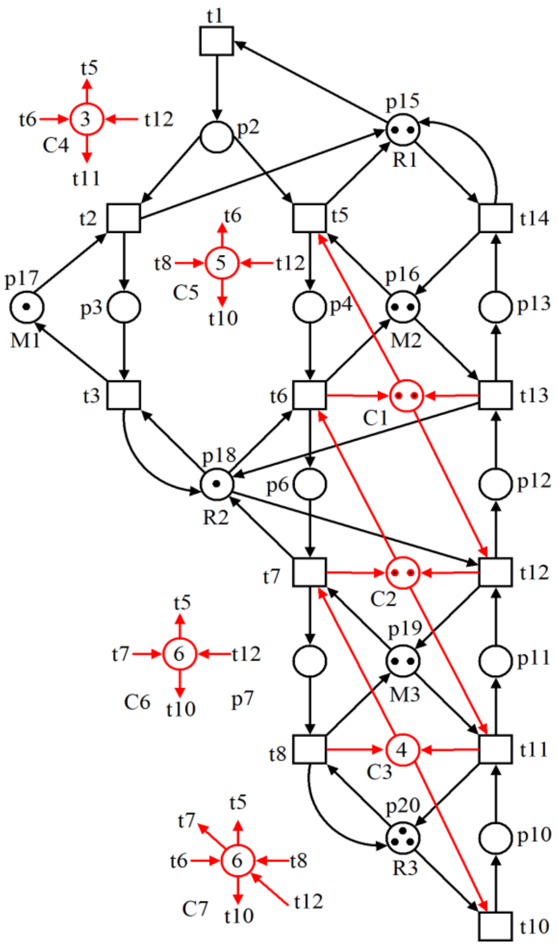


Fig. 4.15. The partially controlled reduced *PNM* (*pCRPNM*) model.

**Step 6.** The *RG* of the *pCRPNM* consists of 1878 states, with $BM_8, ..., BM_{11}$ constituting the *DZ* and 1874 *GS*s representing the *LZ*. It is important to note that by using the Petri net reduction rules, bad markings of the *pCPNM* is greatly reduced from 92 to 4 in the *pcRPNM*. The marking of the activity places of $BM_8, ..., BM_{11}$ are shown in Table 4.18.

Table 4.18. The marking of the activity places of the $BM_8, ..., BM_{11}$ of the *pcRPNM*.

| State nr. | p2 | p3 | p4 | p6 | p7 | p10 | p11 | p12 | p13 |
|---|---|---|---|---|---|---|---|---|---|
| 210 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 262 | 2 | 1 | 0 | 0 | 0 | 2 | 2 | 1 | 2 |
| 354 | 2 | 1 | 0 | 0 | 1 | 3 | 1 | 1 | 2 |
| 999 | 2 | 1 | 1 | 0 | 1 | 3 | 1 | 1 | 1 |

**Step 7.** In order not to reach $BM_8, ..., BM_{11}$, the following place invariants are established:

Table 4.19. Place invariants for the $BM_8, ..., BM_{11}$.

| | |
|---|---|
| $PI_8$ | $\mu_2 + \mu_3 + \mu_4 + \mu_{12} + \mu_{13} \leq 5$ |
| $PI_9$ | $\mu_2 + \mu_3 + \mu_{10} + \mu_{11} + \mu_{12} + \mu_{13} \leq 9$ |
| $PI_{10}$ | $\mu_2 + \mu_3 + \mu_7 + \mu_{10} + \mu_{11} + \mu_{12} + \mu_{13} \leq 10$ |
| $PI_{11}$ | $\mu_2 + \mu_3 + \mu_4 + \mu_7 + \mu_{10} + \mu_{11} + \mu_{12} + \mu_{13} \leq 10$ |

**Step 8.** To enforce $PI_8, ..., PI_{11}$, the following monitors are computed as follows:

Table 4.20. The computed monitors of the *PNM*.

| $C_i$ | $^\bullet C_i$ | $C_i^\bullet$ | $\mu_0(C_i)$ |
|---|---|---|---|
| C8 | t3, t6, t14 | t1, t12 | 5 |
| C9 | t3, t6, t8, t14 | t1, t7, t10 | 10 |
| C10 | t3, t5, t14 | t1, t10 | 9 |
| C11 | t3, t5, t8, t14 | t1, t7, t10 | 10 |

**Step 9.** It is verified that the controlled *PNM*, obtained by adding monitor *C8* to the *pCRPNM* is live with 12963 good states. This is the optimal live behavior for the controlled *PNM*. The controlled PNM is shown in Figure 4.16. Table 4.21 briefly shows the liveness enforcing procedure applied for the $S^3PR$ model. In this table the last column shows the

number of unreachable states. As the elements of this column are all zero this indicates that there are no good states lost due to included monitors.

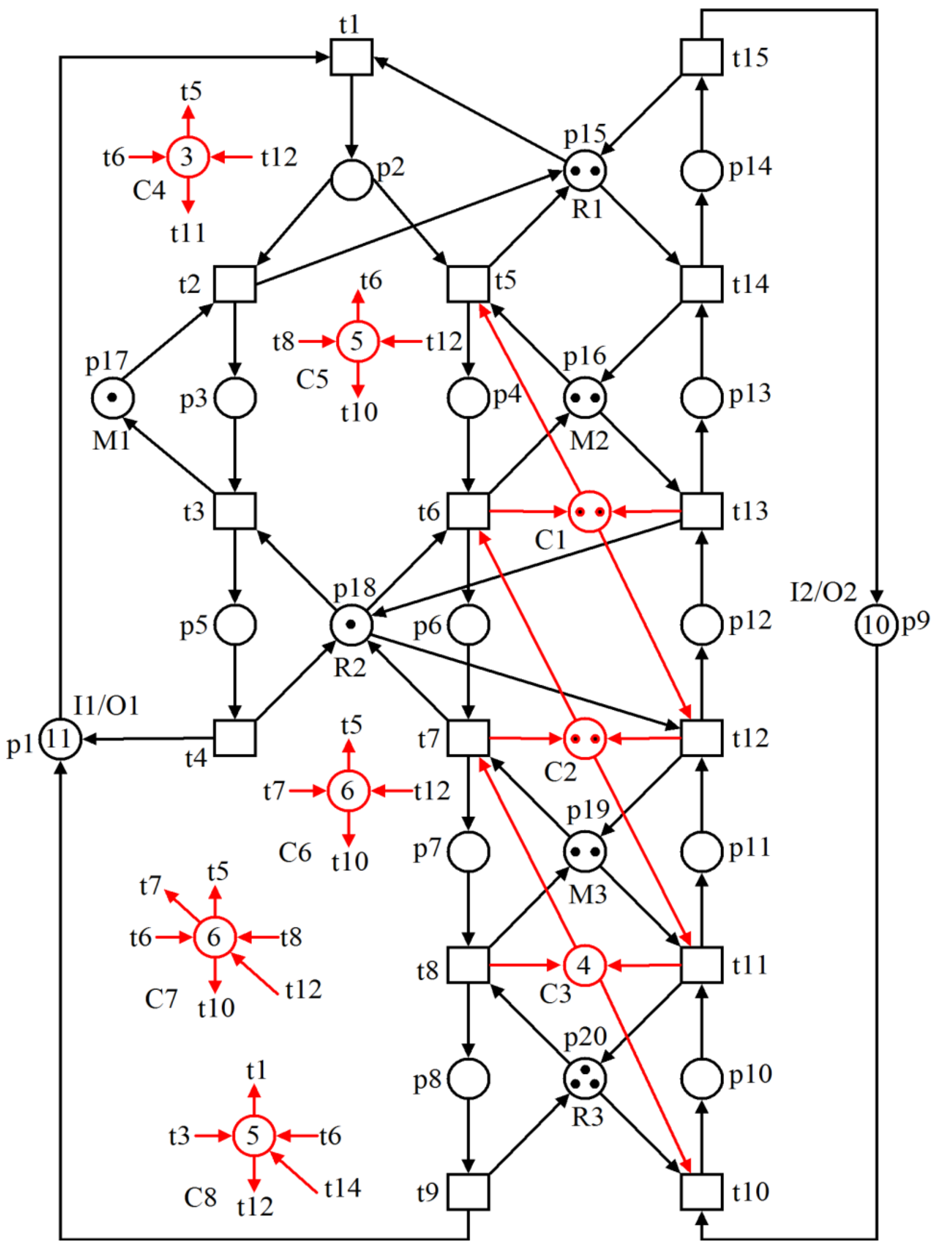


Fig. 4.16. The controlled *$S^3PR$ PNM*.

Table 4.21. The divide-and-conquer based liveness enforcing procedure applied for $S^3PR$ net.

| *steps* | *net* | *included C* | *#states within controlled net* RG | DZ | LZ | *computed C* | *# states within controlled net* RG | UR |
|---|---|---|---|---|---|---|---|---|
| *4.1* | $SN_1$ | - | *23* | *1* | *22* | *C1* | *22* | *0* |
| *4.2* | $SN_2$ | - | *23* | *1* | *22* | *C2* | *22* | *0* |
| *4.3* | $SN_3$ | - | *59* | *1* | *58* | *C3* | *58* | *0* |
| *4.4* | $SN_4$ | *C1* | *128* | - | *128* | - | *128* | *0* |
| *4.5* | $SN_5$ | *C1, C2* | *119* | *1* | *118* | *C4* | *118* | *0* |
| *4.6* | $SN_6$ | *C2, C3* | *213* | *1* | *212* | *C5* | *212* | *0* |
| *4.7* | $SN_7$ | *C1, C2, C5* | *685* | - | *685* | - | *685* | *0* |
| *4.8* | $SN_8$ | *C1, C2, ..,C5* | *1137* | *3* | *1134* | *C6, C7 Rd:1* | *1134* | *0* |
| *5-9* | *pCRPNM* | *C1, C2,..,C7* | 1878 | *4* | *1874* | *C8 Rd:3* | *1874* | *0* |
| *9-10* | *PNM* | *C1, C2, ..,C8* | *12963* | - | *12963* | - | *12963* | *0* |

*Rd.: the number of redundant monitors.*

**Step 10.** The redundancy check carried out show that the computed monitors *C8* is necessary while monitors *C9*, *C10* and *C11* are redundant.

**Step 11.** Exit.

## 4.4 DISCUSSION

In this section, the performance of the method proposed in this thesis is compared with the performance of the previously proposed methods in terms of behavioral permissiveness. In these comparisons, the results obtained for three examples are used. In the 1st example, it is verified that 322 good states are achieved (near optimal solution) with three necessary monitors without applying the Petri net reduction rule. Therefore the proposed method is simpler and straight forward compared with the methods in [15, 32]. In the second application example, it is verified that 205 good states are achieved (optimal solution) with eight computed monitors. Table 4.22 shows the performance comparison of the second application example using the methods of [22], [11] and our method.

Table 4.22. Performance comparison of 2nd application example using different policies

| policy | # reachable states | # arcs | # monitors |
|---|---|---|---|
| [22] | 210 | 23 | 5 |
| [11] | 205 | 42 | 9 |
| Our method | 205 | 33 | 8 |

In the third application example, it is verified that 12963 good states are achieved (optimal solution) with eight computed monitors. Table 4.23 shows the performance comparison of the third application example using the method [23], [10] and our method.

Table 4.23. Performance comparison of 3rd example using different policies

| policy | # reachable states | # arcs | # monitors |
|---|---|---|---|
| [23] | 7608 | 16 | 4 |
| [10] | 4217 | 35 | 9 |
| Our method | 12963 | 36 | 8 |

In Table 4.22, the policy in [22] provides 210 reachable states with five monitors which cannot totally eliminate deadlocks. Secondly, in [11], 205 reachable states can be obtained with nine monitors, but with the method proposed in this thesis, 205 reachable states can be achieved with eight monitors which is of less complexity and easy to handle. In Table 4.23, it has been observed that our policy can lead to more reachable states with eight monitors then the other two policies. This indicates that the two methods cannot lead to maximally permissible supervisors.

# CHAPTER 5

## CONCLUSION

Deadlock is a highly undesirable situation, where each of a set of two or more jobs keeps waiting indefinitely for the other jobs in the production sequence to release resources. Deadlocks may arise as a number of jobs flow concurrently through FMS and are generally difficult to predict. Therefore, it is necessary for an effective FMS control policy to ensure that deadlocks never occurs [14].

In this study, a divide and conquer approach is proposed for the synthesis of Petri net based liveness enforcing supervisors in FMS. It is an improved version of the method proposed in [15] and the policy also includes Petri net reduction approach [12], redundancy test for liveness enforcing supervisors of FMS [15] and simplified controller computation for a place invariant [15, 17] so as to carry out necessary computation easily. The proposed method is generally applicable, easy to use, effective and straightforward although its off-line computation is of exponential complexity in theory.

The divide and conquer policy proposed in this Thesis can lead to a liveness enforcing supervisor with maximally permissive results with easy computation compared to the other policies. It can also be applied to all classes of Petri net models that are proposed in the literature for the study of deadlock in FMSs.

# REFERENCES

[1] Z.W. Li and Zhou M.C., Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach. 2009.

[2] Murali M. and Bhatwadekar. Petri nets based Modeling of FMS and its Deadlock Prevention. *International Journal of Mechanical and Production Engineering (IJMPE) ISSN* No. 2315–4489, Vol-2, Iss–1, 2013.

[3] Y. Narahari and N. Viswanadham. Performance modeling of Automated manufacturing Systems. Prentice Hall of India, New Delhi. ISBN 81–203–0870–0.

[4] Murata, T., 1989. *Petri nets: Properties, analysis and application. Proceedings of IEEE*, 77(4), 541–579.

[5] Z.W. Li and Zhou M.C., "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on System, Man and Cybernetics. Part A: Systems and Humans*, Vol. 34, 2004, pp. 38–51.

[6] Wysk R. A., Yang N. S., and Joshi S., "Detection of deadlocks in flexible manufacturing cells," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 853859, Dec. 1991.

[7] Banaszak Z. and Krogh B. H., "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Transactions on Robotics and Automation*, Vol. 6, no. 6, pp. 724–734, Dec. 1990.

[8] Z.W. Li and D. Liu, "A correct minimal siphons extraction algorithm from a maximal unmarked siphon of a Petri net," *International Journal of Production Research* vol. 45, Issue: 9, pp. 2161–2165, 2007.

[9] Uzam M. and Jones A.H., (July 1996), "Equivalence of Control Places and Control Transitions in Petri Net Controllers", *Proc. of the Circuits, Systems and Computers -CSC'96*, Athens, Greece, July 15 – 17, pp. 535–539.

[10] Ezpeleta J., J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, Vol. 11, no. 2, pp. 173–184, Apr. 1995.

[11] Uzam M., 2002. An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions. *International Journal of Advanced Manufacturing Technology*, 19 (3), 192–208.

[12] Uzam, M., 2004. The use of Petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology,* 23 (3-4), 204–219.

[13] Y.F. Chen, Z.W. Li, M. Khalgui, O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Trans. on Auto. Sci. and Eng*. vol. 8, Issue: 2, pp. 374–39, 3 April 2011.

[14] Uzam M. and Zhou M.C., 2007. An iterative synthesis approach to Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on System, Man and Cybernetics – Part A: Systems and Humans*, 37 (3), 362–371.

[15] Uzam M. and Zhou M.C., 2006. An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems. *International Journal Production Research*, 44 (10), 1987–2030.

[16] B. Hruz and Zhou M.C., Modeling and Control of Discrete Event Dynamic Systems. Springer-Verlag: London, U.K., 2007.

[17] K. Yamalidou, J. Moody, M. Lemmon, and P.Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, Issue: 1, pp. 15–28, 1996.

[18] Zhou M.C. and F. Di Cesare, Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Boston, MA: Kluwer, 1993.

[19] Uzam M. and Zhou M.C., 2006. Identification and elimination of redundant control places in Petri net based liveness enforcing supervisors of FMS. *International Journal Advanced Manufacturing Technology*, 35:150–168.

[20] INA, 31.07.2003, Integrated Net Analyzer, a software tool for analysis of Petri nets, Version 2.2. At: http://www.informatik.hu-berlin.de/_starke/ina.html

[21] Huang, Y., Jeng, M.D., Xie, X. and Chung, S., Deadlock prevention based on Petri nets and siphons. *Int. J. Prod. Res*., 2001, 39(2), 283–305.

[22] Abdallah, I.B. and Elmaraghy, A., 1998. Deadlock prevention and avoidance in FMS: A Petri net based approach. *International Journal of Advanced Manufacturing Technology*, 14 (10), 704–715.

[23] Chung-Fu Zhong, Z.W. Li, Design of liveness-enforcing supervisors via transforming plant Petri net models of an FMS, *Asian Journal of Control*, Vol. 12, No. 3, pp. 240 252, May 2010.

[24] Z.W. Li, and Zhou M.C., Two-Stage Method for Synthesizing Liveness-Enforcing Supervisors for Flexible Manufacturing Systems Using Petri Nets, *IEEE Transactions on Industrial information Part A: Systems and Humans*, vol.2, 2006.

[25] Z.W. Li and Zhou M.C., A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems, *IEEE Transactions on Man and Cybernetics*, Vol. 38, no. 2, March 2008.

[26] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Boston, MA: Kluwer, 1993.

[27] Z.W. Li and Y.F. Chen, Optimal Supervisory Control of Automated Manufacturing Systems. CRC Press, 2013.

[28] Y.S. Huang, M.D. Jeng, X.L. Xie, et al., “Siphon-based deadlock prevention policy for flexible manufacturing systems,” *IEEE Transactions on Systems, Man and Cybernetics Part A–Systems and Humans* vol. 36, Issue: 6, pp. 1248–1256 November 2006.

[29] Z.W. Li, S. Zhu, and Zhou M.C., “A Divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems,” *IEEE Transactions on Systems, Man and Cybernetics. Part C–Applications and Reviews vol*. 39, Issue: 2, pp. 156–169, March 2009.

[30] Uzam M. and Jones A.H., (1997), “Real-Time Implementation of Automation Petri Net Controllers Using Programmable Logic Controllers”, *Preprints of International*. IFAC Conference on Algorithms and Architectures for Real–Time Control (AARTC’97), Vilamoura, Portugal, 9–11 April, pp. 421–426.

[31] Zhou M.C. and DiCesare F., (1993), “Petri Net Synthesis for Discrete Event Control of Manufacturing Systems”, Boston, MA, Kluwer Academic Publishers, 234 pages.

[32] Park, J. and Reveliotis, S.A., Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems. *IEEE Trans. Robot. Autom.*, 2001, 16(2), 190–195.

[33] D. Y. Chao, “Reachability and firing sequences of homogeneous synchronized choice Petri nets,” *Journal of Information Science and Engineering*, Vol. 21, 2005, pp. 129–152.

[34] M. V. Iordache, J. O. Moody, and P. J. Antsaklis, “Synthesis of deadlock prevention supervisors using Petri nets,” *IEEE Transactions on Robotics and Automation*, Vol. 18, 2002, pp. 59–68.

[35] Z.W. Li, Zhou, M. C., and Uzam, M. 2007. Deadlock control policy for a class of Petri nets without complete siphon enumeration. IET Control Theory Appl.1594–1605.