

# Extending the Knitting Technique to Petri Net Synthesis of Automated Manufacturing Systems

DANIEL Y. CHAO<sup>†</sup>, MENGCHU ZHOU<sup>\*</sup> AND DAVID T. WANG

*Computer & Information Science Department, Institute for Integrated Systems Research and*

*\*Department of Electrical & Computer Engineering, Center for Manufacturing Systems, New Jersey  
Institute of Technology, Newark, NJ 07102, USA*

The modelling or synthesis of ordinary Petri nets has been recognized as a key step for applying Petri nets to performance analysis, control and simulation of industrial production systems. This paper addresses this synthesis problem by adopting a global and incremental synthesis approach: the knitting technique for manufacturing systems. The knitting technique has been applied to Petri net modelling and analysis of communication networks. The idea behind this technique is to introduce details in an incremental way, i.e. by adding new paths and/or cycles to a Petri net. At each step the Petri net model grows according to certain rules which will guarantee such system properties as boundedness, liveness and reversibility. Thus the cumbersome analysis for these properties can be avoided while designers can still build up a Petri net model for a complicated system. The knitting rules are divided into two types: TT and PP with a number of variations. This paper formulates and discusses these rules in the context of manufacturing. The theoretical results on the preservation of system properties using the proposed rules are presented. The results are illustrated through Petri net modelling of an automated manufacturing system which consists of two robots, five machines and two automatic guided vehicles. This approach is compared with other existing synthesis approaches and its distinguishing advantages are indicated in the conclusion. Future research along this direction is also discussed.

## 1. INTRODUCTION

Petri net theory (Peterson 1981) has been applied to specifications, validation, performance analysis (Chao and Wang 1992a,b, 1993a–c, 1994b; Chao *et al.*, 1993; Chen *et al.*, 1993), control and simulation for industrial production systems (Murata *et al.*, 1986; Villarroel *et al.*, 1988; Zhou, 1990; Zhou *et al.*, 1990). The first step toward these applications is modelling or synthesis of Petri nets for integrated manufacturing systems. This synthesis problem has been identified as a key one for further successful industrial applications (Narahari and Viswanadham, 1985; Krogh and Beck, 1986; Zhou *et al.*, 1990). Early work has paid little attention on the detailed modelling process of Petri nets (Chao *et al.*, 1992a,b; Zhou *et al.*, 1991a). A Petri net model is given for a system, the analysis of the model is conducted and system properties are claimed. However, as the system grows in complexity, this modelling process becomes crucial and the analysis becomes no longer an easy problem. Though such analysis methods as reachability graph, reduction (Valette, 1979; Suzuki and Murata, 1983; Berthelot, 1985; Lee and Favrel, 1985; Silva, 1985; Chao and Wang, 1992c) and linear algebra based methods (Narahari and Viswanadham, 1985; Silva, 1985) are available, they are of limited use due to their limited capacity. Another disadvantageous fact is that modification and re-analysis may have to be conducted if the analysis methods have detected some undesired properties. Therefore, it is desired to have a synthesis approach

which can build up a Petri net systematically such that the net has desired logical properties, e.g. boundedness, liveness and reversibility. These properties are critical for a manufacturing system to operate in a stable, deadlock-free and cyclic way.

Since Petri net synthesis research started in late 1970s, significant results have been developed for special classes of Petri nets such as marked graphs (Murata, 1977, 1980, 1989; Murata and Koh, 1980), free-choice Petri nets (Esparza and Silva, 1990), and safe and live Petri nets (Krogh and Beck, 1986). Two dominant synthesis approaches for general classes of Petri nets are bottom-up and top-down. Bottom-up approaches start with decomposition of systems into subsystems, construct sub-Petri nets for subsystems, and merge these subnets to reach a final Petri net by sharing places (Agerwala and Choed-Amphai, 1978; Narahari and Viswanadham, 1985), transitions (Jeng and DiCesare, 1991) and/or elementary paths (Krogh and Beck, 1986; Valavanis, 1990; Koh and DiCesare, 1991) or by linking subnets (Datta and Ghosh, 1986). Top-down approaches begin with a first-level Petri net and refine the net to satisfy the system specification until a certain level is reached such as refinement of transitions and places by general modules with boundedness and liveness (Valette, 1979; Suzuki and Murata, 1993) or well-defined modules (Zhou and DiCesare, 1989). Based on top-down refinement of activities and bottom-up modelling of shared resources, Zhou (1990) and Zhou *et al.* (1992) have recently formulated a hybrid synthesis approach. Parallel and sequential mutual exclusions are used for resource-

<sup>†</sup> D. Y. Chao published papers prior to 1990 under the name Y. Yaw.

sharing modelling (Zhou and DiCesare, 1991) and design of first-level Petri nets is also discussed (Zhou *et al.*, 1991).

However, the existing bottom-up approaches suffer from the difficulty of analysing the global system properties although such useful information as invariants of the global Petri net can be derived from those of its subnets. In the top-down approaches, there is no easy way to validate all general modules and their design is still a problem given the system specification. In the hybrid approach (Zhou, 1990), the validation for mutual exclusions can be a formidable task for large systems. On the other hand, many reduction rules (Berthelot, 1985; Lee and Favrel, 1985; Silva, 1985; Chao and Wang, 1992c) are very powerful in reducing Petri nets. Nevertheless, they are often difficult to apply to synthesis directly, except the work by Chao and Wang (1992c). All these difficulties and problems motivate us to devise some simple but effective rules which can guide the synthesis of Petri nets with desired properties. These rules constitute a knitting technique which has been applied to the synthesis of communication protocols (Yaw, 1987; Yaw *et al.*, 1988; Chao and Wang, 1994a).

In the knitting technique (Yaw, 1987), designers start with a basic process which is modelled by a set of closed-loop sequentially-connected places and transitions with a so-called home-place marked with a certain number of tokens. The tokens may represent the number of raw materials which can be present in a system each time. Then parallel, alternative and exclusive processes or operations are added according to the system specification. Closed loops for the operations are added according to the resources required by the operations involved. The knitting technique is so-called since expansions are conducted among nodes (either transitions or places) in a global way. This approach is easy to use due to the simple rules, and leads to the final net which is bounded, live and reversible. The other advantage is that the approach is easily adapted to the computer implementation of automatic synthesis of ordinary Petri nets (Yaw and Foun, 1989).

Esparza and Silva (1991b) proposed two rules to synthesize live and bounded free-choice Petri nets. Their TT and PP handles (Esparza and Silva, 1991a) are similar to the TT and PP paths in Yaw (1987). Similar to Yaw (1987) and, rephrased in another way, they also showed that circuits without TP and PT handles have good structural properties. However, they do not explicitly apply them to synthesis using the notion of a structural relationship. In Esparza and Silva (1991b), Rule RF1 refines a macroplace by a state machine and RF2 adds a marking structurally implicit place (MSIP) to a free-choice net. RF1 corresponds to the PP rule in Yaw (1987) to add paths between places. RF2 corresponds to the TT rule in Yaw (1987) to add paths between transitions and increase the degree of concurrency. However one needs to decide whether the subnet can be reduced to macroplace (in RF1) and

whether a place is an implicit place (in RF2). The knitting technique (Yaw, 1987) requires no checking on the paths generated; rather, it uses the T-Matrix to check whether the generation points and joints satisfy certain constraints. This is simpler than checking many different subnets and a linear algebra equation for MSIP. In addition, the T-Matrix can record self-loops and find maximum concurrency with linear time complexity. Although Esparza and Silva (1991a) added the RF3 rule to synthesize EFC nets, Esparza and Silva (1991a) is unable to synthesize ACs.

The goal of this research is to formulate a powerful and easy-to-use synthesis methodology and computer environment for Petri net synthesis for various concurrent systems such as automated manufacturing systems and communication networks. The objectives of this paper include:

1. Introducing the knitting technique by presenting and discussing useful rules in the context of Petri net synthesis.
2. Presenting a theorem which will guarantee the preservation of desired properties of the proposed rules.
3. Exemplifying the procedure through Petri net synthesis of an automated manufacturing system.

Section 2 presents the rules and the theoretical results. Section 4 illustrates the rules for a manufacturing application. Finally, we present the conclusion and the future research along this line in Section 4.

## 2. KNITTING TECHNIQUE

For a fundamental knowledge of Petri net theory, the reader is referred to (Peterson, 1981; Murata, 1989; Silva, 1985). To be consistent, we introduce the following definition and notation (Zhou *et al.*, 1989, 1990; Zhou, 1990).

**Definition 1.** A marked Petri net  $N = (P, T, I, O, m_0)$ , where

1.  $P = \{p_1, p_2, \dots, p_n\}$ ,  $n > 0$ .
2.  $T = \{t_1, t_2, \dots, t_s\}$ ,  $s > 0$  with  $P \cup T \neq \emptyset$  and  $P \cap T = \emptyset$ .
3.  $I: P \times T \rightarrow \{0, 1\}$ .
4.  $O: P \times T \rightarrow \{0, 1\}$ .
5.  $m_0: P \rightarrow \{0, 1, 2, \dots\}$ .

In this definition,  $p_i$  ( $1 \leq i \leq n$ ) is called a place,  $t_i$  ( $1 \leq i \leq s$ ) a transition,  $I$  an input function defining the set of directed arcs from  $P$  to  $T$ ,  $O$  an output function defining the set of directed arcs from  $T$  to  $P$ , and  $m_0$  an initial marking whose  $i$ th component represents the number of tokens in place  $p_i$ .

**Definition 2.** The firing rules are:

1. A transition  $t \in T$  is *enabled* if and only if  $m(p) > 0$  when  $I(p, t) = 1$ ,  $\forall p \in P$ .
2. An enabled transition  $t$  fires at marking  $m'$ , yielding

the new marking,

$$m(p) = m'(p) + O(p, t) - I(p, t), \forall p \in P.$$

The marking  $m$  is said to be immediately *reachable* from  $m'$ . Given  $N$  and its initial marking  $m_0$ , the *reachability set* is the set of all markings immediately reachable starting from  $m_0$  through various sequences of transition firings and is denoted by  $R(N, m_0)$ .

**Definition 3.** A marked Petri net  $N$  is *B-bounded* if and only if  $m(p) \leq B, \forall p \in P$  and  $m \in R(N, m_0)$  where  $B$  is a positive integer. If  $B = 1$ ,  $N$  is *safe*.  $N$  is *live* if and only if  $\forall t \in T$ , and  $\forall m \in R(N, m_0)$ ,  $\exists$  a sequence of transitions  $f$ , firing  $f$  leads to a marking which enables  $t$ .  $N$  is *reversible* if and only if  $\forall m, m \in R(N, m_0) \Rightarrow m_0 \in R(N, m)$ . The synchronic distance between  $t$  and  $t'$  is  $\max\{\#(\sigma, t) - \#(\sigma, t'), \#(\sigma, t') - \#(\sigma, t) \mid \sigma \in L(N, m), m \in R(N, m_0)\}$  where  $\#(\sigma, t)$  denotes the number of times  $t$  appears in a firing sequence  $\sigma$  and  $L(N, m)$  is the set of all firing sequences from  $m$ .  $N$  is *conservative* if and only if  $\forall m \in R(N, m_0), \sum_{p \in P} m(p) = \sum_{p \in P} m_0(p)$ .

The synchronic distance between any two transitions can be obtained as follows:

1. For each firing sequence of the Petri net in one iteration, calculate the absolute value of the difference between the numbers of the firing of these two transitions.
2. For all the firing sequences of the synthesized Petri net, the maximum such value is the synchronic distance.

The complexity of this procedure with the Petri net synthesized using the knitting technique might be polynomial and is an open problem.

The implications of these properties of Petri nets have been explored in manufacturing (Narahari and Viswanadham, 1985; Krogh and Beck, 1986; Valvanis, 1990; Zhou *et al.*, 1989). The synchronic distance is a measure of dependence between transitions and more detailed explanation can be found in Silva (1987).

**Definition 4.** The home place(s) is defined as the place(s) with token(s) which can enable a transition at the initial marking.

For example, place  $p_1$  in Figure 1 is a home place.

**Definition 5.** A basic process is defined as a Petri net

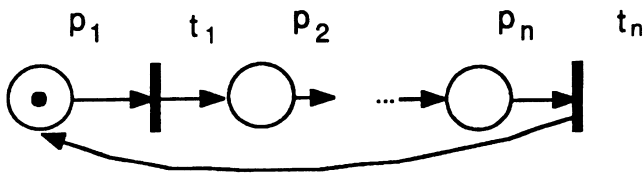


FIGURE 1. A basic process with home place ( $n > 0$ ).

shown in Figure 1 such that

$$I = I_n, O = \begin{pmatrix} 0_{n-1} & I_{n-1} \\ 1 & 0_n^t \end{pmatrix}, \quad n \geq 1, \quad m_0(p_1) > 0$$

and

$$m_0(p_i) = 0, \quad 2 \leq i \leq n,$$

where  $I_n$  is an  $n \times n$  identity matrix,  $0_{n-1}$  is a  $(n-1)$ -zero-vector, and  $\tau$  denotes the transpose.

Numerous applications can be synthesized as a Petri net by starting with the basic process which exhibits the desired properties such as boundedness, liveness, and reversibility.

**Definition 6.** Given  $N = (P, T, I, O, m_0)$ , a node is either a place in  $P$  or a transition in  $T$ . An *elementary path* is a sequence of nodes:  $x_1 x_2 \dots x_n, n \geq 1$ , such that  $\exists$  an arc  $(x_i, x_{i+1}), 1 \leq i < n$  if  $n > 1$ , and  $x_i = x_j$  implies that  $i = j, \forall 1 \leq i, j \leq n$ . An *elementary circuit* is  $x_1 x_2 \dots x_n, n > 1$  such that  $x_i = x_j, 1 \leq i \leq j \leq n$ , implies that  $i = 1$  and  $j = n$ . The post-set of node  $x$  is  $x^+ = \{y : \exists \text{ an arc } (x, y)\}$  and its pre-set  $^+x = \{y : \exists \text{ an arc } (y, x)\}$ .

Motivated by the A-place and A-path concept, operation places defined in Zhou and DiCesar (1991b), we define the following A-path concept.

**Definition 7.** An A-path is an elementary path whose places initially have no tokens.

**Definition 8.** An elementary path,  $x_1 x_2 \dots x_n, n \geq 2$  is called a pseudo process (PSP) if  $|^+x_i| = |x_i^+| = 1, 2 \leq i \leq n-1$  and if there is no elementary path,  $y_1 y_2 \dots y_m$ , such that  $|^+y_i| = |y_i^+| = 1, 2 \leq i \leq m-1, \{y_i, 1 \leq i \leq m\} \supset \{x_i, 1 \leq i \leq n\}$  and  $m > n$ . The length of a PSP is  $|\text{PSP}| = n$ .  $x_1$  is a generation point of a PSP and  $x_n$  is a joint point. A PSP is marked iff a place in PSP is marked.

Thus in the basic process (Figure 1), if  $n = 3$ , then  $p_1 t_1 p_2 t_2 p_3 t_3$  and  $t_1 p_2 t_2 p_3 t_3 p_1$  are two PSP while  $t_1 p_2 t_2 p_3$  is not since it is included in  $p_1 t_1 p_2 t_2 p_3 t_3$ . For convenience, we use a PSP as a set of nodes, i.e. if  $x$  is on the PSP, we say  $x \in \text{PSP}$ . In the following discussion,  $\text{psp}, \text{psp}', \text{PSP}, \text{PSP}'$  and  $\text{PSP}_k$  ( $k$  is a positive integer) are used to represent different pseudo processes.

**Definition 9.** If a PSP is added between two places (transitions), this generation of a new PSP is called *PP (TT) generation* and the resulting PSP is called *PP path* or *TT path*. A virtual path is a two-node PSP, i.e. one with no node except the generation and joint points.

The length of any PP path or TT path is at least 3 since there has to exist at least one transition or place between any two places or two transitions.

Most manufacturing systems are designed to achieve cyclic production. Thus its Petri net model should be reversible, i.e. its initial marking should be recovered from any other marking. We call the period from the initial marking to other markings and back to the initial

marking a cycle. The relationships among the pseudo processes within a cycle can be sequential, concurrent, or exclusive.

**Definition 10.** PSP is *sequentially earlier* than PSP', denoted by  $PSP \rightarrow PSP'$ , iff  $\exists$  an A-path AP, such that  $PSP-AP-PSP'$  is an elementary path. PSP is *concurrent* to PSP', denoted by  $PSP \parallel PSP'$  iff (1) neither  $PSP \rightarrow PSP'$  nor  $PSP' \rightarrow PSP$ , (2) PSP and PSP' are both marked during each iteration. PSP is *exclusive* to PSP', denoted by  $PSP|PSP'$  iff (1) neither  $PSP \rightarrow PSP'$  nor  $PSP' \rightarrow PSP$ , (2) PSP is not concurrent to PSP'.

**Definition 11.** Local concurrent set for PSP with respect to PSP' is  $LCN(PSP, PSP')$  satisfying: (1)  $\forall psp, psp' \in LCN(PSP, PSP')$ , if  $psp \neq psp'$ , then  $psp \parallel psp'$ , and (2)  $\forall psp \in LCN(PSP, PSP')$ ,  $psp \parallel PSP$ ,  $\neg(psp = PSP)$  and  $\neg(psp \parallel PSP')$ .

**Definition 12.** Local exclusive set for PSP with respect to PSP' is  $LEX(PSP, PSP')$  satisfying: (1)  $\forall psp, psp' \in LEX(PSP, PSP')$ , if  $psp \neq psp'$ , then  $psp|psp'$  and (2)  $\forall psp \in LEX(PSP, PSP')$ ,  $psp|PSP$ , neither  $psp = PSP$ , nor  $psp|PSP'$ .

Both local concurrent and local exclusive sets are not unique and will be discussed in the next section. Let the set of home places be  $H$ . Initially  $H = \{p_1\}$  for the basic process in Figure 1.

TT rules are motivated by the need of increasing the concurrent activities in the system while PP rules are used to offer the choices for the existing processes.

**Definition 13 (TT rule).** Select pseudo processes PSP and PSP' and two transitions  $t \in PSP$  (as a generation point) and  $t' \in PSP'$  (as a joint PP). Generate a new pseudo process psp between  $t$  and  $t'$  if  $t$  is not exclusive to  $t'$ .

TT1: if  $PSP = PSP'$ ; go to TT4.

TT2: if  $PSP \parallel PSP'$ ,  $PSP \neq PSP'$  and synchronic distance between  $t$  and  $t'$  is bounded, (1) for each  $psp' \in LEX(PSP, PSP')$ , find a transition  $t'' \in psp'$  and a place  $p \in psp$  and add a pseudo process from  $t''$  to  $p$  (TT2.1); (2) for each  $psp'' \in LEX(PSP', PSP)$ , find a transition  $t'' \in psp''$  and a place  $p \in psp$  and add a virtual PT path from  $p$  to  $t''$  (TT2.2); go to TT4.

TT3: if  $PSP \parallel PSP'$ ,  $PSP \neq PSP'$  and synchronic distance between  $t$  and  $t'$  is infinite, find  $t^1 \in PSP$  and  $t^2 \in PSP'$  and generate a  $psp'$  between  $t^1$  and  $t^2$  such that  $psp, psp', PSP$  and  $PSP'$  constitute a cycle; go to TT4.

TT4: if  $t \leftarrow t'$  and if the resulting cycle has no tokens, find a place  $p \in psp$ , insert at least one token in  $p$  and add  $p$  to set  $H$ .

Note that in TT3,  $t^1$  ( $t^2$ ) can be the same as  $t$  ( $t'$ ). In TT2.1 (TT2.2), if  $LEX(PSP, PSP') = \emptyset$  ( $LEX(PSP', PSP) = \emptyset$ ), we add nothing for the net. It is not allowed for a path to be added between two exclusive transitions.

**Definition 14 (PP rule).** Select pseudo processes PSP and PSP' and two places  $p \in PSP$  (as a generation point) and  $p' \in PSP'$  (as a joint point). Generate a new pseudo process psp between  $p$  and  $p'$  if  $p$  is not concurrent to  $p'$ .

PP1: if  $PSP = PSP'$ , it is done.

PP2: if  $PSP \neq PSP'$ , (1) for each  $psp' \in LCN(PSP', PSP)$ , find a place  $p'' \in psp'$  and a transition  $t \in psp$  and add a pseudo process from  $t$  to  $p''$  (PP2.1) and (2) for each  $psp'' \in LCN(PSP, PSP')$ , find a place  $p'' \in psp''$  and transition  $t \in psp$  and add a pseudo process from  $p''$  to  $t$  (PP2.2).

**THEOREM.** Any Petri nets resulting from applications of TT and PP rules to a basic process are bounded, live, reversible and conservative.

The complete proof of this theorem requires much space and is omitted in this paper. The idea is to create the conditions for the net to be bounded, live, reversible and conservative. The applications of these rules will not affect these conditions. Then we can preserve the properties of boundedness, liveness, reversibility and conservativeness.

Here we discuss the physical meaning of interactive generations in terms of manufacturing systems. An interactive TT generation models a concurrent interaction such as a resource exchange. PSP<sub>g</sub> sends a token (or a resource) to PSP<sub>j</sub>. Resources received by an entity must be properly handled. If a received resource is not handled by any process of the entities, the model is not a complete one. Thus, each PSP in  $LEX(PSP_j, PSP_g)$  must be able to receive token from  $LEX(PSP_g, PSP_j)$ . This is Rule TT2. An interactive PP generation models an exclusive interaction such as execution transfer from process 1 to process 2 in a resource-sharing system. If only part of the preconditions of one execution in process 2 is satisfied then process 2 cannot even start any execution. Thus, by Rule PP2, when PSP<sub>1</sub> switches to PSP<sub>2</sub>, all PSPs of  $LCN(PSP_g, PSP_j)$  must switch together and all PSPs of  $LCN(PSP_j, PSP_g)$  must be activated.

Given the specification of a manufacturing system, we first need to design a basic process which models a sequence of activities. Then PP rules and TT rules can be applied to increase the detail based on the specifications. Each time the set of home places,  $H$ , may be expanded according to TT4. The places in  $H$  often represent the availability of resources in the context of manufacturing systems.

Given a synthesized net, N<sub>1</sub>, a set of new paths are generated using the synthesis rules to form another net, N<sub>2</sub>. Figure 2 illustrates an example of Rules TT2 and TT3, a TT path,  $psp = t_2 p_6 t'_6$ , is generated from  $t_2$  of PSP<sub>1</sub> ( $= p_2 t_2 p_3 t_3 p_4$ ) to  $t'_6$  of PSP<sub>4</sub> ( $= p'_2 t'_2 p'_3 t'_3 p'_4$ ).  $LEX(PSP_1, PSP_4) = PSP_3$  ( $= p_2 t_5 p_5 t_6 p_4$ ) and  $LEX(PSP_4, PSP_1) = PSP_2$  ( $= p'_2 t'_2 p'_3 t'_3 p'_4$ ). Applying Rule TT2, we connect from  $t_5$  of PSP<sub>3</sub> to  $p_6$  and from  $p_6$  to  $t'_3$  of PSP<sub>2</sub>. In addition, the synchronic distance between  $t_2$  and  $t'_6$  is infinite, hence, by Rule TT3, we

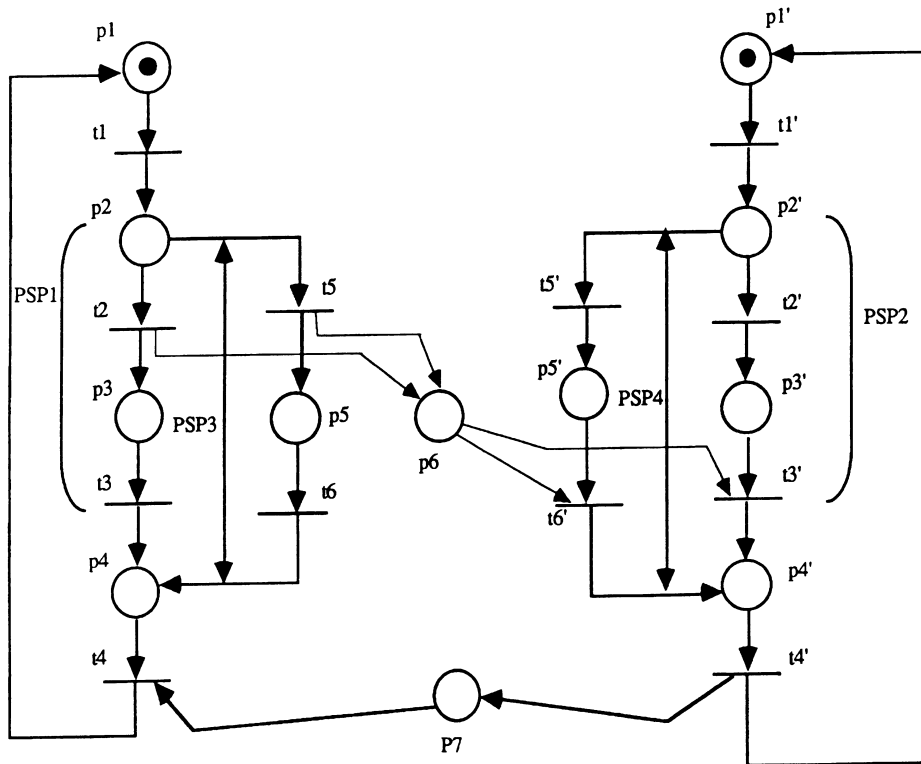


FIGURE 2. An example of Rules TT2 and TT3.

generate another TT path  $PSP' (= t_4' p_7 t_4)$  such that  $PSP1$ ,  $psp$   $PSP4$  and  $psp'$  constitute a cycle.

Rule PP2 is illustrated in Figure 3. We generate a PP path,  $psp (= p_7 t_g p_{11} t_j p_{12} t_{10} p_2)$ , from  $p_7$  of  $PSP3 (= t_2 p_7 t_7 p_9 t_8)$  to  $p_2$  of  $PSP1 (= t_1 p_2 t_3 p_4 t_5)$ .  $LCN(PSP3, PSP1) = PSP4 (= t_2 p_6 t_6 p_8 t_8)$  and  $LCN(PSP1, PSP3) = PSP2 (= t_1 p_3 t_4 p_5 t_5)$ . Hence, applying Rule PP2.1, we add a pseudo process,  $t_j p_5$ , from  $t_j \in psp$  to  $p'' \in PSP2$ . Further, applying Rule PP2.2, we add a pseudo process,  $p_6 t_g$ , from  $p_6 \in PSP4$  to  $t_g \in psp$ .

### 3. APPLICATION ILLUSTRATION

#### 3.1. Description of a manufacturing system

The automated manufacturing system, as illustrated in Figure 4, consists of the following major components: two entries, two exits, five machines, two robots, two automatic guided vehicles (AGVs), and related conveyors. It can produce two types of products.

**Entries.** There are two entries for the input of two types of raw materials, called types A and B, which are produced into two different kinds of products.

**Exits.** There are two exits for final A-parts and B-parts, respectively.

**Machines.** Machines  $M_1$  and  $M_2$  process A-raw materials from Entry 1. Both machines produce identical A-parts which are unloaded via  $AGV_1$ . These intermediate parts are further machined by Machine  $M_3$  to produce final A-parts. Machine  $M_4$  processes B-raw

materials from Entry 2 and produces B-parts which are unloaded via  $AGV_2$ . Machine  $M_5$  machines those intermediate B-parts.

**Robots.** Robot  $R_1$  serves three machines,  $M_1$ ,  $M_2$  and  $M_4$ , and none of these three machines has any priority to use this robot. In a non-deterministic manner,  $R_1$  chooses one machine if raw materials are available and these machines are ready. Robot  $R_2$  loads Machines  $M_3$  and  $M_5$ .

**AGV System.** Two AGVs are designed for the delivery of intermediate parts. From Machines  $M_1$  and  $M_2$ ,  $AGV_1$  sends intermediate A-parts to  $M_3$ , and  $AGV_2$  sends intermediate B-parts to  $M_5$ .

An unlimited source of raw materials is assumed. Once machines, robots or AGVs start any operation, they cannot be interrupted until the work is complete. We will build up a Petri net model which guarantees the boundedness, liveness and reversibility next.

#### 3.2. Modelling process

First we identify a sequence of operations and represent it as a basic process. For this system, availability of A-raw materials and the sequence of operations, i.e. loading by  $R_1$ , machining by  $M_1$ , unloading via  $AGV_1$ , loading by  $R_2$ , machining by  $M_3$ , are modelled as shown in Figure 5.

Machine 2 offers an alternative way to produce A-parts. Thus Rule PP-1 is used since two places  $p_2$  and  $p_4$  can be selected as the generation and joint points, respectively, from the same  $PSP = p_1 t_1 p_2 t_2 \dots p_6 t_6$ . A

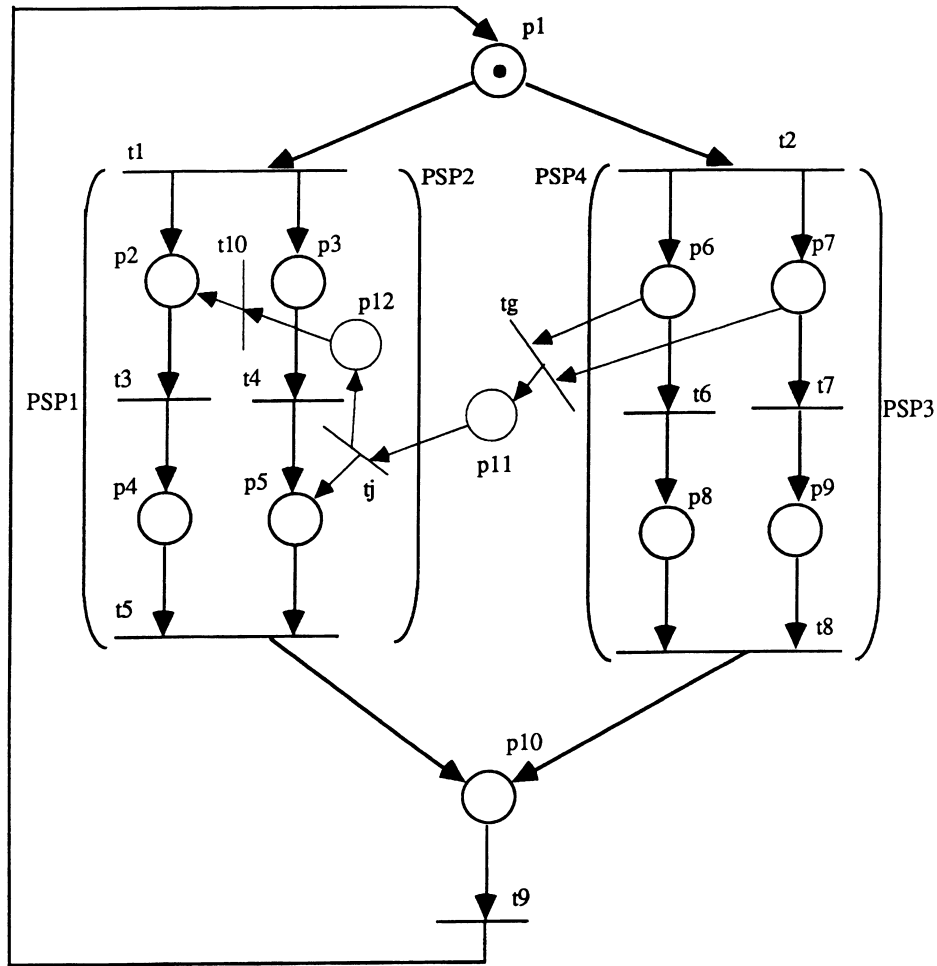


FIGURE 3. An example of the Rule PP2.

new  $psp = p_2 t_7 p_7 t_8 p_4$  is generated as shown in Figure 6. For this net, we have three unique processes, i.e.  $PSP1 = p_2 t_7 p_7 t_8 p_4$ ,  $PSP2 = p_2 t_2 p_3 t_3 p_4$  and  $PSP3 = p_4 t_4 p_5 t_5 p_6 t_6 p_1 t_1$ . It is easy to realize that the relations among these processes are as follows:

$PSP|PSP2$  and  $PSP3$  is sequential to  $PSP1$  and  $PSP2$ , respectively. Now since the operation in  $p_2$  requires Robot 1, a TT path is generated from  $t_2$  (of  $PSP2$ ) to  $t_1$  (of  $PSP3$ ), i.e.  $t_2 p_8 t_1$ . Applying TT4, we must insert a token in place  $p_8$  since there is no token in the new circuit  $t_1 p_2 t_2 p_8 t_1$ . This token in  $p_8$  actually stands for Robot 1. Furthermore, since the local exclusive set  $LEX(PSP2, PSP3) = \{PSP1\}$ , a path  $t_7 p_8$  is added via TT2.1. Since  $LEX(PSP3, PSP2) = \emptyset$ , nothing more needs to be added via TT2.2. The result is shown as the dotted lines in Figure 4.  $H = \{p_1, p_8\}$ .

Machines 1, 2 and 3, and  $R_1$  are required to start their corresponding operations. Thus using TT2 and TT4 we are able to produce the new paths as follows:  $t_3 p_{10} t_2$ ,  $t_8 p_{11} t_7$ ,  $t_5 p_{12} t_4$  and  $t_6 p_{13} t_5$  as shown in Figure 7.

Consider the  $AGV_1$ , we first generate a TT path between  $t_4$  (of  $PSP = p_4 t_4$ ) and  $t_3$  (of  $PSP' = t_3 p_4$ ), i.e.  $t_4 p_9 t_3$ . Using TT4 we add a token to  $p_9$ . Since  $LEX(p_4 t_4, t_3 p_4) = \emptyset$ , we add nothing.  $LEX(t_3 p_4, p_4 t_4)$  can be any one of  $\{t_8 p_4\}$ ,  $\{t_7 p_7 t_8\}$  and  $\{p_2 t_7\}$ . By using

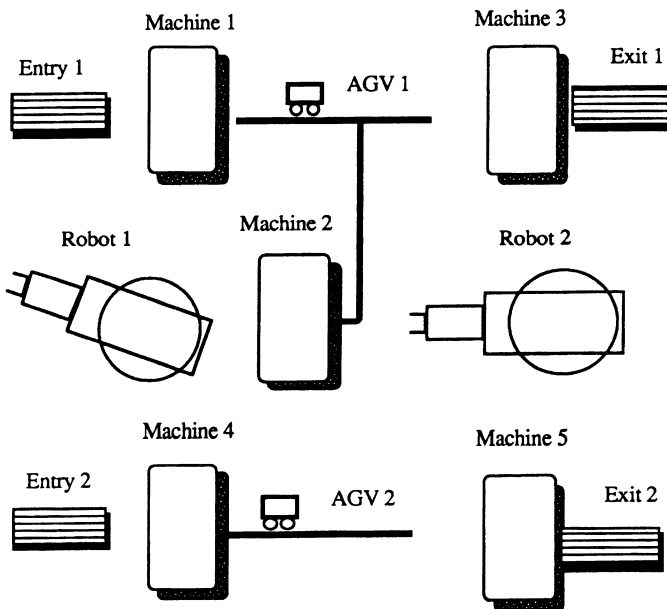


FIGURE 4. Layout of an automated manufacturing system.

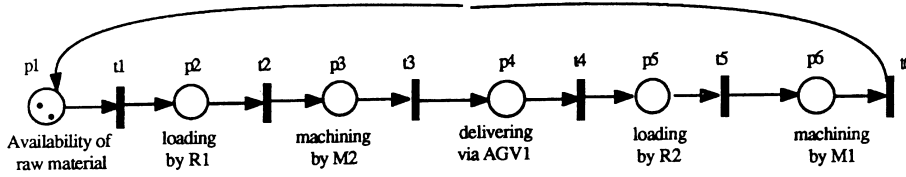
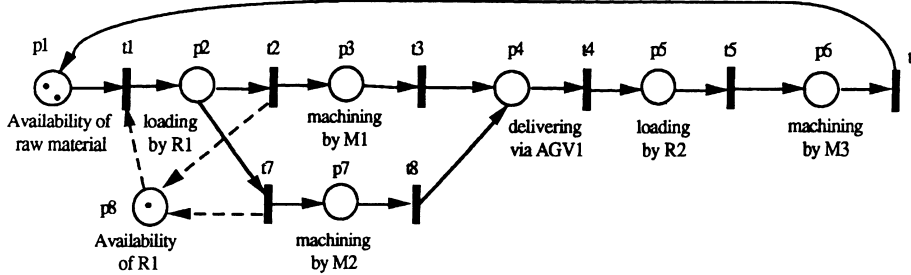
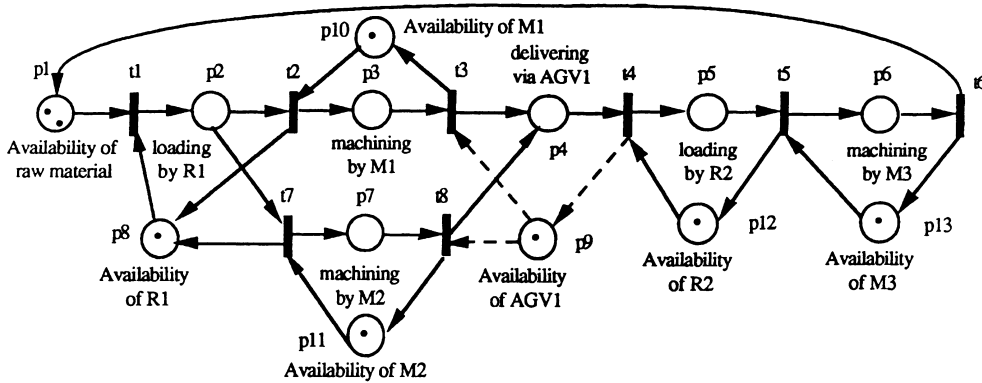


FIGURE 5. A basic process.

FIGURE 6. Add  $p_2t_7p_7t_8p_4$  via PP1 and  $t_2p_8t_1$  and  $t_7p_8$  via TT2 and TT4.FIGURE 7. Add the paths  $t_3p_{10}t_2$ ,  $t_8p_{11}t_7$ ,  $t_5p_{12}t_4$ ,  $t_6p_{13}t_5$ ,  $t_4p_9t_3$  and  $p_9t_8$  via TT2 and TT4.

either  $\{t_8p_4\}$  or  $\{t_7p_7t_8\}$ , the virtual PT path from  $p_9$  to  $t_8$  is added as the dotted lines in Figure 7. This implies that once  $M_2$  completes its processing,  $AGV_1$  is used to deliver the part. Theoretically, we may use  $\{p_2t_7\}$  and then a PT path  $p_9t_7$  will be added.  $H = \{p_1, p_8-p_{13}\}$ .

We have synthesized the partial system shown in Figure 4, i.e. production portion from Entry 1 to Exit 1. Now we synthesize the rest portion, from Entry 2 to Exit 2, by applying the rules. For  $t_7p_8t_1$  and  $t_5p_{12}t_4$ , applying PP1 leads to the two loops  $p_8t_9p_{14}t_{10}p_8$  and  $p_{12}t_{11}p_{15}t_{12}p_{12}$ . The meaning of the added places is shown in Figure 8.

Consider two processes:  $PSP4 = p_8t_9p_{14}t_{10}p_8$  and  $PSP5 = p_{12}t_{11}p_{15}t_{12}p_{12}$ . We choose  $t_{10}$  from  $PSP4$  and  $t_{11}$  from  $PSP5$  and generate a new process,  $psp = t_{10}p_{16}t_{13}p_{17}t_{11}$ , as shown in Figure 9. Since  $PSP4 \neq PSP5$ ,  $PSP4 \parallel PSP5$ , the synchronic distance between  $t_{10}$  and  $t_{11}$  is infinite, TT3 is applied by picking up  $t_9$  from  $PSP4$  and  $t_{12}$  from  $PSP5$  and generating  $psp' = t_{12}p_{18}t_{14}p_{19}t_9$ . It is easily verified that  $psp$ ,  $psp'$ ,  $PSP4$  and  $PSP5$  constitute a loop. Furthermore, we insert the tokens in  $p_{19}$  to represent the availability of raw material from Entry 1 to fulfill

Rule TT4. The synchronic distance between  $t_{10}$  and  $t_{11}$  is now bounded by the initial tokens given in  $p_{19}$ .  $H = \{p_1, p_8-p_{13}, p_{19}\}$ .

Applying TT2 and TT3 we obtain the paths:  $t_{13}p_{20}t_{10}$ ,  $t_{11}p_{21}t_{13}$  and  $t_{14}p_{22}t_{12}$  which model the availability of  $M_4$ ,  $M_5$  and  $AGV_2$ . This completes the modelling process and the final net is depicted in Figure 10 as a bounded, live and reversible net if  $m_0(p_1) > 0$ ,  $m_0(p_{19}) > 0$ ,  $m_0(p_i) = 0$  for  $8 \leq i \leq 13$  or  $20 \leq i \leq 22$ , and  $m_0(p) = 0$  for  $p \in \{p_i, 1 \leq i \leq 22\} - H$ , i.e. the places except those in  $H$  in the net have no token.  $H = \{p_1, p_8-p_{13}, p_{19}-p_{22}\}$ .

#### 4. CONCLUSIONS

This paper proposes the knitting technique of Petri net synthesis for manufacturing systems. This methodology insures the desired qualitative properties of the systems, and precludes the need for analysis of boundedness, liveness and reversibility by adopting several useful rules which are easy to use. Application of these rules is illustrated through a production system of five machines, two robots and two AGVs. The proposed technique has two basic advantages over previous synthesis

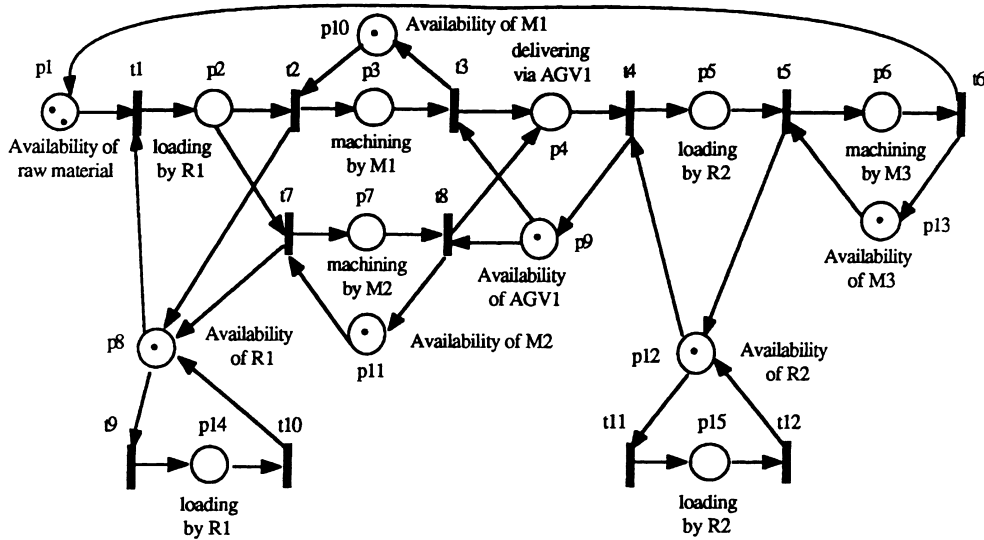


FIGURE 8. Add  $p_8t_9p_{14}t_{10}p_8$  and  $p_{12}t_{11}p_{15}t_{12}p_{12}$  via PP1.

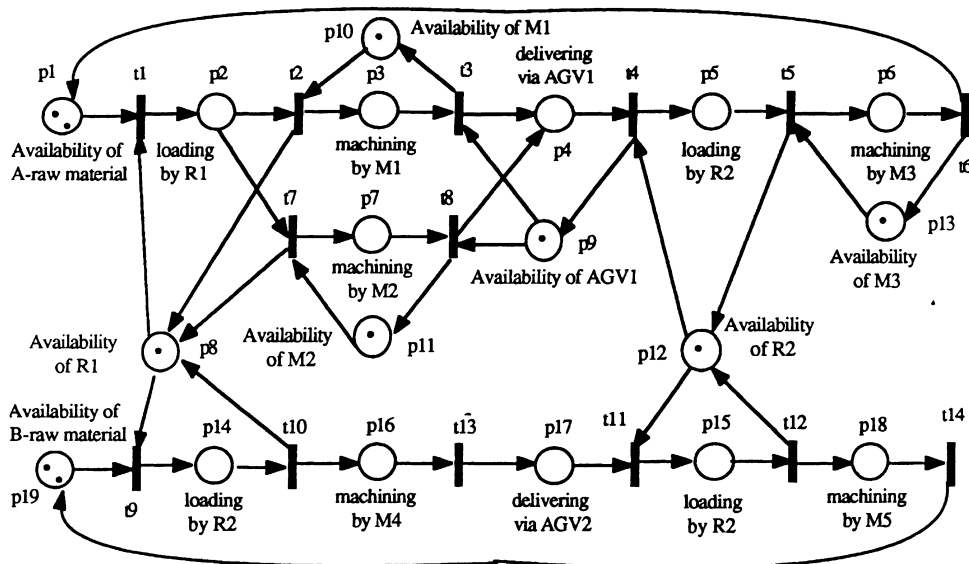


FIGURE 9. Adding  $t_{10}p_{16}t_{13}p_{17}t_{11}$  and  $t_{12}p_{18}t_{14}p_{19}t_9$  via TT3.

approaches, i.e. top-down (Valette, 1979; Suzuki and Murata, 1983; Berthelot, 1985; Zhou and DiCesare, 1989), bottom-up (Narahari and Viswanadham, 1985; Datta and Ghosh, 1986; Krogh and Beck, 1986; Koh and DiCesare, 1991) or hybrid approaches (Zhou *et al.*, 1992), in the following aspects:

1. At each step, the rules can be easily checked and used for incremental design of Petri net models for a concurrent system. The boundedness, liveness and reversibility can be preserved, and thus no analysis is required for the complicated net.
2. The procedure introduced can be easily automated in the environment of computers (Yaw and Foun, 1989). The feature is especially important for large-scale manufacturing systems.

However, the current approach needs to be further

extended to include more general resource sharing cases discussed in Zhou and DiCesare (1991b). Second, the research should focus on the system whose first-level models are no longer the basic process discussed, e.g. the choice-synchronization structure discussed in Zhou *et al.*, 1992. Third, this technique has been extended to general Petri nets (Chao and Wang, 1994c; Chou *et al.*, 1993). Finally, a computerized environment should be created for further industrial applications (Yaw and Foun, 1989).

#### ACKNOWLEDGMENTS

The authors acknowledge the support of SBR Program at New Jersey Institute of Technology (NJIT). The second author also acknowledges the support of the Center for Manufacturing Systems at NJIT. The authors

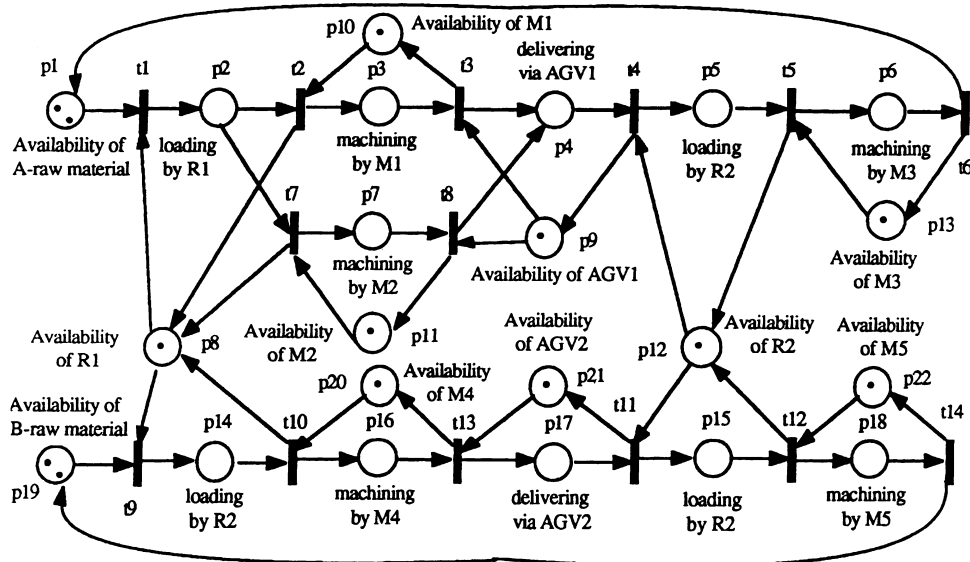


FIGURE 10. A final Petri net.

are indebted to Moses R. Kodali for his work in preparation of the final version of this paper.

## REFERENCES

- Agerwala, T. and Choed-Amphai, Y. (1978) A synthesis rule for concurrent systems. In *Proc. Design Automation Conf.*, pp. 305–311.
- Berthelot, G. (1985) Checking properties of nets using transformations. In Rozenberg, G. (ed.), *Advances in Petri Nets 1985*. Springer-Verlag, Berlin.
- Chao, D. Y., Zhou, M. C. and Wang, D. T. (1992a) Extending knitting technique to Petri net synthesis of automated manufacturing systems. In *Proc. Rensselaer's 3rd Int. Conf.*, pp. 56–63, Troy, New York.
- Chao, D. Y. and Wang, D. T. (1992a) Performance evaluation using data flow graphs for concurrent processing. *Proc. 1992 IEEE Int. Conf. SMC*, pp. 638–643. Chicago, IL.
- Chao, D. Y. and Wang, D. T. (1992b) Theory, parallelization and scheduling of critical loop, subcritical loops, and next-critical loops. In *Proc. 1992 Int. Comp. Symp.*, pp. 285–292. Taichung, Taiwan.
- Chao, D. Y. and Wang, D. T. (1992c) A reduction algorithm of Petri net. In *Proc., Int. Comp. Symp.*, pp. 16–23. Taichung, Taiwan.
- Chao, D. Y., Chen, T. H., Wang, D. T. and Zhou, M. C. (1992b) X window implementation of Petri net based animation for FMS. In *Proc. 1992 IEEE Int. Conf. SMC*, pp. 1651–1656. Chicago, IL.
- Chao, D. Y., Zhou, M. C. and Wang, D. T. (1993d) Multiple-weighted marked graphs. In *Preprints of 12th IFAC 1993 World Congr.* Sydney, Australia. pp. 259–263.
- Chao, D. Y. and Wang, D. T. (1993a) Minimum marking for no loop, next-critical loops and subcritical loops and iteration bounds for data flow graphs. In Schwetman, H., Walrand, J., Bagchi, K. and DeGrot, D. (eds.), *MASCOTS' 93*. Society for Computer Simulation. Vol. 25 No. 1, San Diego, CA.
- Chao, D. Y. and Wang, D. T. (1993b) X window-implementation of finding critical loop, next-critical loops and subcritical loops and iteration bounds for data flow graphs. In Schwetman, H., Walrand, J., Bagchi, K., and DeGrot, D. (eds.), *MASCOTS' 93*. Society for Computer Simulation, Vol. 25 No. 1, San Diego, CA.
- Chao, D. Y. and Wang, D. T. (1993c) Iteration bounds of single-rate data flow graphs of concurrent processing. *IEEE Trans. Circuits and Systems*, **40**, pp. 629–634.
- Chao, D. Y. and Wang, D. T. (1994a) The knitting technique and its application to communication protocol synthesis. To appear in *Proc. MASCOTS' 94*, Durham, NC.
- Chao, D. Y. and Wang, D. T. (1994b) An interactive tool for design, simulation, verification and synthesis of protocols. To appear in *Proc. MASCOTS' 94*, Durham, NC.
- Chao, D. Y. and Wang, D. T. (1994c) A synthesis technique of general Petri Nets. *J. Syst. Integration*, **4**, 67–102.
- Chen, Y., Tsia, W. T. and Chao, D. Y. (1993) Dependency analysis—a compositional technique for building large Petri net. *IEEE Trans. Parallel and Distributed Syst.*, **4**, 414–426.
- Datta, A. and Ghosh, S. (1986) Modular synthesis of deadlock-free control structures. In Goos, G. and Hartmanis, J. (eds.), *Foundations of Software Technology and Theoretical Computer Science*. Springer-Verlag, Berlin. pp. 288–318.
- Jeng, M. D. and DiCesare, F. (1991) A modular synthesis techniques for Petri nets. *Proc. 1992 Japan-USA Symp. on Flexible Automation*. 1163–1179, San Francisco, CA.
- Esparza, J. and Silva, M. (1991a) On the analysis of free choice systems. LNCS, *Advances in Petri Nets*. Springer-Verlag, pp. 243–286.
- Esparza, J. and Silva, M. (1991b) Circuits, handles, bridges and nets. LNCS, *Advances in Petri Nets*. Springer-Verlag. pp. 210–242.
- Koh, I. and DiCesare, F. (1991). Transformation methods for generalized Petri nets and their applications in flexible manufacturing systems. *IEEE Trans. System, Man, Cybernet.*
- Krogh, B. H. and Beck, C. L. (1986) Synthesis of place/transition nets for simulation and control of manufacturing systems. In *Proc. 4th IFAC/IFORS Symp. Large Scale Systems*, pp. 661–666. Zurich, Switzerland.
- Lee, K. H. and Favrel, J. (1985) Hierarchical reduction method for analysis and decomposition of Petri nets. *IEEE Trans. Systems, Man, Cybernet.*, **SMC-15**, 272–280.
- Murata, T. (1977) Circuit theoretic analysis and synthesis of marked graphs. *IEEE Trans. Circ. Sys.*, **CAS-27**, 400–405.
- Murata, T. (1980). Synthesis of decision-free concurrent systems for prescribed resources and performance. *IEEE Trans. Software Eng.*, **SE-6**, 525–530.
- Murata, T. and Koh, J. Y. (1980). Reduction and expansion

- of live and safe marked graphs. *IEEE Trans. Circ. Sys., CAS-27*, 68–70.
- Murata, T., Komoda, N. and Matsumoto, K. (1986) A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Trans. Ind. Electron., IE-23*, 1–8.
- Murata, T. (1989) Petri nets: properties, analysis and application. *Proc. IEEE*, **77**, 541–579.
- Narahari, Y. and Viswanadham, N. (1985) A Petri net approach to the modelling and analysis of flexible manufacturing systems. *Ann. Operat. Res.*, **3**, 449–472.
- Peterson, J. L. (1981) *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, NJ.
- Silva, M. (1985) *Las redes de Petri en la Automatica y la Informatica*. Editorial AC, Madrid.
- Silva, M. (1987) Toward a synchrony theory for P/T nets. In Voss, K., Genrich, H. J. and Rozenberg, G. (eds.), *Concurrency and Nets*. Springer-Verlag, Berlin.
- Suzuki, I. and Murata, T. (1983a) A method for stepwise refinements and abstractions of Petri nets. *J. Comp. Syst. Sci.*, **27**, 51–76.
- Valvanis, K. S. (1990) On the hierarchical analysis and simulation of flexible manufacturing systems with extended Petri nets. *IEEE Trans. Syst., Man, Cybernet.*, **SMC-20**, 94–100.
- Valette, R. (1979) Analysis of Petri nets by stepwise refinements. *Comp. Syst. Sci.*, **18**, 35–46.
- Villarroel, J. L., Martinez, J. and Silva, M. (1988) GRAMAN: a graphic system for manufacturing system design. *IMACS Int. Symp. Systems Modelling & Simulation (SMS '88)*, Cetraro, Italy.
- Yaw, Y. (1987) *Analysis and Synthesis of Distributed Systems and Protocols*. Doctoral Dissertation, Department of Electrical and Computer Science, University of California, Berkeley.
- Yaw, Y., Ramamoorthy, C. V. and Tsai, W. T. (1988) Synthesis rules for cyclic interactions among processes in concurrent systems. *Comsac Symposium*, Oct. 1988, pp. 496–504.
- Yaw, Y. and Foun, F. L. (1989) The algorithm of a synthesis technique for concurrent systems. *IEEE Int. Workshop on Petri Nets and Performance Models*, Tokyo, pp. 266–279.
- Zhou, M. C. and DiCesare, F. (1989). Adaptive design of Petri net controllers for error recovery in automated manufacturing systems. *IEEE Trans. Syst., Man, Cybernet.*, **SMC-19**, 963–973.
- Zhou, M. C., DiCesare, F. and Desrochers, A. A. (1989). A top-down modular approach to synthesis of Petri net models for manufacturing systems. In *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 534–539. Phoenix, AZ.
- Zhou, M. C. (1990) *A Theory for the Synthesis and Augmentation of Petri Nets in Automation*. Doctoral Dissertation, Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY.
- Zhou, M. C., DiCesare, F. and Rudolph, D. (1990) Control of a flexible manufacturing system using Petri nets. In *1990 IFAC World Congress*, Vol. 9, pp. 43–48. Tallin.
- Zhou, M. C., McDermott, K., Patel, P. and Tang, T. (1991a) Construction of Petri net based mathematical models for an FMS cell. In *1991 IEEE. Int. Conf. Systems, Man, and Cybernetics*, pp. 367–372. Charlottesville, VA.
- Zhou, M. C. and DiCesare, F. (1991b) Parallel and sequential mutual exclusions for Petri net modeling for manufacturing systems with shared resources. *IEEE Trans. Robotics Automation*, **7**, 515–527.
- Zhou, M. C., DiCesare, F. and Desrochers, A. A. (1992) A hybrid methodology for Petri net synthesis of manufacturing systems. *IEEE Trans. Robotics Automation*, **8**, 350–360.