continuous and discrete time control engineering, spanning adaptive control, classical and modern controller design, linear-quadratic-Gaussian (LQG) design, Kalman filtering, parameter identification, robotics, and stability and aperiodicity. The delta operator model thereby lays the foundation for a unified approach to control engineering education, research and industrial practice [3]. Finally, the delta operator viewpoint opens vistas for the design of reliable and robust algorithms throughout engineering computation. What remains is a comparative case-study evaluation of delta operator and w' plane control system design and implementation.

REFERENCES

[1] R. H. Middleton and G. C. Goodwin, "Improved finite word length characteristics in digital control using delta operators," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 1015–1021, Nov. 1986.

[2] G. C. Goodwin and R. H. Middleton, "Continuous and discrete adaptive control," in *Control and Dynamic Systems*, C. T. Leondes, Ed., vol. 25: *System Identification and Adaptive Control* (Part 1 of 3). Orlando, FL: Academic, 1987, pp. 151–185.

[3] R. H. Middleton and G. C. Goodwin, *Digital Control and Estimation: A Unified Approach*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[4] J. Tschauner, *Introduction à la Théorie des Systèmes Échantillonnés*. Paris: Dunod, 1963, p. 42.

[5] J. M. Edmunds, "Identifying sampled data systems using difference operator models," UMIST Contr. Syst. Rep. No. 601, Univ. Manchester, 1985.

[6] G. C. Goodwin, M. E. Salgado and R. H. Middleton, "Indirect adaptive control: An integrated approach," in *Proc. 1988 American Contr. Conf.*, Atlanta, GA, June 1988, pp. 2440–2445.

[7] D. Williamson, "Delay replacement in direct form structures," *IEEE Trans. Acoust., Speech Signal Processing*, vol. 36, pp. 453–460, Apr. 1988.

[8] F. Jabbari, "Lattice filters for RLS estimation of a delta operator-based model," *IEEE Trans. Automat. Contr.*, vol. 36, July 1991, pp. 869–875.

[9] R. Vijayan *et al.*, "A Levinson-type algorithm for modeling fast-sampled data," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 314–321, Mar. 1991.

[10] M. Salgado, R. H. Middleton and G. C. Goodwin, "Connection between continuous and discrete Riccati equations with applications to Kalman filtering," *IEE Proc.*, vol. 135, pt. D, no. 1, Jan. 1988, pp. 28–34.

[11] G. C. Goodwin, R. H. Middleton and H. V. Poor, "High-Speed Digital Signal Processing and Control," *Proc. IEEE*, vol. 80, no. 2, Feb. 1992, pp. 240–259.

[12] C. P. Neuman and V. D. Tourassis, "Discrete dynamic robot models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 193–204, Mar./Apr. 1985.

[13] V. Peterka, "Control of uncertain processes: Applied theory and algorithms," *Kybernetika*, vol. 22, pp. 1–102, 1986.

[14] B. Wahlberg, "Limit results for sampled systems," *Int. J. Control*, vol. 48, no. 3, pp. 1267–1283, Sept. 1988.

[15] B. Wahlberg, "The effects of rapid sampling in system identification," *Automatica*, vol. 26, no. 1, pp. 167–170, Jan. 1990.

[16] C. P. Neuman, "Transformations between delta and forward shift operator transfer function models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-23, pp. 295–296, Jan./Feb. 1993.

[17] R. F. Whitbeck and L. G. Hofmann, "Digital control law synthesis in the w' domain," *AIAA J. Guidance Contr.*, vol. 1, no. 5, pp. 319–326, Sept./Oct. 1978.

[18] M. B. Tischler, "Digital control of highly augmented combat rotorcraft," *NASA TM 88346*, Ames Res. Center, Moffett Field, CA, May 1987.

[19] K. J. Astrom and B. Wittenmark, *Computer Controlled Systems: Theory and Design. second ed.*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[20] C. P. Neuman and C. S. Baradello, "Digital transfer functions for microcomputer control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, pp. 856–860, Dec. 1979.

[21] Y. Takahashi, M. J. Rabins, and D. M. Auslander, *Control and Dynamic Systems*. Reading, MA: Addison-Wesley, 1970, p. 103.

[22] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Rev.*, vol. 20, no. 4, pp. 801–836, Oct. 1978.

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations. second ed.*. Baltimore: The Johns Hopkins Univ. Press, 1989.

[24] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Trans. Automat. Contr.*, vol. AC-23, pp. 395–404, June 1978.

[25] E. S. Armstrong, "Series representation for the weighting matrices in the sampled-data optimal linear regulator problem," *IEEE Trans. Automat. Contr.*, vol. AC-23, pp. 478–479, June 1978.

[26] E. S. Armstrong and A. K. Caglayan, "An algorithm for the weighting matrices in the sampled-data optimal linear regulator problem," *NASA TN D-8372*, NASA Langley Res. Center, Hampton, VA, Dec. 1976.

[27] S. Z. Qi, B. Lennartson and B. Qvarnstrom, "Algorithms for the design of optimal control of time continuous processes using long control and sampling intervals," Tech. Rep., Dept. Contr. Eng., Chalmers Univ. Technol., Gothenburg, Sweden, 1981.

[28] B. C. Kuo, *Digital Control Systems*. second ed. Fort Worth: Saunders, 1992.

[29] A. G. J. McFarlane and N. Karcanias, "Poles and zeros of linear multivariable systems: A survey of the algebraic, geometric and complex-variable theory," *Int. J. Con.*, vol. 24, no. 1, pp. 33–74, July 1976.

[30] A. Emami-Naeini and P. Van Dooren, "Computation of zeros of linear multivariable systems," *Automatica*, vol. 18, no. 4, pp. 415–430, July 1982.

[31] K. J. Astrom, P. Hagander and J. Sternby, "Zeros of sampled systems," *Automatica*, vol. 20, no. 1, pp. 31–38, Jan. 1984.

A Review of Synthesis Techniques for Petri Nets with Applications to Automated Manufacturing Systems

Mu Der Jeng and Frank DiCesare

Abstract— **The paper presents a review of research results in both bottom–up and top–down synthesis techniques for Petri net modeling. These methods can be adopted for representing parallel and distributed application environments such as automated manufacturing systems. They provide a solution for handling complex and concurrent interactions. Bottom–up techniques, consisting of the merging of places and sharing of simple elementary paths, have the advantage of ease of system description since the modeled subsystems usually have real-life correspondences. Nevertheless, with existing bottom–up techniques, the synthesized system may not exhibit the same control properties as the subsystems. Top–down methods, including refinement of transitions and refinement of places, have the advantage of viewing the system globally, which may generate more structured designs. However, it is difficult to apply these methods to the environments with highly shared resources. Examples in the context of automated manufacturing systems are given to demonstrate application of these techniques. Finally, this correspondence discusses Petri net reduction techniques and their relationship to synthesis methods.**

I. INTRODUCTION

Since Petri nets were introduced, they have been widely studied and applied for modeling and analyzing concurrent systems such as

Manuscript received June 18, 1990; revised June 8, 1991 and August 5, 1992. This work was supported by the New York State Center for Advanced Technologies in Automation and Robotics and by the Computer Integrated Manufacturing program of the Center for Manufacturing Productivity and Technology Transfer, both at Rensselaer Polytechnic Institute.

M. D. Jeng is with the Department of Electrical Engineering, National Taiwan Ocean University, Keelung, Taiwan, Republic of China.

F. DiCesare is with the Department of Electrical, Computer, and Systems Engineering and the Center for Manufacturing Productivity and Technology Transfer, Rensselaer Polytechnic Institute, Troy, NY 12180.

IEEE Log Number 9205800.

computers [6], [7], [22], [36], [44], [52], communication protocols [11], [15], [35], and manufacturing [31], [33], [41], [50], [51]. Informally, a Petri net can be depicted as a graph-based static structure and a dynamic behavior, which represents the information (control) flow within the modeled system. Using Petri nets to represent concurrency, conflict, and mutual exclusion in a system is convenient and straightforward, but problems arise when the system to be modeled is very complex. In these cases, the final models will have a large state space and will therefore be difficult to analyze. Two approaches exist to address the issue of complexity caused by state space explosion. One method is to reduce the Petri net while maintaining the logical properties of interest. Then the analysis of the reduced net will provide the same results as the original net. Powerful methods have been developed to do this [9], [10], [27]–[29]. An alternate approach is the development of systematic methods for the synthesis of models such that important system properties are either built in the resulting model or can be easily analyzed using the model. This paper presents a review of research results for such synthesis methods as they relate to automated manufacturing systems. In addition, examples of use of these techniques are provided.

In the following sections, major advances concerning top–down and bottom–up Petri net synthesis methods are discussed. We also briefly review Petri net reduction techniques, which are primarily used for analysis, and their relation to synthesis techniques. Emphasis is placed on application of synthesis and reduction methods for the design and control of concurrent systems. Section II describes Petri net models, including their formal definitions, properties, advantages, disadvantages, and limitations. Section III, IV, and V review bottom–up, top–down, and reduction techniques, respectively. Throughout the discussion of synthesis methods, their applications will be demonstrated using examples in the context of automated manufacturing systems. This domain reveals many typical and important characteristics of discrete event systems such as concurrency, mutual exclusion, sequential dependence, and resource sharing. Section VI gives a conclusion and future directions.

II. Petri Net Models

Petri nets and their properties are now defined so that Petri nets can be discussed in a formal manner. The notation used in this section will be adopted throughout the paper. For more detail please refer to [2], [4], [40], [42], and [43] .

A. Formal Definitions and Properties

Definition 1: A Petri net (PN) is a directed graph consisting of two types of nodes, places and transitions, in which places are always followed by transitions, and vice versa. A Petri net is defined as a quadruple, $PN = (P, T, I, O)$ in which

1) P and T are finite sets of *places* and *transitions*, respectively, such that $P \cap T \neq \phi$;
2) I: $P \times T \rightarrow \{0, 1\}$ is called the input function which specifies input places of transitions;
3) O: $P \times T \rightarrow \{0, 1\}$ is called the output function which specifies output places of transitions.

If 2) and 3) are defined as I: $P \times T \rightarrow N$ and I: $P \times T \rightarrow N$, where N is the set of natural numbers, i.e., multiple arcs are allowed, then it is called a generalized Petri net.

Definition 2: A *marking* m of a PN is a function m: $P \rightarrow N$, which gives the number of *tokens* in each place $p \in P$. The presence or absence of tokens indicates the status of a place, and the marking of places represents either the availability of resources or the occurrence of operations.

Definition 3: A marked PN is a 5-tuple (P, T, I, O, m_0) where m_0 is the initial marking.

Definition 4: A transition t is said to be *enabled* for a marking m if and only if $\forall p \in P$, $m(p) \geq I(p, t)$. If a transition is enabled, then it can be *fired.*

Definition 5 (Firing Rule): Petri nets model system dynamics by using the firing rule. Firing is interpreted as a change of state or the occurrence of an event. To fire an enabled transition t, $I(p, t)$ tokens are removed from each of its input places, and $O(p, t)$ tokens are added to each of its output places. That is

$$m_{k+1}(p_i) = m_k(p_i) + O(p_i, t) - I(p_i, t) \quad \text{for } i = 1, 2, \cdots, |P|$$

where m_k and m_{k+1} are the marking functions before and after the firing, respectively. If there are several enabled transitions in a Petri net, firing these transitions may occur nondeterministically. In other words, any firing sequence for these transitions will be possible.

Let m and m_0 be the vector notation of marking functions, and I and O be the matrix notation of the input and output functions, respectively. Using the enabling and firing rule iteratively from the initial marking m_0, the following matrix state equation for a marking m can be derived.

$$m = m_0 + (O - I) \cdot u = m_0 + C \cdot u \tag{1}$$

where C is called the *incidence* (or *flow*) *matrix*, and u is called the *firing count vector*, which represents the number of firings of each transition. If there exists a firing sequence σ from m to m_0, then m is said to be *reachable* from m_0. The reachability set $R(m_0)$ is defined as the set of all markings reachable from m_0.

Definition 6: $\cdot p$ is the *preset* of p, and $p \cdot$ is the *postset* of p, where

$$\cdot p = \{t | O(p, t) = 1, \quad \forall t \in T\}$$

$$p \cdot = \{t | I(p, t) = 1, \quad \forall t \in T\}$$

that is, $\cdot p$ is the set of input transitions of a place p and $p \cdot$ is the set of output transitions of a place p.

Definition 7: A transition t is said to be *live* if for each marking $m \in R(m_0)$ there exists a marking reachable from m in which t is enabled. A marked PN is live if every transition is live.

Definition 8: A place p is said to be *bounded* if and only if there exists a fixed k such that $m(p) \leq k$ for all $m \in R(m_0)$. If k is equal to 1 then the place is said to be *safe.* A marked PN is bounded (safe) if every place is bounded (safe).

Definition 9: A marked PN is said to be *reversible* if for any $m \in R(m_0)$ there exists a firing sequence of transitions such that $m_0 \in R(m)$.

Definition 10: A *P-invariant* is a column vector x such that

$$x^T C = O \tag{2}$$

where C is the incidence matrix. Considering (1) and (2), we have the following result:

$$x^T m = x^T m_0. \tag{3}$$

This equality implies that for any reachable marking m from m_0, the total number of tokens, weighted by x, is constant. Note that the set of places corresponding to nonzero entries in a P-invariant $x \geq 0$ is called the *support* of x, denoted by $\|x\|$.

P-invariants can be used for checking the boundedness and liveness of a Petri net [4]. If each place of a net is covered by some P-invariant, then the net is bounded. Unfortunately, P-invariants only assist verification of liveness. We will explain this in detail in the next section.

Definition 11: A *T-invariant* is a column vector y such that

$$Cy = 0. \tag{4}$$

Considering (1) and (4) with $y = u$, we obtain the following equation.

$$m = m_0 \tag{5}$$

There may exist a firing sequence σ from m_0 back to m_0 such that y is the firing count vector of σ. However, this σ may not exist. In other words, (4) is a necessary but not a sufficient condition for the existence of such a sequence.

T-invariants can be used for checking the reversibility of a Petri net [4]. Obviously, if a net has no T-invariants, then it is not reversible.

B. Marked Graphs

Marked graphs [13], which have been frequently used for modeling decision-free (i.e., nonconflicting) concurrent systems, form a subclass of Petri nets. The following definition describes a Petri-net type of notation, although marked graphs are usually represented with a different notation [20], [38]–[40].

Definition 12: A *marked graph* is a marked Petri net (P, T, I, O, m_0), where

1) P and T are finite sets of places and transitions, respectively, such that $P \cap T = \phi$ and $P \cup T \neq \phi$;
2) I: $P \times T \rightarrow \{0, 1\}$ is the input function;
3) O: $P \times T \rightarrow \{0, 1\}$ is the output function;
4) m_0: $P \rightarrow N$ is the initial marking;
5) for each $p \in P$, $|\cdot p| = 1$ and $|p \cdot| = 1$, i.e., each place is an input for exactly one transition and an output for exactly one transition.

C. Advantages, Disadvantages, and Limitations

Petri nets are an effective representation for concurrent processes for several reasons. First, the graphical notation of Petri nets allow for a concise visualization of system dependencies; and with tokens, we can use a simple Petri net graph structure to generate very complex dynamic behavior. In addition they provide the modeler with the ability to deal with complexity by focusing on local information. Second, the mathematical theory of Petri nets provides for analysis techniques, such as the invariant method [41], the reachability graph method [4], [40], [42], [43], and the reduction method [9], [10], [27], [28], [29]. These allow for model verification of properties such as boundedness, liveness, and reversibility. Third, since a bounded Petri net contains an embedded Markov chain, we are also able use Petri nets to evaluate the performance of a system [37], [55]. Last but not least is the ability to use the net description to directly implement computer control for the system [14], [21], [32], [49]. In summary, Petri nets are a very promising tool for specification, analysis, performance evaluation, and implementation of concurrent systems.

The main disadvantage of Petri nets lies in the size of the nets produced by modeling very complex discrete event systems, such as flexible manufacturing systems [31]. In order to tackle the complexity found in these cases, the modeling procedure must be guided by systematic rules or methodologies. For these large nets, although analytic techniques are available for their verification, the computation involved in analyzing them is still very substantial, and, in many cases, it may not be practical.

The modeling capability of Petri nets is less powerful than that of universal Turing machines. Therefore, some phenomena cannot be described with Petri nets. For example, as we discussed above, if two or more transitions are firable, then their actual firing sequence is nondeterministic. That is, it is impossible to impose firing priorities on the sequence. To eliminate this limitation, extensions of Petri nets [1] have been developed. One of them is the addition of inhibitor arcs to Petri nets. It is proven that this type of Petri net has the modeling power of Turing machines [1]. However, some of the previously mentioned analytic techniques are no longer applicable [42], [43] for these nets.

The ability to use analytical methods is well worth the compromise of modeling power. Petri nets are a powerful tool with which to address complexity because they allow for bottom–up or modular synthesis, top–down or step-wise refinement, and hybrid methods as reviewed here.

III. Bottom–Up Synthesis Techniques

Bottom–up methods with Petri nets are applied as follows. First, each subsystem is separately modeled, ignoring interactions with other subsystems. These subsystem models are usually very simple and easy to verify. Different subsystem models may have places or transitions in common, e.g., places which represent the same resource availability or activity status in the different subsystems. These common places or transitions represent the interactions among the subsystems. At each synthesis step, these interactions are considered, and the corresponding subsystems are combined through merging these places and/or transitions into a larger subsystem. Analysis of the combined net is usually done immediately after each step, so that the final analysis is simplified. At the end of the synthesis, the final system and some of its important properties are obtained.

A. Merging of Places

Agerwala and Choed-Amphai [3] have proposed a systematic bottom–up approach for synthesizing concurrent systems modeled by Petri nets. At each synthesis step, subnets can be merged in such a way that a set of places are merged into a new place. This is called a one-way merge, as described in the following paragraph.

Given a Petri net $N = (P, T, I, O)$, select a set of places to be merged $P_m \subseteq P$, such that

1) For any $p_i, p_j \in P_m$, if $I(p_i, t) = 1$ and $I(p_j, t) = 1$, then $i = j$.
   (If p_i and p_j are input places for the same transition, then they cannot be merged.)
2) For any $p_i, p_j \in P_m$, if $O(p_i, t) = 1$ and $O(p_j, t) = 1$, then $i = j$.
   (If p_i and p_j are output places for the same transition, then they cannot be merged.)

Then, construct the resultant net $N' = (P', T', I', O')$, such that

1) $T' = T$
2) $P' = (P - P_m) \cup \{p\}$ where $p \notin P$
3) I' and O' are obtained by replacing every occurrence of each $p_i \in P_m$ in I and O by p.

In other words, all places in P_m are merged to a place named p. For example, Fig. 1(b) is obtained by merging p_3 and p_6 of Fig. 1(a) into p_3'. At the end of the synthesis, we obtain the system model, the merged Petri net which results from all the one-way merges.

In order to facilitate the analysis, Agerwala and Choed-Amphai [3] have provided a theorem that states that after every one-way merge, the P-invariants of the resultant net can be derived from the P-invariants of the subnets, as described in the following.

Theorem 1 [3]: Consider a net $N = (P, T, I, O)$ on which the one-way merge operation is applied to construct a net $N' = (P', T', I', O')$. All the P-invariant supports of N' are obtained from the P-invariant supports of N as follows.

$PI' \subseteq P'$ is a P-invariant support of N' if and only if there exists a P-invariant support PI on N, such that

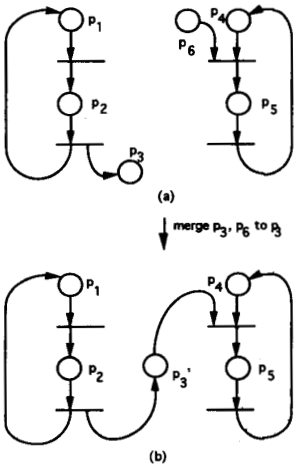


Fig. 1. An example of one-way merge (modified from [3]).

1) If $P_m \subseteq PI$ then $PI' = (PI - P_m) \cup \{p\}$
2) If $P_m \cap PI = \phi$ then $PI' = PI$.

Note that if the synthesis rule is restricted in such a way that a merge operation is allowed only if each place in the resultant net is in some P-invariant, then the class of nets synthesized in this way is equivalent to the class of bounded Petri nets, because all the places are in some P-invariants.

Narahari and Viswanadham [41] have extended this theorem and developed a similar theorem for T-invariants for a net obtained by merging transitions. They use these theorems to verify important qualitative properties of nets, such as existence/absence of deadlock (liveness), conservativeness, and boundedness after each synthesis step. The theorem for P-invariants allows more than one set of places to be merged at each synthesis step, and allows simplified computation of the P-invariants of the merged net when the P-invariants of the subnets are known. This research was done in the context of modeling and analysis of flexible manufacturing systems. In their approach, a Petri net is first created for every basic operation such as a machine operation for a product. A product might need more than one machine operation, and there might be several choices for selecting machines to execute these machine operations. In addition, several different types of products might be manufactured in the system. The subnets that represent the products can be obtained by merging the places of the subnets that denote the machine operations. Therefore, the system Petri net can be constructed from merging the places of the subnets that represent the products.

Example 1: In this illustration, we show how to design the coordination for a simple automated manufacturing system consisting of two robots: R_1, R_2, and three machines: M_1, M_2, M_3, as shown in Fig. 2, using Agerwala and Choed-Amphai's method. This example could also be accomplished by using Narahari and Viswanadham's method. The system that we are going to model only produces one type of product, and requires two machine operations: a product is produced by a machine operation 1 on M_1 followed by a machine operation 2 on M_2 or M_3.

The procedure to manufacture a product is described as follows.

1) R_1 takes raw stock from storage S_1 and loads M_1.
2) M_1 starts machining.
3) After M_1 finishes its operation, R_2 takes the intermediate product from M_1 to M_2 or M_3 for further machining.
4) Any finished product on M_2 or M_3 will be moved by R_2 to storage S_2.

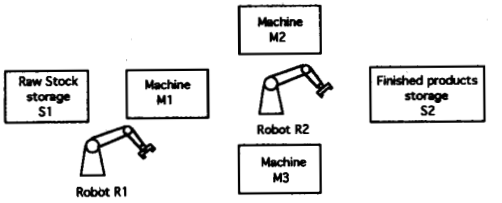


Fig. 2. A simple automated manufacturing system.

Although simple in its configuration this example demonstrates some important features of concurrent systems: 1) concurrency: any robot or machine is an autonomous system which can operate in parallel with others; 2) choice: a second machine operation can be done on either M_2 or M_3; 3) resource sharing: robot R_2 is shared; and 4) sequential dependence: the second operation on a product cannot occur until the completion of the first operation. In addition, it demonstrates some flexibility in the context of manufacturing since the system produces one product type but employs two possible sequences for making it.

There are two assumptions in this example: 1) the supply of raw stock is never exhausted; that is, there is always raw stock in its storage place ready for use. 2) the finished product will be taken away so that there is no overflow problem with its storage place.

Before using one-way merges, we specify the individual subsystems separately. Remember that at this stage any interaction among the subsystems is neglected. Fig. 3(a) shows operations on M_1 and an initial merge of p_{12} and p_{15} into p'_{12}. Using the state equation of the model in Fig. 3(a) it is easy to find that the P-invariant supports of the model after the initial merges in Fig. 3(a) are $\{p_{11}, p_{13}, p_{14}, p_{16}\}$ and $\{p'_{12}, p_{13}\}$. Fig. 3(b) denotes operations on M_2, initial merges of p_{23}, p_{27}, p_{28}, and p_{211} into p'_{23}, and an initial merge of p_{22} and p_{210} into p'_{22}. Its P-invariant supports are $\{p_{21}, p_{24}, p_{25}\}$, $\{p'_{22}, p_{24}, p_{26}, p_{29}\}$, and $\{p'_{23}, p_{24}, p_{29}\}$. Fig. 3(c) describes operations on M_3, initial merges of p_{33}, p_{37}, p_{38}, and p_{311} into p'_{33}, and an initial merge of p_{32} and p_{310} into p'_{32}. Its P-invariant supports are $\{p_{31}, p_{34}, p_{35}\}$, $\{p'_{32}, p_{34}, p_{36}, p_{39}\}$, and $\{p'_{33}, p_{34}, p_{39}\}$. Note that places with the same description will be merged to a single place after a one-way merge.

After several one-way merges, each of which merges one set of places, the resultant model can be obtained as shown in Fig. 3(d). By using Theorem 1, the net has the following P-invariant supports (not necessarily minimal): $\{p'_{11}, p_{13}, p_{14}, p'_{16}, p_{24}\}$, $\{p'_{11}, p_{13}, p_{14}, p'_{16}, p_{34}\}$, $\{p'_{12}, p_{13}\}$, $\{p'_{22}, p_{24}, p_{26}, p_{29}\}$, $\{p'_{32}, p_{34}, p_{36}, p_{39}\}$, and $\{p''_{23}, p_{24}, p_{29}, p_{34}, p_{39}\}$. If each of p'_{11}, p'_{12}, p'_{22}, p'_{23}, and p'_{32} is marked with one token initially, then from Definition 10, we have the following equations:

$$m(p'_{11}) + m(p_{13}) + m(p_{14}) + m(p'_{16}) + m(p_{24}) = 1 \quad (6)$$

$$m(p'_{11}) + m(p_{13}) + m(p_{14}) + m(p'_{16}) + m(p_{34}) = 1 \quad (7)$$

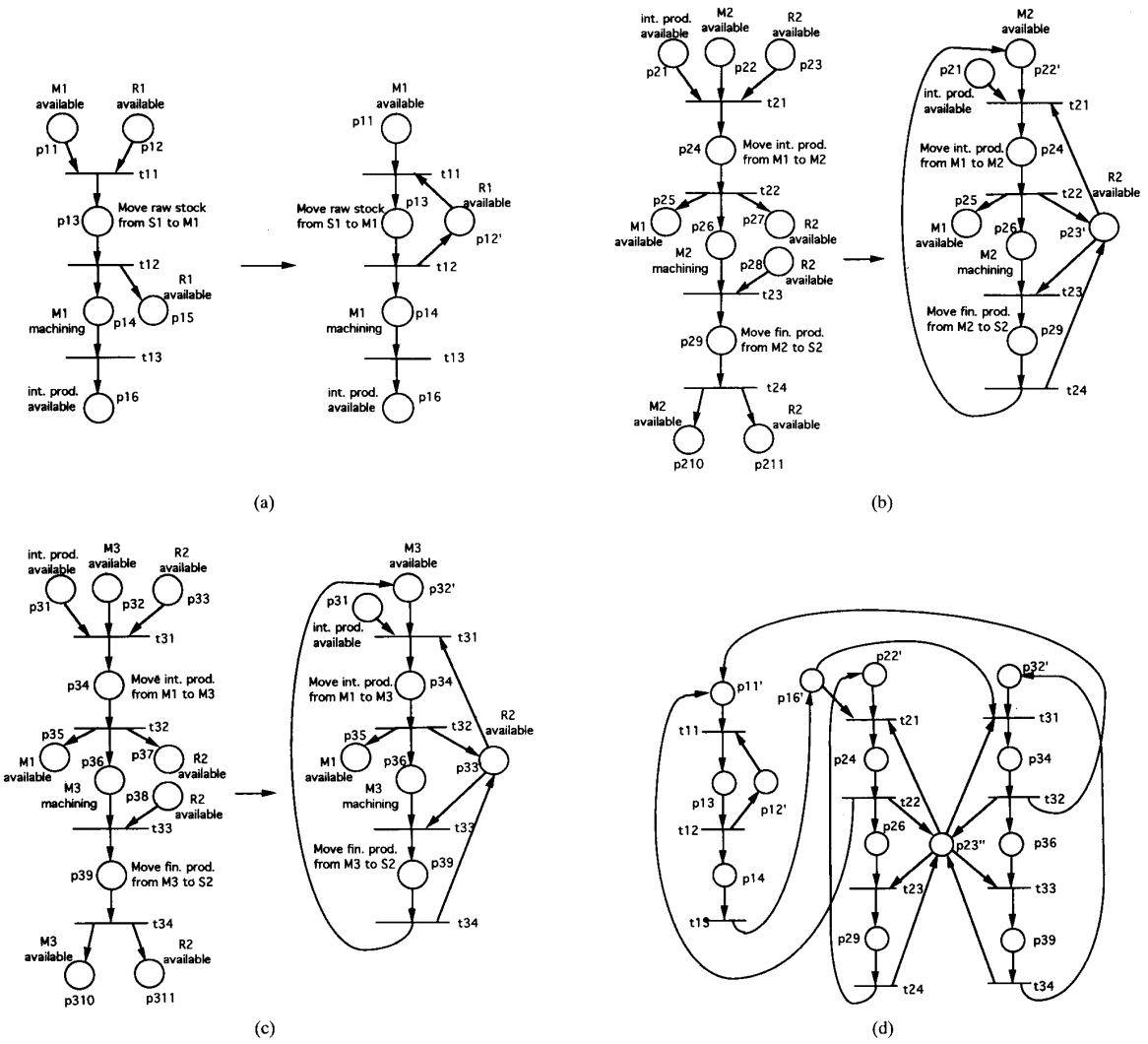
$$m(p'_{12}) + m(p_{13}) = 1 \quad (8)$$

$$m(p'_{22}) + m(p_{24}) + m(p_{26}) + m(p_{29}) = 1 \quad (9)$$

$$m(p'_{32}) + m(p_{34}) + m(p_{36}) + m(p_{39}) = 1 \quad (10)$$

$$m(p''_{23}) + m(p_{24}) + m(p_{29}) + m(p_{34}) + m(p_{39}) = 1. \quad (11)$$

We shall prove that the net is live, bounded, and safe for this initial marking. The boundedness is easy to verify because each place is covered by some P-invariant, but checking liveness takes more effort, because we have to verify that each transition is live. Fortunately, with the help of the P-invariants this job is simplified. First consider the liveness of t_{11}.



(a) (b) (c) (d)

Fig. 3. (a) Operations on $M1$ (an initial merge). (b) Operations on $M2$ (initial merges). (c) Operations on $M3$ (initial merges). (d) Final model after one-way merges.

1) If $m(p'_{11}) = 1$ and $m(p'_{12}) = 1$, then t_{11} is enabled. Otherwise, if $m(p'_{11}) = 0$ or $m(p'_{12}) = 0$, then t_{11} is disabled.

2) For $m(p'_{12}) = 0$, from (8), $m(p_{13}) = 1$, which enables t_{12}. After t_{12} fires, $m(p'_{12}) = 1$. If $m(p'_{11}) = 1$, then t_{11} is enabled.

3) For $m(p'_{11}) = 0$, from 6 and (7), we have

$$m(p_{13}) + m(p_{14}) + m(p'_{16}) + m(p_{24}) = 1 \tag{12}$$

$$m(p_{13}) + m(p_{14}) + m(p'_{16}) + m(p_{34}) = 1. \tag{13}$$

From the previous two equations, we know that one token resides in either p_{13}, p_{14}, p'_{16}, or p_{24}, and one token resides in either p_{13}, p_{14}, p'_{16}, or p_{34}. For the case where a token is in p_{13}, the token will reach p_{14}, and then p'_{16} by firing t_{12} and t_{13}, which are enabled when the token is in p_{13} and p_{14}, respectively.

3a) Now, if $m(p'_{22}) = 1$ and $m(p''_{23}) = 1$, then t_{21} is enabled. However, if $m(p'_{22}) = 0$ or $m(p''_{23}) = 0$, then t_{21} is disabled.

3b) For $m(p'_{22}) = 0$, by (9) and (12), we know that either $m(p_{26}) = 1$ or $m(p_{29}) = 1$ (because $m(p'_{16}) = 1$, $m(p'_{24}) = 0$). If $m(p_{26}) = 1$, by (11), (12), and (13), either $m(p''_{23}) = 1$ or $m(p_{39}) = 1$. But if $m(p_{39}) = 1$ then by firing t_{34}, $m(p''_{23}) = 1$. After t_{23} fires, $m(p_{29}) = 1$. Thus, once t_{24} fires, $m(p'_{22}) = 1$.

3c) For $m(p''_{23}) = 0$, by (11), (12), and (13), we have either $m(p_{29}) = 1$ or $m(p_{39}) = 1$, which enables t_{24} and t_{34}, respectively. So, eventually $m(p''_{23}) = 1$.

Therefore, for any case, p_{24} will have a token, which enables t_{22}, and after t_{22} fires, $m(p'_{11}) = 1$. Consequently, from 1), 2), and 3) above, t_{11} is live (t_{12}, t_{13}, t_{21}, and t_{22} are also proved to be live by similar argument). The rest of the proof is left to the reader.

B. Sharing of Simple Elementary Paths

Krogh and Beck [8], [25] developed a bottom–up technique for synthesizing live and safe Petrl nets. Their method shares simple elementary paths in which no place or transition appears more than once. They defined two types of simple elementary paths for synthesis purposes: 1) a solitary transition path (STP) is a simple elementary path terminated on both ends by a place, for which each transition in

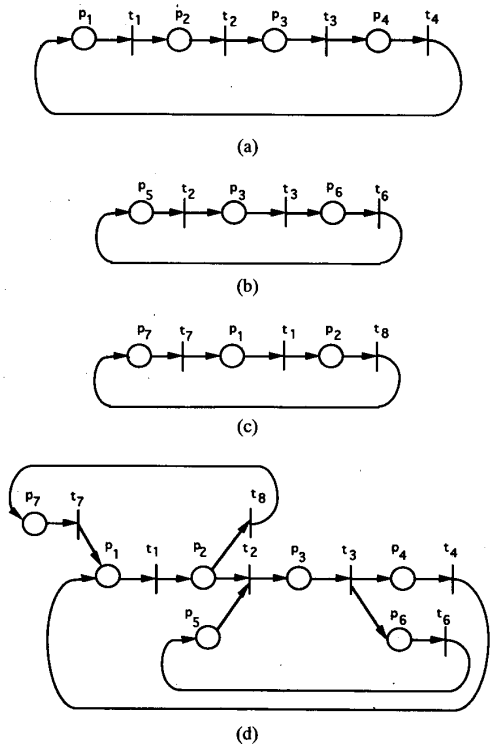


Fig. 4. (a)–(c) Three simple elementary circuits. (d) Three SEC's combined along an STP and an SPP.

the path has exactly one input place and exactly one output place; and 2) a solitary place path (SPP) is a simple elementary path terminated on both ends by a transition, for which each place in the path is an input place for exactly one transition and an output place for exactly one transition.

The synthesis procedure starts with a collection of simple elementary circuits (SEC), Petri nets constructed from finite-length paths with coincident initial and terminal places, e.g., Fig. 4. These SEC's correspond to basic activity cycles in the system with the assumption that the system is designed to perform a repetitive operation. Next, we choose one SEC as the initial Petri net, and combine one of the remaining SEC's with the initial Petri net along an STP or an SPP. We repeat the combining step until all SEC's are included.

Fig. 4 illustrates the synthesis of a Petri net by joining three SEC's. First, the SEC's in Fig. 4(a) and (c) are combined along a common STP $\{p_1, t_1, p_2\}$ and then the resultant net is combined with Fig. 4(b) along a common SPP $\{t_2, p_3, t_3\}$. The final combined net is shown in Fig. 4(d).

The Petri net obtained in this fashion will be live and safe with respect to any initial marking for which there is exactly one token in each of the P-invariants of the system. This conclusion is supported by Krogh and Beck's theorems. For example, if we mark each of p_1 and p_5 above with exactly one token, then the combined net in Fig. 4(d) is live and safe. This is a very strong result. In their method, Krogh and Beck also show that after each synthesis step, the P-invariants of the combined Petri net can be easily calculated [8], [25].

Because their method allows only safe places to model resources such as buffers, it is necessary to use a set of places to describe all the possible states of a buffer. If the buffer is bounded by a large number, then the number of states might be large. This causes the net structure to be large. In addition, because at each synthesis step only one SEC is considered, it is more difficult to use Krogh and Beck's method than to use the two former methods for modeling certain shared resources.

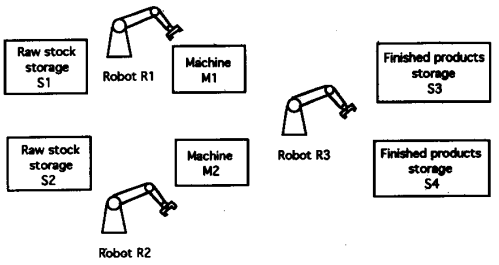


Fig. 5. A simple automated manufacturing system.

Example 2: In this example, we apply Krogh and Beck's method to model the following simple automated manufacturing system (see Fig. 5) which produces two products. The two product flows within the system are independent except that R_3 is shared between them. Again, the same assumptions as those in Example 1 apply to the raw stock storages and finished product storages. In the description below, the names in the parentheses are for product 2. The procedure to make product 1 (product 2) is depicted as follows: robot R_1 (robot R_2) takes raw stock1 (stock 2) from storage S_1 (storage S_2) and loads machine M_1 (machine M_2). Then M_1 (M_2) starts machining. After M_1 (M_2) is finished, robot R_3 unloads M_1 (M_2) and moves product 1 (product 2) to storage S_3 (storage S_4).

In the following figures, basic activity cycles are modeled as SEC's. Two or more simple elementary paths to be merged together are depicted using same place names and same transition names along the paths. Fig. 6(a) describes a cycle where raw stock 1 is machined on M_1. Fig. 6(b) denotes a similar cycle for M_2. Fig. 6(c)–(f) describes the moving cycles for robot R_1, R_2, and R_3.

Fig. 6(g) shows the final model after merging simple elementary paths (STP or SPP). Table I lists these merges. By using Krogh and Beck's theorems, the model in Fig. 6(g) is live and safe if we mark it in such a way that each of its P-invariants contains exactly one token. For example, if each of the places p_{11}, p_{21}, p_{31}, p_{41}, p_{51} is marked with exactly one token, then the model is live and safe.

Krogh and Beck's result is extended by Koh and DiCesare [23], who invented a similar synthesis method that can be applied to model nets with bounded places and generalized Petri nets, i.e., Petri nets with multiple arcs.

C. Summary of Bottom–Up Synthesis Techniques

In terms of interactions in a concurrent environment, bottom–up methods provide some freedom in specifying the system. At the beginning of the synthesis, we can treat the system as the composition of independent subsystems and neglect any interaction. This has the advantage of ease in describing the system, since the subsystems usually have real-life correspondences such as robots, machines, and transport devices. The tradeoff is that we may lose some control in the behavior of the composed net so that some important properties, e.g., liveness, boundedness, etc., may not be obtained. Nevertheless, using bottom–up synthesis methods can still be advantageous, because the analysis of the final net is simplified. At each synthesis step, some important properties of the resultant net can be easily verified from the information known from the individual subnets.

From the bottom–up synthesis methods discussed previously, we see that the invariant method is often used to analyze the properties of the combined net after each synthesis step. However, the disadvantage

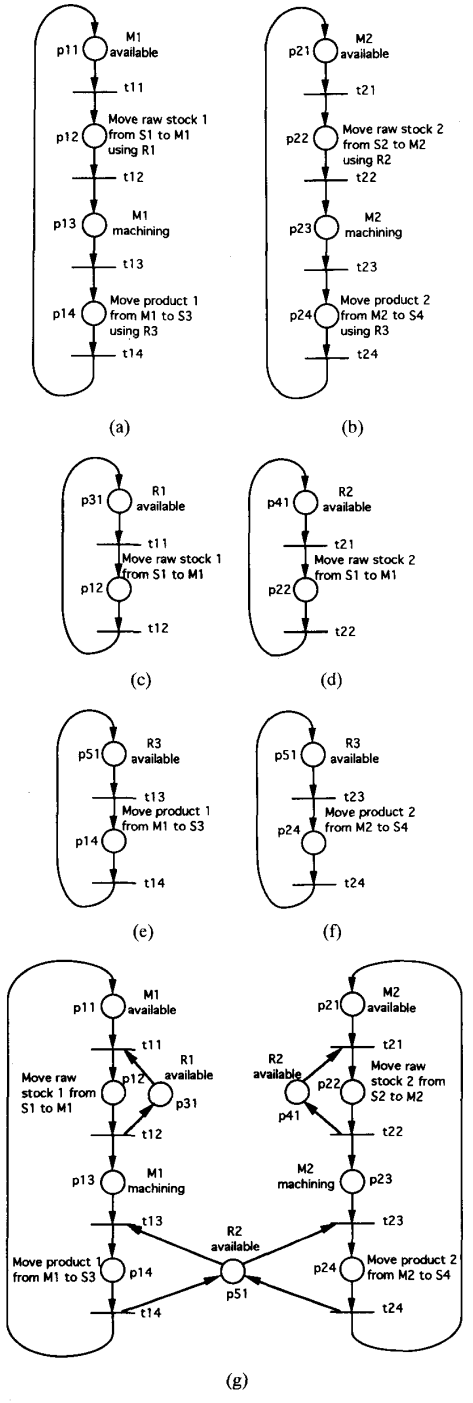


Fig. 6. (a) Cycle $S1$–$M1$–$S3$. (b) Cycle $S2$–$M2$–$S4$. (c)–(f) Moving cycles of Robot $R1$, $R2$, and $R3$. (g) Final model of Example 2.

of using invariants is that they do not convey complete information about the net, which makes investigating some properties, e.g., liveness and reversibility, difficult. Besides, the methods usually do not guarantee that the resultant net will preserve important properties. The exception is that of Krogh and Beck [8], [25], and of Koh and DiCesare [23] in which liveness and safeness or boundedness are preserved. Nevertheless, these bottom–up methods have provided successful solutions in some applications [8], [25], [41]. Top–down synthesis as described in the next section provides a complementary methodology.

TABLE I
MERGES FOR THE NETS IN FIG. 6

SEC's Merged	Common Path	STP or SPP
(a), (c)	$\{t_{11}, p_{12}, t_{12}\}$	SPP
(a), (c), (e)	$\{t_{13}, p_{14}, t_{14}\}$	SPP
(a), (c), (e), (f)	$\{p_{51}\}$	STP
(a), (c), (e), (f), (b)	$\{t_{23}, p_{24}, t_{24}\}$	SPP
(a), (c), (e), (f), (b), (d)	$\{t_{21}, p_{22}, t_{22}\}$	SPP

IV. TOP–DOWN SYNTHESIS TECHNIQUES

Top–down synthesis usually begins with an aggregate model of the system and neglects low-level detail. Then, refinement is done in a stepwise manner to incorporate more detail into the model. The two commonly used schemes for refinements are expanding places and expanding transitions. The refinements continue until the level of detail satisfies the specification of the system. Top–down methods have the advantage of starting with a global view of the system. In addition, many researchers [48], [47], [53], [54] have made efforts to provide methods that guarantee that each synthesis step does not lose important properties of the system so that final analysis will not be necessary.

A. Refinement of Transitions

Valette [48] first studied top–down techniques and proposed a method for stepwise refinement of transitions for synthesizing Petri nets. The argument is that if it is assumed that the firing of a transition is not instantaneous (relaxing the original definition of Petri nets), and is made up of two steps, then it is possible to associate a transition with a complex operation that can later be detailed by means of another Petri net. In order to insure that the important properties of boundedness, safeness, and liveness are preserved after each refinement step, Valette defined the conditions with which a transition can be replaced by a Petri net. Valette's method can be considered a generalization of the results of Bruno and Altman [12] who developed a theory of *asynchronous control networks* for modeling the control structure of digital systems. These are similar in modeling power to Petri nets. Similar work for the reduction and synthesis of marked graphs has been explored by Murata, Koh, and Johnsonbaugh [20], [38], [39].

Suzuki and Murata [47] further generalized Valette's work and defined the concept of *k-well-behaved* Petri nets. They [47] also defined *k-enabled* as follows: a transition t is said to be k-enabled if and only if there exists a marking m reachable from the initial marking such that $\forall p \in P,\ m(p) \geq k \cdot I(p, t)$ where $k \in N$. By using Suzuki and Murata's method, a transition which is not $k+1$-enabled ($k \geq 1$; i.e., it may not be more than k-enabled) can be replaced by a *k-well-behaved* Petri net while preserving the properties of boundedness, safeness, and liveness. A Petri net N is said to be k-well-behaved with respect to two distinct transitions, t_{in} (input transition) and t_{out} (output transition), if and only if the following three conditions hold.

1) t_{in} is live in its associated Petri net $B(N, t_{\text{in}}, t_{\text{out}}, k)$, which is similar to the concept of associated Petri nets defined by Valette, but the idle place may contain up to k initial tokens (refer to Fig. 7 and [47]).

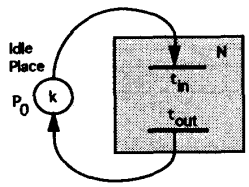


Fig. 7. A Petri net N and its associated $B(N, t_{in}, t_{out}, k)$ [47].

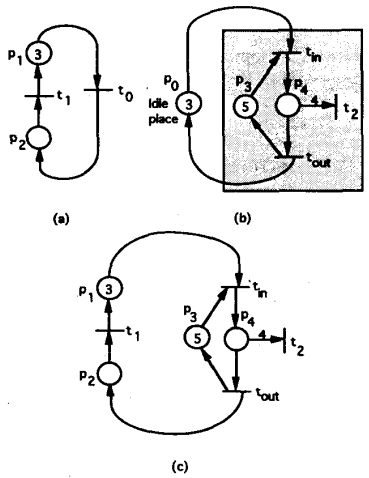


Fig. 8. Example of refinement. (a) Petri net. (b) 3-well-behaved PN and its idle place. (c) Combined Petri net [47].

2) Let $L(B(N, t_{in}, t_{out}, k))$ denote the set of all firing sequences in $B(N, t_{in}, t_{out}, k)$ starting from the initial marking and $\#(\sigma, t)$ denote the number of firing of t in the firing sequence σ. Then, for each $\sigma_1 \in L(B(N, t_{in}, t_{out}, k))$ such that $\#(\sigma_1, t_{in}) > \#(\sigma_1, t_{out})$, there exists $\sigma_2 \in (T - \{t_{in}\})^+$ such that $\sigma_1\sigma_2 \in L(B(N, t_{in}, t_{out}, k))$ and $\#(\sigma_1, t_{in}) = \#(\sigma_1\sigma_2, t_{out})$.

3) $\#(\sigma, t_{in}) \geq \#(\sigma, t_{out})$ for any $\sigma \in L(B(N, t_{in}, t_{out}, k))$.

Condition 1) means that t_{in} can never "get blocked." Conditions 2) and 3) say that t_{in} may "get ahead" of t_{out} in firing, and t_{out} will always "catch up."

Consider the example below. It is easy to verify that transition t_0 in Fig. 8(a) is 1-, 2-, and 3-enabled but not k-enabled for $k \geq 4$. The Petri net (excluding the idle place) in Fig. 8(b) is 1-, 2-, and 3-well-behaved PN with respect to t_{in} and t_{out}. Hence, transition t_0 can be replaced by the Petri net in Fig. 8(b), which results in Fig. 8(c). Note that the numbers in the places stand for token numbers, and the numbers beside the arcs denote the number of arcs (multiple arcs).

Although the results of Valette, Suzuki, and Murata are very useful for constructing a net with important properties, applications of their top–down methods are limited to the systems that can be described as independent modules, i.e., there are only shared resources inside blocks. The reason for this is that a well-behaved Petri net module has only one input and one output.

B. Refinement of Places

Suzuki and Murata [47] also present a refinement method for places, by which the place to be refined, p_0, is first replaced by two places, p_{01} and p_{02}, and by a transition t_0 such that the following conditions are satisfied.

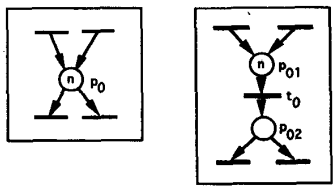


Fig. 9. First step of the refinement of a place [47]. (a) A Petri net. (b) p_0 is replaced by p_{01} and p_{02}.

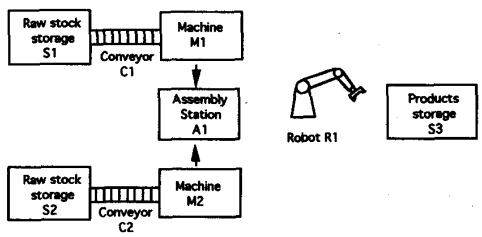


Fig. 10. A simple automated machining and assembly system.

1) The input (output) transitions of p_{01} (p_{02}) are the input (output) transitions of p_0.

2) P_{01} (p_{02}) is the only input (output) place of t_0.

3) The initial marking of p_{01} is equal to the initial marking of p_0.

4) The initial marking of p_{02} is zero (see Fig. 9).

After these conditions are satisfied, the refinement technique for transitions can be applied to t_0. Suzuki and Murata have proven that the method for refining places also preserves the properties of boundedness, safeness, and liveness after each refinement step.

Zhou and DiCesare [53], [54] developed a method which performs stepwise refinements directly on places instead of the two-step method by Suzuki and Murata. The method uses specialized blocks, such as sequence PN, parallel PN, conflict PN, etc., to substitute places. Their method is like those of Murata, Koh, and Johnsonbaugh but not limited to marked graphs. In their papers, Zhou and DiCesare also incorporate two additional concepts called parallel mutual exclusion and sequential mutual exclusion with their top–down synthesis technique. They consider this method a hybrid synthesis approach which is not strictly top–down. The parallel and sequential mutual exclusion modules (the shared resources) are considered in the final synthesis stage, the stage at which all system functions excluding shared resources have been refined in final detail.

Example 3: We use a simple automated machining and assembly system to demonstrate the application of these methods. This system only makes one type of product which needs two machining operations and one assembly operation. As shown in Fig. 10, the system has one assembly station A_1, one robot R_1, and two conveyors for transferring raw stock type 1 and type 2 to two machines, machine M_1 and machine M_2, respectively. Similar to Example 1, we have the following two assumptions for the system.

1) The supply of raw stock 1 and 2 in storage S_1 and storage S_2 is never exhausted.

2) The finished product will be taken away so that there is no overflow problem with storage S_3.

The production procedure is depicted as follows.

1) The conveyor C_1 (C_2) transfers raw stock type 1 (type 2) to M_1 (M_2).

2) M_1 (M_2) start machining.

3) After M_1 and M_2 finish their operations, the robot R_1 takes

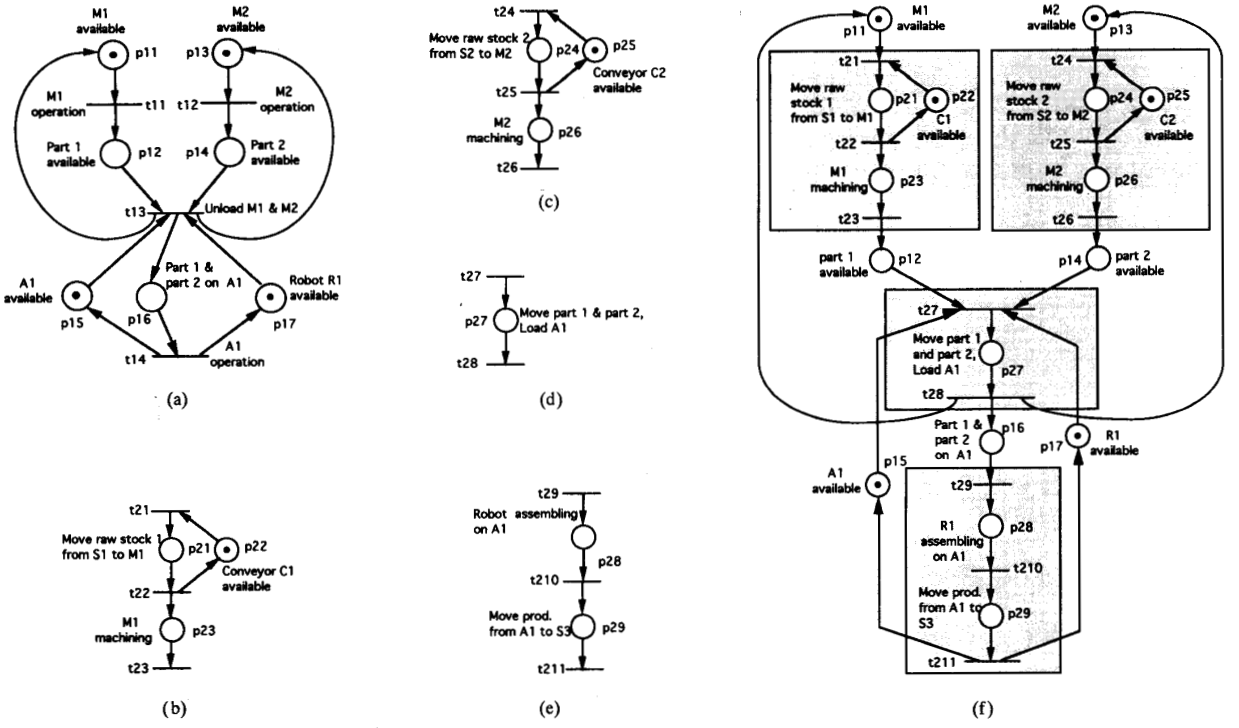


Fig. 11. (a) First level of the system in Example 3. (b) 1-well-behaved Petri net for t_{11}. (c) 1-well-behaved Petri net for t_{12}. (d) 1-well-behaved Petri net for t_{13}. (e) 1-well-behaved Petri net for t_{14}. (f) Final model after refinements of transitions.

part 1 from M_1 to A_1, and then takes part 2 from M_2 to A_1.

4) R_1 begins the assembly on A_1 using part 1 and part 2.
5) After R_1 finishes the assembly on A_1, it moves the product to storage S_3.

Because we refine transitions, the activities in the system are defined intially as transitions. At the first step of the modeling, an abstract model of the system is specified as described in Fig. 11(a) which states that the production procedure needs an M_1 machine operation and an M_2 machine operation, followed by unloading of parts and an A_1 assembly operation. It is easy to verify that this model is live and bounded (safe).

Then, we substitute, respectively, the 1-well-behaved Petri nets that denote M_1 operation, M_2 operation, unloading of parts, and A_1 operation in Fig. 11(b), (c), (d), and (e) for the transitions t_{11}, t_{12}, t_{13}, and t_{14} in Fig. 11(a). For this simple system, one level of refinement is sufficient.

The final model, which is live and bounded (safe), as shown in Fig. 11(f), is obtained after these refinements. The shaded areas in the figure are the 1-well-behaved Petri nets of Fig. 11(b), (c), (d), and (e).

It is not easy to apply Valette, Suzuki, and Murata's top–down methods to model the system in Example 1. The problem is that we have difficulties in specifying the interactions with shared resources by using refinement. Nevertheless, in situations in which shared resources exist within k-well-behaved Petri nets, their methods are still very useful.

C. Summary of Top–Down Synthesis Techniques

The motivation for top–down synthesis methods is that by using some principles, e.g., modularization, information hiding, etc., we restrict the synthesis methods so that certain "good" behaviors can be guaranteed in the final synthesized model. These good behaviors may be qualitative properties such as liveness, boundedness, etc., or just characteristics such as structured design, modularity, etc. This kind of methodology has been used widely in sequential environments such as structured programming techniques for simplifying costly debugging, maintenance, and verification. However, if we want to preserve properties, the same principles are not easily applied to concurrent environments with highly shared resources, because of the complexity of interactions within these environments. The reason is simply that interactions among the subsystems are coupled throughout all steps of synthesis, which makes it more difficult to specify the system using the top–down approach.

On the other hand, if the system can be designed in such a way that shared resources are separated and embodied within modules, then the top–down methods that we have mentioned still provide some solutions. Actually, their concept of k-well-behaved Petri nets decoupled further interactions between subsystems inside the well–behaved module and subsystems outside the well-behaved module. Future extensions of their methods might consider multiple inputs and outputs for more design flexibility.

V. Reduction Techniques

Reduction techniques are primarily used to mitigate the effort of checking Petri net properties, because most reduction steps do not lose important information about the net. Moreover, if we consider the converse process of reduction, i.e., expansion (refinement), then the same techniques may be used to synthesize Petri net models in a top–down manner while preserving important properties. This is obviously very attractive, so in the following sections, a brief review of important reduction methods will be given first, and then the relationship between reduction and synthesis will be discussed.

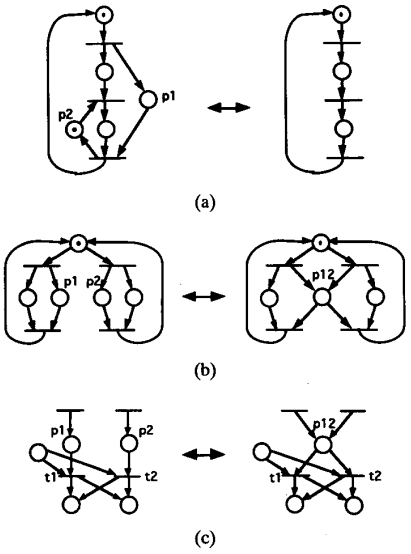


Fig. 12. Examples of transformations of places (modified from [9], [10]). (a) Simplification of redundant places p_1 and p_2. (b) Fusion of doubled places p_1 and p_2 into p_{12}. (c) Fusion of equivalent places p_1 and p_2 into p_{12}.

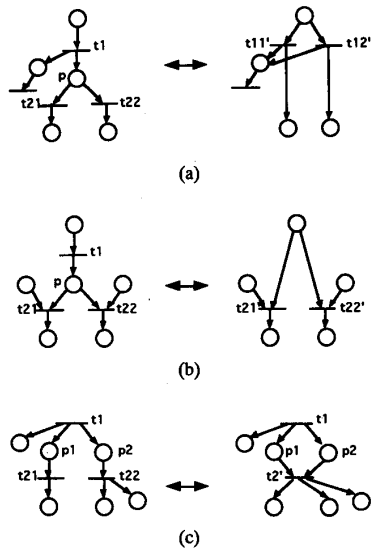


Fig. 13. Examples of fusions of transitions (modified from [9], [10]). (a) Postfusion of transitions t_1 and t_{21} into t'_{11} and t_1 and t_{22} into t'_{12} (p is the sole input of t_{21} and t_{22}. (b) Prefusion of transitions t_1 and t_{21} into t'_{21} and t_1 and t_{22} into t'_{22}. (p is the sole output of t_{11}). (c) Lateral fusion of transitions t_{21} and t_{22} into t'_2.

We will see that although the theoretical results of reduction may be applied for synthesis, and vice versa, there is still one distinction between them—context or interpretation.

A. Reduction by Transformations and Decompositions

Berthelot [9] has extended the work of reduction and verification of parallel programs by Lipton [30], Kwong [26], and Kowalk and Valk [24], and has presented a powerful set of transformation rules for reduction and refinement of Petri nets. In his approach, the reduction procedure is realized by eliminating arcs and by replacing subnets with places and transitions. These rules are classified into three categories:

1) place transformations
2) fusion of transitions
3) addition of nets.

Each transformation rule may be used if some conditions, called application conditions, are satisfied. Most application conditions of the rules are based on the structure of the net, but some depend on its behavior. Although it is easier to verify conditions on static structure than dynamic behavior, the former tends to require much more detail. As for the rules, each category contains a number of subrules. Place transformations consist of three rules: *simplification of redundant places, fusion of doubled places*, and *fusion of equivalent places*. Fusion of transitions is divided into three rules: *postfusion, prefusion*, and *lateral fusion*. As shown in Figs. 12 and 13, examples of place transformations and fusion of transitions are illustrated. Addition of nets differs from the place transformations and fusion of transitions in that they add an external net rather than combine places or transitions. That is, the net is expanded. Addition of nets comprises four rules: *addition of a derivative net, alternation of laterally fusible transitions, identical transition regulation*, and *addition of nonrestricting nets*. The reader is referred to Berthelot's paper [9] for examples. In another of his papers, Berthelot [10] generalized the first two categories of the preceding transformation rules using net's dynamic behavior as much as possible to describe the application conditions of the rules. These rules are general in the sense that they can include the structurally defined reduction rules as subsets.

He also reviewed and organized a number of decomposition schemes [10] to split a system into subsystems which can be analyzed separately. Decomposition is compared to reduction because its purpose is to analyze and decompose the system into simple subsystems. The converse process of decomposition, composition, is compared to refinement (synthesis): starting from a collection of systems specifying abstract properties we refine and compose in order to obtain a detailed model. The method of Suzuki and Murata [47] and the method of Valette [48] are Berthelot's first and simple example, in which we continue to refine nodes, i.e., places or transitions, in a net, until a detailed net that completely describes the system is obtained. These methods are considered to be simple because in many applications there are not always occasions in which the specification of a total system can be reached only be refining nodes. He stated that the composition of subsystems may always be defined in terms of fusion of (common) places, which correspond to synchronization by communication channels or common variables, and in terms of fusion of (common) transitions, which represents synchronization by the "rendezvous" technique. Because of the complexity of handling interactions, it is not necessary to combine places and transitions in a single step. Actually, in a system, mixing these two techniques can always be avoided. Hence, Berthelot divided the decomposition techniques proposed by a number of researchers into S-decompositions and T-decompositions. An S-decomposition is a partition of the set of places in a net, while transitions are split, i.e., shared with others. A T-decomposition is a partition of transition of the set of transitions in a net, while places are split. Refer to the paper [10] for more detail.

B. Hierarchical Reduction

Lee and Farvel [27] noticed two problems with existing reduction methods:

1) There are no hierarchical reduction and decomposition methods, which facilitates the study of the relationship between the reducible subsystems and the original modeled system;

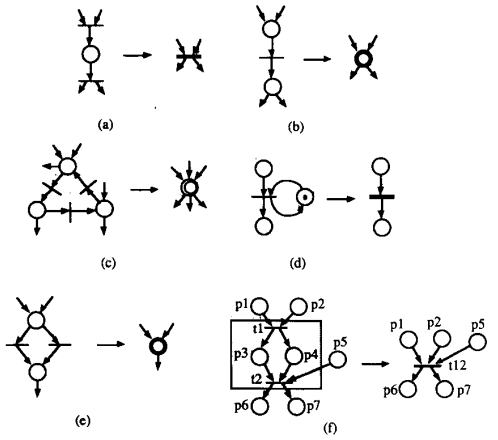


Fig. 14. Hierarchical reduction rules [27]. (a) RSN–1T. (b) RSN-1P. (c) RSN-2P. (d) RSN-2T. (e) RSN-3P. (f) RSN-4T.

2) The conditions for a reducible subnet are defined on the dynamic behavior of the net, so that the test of a reducible subnet is often complex.

Hence, Lee and Favrel have developed a hierarchical reduction method for Petri nets. The method is based on a set of reduction rules, and reduces the reducible subnets (RSN) in a Petri net into the macronodes (i.e., macroplaces and macrotransitions) without changing the important properties of liveness, boundedness, and proper termination (see Fig. 14). The set of RSN is partially ordered by the degree of the subnets [27]. In each iteration, the lowest RSN is reduced. The advantage of their method is that the application conditions of these rules depend solely on the static structure of the net and not directly on its dynamic behavior. This makes the rules easy to use because we can observe the net and do the reduction without deeper investigation of the net behavior. Moreover, through a sequence of reduction, the method produces a hierarchy of reduced Petri nets. This allows us to decompose a large system into multilevel subsystems and to study step-by-step the structure and properties of the system. In a later paper [29], they have extended their work to include the reduction rules for generalized Petri nets, i.e., Petri nets with multiple arcs. The reader is referred to the following papers for other research on reduction of Petri nets: [5], [16], [20], [34], [38] and [46].

C. Relationship Between Synthesis and Reduction

Theoretically, the converse of a reduction rule can also be used as a synthesis rule. However, for synthesis, we also consider rule interpretation with respect to the modeling domains. This interpretation is very important because the goal of synthesis is to specify and design a system. Therefore, a synthesis method should provide all the information for helping the modeler to construct the net model of the target system. In other words, the rule interpretation provides guidelines for choosing proper rules so that we can model portions of the system behavior at each specific synthesis stage. Gradually, we should be able to synthesize the whole system model. For example, if we use Berthelot's reduction rules for synthesis, each rule interpretation should tell us when to use it so that, for example, we can add a redundant place or decompose a double place properly. When we reduce a net, we examine the net structure and apply whatever reduction rule that can reduce the net as much as possible. Usually, the interpretation of a reduction rule is not that essential as long as the net structure can be simplified.

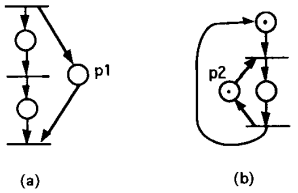


Fig. 15. Two possible interpretations of redundant places. (a) A parallel process that does not prevent a synchronizing event from occurring. (b) An available resource.

Most reduction techniques that we have seen in this paper can be used for synthesizing nets by adopting their converse process, expansion. As shown in Fig. 15, Berthelot's redundant places have the following possible interpretations (there may be others): (a) adding a parallel process that does not prevent a synchronizing event from occurring, and (b) adding an available resource. Take another example. The converse of Lee and Favrel's RSN-3P can be interpreted as generation of two parallel processes. Similar situations may occur using Valette, and Suzuki, and Murata's top–down synthesis methods. For application purposes, specialized and meaningful blocks, such as the modules defined by Bruno and Altman, may be more appropriate than k-well-behaved Petri nets.

VI. Conclusion

This paper has presented a review of synthesis techniques for Petri nets as well as reduction methods. The synthesis methods are divided into two categories: bottom–up and top–down. Top–down techniques have the advantage of viewing the system globally, which may generate more structured designs. For systems without highly shared resources, methods exist to preserve important properties of the net after the synthesis steps by means of restricting the synthesis rules such that concurrent interactions among the subnets are properly controlled. However, it is not easy to find rules and methodologies that are appropriate for some applications. In other words, developing techniques that satisfy the requirement of both guaranteeing net's properties and the flexibility of specifying systems is very difficult. This is especially true if the system has highly shared resources. On the other hand, bottom–up techniques have the advantage of convenience for specifying systems on describing the concurrent interactions such as resource sharing, but in most cases, these methods do not guarantee that the synthesized net preserves important properties.

Reduction methods have provided a solution for simplifying the computational effort of analyzing large and complex Petri nets, and have been proven to be very useful in some applications [27]. We claimed that the converse of a reduction rule is more useful as a synthesis rule if the context or interpretation relating to the modeling domains are stated, where the conditions for applying the rule specify some behavior of the system, e.g., fork, join, mutual exclusion, resource sharing, etc.

There is still much work to be done in the synthesis methods for Petri nets, so that these methods can be applied to a broader domain. One possible future direction is the study of hybrid synthesis, which combines the advantages of bottom–up and top–down methods [54]. The shared resource problem may be a focus in this research. Another possible direction is the synthesis of high level Petri nets [17], [18], [19], [45] such as colored Petri nets, which are more concise models than ordinary Petri nets.

ACKNOWLEDGMENT

The authors wish to thank Prof. A. Desrochers, M. C. Zhou, and I. Koh for their comments on this paper.

REFERENCES

[1] T. Agerwala, "A complete model for representing the coordination of asynchronous processes," Hopkins Comput. Sci. Program, Johns Hopkins Univ., Baltimore, MD, Computer Res. Rep. 32, July 1974.

[2] ——, "Putting Petri nets to work," *IEEE Computer*, pp. 85–94, Dec. 1979.

[3] T. Agerwala and Y. Choed-Amphai, "A synthesis rule for concurrent systems," in *Proc. 15th Design Automation Conf.*, Las Vegas, NV, June 1978, pp. 305–311.

[4] R. Y. Al-Jaar and A. Desrochers, "Petri nets in automation and manufacturing," in *Advances in Automation and Robotics*, vol. 2, G. N. Saridis, Ed. Greenwich, CT: JAI Press, 1991, pp. 153–225.

[5] C. Andre, F. Boeri, and J. Marin, "Synthese et realisation des systemes logiques a evolution simultanees," *PAIRO*, vol. 10, pp. 67–86, Apr. 1976.

[6] J. M. Ayache, J. P. Courtiat, and M. Diaz, "REBUS, a fault-tolerant distribution system for industrial real-time control," *IEEE Trans. Comput.*, vol. C-31, pp. 637–674, 1982.

[7] J. L. Baer and C. S. Ellis, "Model, design and evaluation of a compiler for a parallel processing environment," *IEEE Trans. Software Eng.*, vol. SE-3, pp. 394–405, 1977.

[8] C. L. Beck, "Modeling and simulation of flexible control structures for automated manufacturing systems," M.S. thesis and Robotics Inst. Tech. Rep., Carnegie-Mellon Univ. Pittsburgh, PA, 1985.

[9] G. Berthelot, "Checking properties of nets using transformations," in *Advances in Petri Nets 1985 (Lecture Notes in Computer Science 222)*. New York: Springer-Verlag, 1985, pp. 19–40.

[10] ——, "Transformations and decompositions of nets," *Advances in Petri Nets 1986 (Lecture Notes in Computer Science 254)*. New York: Springer-Verlag, 1986, pp. 359–376.

[11] G. Berthelot and R. Terrat, "Petri nets theory for the correctness of protocols," *IEEE Trans. Commun.*, vol. COM-30, pp. 2497–2509, 1982.

[12] J. Bruno and S. M. Altman, "A theory of asynchronous control networks," *IEEE Trans. Comput.*, vol. C-20, June 1971, pp. 629–638.

[13] F. Commoner, A. W. Holt, S. Even, and A. Pnueli, "Marked directed graphs," *J. Comput. Syst. Sci.*, vol. 15, pp. 511–523, 1971.

[14] D. Crockett, A. Desrochers, F. DiCesare, and T. Ward, "Implementation of a Petri net controller for a machining workstation," in *Proc. IEEE Conf. Robotics Automation*, Raleigh, NC, Apr. 1987, pp. 1861–1867.

[15] M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models," *Comput. Net.*, vol. 6, 1982.

[16] S. T. Dong, "The modelling, analysis and synthesis of communication protocols," Ph.D. dissertation, Univ. California, Berkeley, 1983.

[17] H. J. Genrich and K. Lautenbach, "System modelling with high-level Petri nets," *Theoret. Comput. Sci.*, vol. 13, pp. 109–136, 1981.

[18] K. Jensen, "Colored Petri nets and the invariant method," *Theoret. Comput. Sci.*, vol. 14, pp. 317–336, 1981.

[19] ——, "Coloured Petri nets," in *Advances in Petri Nets 1986, (Lecture Notes in Computer Science 254)*. New York: Springer-Verlag, 1986, pp. 248–299.

[20] R. Johnsonbaugh and T. Murata, "Additional methods for reduction and expansion of marked graphs," *IEEE Trans. Circuit Syst.*, vol. CAS-28, pp. 1009–1119, Oct. 1981.

[21] E. Kasturia, F. DiCesare, and A. Desrochers, "Real time control of multilevel manufacturing systems using colored Petri nets," in *Proc. IEEE Conf. Robotics Automation*, PA, Apr. 1988.

[22] W. E. Kluge and K. L. Lautenbach, "The orderly resolution of memory access conflicts among competing channel processes," *IEEE Trans. Comput.*, vol. C-31, pp. 194–207, 1982.

[23] I. Koh and F. DiCesare, "Modular transformation methods for generalized Petri nets and their applications in manufacturing automation," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 963–973, 1991.

[24] W. Kowalk and R. Valk, "On reduction of parallel programs," in *Lecture Notes in Computer Science*, vol. 71. New York: Springer-Verlag, pp. 356–369, 1979.

[25] B. H. Krogh and C. L. Beck, "Synthesis of place/transition nets for simulation and control of manufacturing systems," in *Proc. IFIP Symp. Large Scale Syst.*, Zurich, Switzerland, Aug. 1986, pp. 661–666.

[26] Y. S. Kwong, "On reduction of asynchronous systems," *Theoret. Comput. Sci.*, vol. 5, pp. 25–50, 1977.

[27] K. H. Lee and J. Favrel, "Hierarchical reduction method for analysis and decomposition of Petri nets," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-15, pp. 272–280, Mar./Apr. 1985.

[28] ——, "Hierarchical reduction and decomposition of graphs for system analysis," in *Proc. IEEE Conf. Syst., Man, Cybern.*, Oct. 1984.

[29] K. H. Lee, J. Favrel, and P. Baptiste, "Generalized Petri net reduction method," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, pp. 297–303, Mar./Apr. 1987.

[30] R. J. Lipton, "Reduction: A method of proving properties of parallel programs," *J. ACM*, vol. 3, pp. 561–567, 1981.

[31] J. Martinez, H. Alla, and M. Silva, "Petri nets for the specifications of FMSs," in *Modelling and Design of Flexible Manufacturing Systems*. Amsterdam, The Netherlands: Elsevier, 1986, pp. 389–406.

[32] J. Martinez, P. Muro, and M. Silva, "Modeling, validation and software implementation of production systems using high level Petri nets," in *Proc. Int. Conf. IEEE Robotics and Automation*, Raleigh, NC, Apr. 1987, pp. 1180–1185.

[33] J. Martinez, P. R. Muro, M. Silva, S. F. Smith, and J. L. Villarroel, "Merging artificial intelligence techniques and Petri nets for real time scheduling and control of production systems," Univ. Zaragoza, Spain, Tech. Rep. GISI-1/88, Jan. 1988.

[34] E. W. Mayr and A. R. Meyer, "The complexity of the finite containment problem for Petri nets," *J. ACM*, vol. 3, pp. 561–567, 1981.

[35] P. M. Merlin, "Specification and validation of protocols," *IEEE Trans. Commun.*, vol. COM-27, pp. 1671–1680, 1979.

[36] R. E. Miller, "A comparison of some theoretical models of parallel computations," *IEEE Trans. Comput.*, vol. C-22, pp. 710–717, Aug. 1973.

[37] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.*, vol. C-31, pp. 913–917, Sept. 1982.

[38] T. Murata and J. Y. Koh, "Reduction and expansion of live and safe marked graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 68–70, Jan. 1980.

[39] T. Murata, "Synthesis of decision-free concurrent systems for prescribed resources and performance," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 525–530, Nov. 1980.

[40] ——, "Petri nets: Properties, analysis, and applications," *Proc. IEEE*, vol. 77, pp. 541–579, Apr. 1989.

[41] Y. Narahari and N. Viswanadham, "A Petri net approach to the modelling and analysis of flexible manufacturing systems," *Ann. Oper. Res.*, vol. 3, pp. 449–472, 1985.

[42] J. L. Peterson, "Petri nets," *Comput. Surveys*, vol. 9, no. 3, pp. 223–252, Sept. 1977.

[43] ——, *Petri Net Theory and The Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[44] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 440–449, 1980.

[45] W. Reisig, "Petri nets with individual tokens," *Theoret. Comput. Sci.*, vol. 41, pp. 185–213, 1985.

[46] M. Silva, "Sur le concept de macro place et son utilisation pour l'analyse des reseaux de Petri," *PAIRO Automat.*, vol. 15, no. 4, pp. 335–345, 1981.

[47] I. Suzuki and T. Murata, "A method for stepwise refinement and abstraction of Petri nets," *J. Comput. Syst. Sci.*, vol. 27, pp. 51–76, 1983.

[48] R. Valette, "Analysis of Petri nets by stepwise refinement," *J. Comput. Syst. Sci.*, vol. 18, pp. 35–46, 1979.

[49] R. Valette, M. Courvoisier, H. Demmou, J. M. Bigou, and C. Desclaux, "Putting Petri nets to work for controlling flexible manufacturing systems," in *Proc. Int. Symp. Circuits Syst.*, Kyoto, Japan, 1985, pp. 929–932.

[50] S. Velilla and M. Silva, "The SPY: A mechanism for safe implementation of highly concurrent systems," in *Proc. 15th IFAC/IFIP Workshop Real-Time Prog.*, Valencia, Spain, May 1988, pp. 95–101.

[51] J. L. Villarroel, J. Martinez, and M. Silva, "GRAMAN: A graphic system for manufacturing system design," in *Proc. IMACS Int. Symp. Syst. to Modeling and Simulation*, Cetraro, Italy, Sept. 1988, pp. 311–316.

[52] K. Voss, "Using predicate/transition-nets to model and analyze distributed database systems," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 539–544, 1980.

[53] M. C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling of manufacturing systems with shared resources," *IEEE Trans. Robotics Automat.*, vol. 7, pp. 515–527, 1991.

[54] ——, "A hybrid methodology for synthesis of Petri nets for manufacturing systems," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 350–361, 1992.

[55] M. C. Zhou, D. Guo, and F. DiCesare, "Integration of Petri nets and moment generating functions for system performance evaluation," in *Proc. Second IEEE Conf. Syst. Integration*, 1991.