Knitting Technique with TP-PT Generations for Petri Net Synthesis

DANIEL YUH CHAO
Department of Management and Information Science
National Chengchi University
Taipei, 116 Taiwan
E-mail: yaw@mis.nccu.edu.tw

The behavior of a Petri net (PN) depends not only on the graphical structure, but also on the initial marking of the net. The knitting technique solves the structural problem. It contains some simple yet effective rules which can guide the synthesis of PN with desired properties. The previous knitting technique admits only TT generations among sequential or concurrent processes and PP generations among sequential or exclusive processes. The synthesized nets, however, are restricted because some generations are prohibited. Recently, we have improved the synthesis rules such that the previously forbidden generations are admissible if they are accompanied with additional generations. This paper deals with how to remove the restrictions of TP-path or PT-path generations.

Keywords: automated manufacturing system, Petri net, concurrent system, cycle, liveness, boundedness, deadlock, reversible, synthesis, structural relationship, rules

1. INTRODUCTION

The behavior of a Petri net (PN) depends not only on the graphical structure, but also on the initial marking of the net. The larger the initial marking (i.e., the more tokens are involved), the larger the reachability graph. It has been shown that the complexity of the reachability analysis of the PN is exponential [12].

Although such analytic methods as reachability graph, reduction [1, 11, 16], and linear algebra based methods [15] 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 analytic methods have detected some undesired properties. Therefore, it is desired to have a synthesis approach which provides a set of guidelines for the designer to build a PN. Various synthesis approaches have been proposed: top-down/bottom-up [8, 10, 15], peer entity generation [14], and knitting [3-7].

The knitting technique is a rule-based interactive approach. It expands the PN in a structural fashion based on a set of rules. While it takes exponential time complexity to determine marking properties; it may take polynomial time to determine structural properties such as structural liveness (SL) and boundedness (SB). It aims to find the fundamental constructions for building any PN. There are two advantages of the knitting technique: (1) reduction of the complexity of synthesis as an interactive tool and (2) providing knowledge about the constructions underlying various classes of nets. It therefore opens a novel avenue to PN analysis.
It, however, cannot synthesize nets with resource. Most synthesis approaches do not deal with resource sharing. It is thus motivated, in this work, to extend the knitting technique to handle resource sharing.

The above limitation springs from the fact that some generations are forbidden. Allowing them would produce new classes of nets. This paper deals with how to remove restrictions of TP-path (from transition to place path) or PT-path generations. It shows that the TPPT rules cover that of synchronization as subsets. In addition, it generates unsafe nets, that is, in the initial marking, any place has at most one token, but after some firing sequence, a place can hold more than one token. Thus, the knitting technique differs from others in that it can be continuously enhanced and the reduction and analysis process becomes more powerful.

Section 2 presents preliminaries that include the terminology, the guidelines, the synthesis rules, and some examples of how to apply these rules. Section 3 derives the structural rules for TPPT generation considering various cases followed by an example in section 4. Section 5 concludes the paper. We follow [2] for the various terminologies of PN and consider only strongly connected nets.

2. PRELIMINARIES

Definition 1  The **synchronic distance** between two transitions $t_1$ and $t_2$ in a Petri net $N$ is defined as $d_{1,2} = \text{Max}\{|\sigma(t_1) - \sigma(t_2)|\}$, where $\sigma$ is a firing sequence and $\sigma(t)$ is the number of times $t$ appears in $\sigma$. $d(W, V)$ is the maximum number of firings of transitions in set $W$ without a transition firing in set $V$.

In Fig. 1(c), $d_{3,4} = 1$, because $t_4$ can fire at most once before $t_3$ does.

![Fig. 1](image-url)

(a) A well behaved Petri net; (b) The PT-generation $[p_3 t_3 p_4 t_5]$; (c) After synchronization by adding a regulation circuit (RC) $[t_4 p_7 t_3 p_6]$. A well behaved Petri net.
Definition 2 A nonnegative, integer $Y(X)$ vector is called an $S$- ($T$-) invariant iff $Y(X) \neq 0$ and $AY = 0$ ($AX = 0$) where $A$ is the incidence matrix. The set of places $p$ such that the component in $Y$, $y(p) > 0$ for all is called the support of the $S$-invariant and is denoted by $|\{y\}|$.

$AY = 0$ implies that the $y$ values be balanced at each transition.

Lemma 1 [4] $\forall \text{PN}, \quad AY = 0 \iff \forall t \in T, \quad Y(\bullet t) = Y(\bullet)$, \quad $Y(Q) = \sum_{p \in Q} y_p \cdot w_i$, where $Q = \bullet t$ or $t \bullet$ and $w_i$ the weight of the (directed) arc between $p_i$ and $t$. The above condition is defined as the $T$-condition.

This $T$-condition is useful for finding $Y$ in a step-by-step fashion. We can assign equal $y$ values for input places of a transition and compute the equal $y$ value of each output place. Continue this until all $y$ values have been computed. We then multiply or divide all $y$ values by a factor such that all $y$ are integers and their greatest common factor is one.

We first present the terminology used throughout this paper.

2.1 Terminology

We now define home place, basic process, pseudo process and the knitting technique related terms for Petri net synthesis. For more details and examples, please refer to [4, 7].

Definition 3 A home place $p_h$ is defined as an input place of a $t_\alpha$ where $t_\alpha$ is enabled in the initial marking of a PN.

Definition 4 A basic process is a special kind of Petri net in which (1) $\forall t \in T, |\bullet t| = |t\bullet| = 1$, and $\forall p \in P, |p\bullet| = |p\bullet| = 1$; and (2) a place $p_h$ is marked with a token.

In Fig. 1 (a), $p_1$ is a home place and $[p_1, t_1, p_3, t_4, p_5, t_5, p_1]$ is a basic process, the first step of synthesis. We begin the synthesis with a well-behaved Petri net (basic process) and preserve the well-behavedness after each synthetic operation without any analysis; hence it can be used to synthesize large and complicated Petri nets. The price we need to pay is the use of structural matrix with polynomial complexity. In the knitting technique, the basic structure of Petri nets is the pseudo process (PSP), which is defined in the following.

Definition 5 A pseudo process (PSP), $\Pi$, in a PN is a directed elementary path in which any node (transition or place) has only one input node and only one output node except its two terminal nodes: the starting node is defined as the generation point $g_\Pi$ and the ending node as the joint $f_\Pi$.

In Fig. 1 (a), path $\Pi_1 = [t_1, p_2, t_2, p_3]$ is a pseudo process, with $g_{\Pi_1} = t_1$ and $f_{\Pi_1} = p_3$. A special kind of pseudo process is defined below.

Definition 6 A virtual PSP (VP) is a two-node PSP.
In Fig. 1 (a), paths \([p_1, t_5]\) is a virtual PSP.

**Definition 7** A transition \(t\) (place \(p\)) is called the *prime * \(t(p_s)\) of PSP \(\Pi_1, \ldots, \Pi_i\), if \(t(p)\) is the starting transition (place) of directed paths, \(\Gamma_i\) (containing \(\Pi_i\)), \(\ldots, \Gamma\) (containing \(\Pi_1\)), which do not share common nodes other than \(t(p)\).

The synthetic rules of knitting technique depend on the relationship among the related PSP. We define the *structural relationship* between two PSP in the following.

**Definition 8** The *structural relationship* between two PSP, \(\Pi_1\) and \(\Pi_2\), is defined as

1) **Exclusive**: \(\Pi_1\) and \(\Pi_2\) are exclusive (abbreviated as ‘EX’) to each other, denoted by \(\Pi_1 \| \Pi_2\), if \(\exists\) prime \(p_s\) of \(\Pi_1\) and \(\Pi_2\).

2) **Concurrent**: \(\Pi_1\) and \(\Pi_2\) are concurrent (abbreviated as ‘CN’) to each other, denoted by \(\Pi_1 \parallel \Pi_2\), if either (a) \(\exists\) prime \(t_i\) of \(\Pi_1\) and \(\Pi_2\) or (b) \(\Pi_1\) and \(\Pi_2\) are respectively in two separate subnets.

3) **Sequential**: If \(\Pi_1\) and \(\Pi_2\) are neither exclusive nor concurrent, and either (a) \(j_{\Pi_1} = g_{\Pi_2}\) or (b) \(\forall p \in \Gamma, t\), the path from \(\Pi_1\) to \(\Pi_2\), \(p \neq p_0\), then \(\Pi_1\) is called sequentially earlier (abbreviated as ‘SL’) than \(\Pi_2\), denoted by \(\Pi_1 \rightarrow \Pi_2\), and \(\Pi_2\) is sequentially later (abbreviated as ‘SL’) to \(\Pi_1\), denoted by \(\Pi_2 \leftarrow \Pi_1\).

**Fig. 2.** Rule TT.3 must also be considered for both TP (e.g. \([t_2, p_4]\), and PT (e.g. \([p_5, t_7]\), generations); (a) We create one more TP-generation \([t_6, p_4]\) since \(t_2 \| t_6\) and Rule TT.3 must consider all transitions in LEX\((t_5, p_4)\)={\(t_5, t_6\)\. (b) \(\forall p \in LCN (TP_{\Pi_1}, PT_{\Pi_2}) (\neq \{p_5, p_4\}\), we create a VPT from \(p_5\) to \(PT_{\Pi_2}\) per Rule PP.2. Otherwise, \(p_5\) can get unbounded; (c) we have to consider all \(t(p)\). A counter example is shown in the right figure where no \(p\) for \(t_8\) in \(p_f = p_8\).

**Definition 9** PSP \(\Pi_i\) is in a PP-circle if \(\exists \Pi, \Pi \circ \Pi_i\).
It should be noted that $\Pi_1$ and $\Pi_2$ are also called cyclic to each other, denoted by $\Pi_1 \circ \Pi_2$, if either $\Pi_1 \rightarrow \Pi_2$ or $\Pi_2 \rightarrow \Pi_1$ since, in this work, the Petri nets are used to model the system which is assumed to operate cyclically.

$[t_1, p_1, t_5]$ and $[t_1, p_3]$ in Fig. 2 (a) share the same prime $t_1$, and hence are concurrent to each other. In Fig. 3 (a), the prime $p_1$ of $\Pi_1 = [p_4, t_3]$ and $\Pi_2 = [p_4, t_3]$ is $p_1$, and $\Pi_1 \circ \Pi_2$. In Fig. 4 (a), $\Pi_1 = [t_4, p_3, t_1, p_1, t_5]$ and $\Pi_2 = [t_1, p_2]$ are on an elementary circle containing $\Pi_1$ and $\Pi_2$; hence $\Pi_1 \rightarrow \Pi_2$.

![Fig. 3. (a) TP$_{ij}$ 'SQ' TP$_{1g}$; (b) TP$_{ij}$ 'CN' TP$_{1g}$.](image)

![Fig. 4. (a) PT$_{2j}$ 'SL' TP$_{1g}$ and PT$_{2j}$ 'SE' PT$_{2g}$, a backward generation by Def. 14; (b) The PT-path interferes with the token flow on the TP-path from TP$_{1g} = t_1$ to TP$_{ij} = p_3$, injected by the firing of $t_1$; (c) PT$_{2j}$ 'SE' PT$_{2g}$, a backward generation by Def. 23. $\forall p_i (= p_6), \neg (p_i \parallel PT_{2j})$ since $p_6 \leftarrow t_1$. It is well-behaved.](image)
Fig. 4. (d) $PT_2 \rightarrow 'SE' PT_{2g}$, a backward generation by Def. 14. $p_v = p_6$ ($p_v \parallel PT_2 = t_2$). The initial token at $p_4$ can leak away by firing $t_4$, $t_2$ is dead and cannot fire to supply a token to $p_4$; (e) Another backward TPPT generation where $PT_2 \rightarrow TP_{1g}$. It is live since $¬(p_v \parallel PT_2)$, $p_v = p_5$, $PT_2 = t_4$ and the leaked token in $p_3$ can be replenished by firing $t_5$; (f) Another backward TPPT $PT_2 \rightarrow TP_{2g}$ and $PT_1 \parallel TP_2$. Unlike Fig. 4 (d), it is live despite $(p_v \parallel PT_2)$, $p_v = p_6$, $PT_2 = t_2$ and the leaked token in $p_4$ can be replenished by firing $t_5$.

Fig. 5. (a) Forward TPPT generation $TP_{1g} \rightarrow 'EX' TP_{g}, PT_2 \rightarrow 'SL' TP_{1g}$; (b) Forward TPPT generation $TP_{1g} \rightarrow 'EX' TP_{g}, PT_2 \rightarrow 'CN' TP_{1g}$. It is not live when $t_2$ never fires to put a token in $PT_1 = p_8$.

In Fig. 5 (a), transitions $t_2$ and $t_7$ are mutually exclusive and places $p_2$ and $p_3$ are mutually concurrent.

**Definition 10**  If two nodes are in the same PSP, they are sequential to each other. If they are in two different PSP, their structural relationship follows that of the two PSP.
A local exclusive set (LEX) of $\Pi_i$ with respect to $\Pi_{i_k}$, $X_{ik}$, is the maximal set (in terms of set cardinality) of all PSP which are mutually exclusive and are equal to or exclusive to $\Pi_i$, but not to $\Pi_{i_k}$, i.e., $X_{ik} = LEX(\Pi_i, \Pi_{i_k}) = \{\Pi_j | \Pi_j \subset \Pi_i \& \Pi_j \cap \Pi_{i_k} = \varnothing\}, \forall \Pi_{i_1}, \Pi_{i_2} \in X_{ik}, \Pi_{i_3} \mid \Pi_{i_2}$. $X_{ik}$ and $X_{i_k}$ are said to be mutually LEX to each other.

A local concurrent set (LCN) of $\Pi_i$ with respect to $\Pi_{i_k}$, $C_{ik}$, is the maximal set of all PSP which are mutually concurrent and are equal to or concurrent to $\Pi_i$, but not to $\Pi_{i_k}$, i.e., $C_{ik} = LCN(\Pi_i, \Pi_{i_k}) = \{\Pi_j | \Pi_j \subset \Pi_i \& \Pi_j \cap \Pi_{i_k} = \varnothing\}, \forall \Pi_{i_1}, \Pi_{i_2} \in C_{ik}, \Pi_{i_3} \mid \Pi_{i_2}$. $C_{ik}$ and $C_{i_k}$ are said to be mutually LCN to each other.

In Fig. 2 (a), $X_{13} = \{\Pi_1 = [p_2,t_2,p_3], \Pi_2 = [p_2,t_0,p_3]\}$ and $X_{13} = \{\Pi_3 = [p_3,t_5,p_4,t_4,p_3]\}$. In Fig. 2 (b), $C_{16} = \{\Pi_4 = [t_4,p_5,t_5], \Pi_5 = [t_4,p_5,t_5]\}$ and $C_{16} = \{\Pi_6 = [t_5,p_6,t_6,p_1,t_1]\}$. Note that $C_{ik}$ and $X_{ik}$ may not be unique. All PSP of $C_{ik}$ ($X_{ik}$) must be involved upon generating a PP(TT)-path between exclusive (concurrent) PSP; otherwise deadlock or unboundedness may occur. The following two tuples are dual to each other: $(LEX, TT, |)$ and $(LCN, PP, |)$.

Similarly, $LEX(t_{i_a}, t_{i_b}) (LCN(p_{a}, p_{b}))$ is a set of transitions or places, instead of PSP. Let $G (J)$ denote the set of all $\Pi_i$ ($\Pi_{i_k}$) involved in a single application of the TT or PP rule. To avoid unbounded, it is necessary to have a new directed path from each PSP in $X_{ik}$ to each PSP in $X_{i_k}$.

Suppose that a Petri net models a set of processes or entities interacting with one another by sending and receiving messages, an entity is an active element (with a token in its home place) such that it can execute some tasks independent of other entities. Initially, each entity has a home place (or places) with a token * indicating that it is ready to start. Each entity conducts a set of tasks in serial (which can be considered as transitions), then returns to the original ready state. Each state can be considered as a place. Executing a task is like firing a transition. At the completion of executing each task, the token leaves the original state(s) and enters the next state(s) by firing transitions thus, the token moves from the home place through consecutive places by firing a sequence of transitions. Eventually, the token will return to the home place, forming a cycle.

**Definition 17:** Pure Generation (PG) generates paths within a single PSP. This basic process can be expanded through a series of synthetic operations. Some of such path definitions defined in the following.

**Definition 13:** Pure Generation (PG) generates paths within a single PSP. Interactive Generation (IG) generates paths between two PSP. The TT (PP) rule generates paths between transitions (places), from the $t_j$ ($p_j$), termed the generation point, to $t_j$ ($p_j$), termed the joint.

Pure-generation processes do not involve interactions; they are pure growth processes. If both the generation point of a $\Pi_i$ are places (transitions), it is a PP (TT) generation; the corresponding path is called a PP-path (TT-path). See Fig. 2(a) for a TT (PP) IG: $[t_j,p_j,t_k] (p_j,t_j,p_j)$. A single process can be expanded by adding a set of PSP using pure generation which does not generate paths of interactions between PSP.

For each of the TT and PP rules, there are two types of generations: forward and backward, defined in the below.

---

1 The synthesis rules still hold if a home place has multiple tokens. We adopt a token here to simplify the discussion.
Definition 14  If \( n_g \rightarrow n_j \) prior to the generation, then it is a forward generation; otherwise it is a backward generation. \( \Pi_g (\Pi_j) \) denotes the PSP which contains the generation point (joint).

In Fig. 3 (a), \([t_1p_2]\) is a forward generation, and \([p_4t_2]\) in Fig. 4 (a) is a backward generation.

2.2 Guidelines for Synthesis

Given a synthesized \( PN, N^1 \); a set of new paths (\( NP \)) are generated using some rules to form a new net, \( N^2 \). The synthesis rules should be such that all transitions and places in the subnet \( N^1 \) and \( NP \) are live and bounded, respectively. This can be ensured by inhibiting any intrusion into normal transition firings of \( N^1 \), which occurs prior to the generation, and any dead transition and unbounded place in the \( NP \).

Based on the concept of no intrusion, the rules are constructed according to

Guidelines for synthesis:
1. From \( M_0 \), each \( t_g \) must be potentially fireable.
2. The tokens generated by firing \( t_g \) must disappear from \( NP \) before it gets infinite number of tokens and return to \( N^1 \) to reach a marking \( M_2 \). The marking of the subnet \( N^1 \) in \( M_2(M_2) \) of all possible sets of such \( M_2 \) are identical to those in \( N^1 \) without the new paths. \( \Pi^* \) cannot get infinite tokens without consuming them (i.e., \( t_g \) and \( t_j \) are synchronized).

Guidelines (1) and (2) guarantee no intrusion; Guideline (2) guarantees no dead transition and no unbounded place in \( NP \) since all tokens can disappear from \( NP \).

The synthesis rules presented below are complete in the sense that depending on the structural relationship between \( t_g \) (\( P_g \)) and \( t_j \) (\( P_j \)), different rules are proposed. Some are forbidden (e.g., when \( t_g | t_j \)) and some require the generation of additional PSP. All \( t_g, t_j, P_g \) and \( P_j \) referred in the rules belong to \( \Pi^* \), rather than those of \( \Pi_j \) and \( \Pi_j \).

2.3 TT and PP Synthesis Rules

We present the synthesis rules below. The correctness (i.e., any synthesized net is bounded, live and reversible) has been established in [8].

The TT and PP rules are formalized below.

TT Rules:

For an \( NP \) from \( t_g \in \Pi_g \) to \( t_j \in \Pi_j \) generated by the designer,

1) TT.1  \textbf{If } \( t_g \mid t_j \text{ or only one of them is in a circle, then } \text{display } \text{“Disallowed, generate another } \Pi \text{ and return.”} \)

2) TT.2  \textbf{If } \( t_g \leftarrow t_j \text{ or } t_g = t_j \text{, then } \text{insert a token in a place of } NP. \) If \( \Pi_g = \Pi_j \), \text{then } \text{display } \text{“You may generate another } \Pi \text{ and return.”}

3) TT.3  \textbf{a) } TT.3.1 \text{ Generate a TP-path from a transition } t_g \text{ of each } \Pi_g \text{ in } X_g \text{ to a place } P_k \text{ in the } NP. \)
\textbf{b) } TT.3.2 \text{ Generate a virtual PT-path from the place } P_j \text{ to a transition } t_j \text{ of each } \Pi_j \text{ in } X_j. \)
4) **TT.4** If there was no path from \( t_g \) to \( t_j \) prior to this generation or both are in a circle, then generate a new TT-path from \( t_g' \) to \( t_j' \) such that \( t_g, t_j, t_g' \) and \( t_j' \) are in a circle in this order.

**PP Rules:**

For an NP from \( p_g \in \Pi_g \) to \( p_j \in \Pi_j \) generated by the designer,

1) **PP.1** If \( p_g \parallel p_j \), then display “Disallowed, generate another \( \Pi \)” and return.

2) **PP.2**

   a) **PP.2.1** Generate a TP-path from a transition \( t_k \) of NP to a place \( p_j \) of each \( \Pi_j \) in \( C_{gj} \).

   b) **PP.2.2** Generate a virtual PT-path from a place \( p_g \) of each \( \Pi_g \) in \( C_{jg} \) to the transition \( t_k \) of NP.

\( p_g(p_j) \) of a TT-path is the output (input) place of its generation point (joint). In Fig. 1 (a), \( p_6 \) is both the \( p_g \) and \( p_j \) of TT-path \( t_1p_6t_4 \). \( t_g(t_j) \) of a PP-path is the output (input) transition of its generation point (joint).

**Structural Derivation of TPPT-Rules:**

The synthesis rules which we have discussed above are limited to the path generations between a pair of transitions or places. Although there are PT- and TP-path generations, they are associated with either TT- or PP-path generations. We need to remove such restrictions by developing the independent rules of TP-path and PT-path generations. The structural theory, considering how the TP- and PT-path should be placed relative to each other, is studied in this section.

We first have

**Observation 1:** Any independent TP-path generation must be accompanied by a PT-path generation and vice versa.

This is because

1) **TP Generation:** Each TP-path generation may cause a place unbounded; hence there must be a PT-path to consume the tokens injected through the TP-path.

2) **PT Generation:** Each PT-path generation adds one more input place to \( j_{PT} \) (joint of the PT-path) which may be disabled, causing deadlock. Hence there must be a TP-path to inject token(s) into \( g_{PT} \) (generation node) for \( j_{PT} \) to fire.

The joint transition \( t \) of the PT-generation \( (j_{PT}) \) may become not firable if its generation place \( p \) misses a token. This disables subsequent transitions \( t_1 \) (\( t_1 \) ‘SL’ \( t \)) and hence permanently vacates \( p \). The net thus turns not live. The guidelines of rules for TPPT-generations are similar to those in section 2.B. Before we study the structural theory, we need to define the following notations:

**Definition 15:** \( TP_i, PT_i, \) or \( PSP_i \) \( (i = 1, 2) \): The ‘\( i \)’th TP, PT or PSP. \( “i = 1” \) stands for the independent generation and “\( i = 2 \)” the accompanying generation.

\( TP_g \) and \( TP_j \): The “Generation Point” and the “Joint” of \( TP_i \), respectively.

\( PT_g \) and \( PT_j \): The “Generation Point” and the “Joint” of \( PT_i \), respectively.

\( \Pi_g \) and \( \Pi_j \): The generation PSP and joint PSP of \( TP_i \) or \( PT_i \), respectively.
Because TP- and PT-generation always occur in pair, we always start first a TP-generation (i.e., $i = 1$) followed by a PT-generation (i.e., $i = 2$) in the sequel.

**Guidelines for TPPT Generation:**

1. From $M_0$, $TP_{1g}$ must be potentially firable.
2. The tokens generated by firing $TP_{1g}$ must disappear from $NP$ before it gets unbounded tokens and return to $N^4$ to reach a marking $M_2$. The marking of the subnet $N^3$ in $N^2$, $M_3(N^3)$ ($N^3 = N^0 \Gamma$, $\Gamma$ is the set of all paths from $TP_{1j}$ to $PT_{2g}$) of all possible sets of such $M_2$ are identical to those in $N^4$ without the new paths. $\Pi^w$ cannot get infinite tokens without consuming them (i.e., $TP_{1g}$ and $PT_{2j}$ are synchronized).

It is the same as those for TT- and PP-generations except with the replacement of $N^1$ by $N^3$. Note that if $PT = [t_2, p_1]$ and there are two arcs from $p_1$ to $t_2$ and the PN becomes a General Petri net, whose synthesis has been discussed in [7]. Therefore, we restrict that, prior to the generation, $PT_{1j}$ ($TP_{1g}$) cannot be an output transition (place) of $PT_{2g}$ ($TP_{1g}$).

Since all arcs in the TPPT-generation are of unit weights, for each firing of $TP_{1g}$, $PT_{2g}$ is enabled at most once. This may disturb the behavior of $N^4$ if the synchronic distance $d(PT_{1g}, PT_{2j}) > 1$ in $N^3$ (subject to further research). Thus, we perform the TPPT-generation only between transitions of synchronic distance 1, which is automatically satisfied with nets synthesized using only TT- and PP-rules.

Fig. 2 (a) shows that Rule TT.3 must also be considered for both TP (e.g. $[t_2, p_4]$), and PT (e.g. $[p_5, t_6]$) generations. We create one more TP-generation $[t_6, p_4]$ since $t_2 | t_6$ and Rule TT.3 must consider all transitions in $LEX(t_2, p_4) = \{t_2, t_6\}$. Since $TP_{1j}$ is a place $p_4$, should we apply Rule PP.2 and consider $p_8$ in Fig. 2 (b) too as a $TP_{1j}$? Yes, $\forall p \in LCN(TP_{1g}, PT_{2g}) = \{p_5, p_8\}$, we create a VPT (virtual PT-path) from $p$ to $PT_{2j}$. Otherwise, $p_8$ can get unbounded. This is in general true that all applicable rules must be considered for any generation. Thus we apply only Rule TT.3.1 ignoring TT.3.2 for TP-generation. The same discussion holds for PT-generation and we apply only Rule PP.2.2 ignoring PP.2.1. To simplify the discussion, in the remainder of this paper, we assume that only the TPPT rules need to be considered. The most important issue of these rules is the addition of the PT and TP path to keep the Petri net logically correct. This can be subdivided into the following sub problems:

1. The relative location of $TP_{1j}$ with respect to $TP_{1g}$.
2. The choice of the $PT_{2g}$ and $PT_{2j}$.

In the sequel, we discuss how to do the placement of $TP_{1j}$, $PT_{2g}$ and $PT_{2j}$ and in the mean time, still maintain rule simplification.

**Placement of $TP_{1j}$**

There is no restriction about the placement of $TP_{1j}$ relative to $TP_{1g}$. There are two cases: (A) ‘SQ’ (Fig. 3 (a)) or ‘CN’ (Fig. 3 (b)), and (B) ‘EX’ (Fig. 5 (a)) as discussed below.

(A) $TP_{1j}$ ‘SQ’ or ‘CN’ $TP_{1g}$

For this case, we develop rules considering the placement of $PT_{2g}$ and $PT_{2j}$, respectively.
(a) **Placement of PT$_{2g}$**

PT$_{2g}$ must be a $\overline{p}_v$ as defined below.

**Definition 16**

$p_i \leftrightarrow TP_{ij}$ or $p_i = TP_{ij}$ $\forall t \in (p_i)\bullet$. $\exists p_v \in \bullet t, p_v \neq p_i, TP_{ij} \parallel p,v$, both $p_i$ and $p_i$ are either in a backward TT-generation or not, $p_i$ is not a $\overline{p}_v$ of an earlier TPPT-generation. $\overline{p}_v$ is a nearest $p_v$ such that for all other $p_i$, $\overline{p}_v \rightarrow p_v$.

In Fig. 2 (a), $p_3$ ($p_5$) is a $\overline{p}_v$ ($p_v$). Note that in the absence of $p_v$, the extra token could leak away by firing a $t \in (p_3)\bullet$ ($t = t_5$) without firing PT$_{2g}$ causing $t_3$ not fireable. Otherwise with $p_i \parallel p_v$, the lost token may be replenished since $TP_{ij} \parallel p_v$. However, the replenished token may also leak through by firing the same $t$ if $p_i$ is able to get another token (e.g., from an earlier TPPT-generation or $m_6(p_i = p_v) > 0$ due to an earlier backward TT-generation). Again, the lost token may be replenished if $\overline{p}_v$ was involved in an earlier backward TT-generation.

Note that we have to consider all $t \in (p_3)\bullet$. A counter example is shown in Fig. 2 (c) where no $p_i$ for $t_6 \in (p_2)\bullet$. The extra token could leak away by firing $t_6$. Fortunately, by Rule TT.3.2, a VP = $[p_7; t_6]$ should be generated, solving the problem. Thus, the condition $\forall t \in (p_3)\bullet$ in Def. 16 is somewhat redundant.

Note that in the synthesis, $p_i$ and $p_j$ are either in a backward TT-generation or not; hence this condition is also redundant. Fig. 3 (a) shows a TP$_1$, $\{t_1; p_2\}$. For PT$_{2g}$, $p_3$ or $p_5$ cannot be the PT$_{2g}$ because they will not consume the token injected into $p_2$. Thus PT$_{2g}$ must be ‘SQ’ to TP$_{1j}$ (e.g., $p_4$ in Fig. 3(a)). Fig. 3 (b) illustrates another case where PT$_{2g}$ ‘SQ’ to TP$_{ij}$.

If $p_2$ in Fig. 3 (a), whose output transition has only one input place, is used as PT$_{2g}$, $[p_2; t_3]$ or $[p_2; t_4]$, then the firing sequence $\sigma = t_1; t_2; t_3$ will create a deadlock because $t_1$ or $t_4$, respectively, will never be enabled. We can pick only $p_4$, which is a $\overline{p}_v$, as PT$_{2g}$. Note that $p_2$ or $p_4$ can hold two tokens. Hence the synthesized net is unsafe even though any place in $M_0$ holds at most one token. Similarly, in Fig. 3 (b), PT$_{2g}$ ($= p_5$) is a $\overline{p}_v$.

For the PT-generation $[p_3; t_3]$ in Fig. 1 (c), PT$_{2g}$ ($= p_3$) is not a $\overline{p}_v$. The two output transitions of PT$_{2g}$ are synchronized to have synchronic distance $d = 1$. $m_6(p_3) = 0$ and the net is live. Note that if $m_6(p_3) > 1$ and $m_6(p_4) > 1$ so that $d > 1$, the TPPT-generation may turn it not live. As mentioned earlier, we ignore this for simple presentation.

Thus, the extra token must get blocked at PT$_{2g}$ to be consumed by firing PT$_{2g}$. In other words, PT$_{2g}$ must be a $\overline{p}_v$. Note that $\overline{p}_v$ may not exist. In this case, the TP-generation is disallowed. W now have the following:

**Observation 2:** Given TP$_{ij}$ ‘SQ’ or ‘CN’ TP$_{1j}$, then (1) PT$_{2g}$ ‘SQ’ TP$_{ij}$ or PT$_{2g}$ = TP$_{ij}$; (2) PT$_{2g}$ must be a $\overline{p}_v$ of the TP-path.

Note (1) is redundant since by definition, $\overline{p}_v$ meets (1).

(b) **Placement of PT$_3$**

In order to guarantee the absorption of the extra token, PT$_{2g}$ should not be exclusive to PT$_{2g}$ as the former may not fire to remove the extra token. Once the TP-path injects a token, as long as the PT-path does not interfere with the token flow from TP$_{1j}$ to PT$_{2g}$, the extra token can flow to the PT-path. Otherwise, the extra token may not be absorbed, causing unbounded places or nonlive PT$_{2g}$. The PT-path in Fig. 1 (a) does not interfere
with the token flow from $TP_{1g} (= t_1)$ to $TP_{2g} (= p_1)$. Hence, the $PN$ remains well-behaved.

Such is called auto-interference, while the interference with earlier TPPT-generations is called hetero-interference. Both are disallowed in this section.

In Fig. 4 (a), $PT_{2j} = t_2$ which inhibits the extra token from entering $PT_{2g} = p_4$ if $m(p_4) = 0$. Even with $m(p_4) = 1$, if the firing sequence is $t_1, t_5, t_4, t_6$, the net will become deadlocked. This is because after $t_4$ fires, $m(p_4) = 0$ and the tokens from the two TP-paths are blocked at $p_2$, causing a deadlock. In Fig. 4 (b), the PT-path interferes with the token flow on the TP-path from $TP_{1g} = t_1$ to $TP_{1j} = p_1$, injected by the firing of $t_1$. Note that the $PN$ remains live without the firing of $t_2$ because the extra token prevents the $PN$ from deadlocking. Infinite tokens can accumulate at $p_2$, causing it unbounded.

Note that the interference contradicts the purpose of generating an extra token; hence we prohibit such interference in the TPPT-rules. Also all the above interference involves the backward generation.

Similar to backward TT-generation, a token may need to add to $PT_{2g}$ to avoid infinite waiting. Again, there is the problem of this token leaking away by firing an output transition of $PT_{2g}$ prior to firing $PT_{2j}$. Note that it is possible ($PT_{2j} \rightarrow PT_{1g}$) where the leaked token can be replenished by firing $PT_{1g}$ (Fig. 4 (c)), which does not happen when we have $\forall p_g$ defined in Def. 16, $PT_{2j} \parallel p_1$ (Fig. 4 (d)). $t_2$ is dead for $\sigma = t_1, t_5, t_6$). On the other hand, for the net in Fig. 4 (f), unlike Fig. 4 (d), it is live despite ($p_6 \parallel PT_{2g}$), $p_6 = p_6$, $PT_{2j} = t_2$ and the leaked token in $p_4$ can be replenished by firing $t_5$. Note that the condition $PT_{2j} \rightarrow PT_{1g}$ automatically precludes the interference (Fig. 4 (d)). Fig. 4 (e) shows another backward generation where $PT_{2j} \rightarrow TP_{1g}$ but $PT_{2j} \leftarrow TP_{2g}$. It is live since $\neg (p_4 \parallel PT_{2g})$ and the leaked token can be replenished. Note that $p_4 \parallel PT_{2g}$ in Fig. 4 (b); yet, it is live. However, this case has been precluded because of the interference involved. Thus we have the following observation:

**Observation 3:** (1) If it is a backward TPPT generation, (a) if $TP_{1j}$ ‘SQ’ $TP_{1g}$, then $\forall p_g, \neg (p_g \parallel PT_{2j})$; (b) add a token to $PT_{2g}$, (2) Given $TP_{1j}$ ‘SQ’ or ‘CN’ $TP_{1g}$, then $\neg (PT_{2j}$ ‘EX’ $PT_{2g}$).

**B** $TP_{1j}$ ‘EX’ $TP_{1g}$

Similar to case (A), the extra token must be absorbed; thus $PT_{2g}$ must be a $\overline{p_g}$ of the TP-path. There is a difference, however, in that if $d_{1g, 2j} = \infty$ (i.e., $PT_{2j} \parallel TP_{1g}$), the extra token may get blocked at $\overline{p_g}$ since it is not necessary to consume the extra token to keep it live. Thus, if $TP_{1g}$ fires infinitely relative to $PT_{2g}$ unbounded tokens will accumulate in $\overline{p_g}$. To avoid this, $TP_{1g}$ and $PT_{2j}$ must be synchronized with each other such that $d_{1g, 2j}$ is finite. That is $PT_{2j} (1)$ ‘SQ’ (Fig. 5 (a)); (2) ‘CN’ (Fig. 5 (b)); or (3) ‘SX’ $TP_{1g}$, respectively. However, the net in Fig. 5 (b) (case (2)) is not live when $t_2$ never fires to put a token in $PT_{1j} = p_6$. As mentioned earlier, we do not treat case (3) in this paper. Hence, we have the following observation.

**Observation 4:** Given $TP_{1g}$ ‘EX’ $TP_{1j}$, (1) $PT_{2g}$ must be a $\overline{p_g}$ of the TP-path, and (2) ($PT_{2j}$ ‘SQ’ $TP_{1g}$).

Not that, in this case, we need not be concerned with the interfering by the PT-path as in case (A). This is because condition (2) in the above observation precludes ($PT_{2j}$ ‘SQ’ $TP_{2g}$) which may cause interfering.
4. AN EXAMPLE

In summary, we have the following rules for structural correctness:

Knitting Rules of TPPT-Generation:

TPPT Rules

TPPT.1  TP iff VPT (virtual PT-path) and vice versa. (Observation 1)
TPPT.2  PT_{2g} must be a \( \mathcal{P}_v \) of the TP-path. (Observation 2)
TPPT.3  If it is a backward TPPT generation, then (a) if \( TP_{1j} \) 'SQ' \( TP_{1g} \), then \( \forall p_v, \neg(p_v || PT_{2j}) \), (b) add a token to \( PT_{2g} \). (Observation 3.1)
TPPT.4  Given \( TP_{1j} \) 'SQ' or 'CN' \( TP_{1g} \), then \( \neg(PT_{2j} \text{ 'EX' } PT_{2g}) \). (Observation 3.2)
TPPT.5  Given \( TP_{1g} \) 'EX' \( TP_{1j} \), then \( (PT_{2j} \text{ 'SQ' } PT_{1g}) \). (Observation 4)

Example: The previous PN synthesis technique can not synthesize certain nets such as the one (modeling a communication protocol) by Cindio [9] which involves cyclic interaction (Fig. 6). Here, we demonstrate the application of the TPPT rules to synthesize such a PN. The PSP generated in sequence are as follows:

![Diagram of a net synthesized using TT-, PP-, and TPPT-Rules](image-url)
(1) Construct the basic process \([p_1 t_1 p_3 t_2 p_4 t_4 p_5 t_5 p_6 t_6 p_7 t_7 p_14 t_13 p_15 t_14 p_17 t_16 p_7]\).
(2) Generate PSP (also circuits) \([p_3 t_3 p_5 t_8 p_4 t_7 p_5]\) and \([p_8 t_10 p_1 t_12 t_1 p_5]\) using pure PP-generation.
(3) Generate PSP \([t_14 p_16 t_15 p_18 t_6]\), \([t_4 p_6 t_11]\), and \([t_11 p_13 t_12]\) using pure TT-generation, and PSP \([t_16 p_2 t_4]\) using Rule TT.2, add a token to \(P_2\), and PSP \([p_2 t_3]\) and \([t_7 p_2]\) using Rule TT.3.2.
(4) Generate a pair of PSP \([t_{10} p_{13}]\) and \([p_{11} t_{12}]\) by Rule TPPT.1. Note that \(p_{13} | t_{10} \), thus \(t_{10} \rightarrow SQ' t_{12}\) by Rule TPPT.4. \(p_{13}\) is a \(p_{13}\) by Rule TPPT.2.
(5) Generate a pair of PSP \([t_5 p_6]\) and \([p_6 t_7]\). Note that \(p_6 | t_5\), thus \(t_5 \rightarrow SQ' t_7\) by Rule TPPT.4. \(p_6 = TP_1 p_6 = PT_2 p_6\) is a \(p_6\).
(6) \(t_{12} \rightarrow CN' p_6\) and generate a pair of PSP \([t_{12} p_6]\) and \([p_6 t_{10}]\) using Rules TPPT.1, TPPT.2, and TPPT.3. It is a backward generation since \(t_{10} \rightarrow t_{12}\). Because \(p_6\) was involved in step (5), it is a hetero-interference case.

Due to the space limitation, we refer the reader to [5] for the correctness proof.

5. CONCLUSION

The notions of path generations (rather than refinement) and structural relationship in our knitting technique are the first in its kind. We have removed the restrictions of TP- and PT-generations and developed the rules accordingly. Thus we have completed the last phase of removing forbidden generations.

REFERENCES

Daniel Yuh Chao received the Ph.D. degree from Electrical Engineering and Computer Science from the University of California, Berkeley in 1987. From 1987-1988, he worked at Bell Laboratories. Since 1988, he joined the Computer and Information Science Department of New Jersey Institute. Since 1994, he joined the Management and Information Science Department of National Chengchi University as an Associate Professor. Since February, 1997, he has been promoted to a full professor. His research interest was in the application of Petri nets to the design and synthesis of communication protocols and the CAD implementation of a multi-function Petri net graphic tool. He is now working on the exploration of properties of a new class of Petri nets and implementation of several CAI tools based on Visual C++. He has published 72 (including 15 journals) papers in the area of communication protocols, Petri nets, DQDB, networks, FMS, data flow graphs, and neural networks.