12-31-1991

A synthesis technique of general petri nets for flexible manufacturing and multi-rate digital signal processing

Murthy S. Valluri
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

A SYNTHESIS TECHNIQUE OF GENERAL PETRI NETS FOR FLEXIBLE MANUFACTURING AND MULTI-RATE DIGITAL SIGNAL PROCESSING

By

MURTHY S. VALLURI

Thesis submitted to the Department of Computer Science,
New Jersey Institute of Technology
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

1991

ABSTRACT

Title of Thesis: A Synthesis Technique of General Petri nets for Flexible Manufacturing and Multi-rate Digital Signal Processing.

Name of Candidate: Murthy S Valluri
Master of Science in Computer & Information Sciences, 1991

Thesis Directed by: Dr. Daniel Chao
Professor, Department of Computer Science
New Jersey Institute of Technology, Newark, N J 07102

General Petri nets are useful for modeling flexible manufacturing system with multiple robots and workstations [KOH 90] and for multi-rate digital signal processing systems [CHA 91]. A problem of using Petri nets for modeling various systems is the large number of states generated. Various synthesis approaches have been proposed. Most of them do not deal with general Petri nets. Koh et al [KOH 90] invented a synthesis technique for generalized Petri nets. The pupose of this thesis is to extend their work by modifying the knitting technique by adding the Arc-Ratio rule .

APPROVAL SHEET

Title of Thesis:

A SYNTHESIS TECHNIQUE OF GENERAL PETRI NETS FOR FLEXIBLE MANUFACTURING AND MULTI-RATE DIGITAL SIGNAL PROCESSING

Name of Candidate: **Murthy S Valluri.**

Master of Science in Computer Science, 1991

Thesis and
Abstract Approved:

Dr. Daniel Chao Date
Professor,
Computer Science Department
New Jersey Institute of Technology.

VITA

NAME: **Murthy s. valluri.**

PERMANENT ADDRESS:

DEGREE AND DATE TO BE CONFERRED: M.S., Dec, 1991

DATE OF BIRTH:

PLACE OF BIRTH:

COLLEGE & INSTITUTIONS ATTENDED:

	DATE	DEGREE
M.V.S.R. College of Engineering. (Osmania University) Hyderabad, India	1984-88	B.E.(Mech)
New Jersey Institute of Technology Newark, New Jersey	1990-91	M.S.(CIS)

Major: Computer Science

Positions held:

Technician 1990-1991	Telecommunications and Networks Department, N.J.I.T. Newark, New Jersey
Research Assistant 1991-Present	CMS Department., N.J.I.T Newark, New Jersey

Acknowledgements

I take this opportunity to express my deep gratitude to Dr. Daniel Chao, Professor in Computer Science Department of N.J.I.T for his valuable guidance and help throughout the course of this work. It was his immaculate advice, which led my work to the set mark of culmination.

I am also very thankful to Dr. Wang and Dr.Zhou, for spending their precious time to review my work and to provide constructive suggestions for enhancing my work.

To My Parents

TABLE OFCONTENTS

LIST OF FIGURES

1. INTRODUCTION

Currently, Petri nets are used for modeling FMS instead of differential equations and queueing theory. General Petri nets are useful for modeling the FMS with multiple robots and workstations [KOH 90], and for multirate digital signal processing systems [CHA 91]. A problem of using Petri nets [MUR 89] for modeling various systems is the large number of states generated. Synthesis can eliminate this problem by avoiding analysis. Various synthesis approaches have been proposed: top-down/bottom up [BER 86, DAT 84, SUZ 83, VAL 79], peer entity generation [RAM 85], and knitting [YAW 87]. All of them do not deal with general Petri nets.

Koh et al [KOH 90] invented a synthesis technique. They fuse two live and bounded circuits (LB-circuits) on common paths based on the concepts of arc ratio and overlapping (See Section 3 for examples.) The resultant nets are live and bounded when the common path is a transition-transition path (TTP) or a place-place path (PPP). However, the results cannot apply to the fusion of three LB-circuits. It only applies to special cases (i.e., no overlapping) of the fusion of three nets along two different directed paths (PSPs in the knitting technique). This deficiency can be removed by an extension of the knitting technique [YAW 87].

The knitting technique by [YAW 87,88,89] is a rule-based interactive approach. A net is expanded by inserting new paths bearing some physical meaning such as increasing concurrency, alternatives, and so on; the resulting net is an ordinary Petri net and has properties of *live*, *boundedness*, and *reversibility*.

Examples are shown in Fig. 1 where three LB-circuits are fused along a PPPc and a TTPc. In Fig. 1(a), two LB-circuits $N_1(t_1P_1t_2P_2t_3P_3t_4P_4t_1)$ and $N_2(t_5P_5t_6P_1t_5)$ fuse along path 1 (P_1) which is a common PP-path (PPP_c); two LB-circuits $N_1(t_1P_1t_2P_2t_3P_3t_4P_4t_1)$ and $N_3(t_3P_3t_4P_7t_7P_6t_3)$ fuse along Path 2 ($t_3P_3t_4$) which is a common TT-path (TTP_c). Paths 1 (P_1) and 2 ($t_3P_3t_4$) are non-overlapping.

This net is live and bounded. Whereas, a TTPc and a PPPc are overlapping in Figure 1(b) and 1(c). Note that these two nets violate the PP.2 rule (Section 3) in [YAW 87], which requires the complete switching between two processes exclusive to each other. Each process may contain a number of sub-processes. This is similar to the context switching in a time-sharing system where processes share the CPU resource in a round-robin fashion. The net in Figure 1(b) is not live as explained below. The token in P_7 advances to P_6 by firing t_7 and gets consumed by firing t_2. P_2 now gets 2 tokens and can be diverted to P_5 by firing t_6. P_6 is no longer able to get 2 tokens and t_2 is never firable — not live.

In order for t_2 to fire for the process ps_1 ($t_2P_2t_3P_3t_4$) to get tokens, both processes ps_2 ($t_4P_4t_1P_1t_2$) and ps_3 ($t_4P_7t_7P_6t_2$) must hold tokens. Now the token in ps_1 can be switched to ps_2 through the path ($P_2t_6P_5t_5P_1$). Without any path between ps_1 and ps_3, the latter won't be able to get tokens from ps_1 — no complete switching between ps_1 and ps_4 which contains the two subprocesses ps_2 and ps_3 (referred to as LCN2=LCN(ps_2 , ps_1)=$\{ps_2,ps_3\}$, the local concurrent (LCN) set of ps_2 with respect to ps_1; the local concurrent set of ps_1 with respect to ps_2 LCN1=LCN(ps_1,ps_2)=$\{ps_1\}$). And t_2 will become dead.

The net in Figure 1(c) is live, but not bounded as explained below. After firing t_2, both P_2 and P_6 get a token. The token in P_2 can be diverted to P_5 by firing t_6 and subsequently advance to P_1. Thus, t_2 is always live and can supply tokens to P_6 which eventually become unbounded. Let ps_1 and ps_2 correspond to the same paths as those in Fig. 1(b). ps_3 correspond to the path $(t_2P_6t_7P_7t_4)$. when t_2 in ps_1 fires, both ps_2 and ps_3 get tokens; hence ps_2 and ps_3 are sub-processes of a process ps_4. The path $(P_2t_6P_5t_5P_1)$ switches tokens from ps_2 to ps_1. Without any path from ps_3 to ps_1, tokens in ps_3 would not be able to switch together with those from ps_2. The firing of t_2 in ps_1 keeps pumping tokens into ps_3 which cannot be switched back to ps_1 — places in ps_3 becomes unbounded.

2. PRELIMINARIES

To save space, I assume that the readers are familiar with Petri nets for which [MUR 89, PET 81] are the best reference. I am interested in Petri nets whose initial markings can always be recovered (i.e., the *PN* is *reversible*); the corresponding period is termed as the *iteration period*. Temporal relationships between transitions and places within a single iteration period can be one of the following: *sequential*, *concurrent*, and *exclusive*.

I define temporal relationship between two nodes in the structural sense. That is, I start from an enabled transition and delete all the tokens in the net such that only this transition is firable and it fires once. Then, I fire any transitions subsequently enabled, and so on. If no transitions are enabled, I add minimum amount of tokens such that the blocked tokens can enable some transitions. Continue this process until no more

tokens are needed. Based on this net, I then define the following:

Sequential (*SQ*): $P_i \rightarrow P_j$ (P_i is *sequential earlier* than P_j and P_j is *sequential later* than P_i) if P_j cannot hold a token without earlier presence of a token in P_i, when both hold a token in one iteration of the Petri net.

Similar definitions apply to $P_j \rightarrow t_i$ (e.g, if t_i cannot fire without the earlier presence of tokens in P_j, when t_i fires and P_j hold tokens in one iteration), $t_i \rightarrow P_j$, $P_i \rightarrow t_j$, and $t_i \leftarrow P_j$.

Concurrent (*CN*): $P_i \parallel P_j$ (P_i *is concurrent to* P_j) ($t_i \parallel t_j$) if there always exists a moment such that both hold tokens (fire).

Exclusive (*EX*): $P_i \mid P_j$ ($t_i \mid t_j$) (P_i *is exclusive to* P_j) if P_i (t_i) is neither sequential nor concurrent to P_j (t_j). The synchronic distance [MUR 89] between t_i and t_j, d_{ij}, is ∞. (Murata defines d_{ij} as the maximum number of firings of t_i without any firing of t_j).

A *pseudoprocess* (*PSP*) in a Petri net *PN* is a directed elementary path (DEP) where the in degree and out degree of any node (transition or place) inside the path are one except its two end points. In addition, a *PSP* cannot be a subset of any other *PSPs*. x is a *generation point* of a *PSP*, if $x \rightarrow y$ for any other $y \in PSP$. x is a *joint* of a *PSP*, if $x \leftarrow y$ for any other $y \in PSP$. It means that a *PSP* is a directed path containing two end nodes; the starting node is the generation point and the end node is the joint. Any other node in the *PSP* has a single input node and a single output node.

A new *PSP* (*NP*) connects from PSP_g to PSP_j. If $PSP_g = PSP_j$, it is a pure generation (*PG*); otherwise it is an interaction generation (*IG*). If both the *generation point* and the *joint* of a *NP* are places (transitions), then it is a *PP generation*; the corresponding path is called a *PP–path* (*TT–path*). t_g (t_j) is the output (input) transition of the *generation point* (*joint*) of a *PP–path*. P_g (P_j) is the output (input) place of the *generation point* (*joint*) of a *TT–path*. A *virtual path (VP)* is a *PSP* with no places or transitions other than the *generation point* and the *joint*. $PSP_i \rightarrow PSP_j$ if $\exists P_1 \in PSP_i$ and $\exists P_2 \in PSP_j$ such that $P_1 \rightarrow P_2$.

$PSP_i \parallel PSP_j$ if $\forall$ pair of $P_1 \in PSP_i$ and $P_2 \in PSP_j$ s.t. $P_1 \parallel P_2$.

The local concurrent set of PSP_i with respect to PSP_j, $LCN1 = LCN(PSP_i, PSP_j)$, is the set of all those *PSPs* which are concurrent to each other and which are equal or concurrent to PSP_i but not to PSP_j, i.e., $\forall PSP_k \in LCN1$, $PSP_k \parallel PSP_i$ or $PSP_k = PSP_i$, $PSP_k \parallel$ (not concurrent to) PSP_j, and $\forall PSP_{l \neq k} \in LCN1$, $PSP_k \parallel PSP_l$. $LCN2 = LCN(PSP_j, PSP_i)$. $PSP_i \mid PSP_j$ if $\forall$ pair of $t_1 \in PSP_i$ and $t_2 \in PSP_j$ such that $t_1 \mid t_2$.

The local exclusive set (LEX) of PSP_i with respect to PSP_j, $LEX1 = LEX(PSP_i, PSP_j)$ is the set of all those *PSPs* which are exclusive to each other and which are equal or exclusive to PSP_i but not to PSP_j. That is, $\forall PSP_k \in LEX1$, $PSP_k \mid PSP_i$ or $PSP_k = PSP_i$, $PSP_k \mid$ (*not exclusive to*)PSP_j, and $\forall PSP_{l \neq k} \in LEX1$, $PSP_k \mid PSP_l$. $LEX2 = LEX(PSP_j, PSP_i)$.

3. THE SYNTHESIS RULES

Suppose that a Petri net model is a set of interacting entities. Initially, each entity has a **home place** with tokens indicating that it is ready to start. Each entity conducts consecutive tasks (which can be considered as transitions), and then returns to the original ready state. Each state can be considered as a place. Executing a task means firing a transition. At the completion of executing each task, the token leaves the original state and enters the next state by firing a transition. Thus, in the corresponding Petri net model, 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 as seen in Fig. 3(a). The process for this cycle is termed a **basic process**.

This *basic process* can be expanded through a series of path generations. Each path is a *PSP*. There are two ways to generate paths for both *PG* and *IG*: *TT rule* and *PP rule*. The *TT rule* governs that paths be generated only between transitions, from the *(generation point)* t_g to the *joint* t_j. The *PP rule* governs that paths be generated only between places, from a *generation point* P_g to a *joint* P_j. For each of the *TT* and *PP rules* applied for a *PG*, there can be two kinds of generations: *forward* and *backward* (Fig 3(b)). If $P_g \rightarrow P_j$ $(t_g \rightarrow t_j)$, then it is a forward *PP* (*TT*) generation; it is a backward *PP* (*TT*) generation if the $\rightarrow$ reverses.

A. Guidelines for Synthesis

Given a net PN_1, a set of new paths NP_1 are generated using some synthesis rules to form another net PN_2. The synthesis rules should be such that all transitions

(places) in the subnet NP_1 and NP_1 are all live (bounded). This can be ensured by

1) No intrusion on normal operations of NP_1 prior to the generations. This guarantees no unbounded places and dead transition in PN_1 since it was live and bounded prior to the generations.

2) No dead transitions and unbounded places in NP_1.

There are two kinds of intrusions. One changes the marking on PN_1; the other eliminates some reachable markings. In order for this path to be alive, it must be able to get tokens. When tokens in this NP_1 disappear, the resultant marking must be a reachable marking in PN_1 for no intrusion.

The second intrusion may happen when the joint is a transition. It may never fire (even though it is potentially firable); hence causing some M in PN_1 not reachable. Hence if the joint is a transition, the PN_1 must always be able to get tokens within each iteration.

The synthesis rules are constructed based on the following guidelines: Based on the concept of *no intrusion*, the rules are constructed based on the following guidelines:

(1) Each NP_1 in the set must potentially (always if the joint is a transition) be able to get tokens to fire its transitions.

(2) These token must be able to disappear from the NP_1 and return to PN_1.

(3) After all the tokens inside the NP_1 disappear, subsequent reachable markings must be exactly identical to those in PN_1 prior to this NP_1

generation.

(4) NP_1 cannot get infinite number of tokens without being consumed.

It is easy to see that Petri nets are synthesized, as per the above four guidelines they are live, bounded and reversible [YAW 87]. Guidelines (1) to (3) guarantees no intrusion; guideline (2) guarantee no dead transitions (since all tokens can disapper from the NP); ; and guideline (4) guarantees no unbounded places in the NP.

The *TT* and *PP rules* are developed based on the above four guidelines and are formalized as follows:

B. TT Rule:

Connect a path NP from t_g to t_j.

1) (*TT*. 1) If $PSP_g = PSP_j$, it is a pure *TT generation*.

2) (*TT*. 2) If t_g <- t_j, insert tokens in this new cycle if it does not have any token.

3) (*TT*. 3) If there is no path from t_g to t_j, then generate another new TT-path to synchronize t_g and t_j such that there is a path from t_j to t_g. This rule must be checked concurrently along with other TT rules.

4) Else if there exists a path from t_g to t_j and $t_g \parallel t_j$ or t_g -> t_j or t_j -> t_g,

4.A) Both are not in a cycle which was generated using the PP rule,

If $PSP_g \neq PSP_j$, it is an interactive *TT generation*. There are two alternatives of path generations using the TT rule :

I. (*TT*.4)

a) (*TT*.4.1) Generate a *TP–path* from a transition t_g of each unused PSP_g in *LEX* 1 to a place P_k in the *NP*.

b) (*TT*.4.2) Generate a virtual *PT–path* from the place P_j (the input place of t_j on the *NP*) to a transition t_j of each unused PSP_j in *LEX* 2.

II. (*TT*.5) If all *PSPs* in *LEX* 2 have a common *generation point* (denoted by *CG*) then,

a) (*TT*.5.1) Generate a *TT–path* from an unused PSP in *LEX* 1 to an output transition (in an unused *PSP* in *LEX* 2) of the *CG*. This *TT–path* does not share any nodes with any previously *TT–paths* generated by this instance of *TT*.5 rule.

b) (*TT*.5.2) If all *PSPs* in *LEX* 1 have been used, generate a *TP–path* from a P_j (the input place of t_j on the *NP*) of a *NP* generated previously from this instance of *TT*.5 rule to an output transition (in an unused PSP of *LEX* 2) of the *CG*.

4.B) (*TT*.6) Else both are in a cycle which was generated using the PP rule, then synchronize the two circles such that each circle cannot be traced more than once without the other circle being traced once in any iteration of the net.

4.C) (*TT*.7) If only one token is in a circle, then issue a warning message.

5) (*TT*. 8) If $t_g \mid t_j$, then issue a warning message.

C. PP Rule:

Connect a path *NP* from P_g to P_j.

1) If $P_g \mid P_j$ or $P_g \rightarrow P_j$,

A) (*PP*. 1) If $PSP_g = PSP_j$, it is a pure *PP generation*.

B) (*PP*. 2) else it is an interactive *PP generation*.

a) (*PP*. 2.1) Generate a *TP–path* from a transition t_k of the *NP* to a place p_j of each unused PSP_j in *LCN* 2.

b) (*PP*. 2.2) Generate a virtual *PT–path* from the place P_g of each unused PSP_g in *LCN* 1 to the transition t_g (the output transition of P_g on the *NP*).

2) (*PP*. 3) If $P_g \parallel P_j$, then issue a warning message.

Here I discuss about the physical meanings of interactive generations. An interactive *TT generation* model is a concurrent interaction such as a message exchange. PSP_i sends a token (or a message) to PSP_j. Messages received by an entity must be properly handled. If a received message is not consumed by any process of the entity, the protocol will be *incomplete* [RAM 85]. Thus, each member in *LEX* 2 must be able to receive tokens. By the same token, each member in *LEX* 1 must have the potential to send away tokens. This is the *TT*. 4 or the *TT*. 5 rule.

An interactive *PP generation* model is an exclusive interaction such as context switching in a time-sharing system where processes share the CPU resource in a round-robin fashion. In terms of Petri nets, if only subsets are considered, without special measures, deadlocks may appear. Thus when a *PSP* 1 switches to another *PSP* 2, all the members of *LCN* 1 must switch together and all the members of *LCN* 2 must be activated.

4. EXAMPLES OF SYNTHESIZED PETRI NETS

Examples of Rules:

Pure TT Generation (*TT*. 1): See Fig.3(b)

Pure PP Generation (*PP*. 1): See Fig.3(b)

(*TT*. 2): Fig. 3(b) shows a backward *TT generation* creating a cycle containing t_g and t_j — a deadlock. Inserting tokens in this cycle removes this deadlock.

(*TT*. 3): Fig. 4(a) shows an interactive *TT generation* from t_1 at entity 1 to t_2' at entity 2. Firing t_1 in entity 1 will deposit a token at place P_6 which can be removed only by firing t_2' in entity 2. P_6 can get unbounded by firing transitions in entity 1 an infinite number of times and not firing any transition in entity 2. (The synchronic distance between (t_2 and t_2') is ∞; hence $t_1 \parallel t_2'$). To remove this unboundedness problem, generate a *TT–path* from t_3' to t_4 as shown in Fig. 4(b) such that t_1 and t_2' are in a cycle with a token in P_1. Now transitions in entity 1 cannot fire an infinite number of times without firing transitions in entity 2, thus they are synchronized and

the unboundedness problem disappears.

Interactive TT Generation (*TT*.4): Fig. 5(a) shows an interactive *TT generation* from t_2 of entity 1 to t_6' of entity 2. $LEX1 = \{PSP1, PSP3\}$ and $LEX2 = \{PSP2, PSP4\}$. The PN' is *unbounded* because if $PSP1$ fires an infinite number of times to deposit an infinite number of tokens in place P_6 which can never be consumed and P_6 becomes *unbounded* if the entity 2 always fires $PSP2$ instead of $PSP4$. The virtual path from P_6 to t_3' (Fig. 7(b)) will consume the tokens in P_6, and therefore PN is bounded. Suppose $PSP3$ fires infinitely often without firing $PSP1$ and entity 2 can never fire t_3' and t_6'. This constitutes an unfair situation. This unfairness is remedied by generating a new path connecting PSP3 (Fig. 5(b)) to the *PSP* according to the *TT*.2 rule.

Interactive TT Generation (*TT*.5): Fig. 6 shows another structure; the resulting net is also *live* and bounded. Here each PSP in $LEX1$ connects exactly one in $LEX2$, as against that in Figure 6(a) where all are connected through P_4. Note that each of all these *NPs* must join at an output transition of the CG (P_2').

Interactive PP Generation (*PP*.2): Fig. 7(a) shows an interactive *PP generation* from P_2 of entity 1 to P_2' of entity 2. $LCN1 = \{PSP1, PSP3\}$ and $LCN2 = \{PSP2, PSP4\}$. Note that if a token in $PSP1$ diverts to $PSP2$, none of the transitions in PN is firable, i.e., the *PSP* is not *live*. To maintain *liveness*, the *PP*.1 rule dictates that a path be connected from each *PSP* of these two *LCNs* to the new *PSP* as shown in Fig. 8(b). Note that $P_2\ t_g$ and $P_5\ t_g$ are *VPs* to ensure a complete switching of

tokens from $LCN1$ to $LCN2$. Otherwise, tokens could be trapped in both $LCNs$ which results in a deadlock.

5. ARC-RATIO RULES FOR GENERAL PETRI NETS

The rules for ordinary Petri nets must be modified due to the presence of multiple weights of arcs. The synthesis rules of the knitting technique for ordinary PN should also be observed for general Petri nets in order to meet the guidelines. This set of rules is referred to as the **connection rule** since the absence of any NP will cause some guidelines violated. The weight of arcs in the PN must satisfy some constraints; otherwise, a PN may be nonlive as shown in Fig.8(a) and 8(b). This additional set of rules is called the **Arc-Ratio** rule. Again, one can develop the arc-ratio rules based on the four guidelines.

The following definitions consider a pair of nodes (transitions or places) and all the paths between them in isolation (i.e., all nodes and arcs not in these paths are deleted from PN_1).

Definitions of least ratios are given below :

The least firing ratio $lfr_{ij} = \frac{a}{b}$ between t_i and t_j: is the firing ratio $\frac{a}{b}$ where a is the least firing number of t_i for t_j to fire for b times with no tokens left in path between t_i and t_j.

The least marking ratio of P_a with respect to P_j $lmr_{ij} = \frac{a}{b}$: is the least amount of tokens in P_i to fire transitions between P_i and P_j for P_j to get b tokens with no

tokens left in paths between t_i and t_j.

The least marking–firing ratio of P_i with respect to t_j, $lmfr = \frac{a}{b}$ between P_i and t_j: is the marking-firing ratio $\frac{a}{b}$ where a is the least marking of P_i such that t_j is firable for j times with no tokens left in paths between P_i and t_j.

The least firing–marking ratio $lmfr = \frac{a}{b}$ between t_i and P_j: is the firing-marking ratio $\frac{a}{b}$ where a is the least firing number of t_i such that P_b can get b tokens by firing transitions between t_i and P_j with no tokens left in path between t_i and P_j.

I refer these ratios as *least ratios* q. Let $q = \frac{a}{b}$ be one of these ratios. Then $q^u = a$ and $q^l = b$.

$|q = \frac{a}{b}| = \frac{a'}{b'}$ where a' and b' are prime to each other. $q \in lfr, lmr, lfmr\ lmfr$.

$q_1 >_c$ *literally greater than* q_2: $|q_1| = |q_2|$ and $a_1 > a_2$ and $b_1 > b_2$.

Example: Fig. 8(a) shows a *NP* (the dashed line) from t_2 to t_4 with $lfr_{24} = 2/2$ which is literally greater than ($>_c$) than that ($lfr = 1/1$) for the *TTP* ($t_2P_2t_3P_3t_4$).
Both | lfr |=1/1. The net is nonlive. Fig. 8(b) shows a *NP* (the dashed line) from P_1 to P_2 with $lmr_{12} = 2/2$. which is literally greater than ($>_c$) that ($lmr = 1/1$) for the *PPP* ($P_1t_2P_2$). Both | lmr |=1/1. The net is nonlive.

Partial flow: A path with a generation point of place P_g is said to have partial flow if its t_g (the output transition of P_g on the path) must fire multiple number of times to

have no tokens (consumed by firing t_g) blocked inside the path.

Input ratio of a path: is the ratio of lmr_i^u to the arc weight between P_g and t_g.

Partial flow loses some tokens in some PPPs and will cause deadlocks.

Example: Fig. 9 illustrates an example of partial flow which causes a deadlock as seen in Fig. 9(b). The input ratio of path $(P_2t_2P_3t_3P_4)$ is 2 and that of path $(P_2t_5P_5t_6P_4)$ is 4.

The following observation is useful to find least ratios between nodes that are not sequential to each other.

Observation 1: (1) If x *SQ* y, then there exists a DEP1 from x to y which does not pass through the home place.

(2) If x || y or $x \mid y$, then the DEP1 from x to y contains the home place. If there exists a DEP2 which branches from DEP1 at a transition z (place), then x || y ($x \mid y$), z->LEX(x,y) and z->LEX(y,x).

In the sequel, σ denotes a firing sequence, and $\sigma(t_i)$ denotes the number of firing times of t_i in σ.

Example: In Fig. 4(a), t_1 *SQ* t_3; the DEP1 is $(t_1P_2t_2P_3t_3P_3 \parallel P_5)$; the DEP1 is $(P_3t_3P_4t_4P_1t_1\ P_5)$ where P_1 is the *home place*; the DEP2 is $(t_1P_2t_2P_3)$. In Fig.5, $t_3 \mid t_j$; the DEP1 is $(t_3P_4t_4P_1t_1P_2t_gP_5t_j)$; the DEP2 is $(P_2t_2P_3t_3)$.

Theorem 1: If $t_{g1} \parallel t_{j1}$, $\forall$ σ such that M_0 -> M_0,

$$\sum_{t_{gs} \in LEX(t_{g1}, t_{j1})} \frac{\sigma(t_{gs}) lfr_{g1gs}}{\sum_{t_{js} \in LEX(t_{j1}, t_{g1})} \sigma(t_{js}) lfr_{j1js}} = |lfr_{g1j1}|.$$

Proof: (1) Consider the special case where there is only one member in both $LEX(t_i,t_j)$ and $LEX(t_j,t_i)$. Let $lfr_{ij} = \frac{a}{b}$. From *observation* 1 there exists DEP1 and DEP2 which intersects at a transition t_k. When t_i fires $'a'$ times, t_j is able to fire b times and t_k fires $'a\ lfr_{ki}\ '$ times since t_k is on the path from t_i to t_j. In turn, these firings of t_k induce $a\ lfr_{ki}\ lfr_{ik}$ firings of t_i and $a\ lfr_{ki}\ lfr_{jk}$ firings of t_j. Hence $\frac{\sigma(t_i)}{\sigma(t_j)} = lfr_{ij}$.

(2) There are more than one member in both $LEX(t_i,t_j)$ and $LEX(t_j,t_i)$. When t_k fires w times, then each member t_{gs} (t_l) of $LEX(t_i,t_j)$ $(LEX(t_j,t_i))$ may fire $v_{gs}\ lfr_{gsk}$ times $(v_{js}\ lfr_{jsk})$ as for (1) where v_{gs} (v_{js}) is an integral factor of w. Since each member of $LEX(t_{g1},t_{j1})$ ($LEX(t_{j1},t_{g1})$) is exclusive to each other, the sum of v's must equal to w; i.e., $\sum_{t_{gs}\in LEX(t_{g1},t_{j1})} v_{gs} = w$ ($\sum_{t_{js}\in LEX(t_{j1},t_{g1})} v_{js} = w$) which is equivalent to the formula in the theorem. □

Theorem 2: If $p_i \mid P_j$, $\exists$ M_1 and M_2, $M_1(P_i) \neq 0$, $M_1(P_j) = 0$, $M_2(P_j)\neq 0$, and $M_2(P_i) = 0$, then $\frac{M_1(P_i)}{M_2(P_j)} = lmr_{ij}$.

Proof: (1) Consider the special case where there is only one member in both $LCN(t_i,t_j)$ and $LCN(t_j,t_i)$. Let $lmr_{ij} = \frac{a}{b}$. From *observation* 1 there exists DEP1 and DEP2 which intersect at a place P_k. When P_i gets a tokens, P_j is able to get b tokens via the firing path along DEP1 and P_k is able to get a $\mid lmr_{ki} \mid$ tokens since P_k is on the path from P_i to P_j. Because $P_i \mid P_j$, this set of tokens in P_k flow to either

P_i ($M_1(P_i)$ = a $|lmr_{ki}|$ $|lmr_{ik}|$, $M_1(P_j) = 0$) or to P_j ($M_2(P_j)$ = a $|lmr_{ki}|$ $|lmr_{jk}|$, $M_2(P_i) = 0$).

(2) There are more than one member in both $LCN(t_i,t_j)$ and $LCN(t_j,t_j)$. The proof follows the same idea in (2) of the proof of Theorem 1. □

Similarly the following theorems can be proved in same way as above.

Theorem 3: If $P_i \parallel P_j$,M(P_i) !=0, M(P sub j) !=0, then $\forall$ M, $\frac{M(P_i)}{M(P_j)} = lmr_{ij}$.

Theorem 4: If $P_i \mid P_j$, $\exists$ M_1 and M_2, $M_1(P_i) \neq 0$, $M_1(P_j) = 0$, $M_2(P_j) \neq 0$, and $M_2(P_i) = 0$, then $\frac{M_1(P_i)}{M_2(P_j)} = lmr_{ij}$.

Now the Arc-ratio rule is summarised as follows.

Arc-Ratio Rule:

If an NP connects from x to y,

($ARR.1$) If paths in NP_1 exist between x and y,

($ARR.1.a$) $q_{NP} <_c q_{xy}$,

($ARR.1.b$) If x is a place, all (except one) paths between x and y have input ratio of one.

($ARR.1.c$) Replace the phrase of "enough tokens" in the TT.2 rule by " enough tokens to enable t_g lfr_{gj}^u times".

Possible sets of {x,y, q} are {transition, transition, lfr}, {transition,

place, lfmr}, {place, place, lmr}, and {place, transition, lmfr}.

(*ARR*.2) Otherwise, there must be another accompanied *NP* (TT.3 rule) from x' to y' such that the *NP* from x via y, x' to y' satisfies ARR.1.a.

6. EXAMPLES OF SYNTHESIS RULES FOR GENERALIZED PN

Example: (ARR.2) Fig. 10 shows a *PN* with two disconnected subnets. There is no existing path between t_1 and t_2'. A TTP connects from t_1 and t_2'. To synchronize N_1 and N_2, another TTP is generated from t_3' to t_4. Note that $lfr_{14} = lfr_{12'3} = \frac{2}{1}$ where "12'3" indicates the path $(t_1 t_2' \cdots t_3' t_4)$.

Fig. 11. Shows an *NP* $(t_2 P_6 t_3')$ is generated using the *TT* rule between t_2 of PSP1 and t_3') of PSP2 and $t_2 \parallel t_3'$ and $d_{23'} = \infty$; hence generate another *NP* $(t_4' P_7 t_4)$ such that $d_{23'}=1$. $lfmr_{56}$ =2/8 for the path from t_5 to P_6: $(t_5 P_5 t_6 P_4 t_4 P_1 t_1 P_2 t_2 P_6)$. Thus I connect an arc with weight=4 from t_5 to P_6. $lmfr_{66'}$ =4/4 along the only path from P_6 to t_6': $(P_6 t_3' P_4' t_{4'P_1}' t_1' P_2' t_5' P_5' t_6')$. Thus I connect an arc with weight=1 from P_6 to t_6'.

Fig. 12. An *NP* $(P_2 t_g t_j P_2')$ is generated using the *PP* rule between P_2 of *PSP* 1 and P_2' of *PSP* 2. $P_2 | P_2'$, $lmfr_{22'}$ =4/4 along the only path from $(P_2 t_2 P_3 t_3 P_4 P_1 t_1' P_2')$. $lmfr_{2g}$ =2/1. I connect a *VP* with weight=2 from P_5 to t_g (PP.2.1 rule & Arc-ratio rule ARR.1.a). Based on PP.2.2 rule, I connect a *TPP* from t_j to P_5' with weight=2, since $lfmr_{j5'}$ =1/2.

Note that a synthesis step may involve more than one rule (referred to as a

composite synthesis step, Otherwise, it is a *singular synthesis step*.). For instance, the generation of $(t_2 P_6 t_3')$ must be followed by the generation of $(t_4' P_7 t_4)$ per the TT.3 rule and followed by the generation of $(t_5 P_6)$ and $(P_6 t_6')$ per the TT.4 rule.

An Example of Synthesis:

Fig. 13(a) shows a general petri net from Fig. of [KOH 90] which is live and bounded. First construct a basic process as shown in Fig. 13(b). Fig. 13(c)-(e) show the remaining synthesis steps.

7. CORRECTNESS OF THE SYNTHESIS TECHNIQUE

In the sequel, I prove the correctness of the synthesis rules for the case of *singular synthesis steps*. The proof for the case of a *a composite synthesis step* is similar. The proofs of the following theorems are in appendix.

Theorem 5: After a singular synthesis step of TT.1 forward generation from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

Theorem 6: After a singular synthesis step of TT.2 forward generation from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

Theorem 7: After a singular synthesis step of TT.3 forward generation from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

Theorem 8: After a singular synthesis step of TT.4 forward generation between t_{g1} and t_{j1} from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

Theorem 9: After a PP.1 forward generation from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

Theorem 10: After a PP.2 forward generation from a synthesized PN_1, PN_2 remains live, bounded, and reversible.

APPENDIX A

Appendix A: Proof of Theorem 5

Proof: Since $lfr_{gJ} \geq_c lfr_{NP}$, t_g of the NP is potentially able to fire ($lfr^u_{gJ} > lfr^u_{NP}$) times. Hence NP_1 is potentially able to get tokens to fire its transitions and satisfies guideline (1) These tokens can completely disappear from NP_1 (no blocking inside it again due to $lfr_{gJ} \geq_c lfr_{NP}$) and return to a reachable marking in PN_1. All reachable markings of PN_2 cover those of NP_1. Thus all guidelines are satisfied. □

Appendix A: Proof of Theorem 6

Proof: Because there are enough tokens in the NP to support t_g to fire lfr^u_{gJ} times, guideline (1) is satisfied. (Otherwise, without enough tokens in the NP which is backwardly generated, t_g cannot fire.) Other parts of the proof are similar to that of TT 1

Appendix A: Proof of Theorem 7

Proof: It is easy to see that when t_x fires lfr^u_{xy} times, t_y is always able to fire lfr^l_{xy} times and $t_{y'}$ is in turn always able to fire lfr^l_{xy} times. And afterwards there are no tokens left in the two NPs. Other parts of the proof are similar to that of TT 1

Appendix A: Proof of Theorem 8

Proof: The proof is similar to that for TT.1 when all lmr^u_{gJ} tokens flow through NP_1. When only a portion of these tokens flow through NP_1, Arc-Ration rule ARR.1.b prevents any blocking inside NP_1. □

Appendix A: Proof of Theorem 9

Proof: . Let LEX1=LEX(t_{g1},t_{J1})=t_{g1}, t_{g2}, ..., t_{gm} be the set of generation points and

LEX2=t_{J1}, t_{J2}, ..., t_{Jn} the set of joints. Let σ_1 be a firing sequence prior to the macro generation. I want to show that $\exists$ a σ_2 in PN_2 such that $\sigma_2(t_{xs}) = \sigma_1(t_{xs})$, x=g, s=1,...,m or x=J, s=1,...,n (i.e., no second intrusion). That is, by firing each t_{gs} $\sigma_1(t_{xs})$ times, P_J get enough tokens to support each t_{Js} to fire $\sigma_1(t_{Js})$ times. In addition, the marking after each t_{Js} fires $\sigma_2(t_{Js})$ times must equal to that prior to the macro generation (no first intrusion), which can be satisfied by having no tokens left in some NPs between t_{gh} and t_{jk}.

Firing t_{gs} $\sigma_1(t_{gs})$ times, P_J can get $\sigma_1(t_{gs})lfmr_{gsJ} = \sigma_1(t_{gs})lfr_{g1gs}lfr_{J1g1}a$ amount of tokens. where a_{JJ1} is the weight of the arc between P_J and t_{J1}. The total amount of tokens that P_J get by firing all t_{gs}'s is $\sum_{t_{gs} \in LEX1} \sigma_1(t_{gs})lfr_{g1gs}lfr_{J1g1}a_{JJ1} = \sum_{t_{Js} \in LEX2} \sigma_1(t_{Js})lfr_{J1Js}lfr_{g1J1}lfr_{J1g1}a_{JJ1} = a_{JJ1} \sum_{t_{Js} \in LEX2} \sigma_1(t_{Js})lfr_{J1Js}$. The total amount of tokens in P_J required to support all t_{Js}'s to fire $\sigma_1(t_{Js})$ times is $\sum_{t_{Js} \in LEX2} \sigma_1(t_{Js})\, a_{JJs}$. Substituting $a_{JJs} = lmfr_{JJs} = a_{JJ1}lfr_{J1Js}$ into the above equation, it becomes $a_{JJs} \sum_{t_{Js} \in LEX2} \sigma_1(t_{Js})lfr_{J1Js}$, which is exactly the amount of tokens injected into P_J by firing all t_{gs}'s as derived earlier. This "exact amount of tokens" ensures that no tokens are left in the *NPs* and the same marking as in PN_1 is reached in PN_2 after each t_{Js} fires $\sigma_1(t_{Js})$ times. Thus, guidelines (2)-(4) are thus satisfied. □

Appendix A: Proof of Theorem 10

Proof: The proof is similar to that of TT.4. It is easy to see that all NPs in the macro generation are potentially able to get tokens and when these tokens disappear, the resultant marking is identical to that for NP_1. Hence all guidelines are satisfied. □

FIGURES

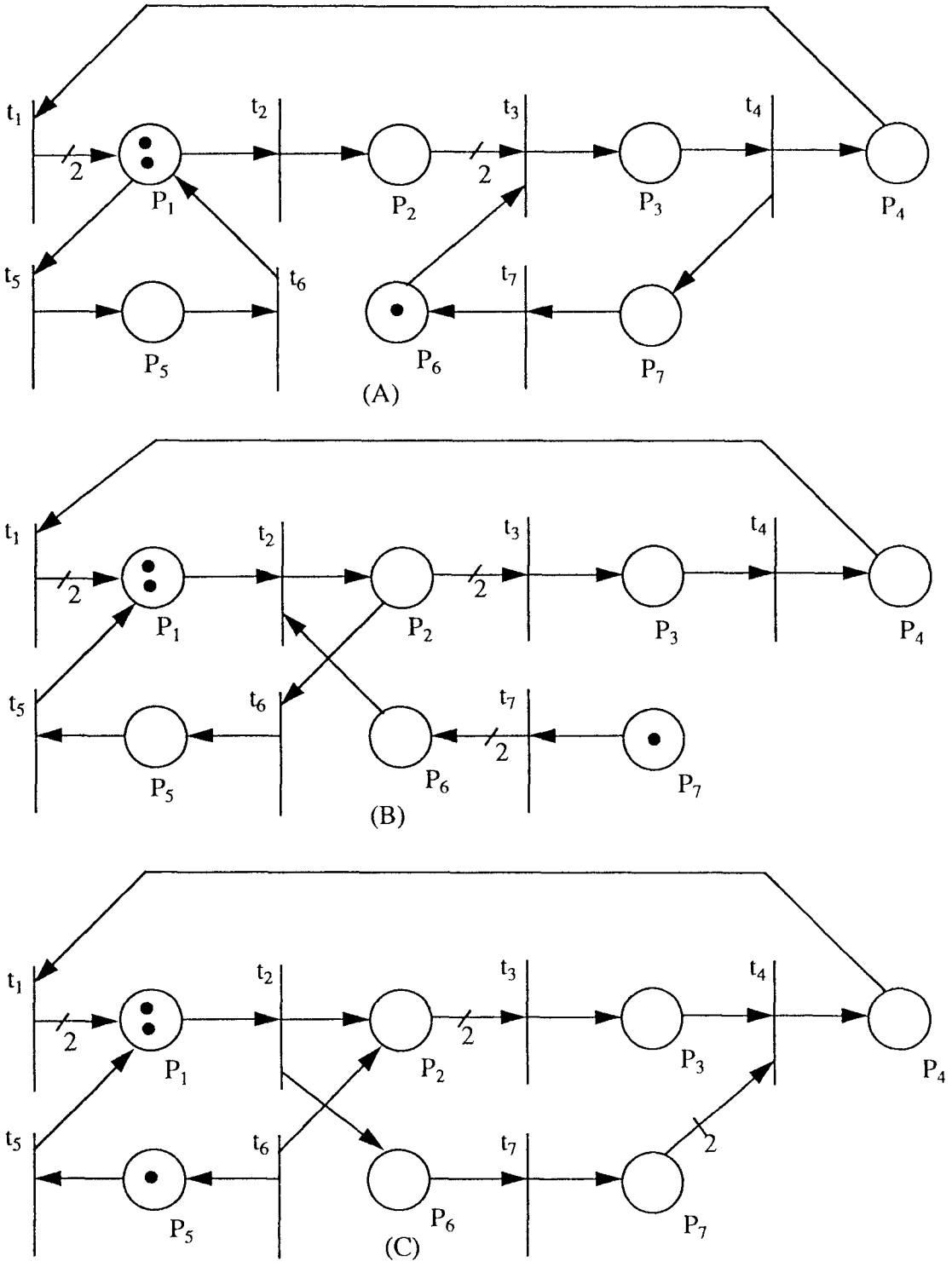
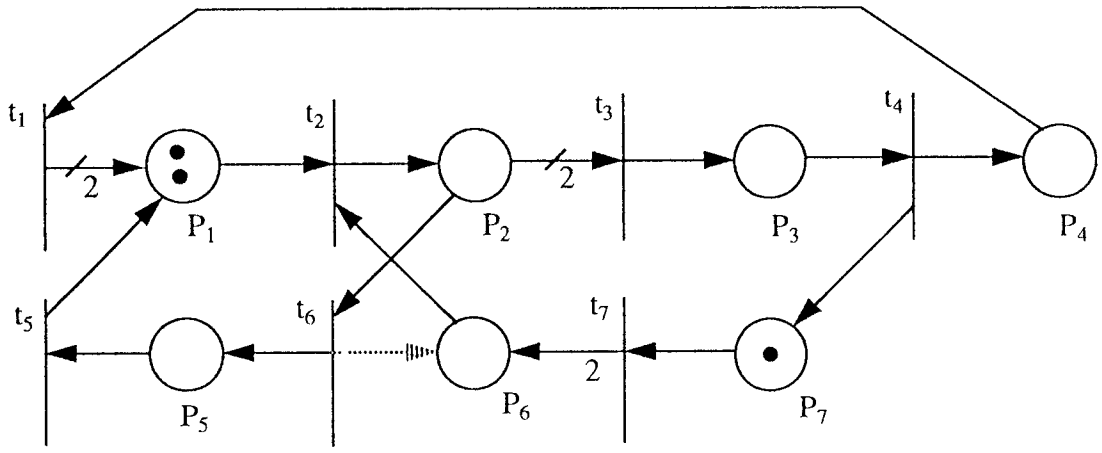
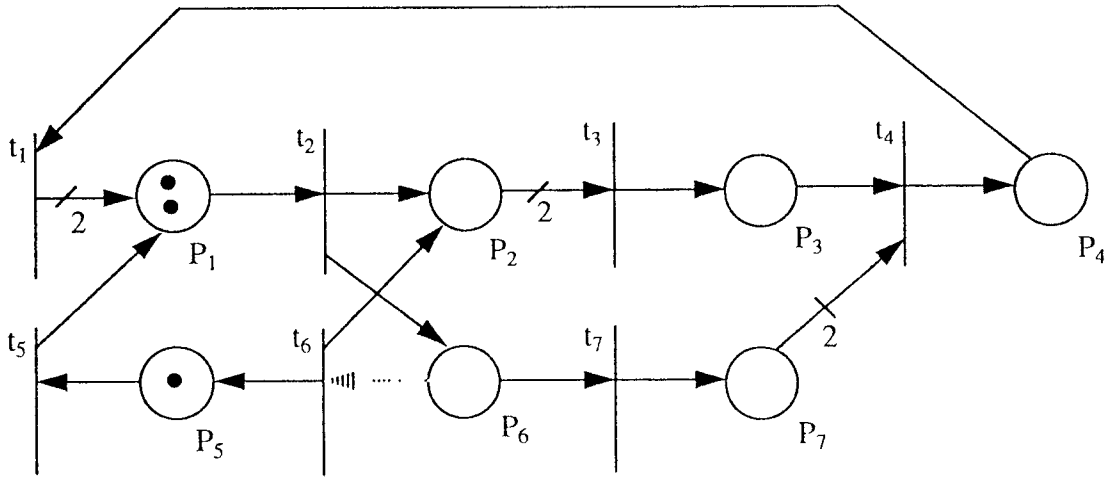


Fig . 1 Examples of non-overlapping and overlapping LB-circuits.
(a) Non-overlapping
(b) Overlapping, not live
(c) Overlapping, not bounded



(A)



(B)

Fig 2. Applications of PP.2 rule

(a) By adding the TPP from t_6 to P_6, the net in fig1(b) becomes live.

(b) By adding the PTP from P_6 to t_6, the net in fig1(c) becomes bounded.

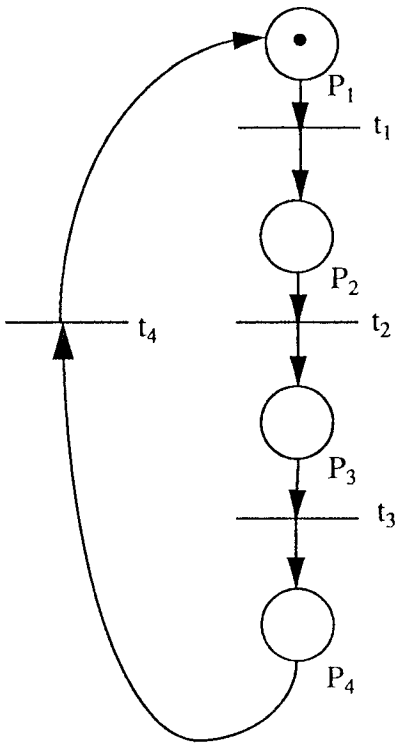


Fig. 3(a) An example of a basic process.

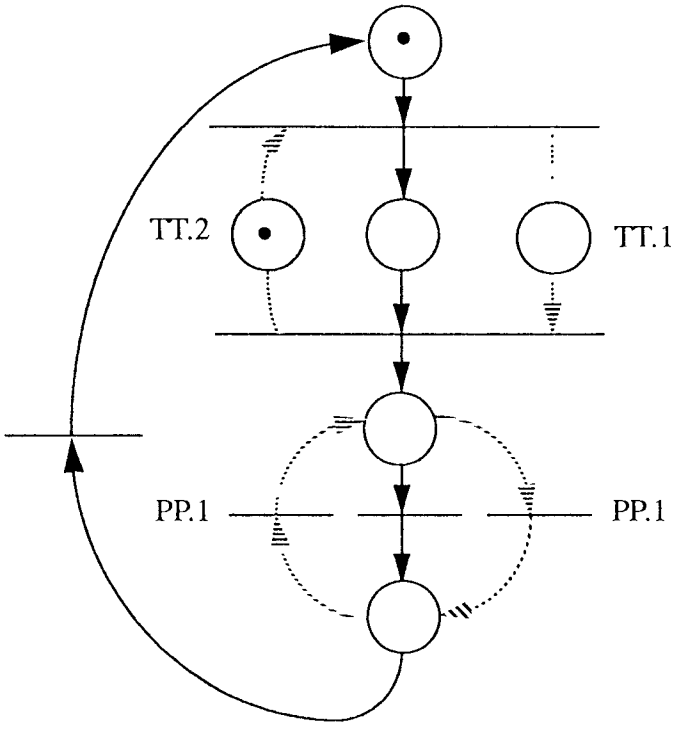


Fig 3(b) An example of the TT-paths and PP-paths.

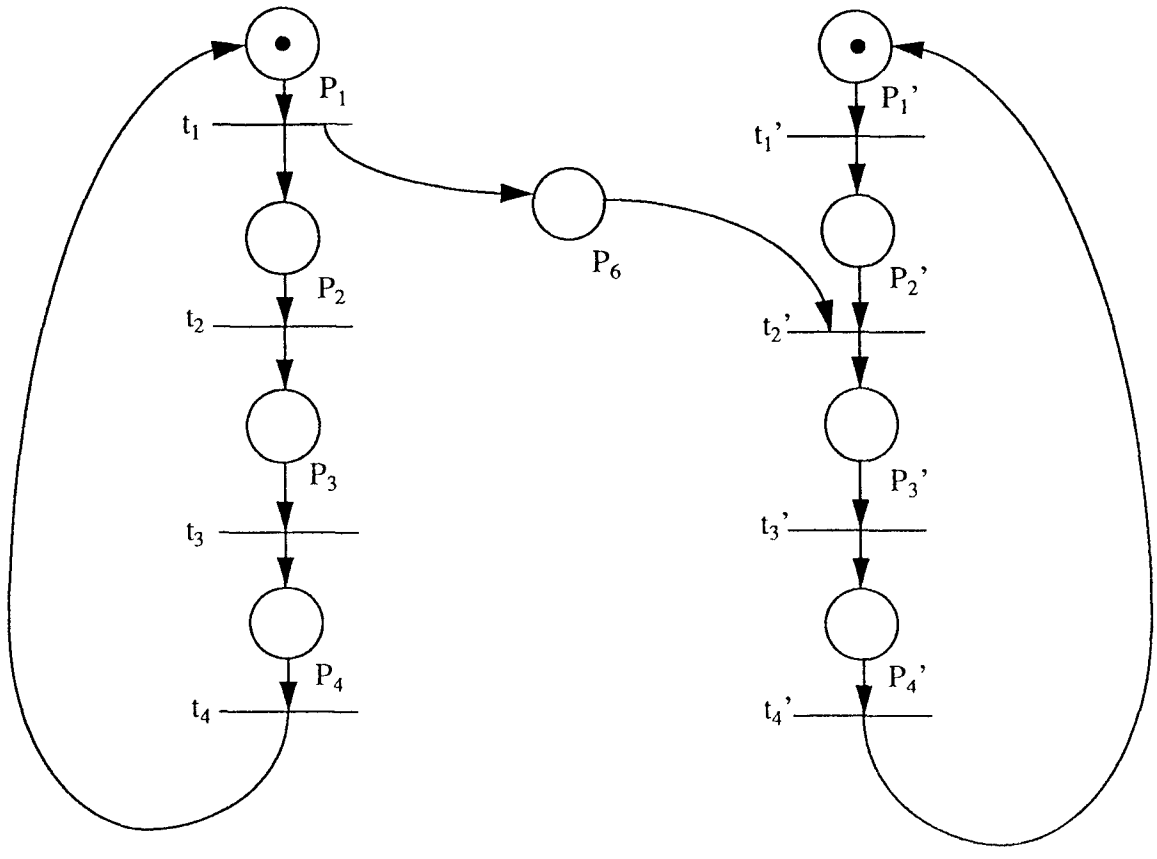


Fig 4. An example of interactive TT generation (TT.3 rule).

Fig 4(a) PSP($t_1P_6t_2'$)is generated between t1 and t_2'; ($t_1 \parallel t_2'$)

The net is unbounded.

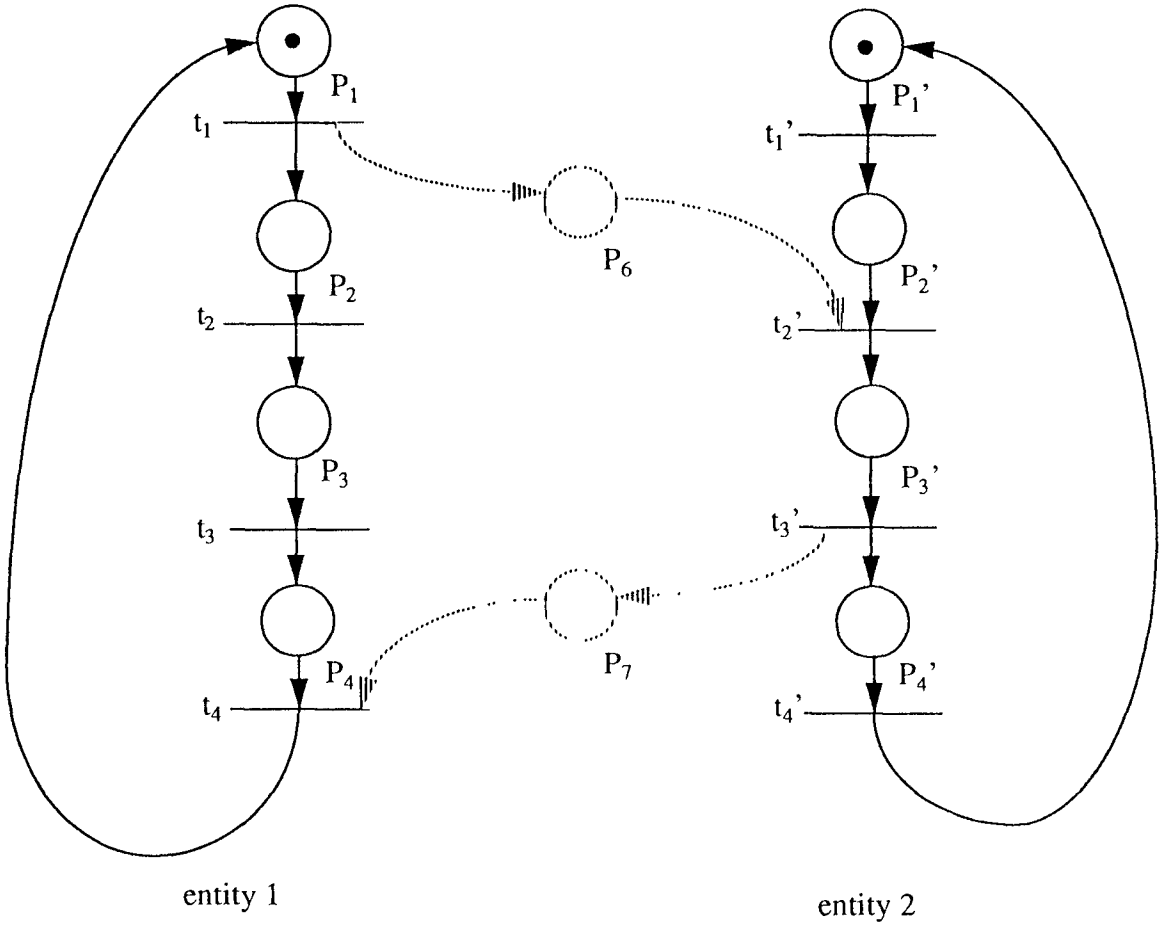


Fig 4. (b) The TT-path from t_1 to t_2' must be accompanied by the TT-path from t_3' to t_4

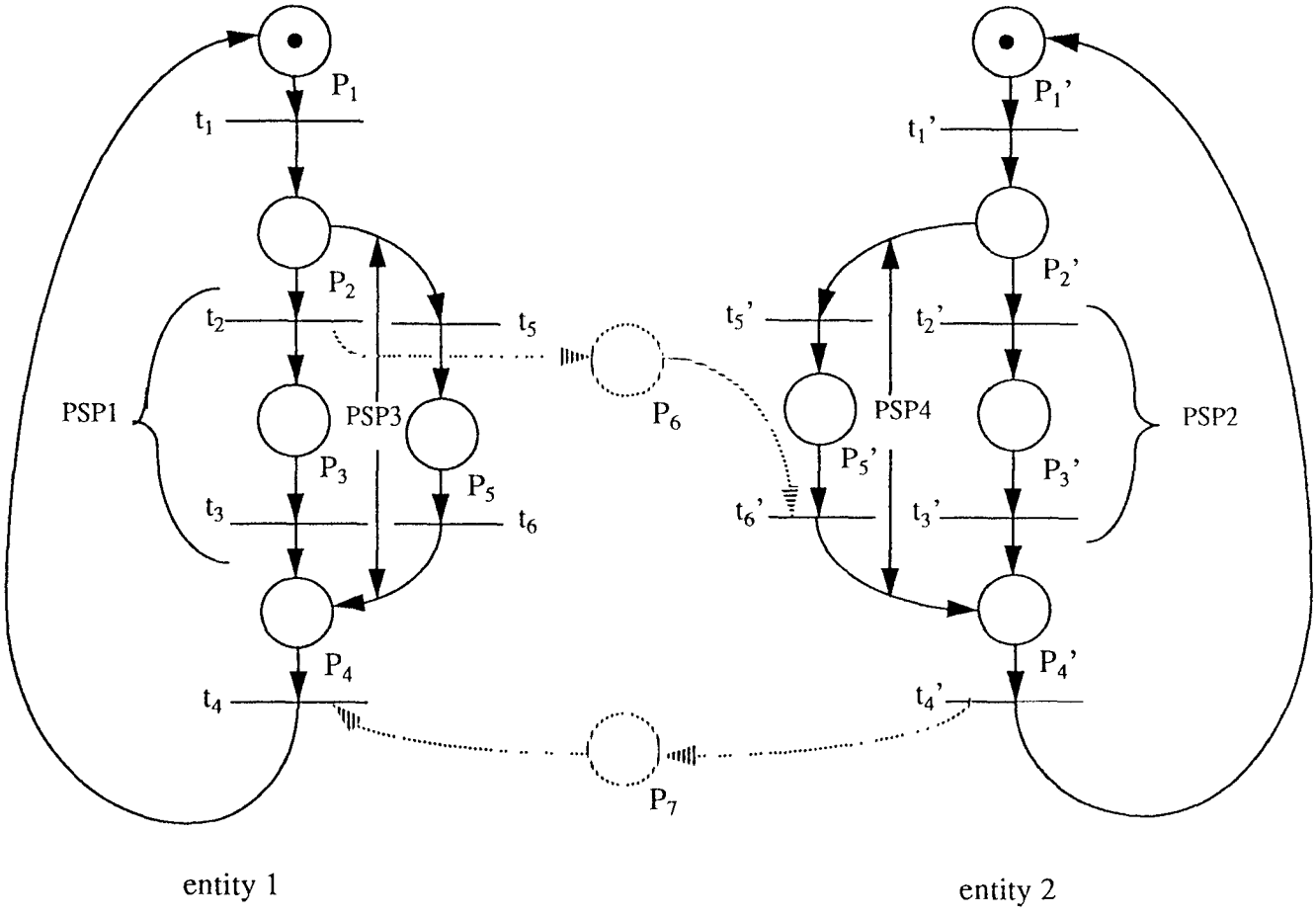


Fig 5. An example of an interactive TT generation (TT.4 rule).

Fig 5 (a). PSP1 | PSP3, PSP2 | PSP4.

PSP (t_2 P_6 t_6') is generated from t_2 to t_6'. The net is deadlock.

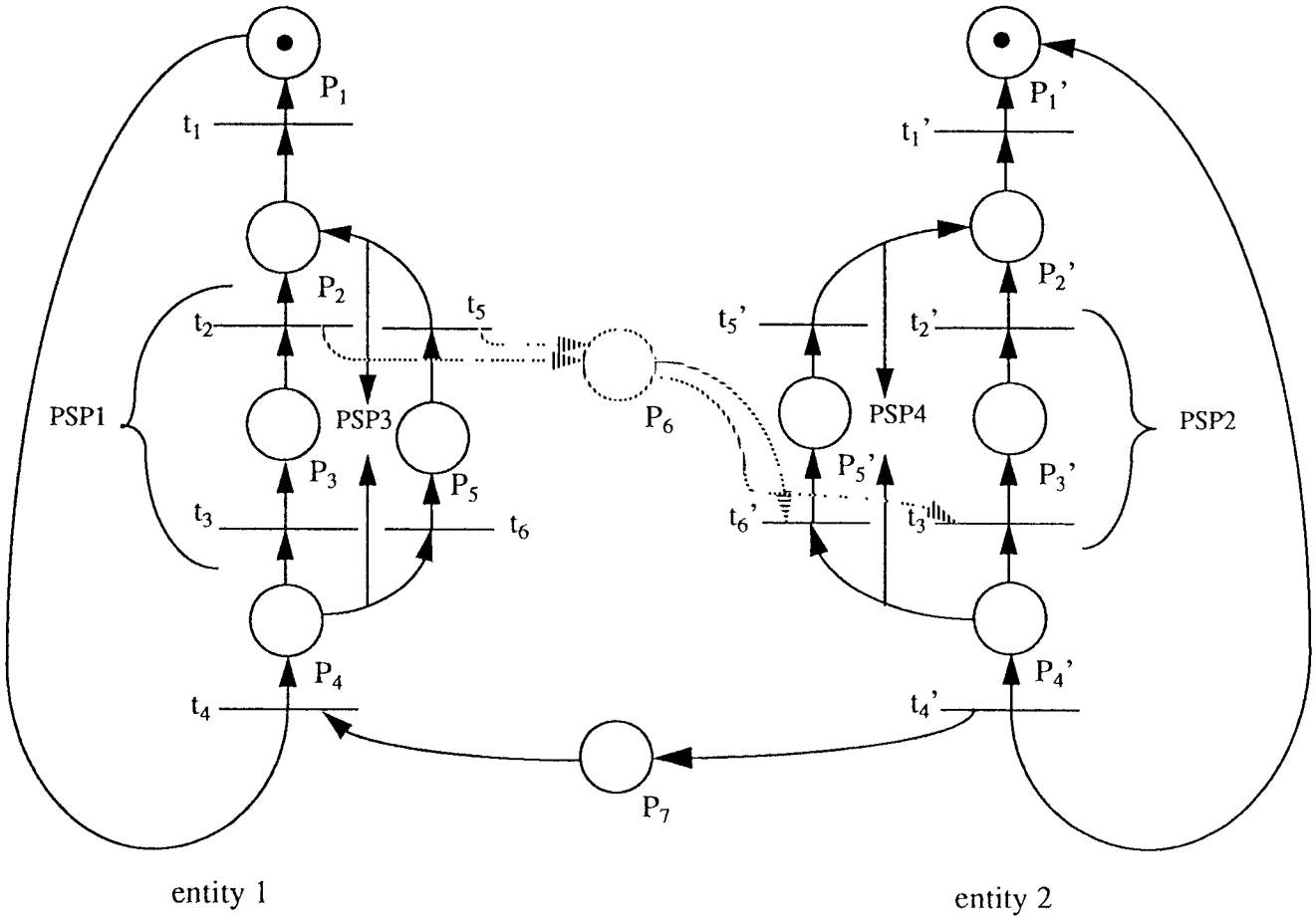


Fig 5 (b). New paths (t_5 P_6) and (P_6 t_3') are generated to fit the TT.4 rule.

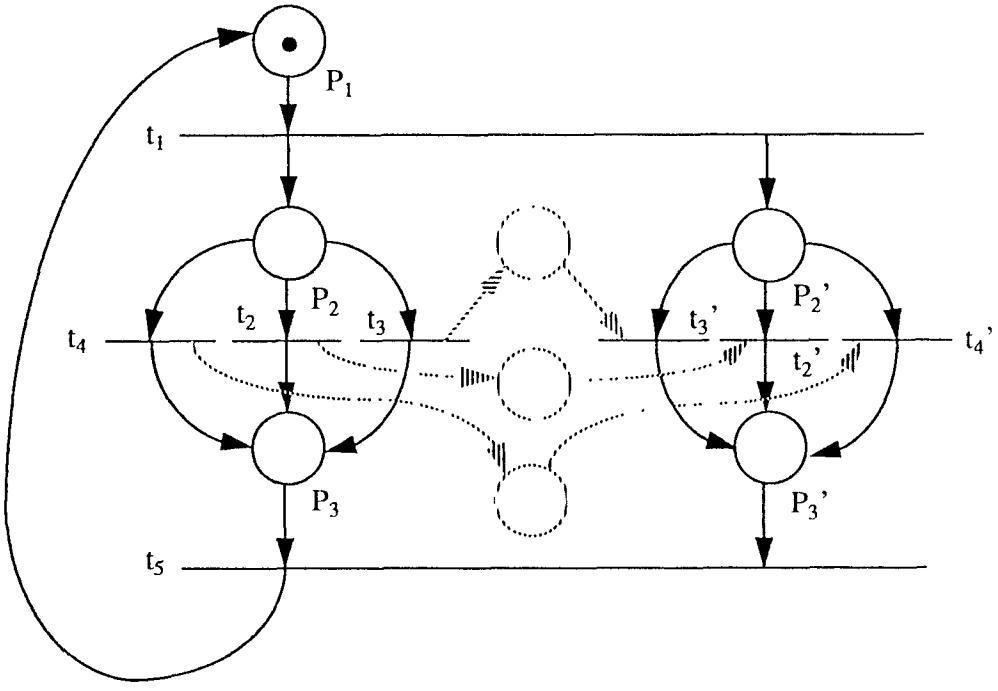


Fig 6. An example of the TT.5 rule. The three dashed paths are separated from each other.

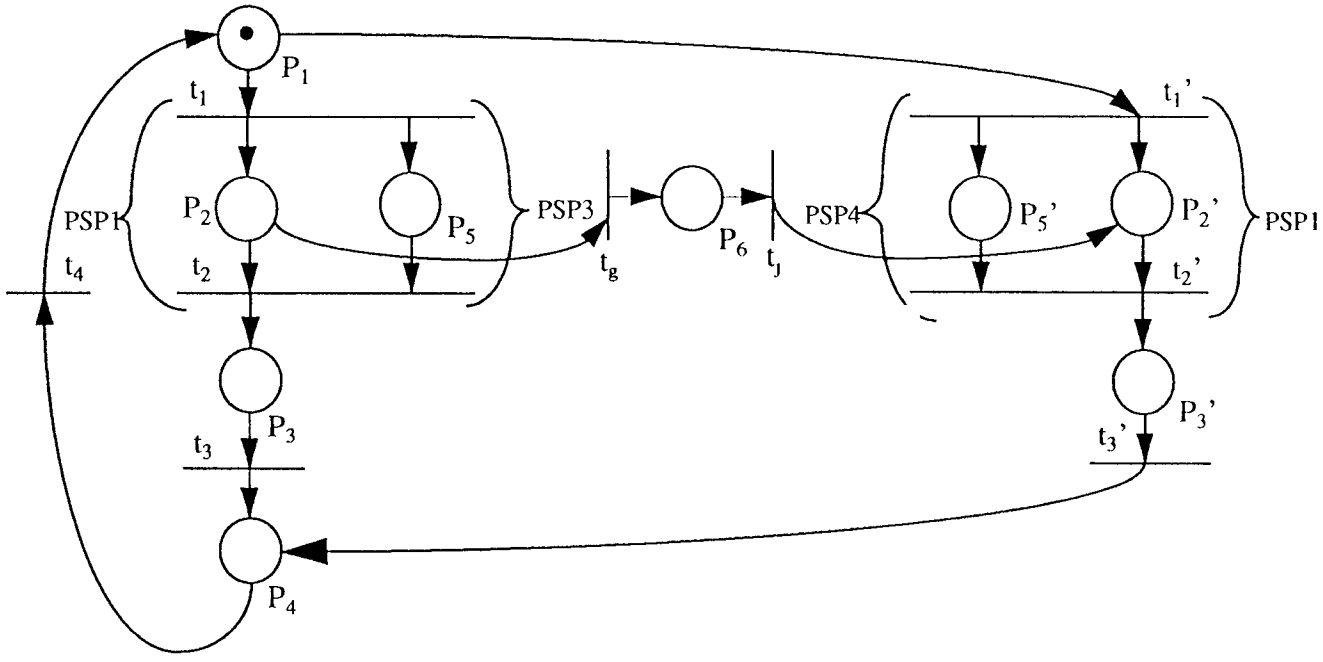


Fig. 7(a). PSP1 || PSP3, PSP2 || PSP4.
PSP (P_2 t_g P_6 t_j P_2') is generated from P_2 to P_2'.
The net has a deadlock.

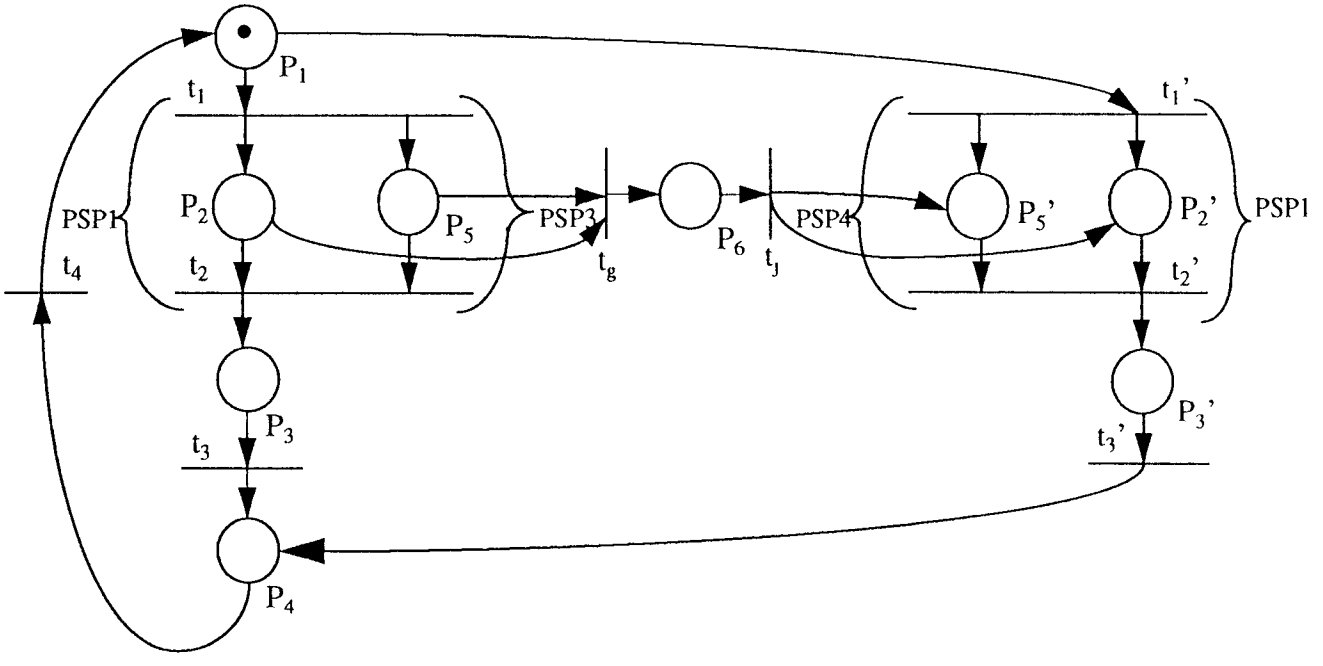
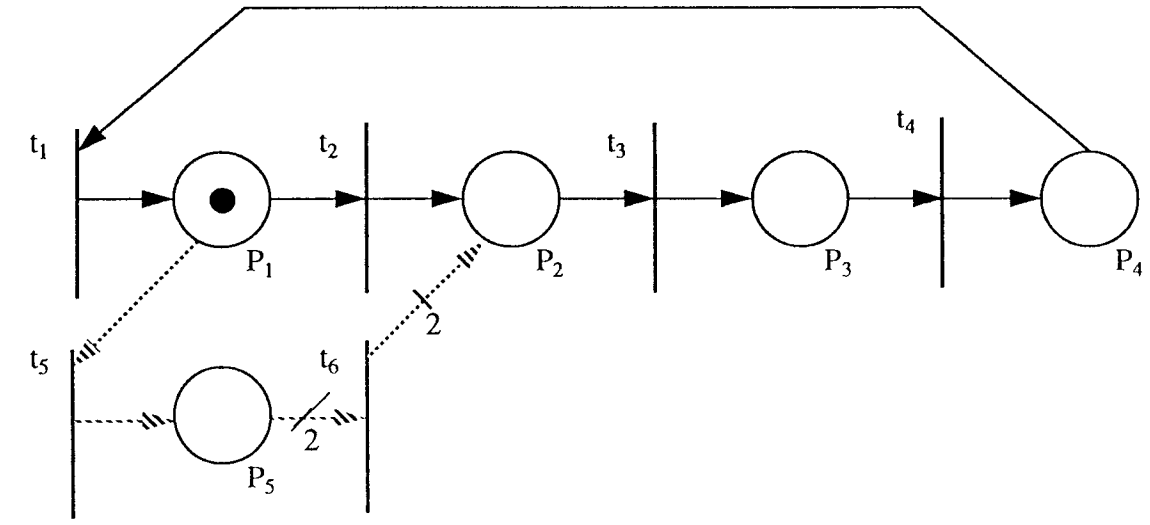
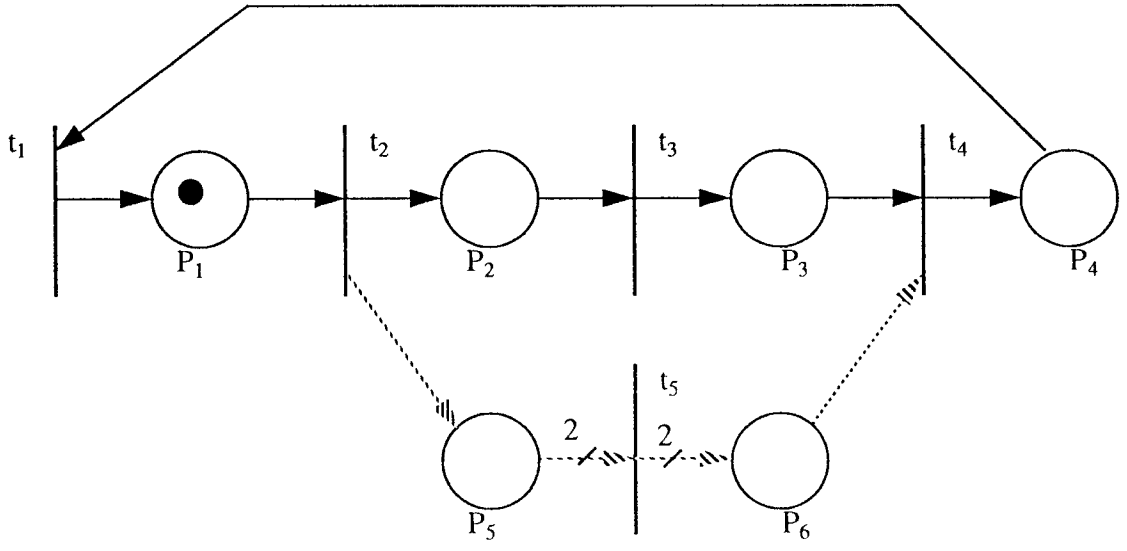


Fig 7(b). New paths (P_5 t_g) and (t_j P_5') are generated to fit the PP.2 rule

Fig 7. An example of an interactive PP generation (PP.2 rule).



(A)



(B)

Fig 8 (a) An example of PP generation which violates the Arc-ratio rule ARR.1.a. $lmr_{12}=1/1$, $lmr_{NP}=2/2$, $lmr_{NP} >_c lmr_{12}$

8 (b) An example of TT generation which violates the Arc-ratio rule ARR.1.a. $lfr_{12}=1/1$, $lfr_{NP}=2/2$, $lfr_{NP} >_c lfr_{12}$

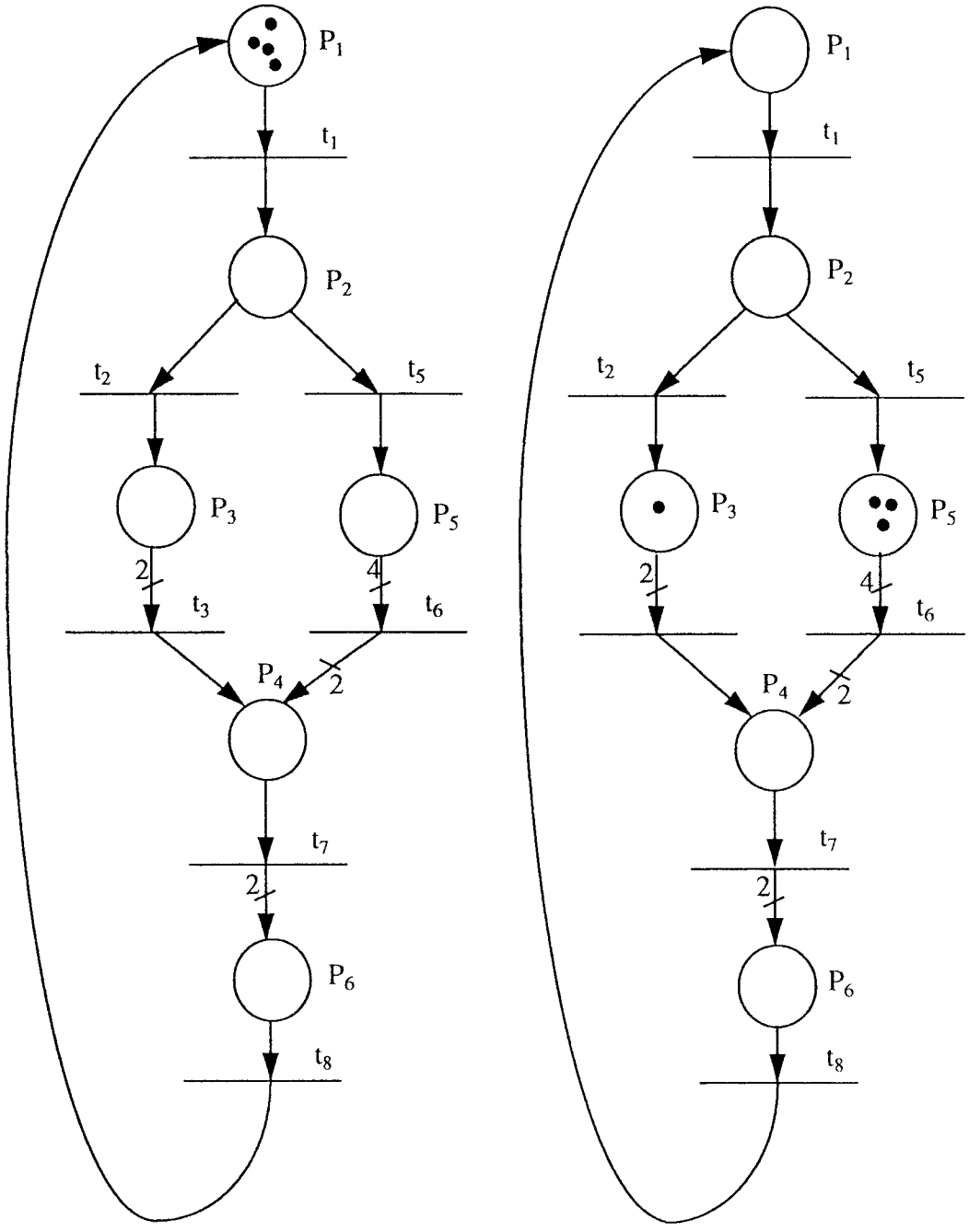


Fig.9 An example of partial flow which causes a deadlock in (b). The input ratio of path ($P_2t_2P_3t_3P_4$) is 2 and that of path ($P_2t_5P_5t_6P_4$) is 4.

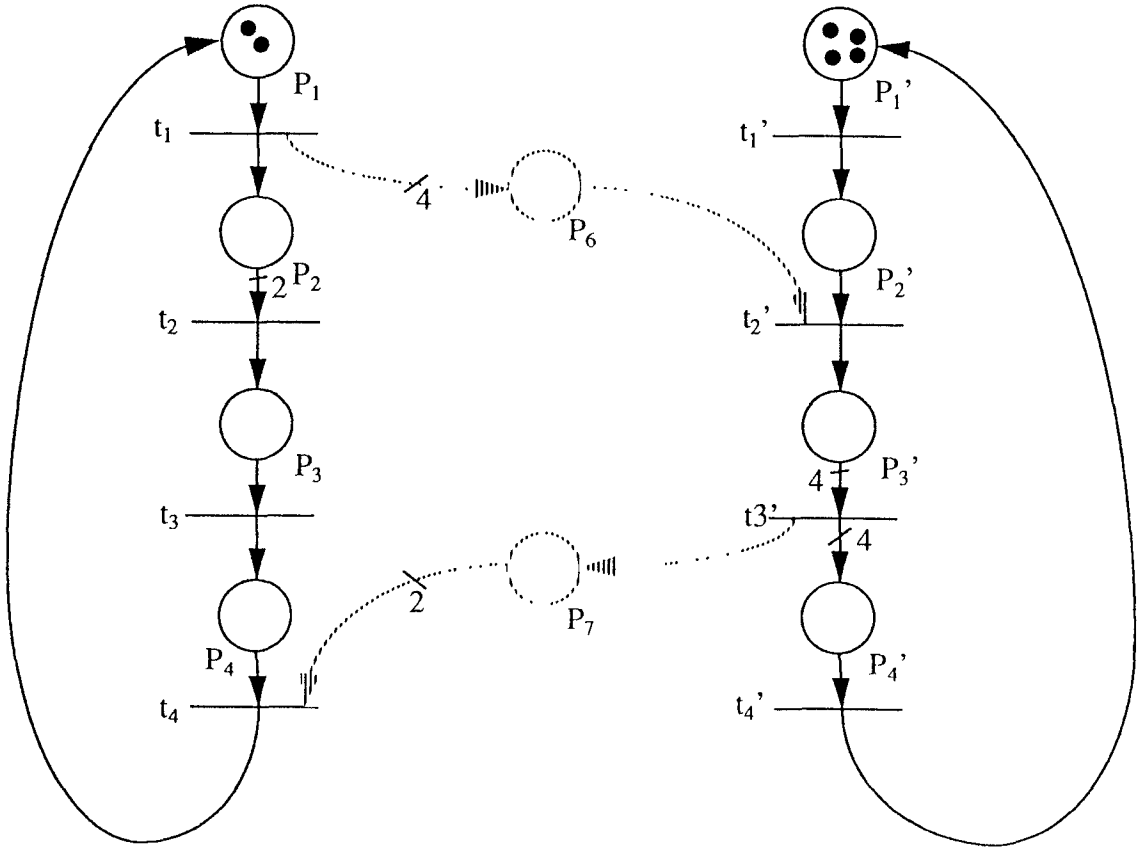


Fig 10. An example of the Arc-ratio rule ARR.2.

PN_1 consists of all solid lines and holes. There are no paths in PN_1 from t_1 to t_2'. The NP from t_1 to t_2' must be accompanied by the NP from t_3' to t_4. There are two paths from t_1 to t_4 in PN_2; both lfr=2/1.

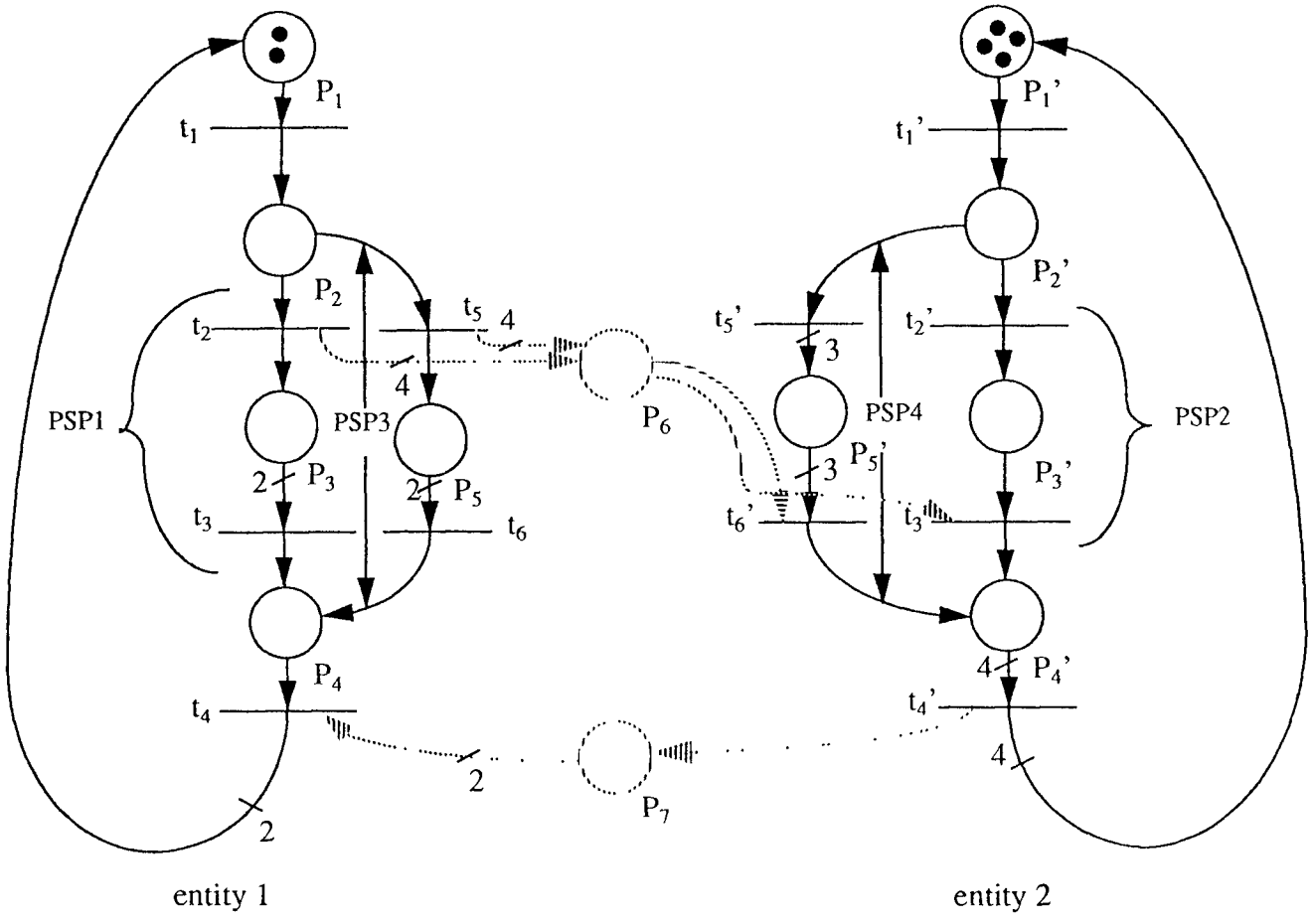


Fig 11. An example of the Arc-ratio rule ARR.1.a on top of the TT.4 rule.

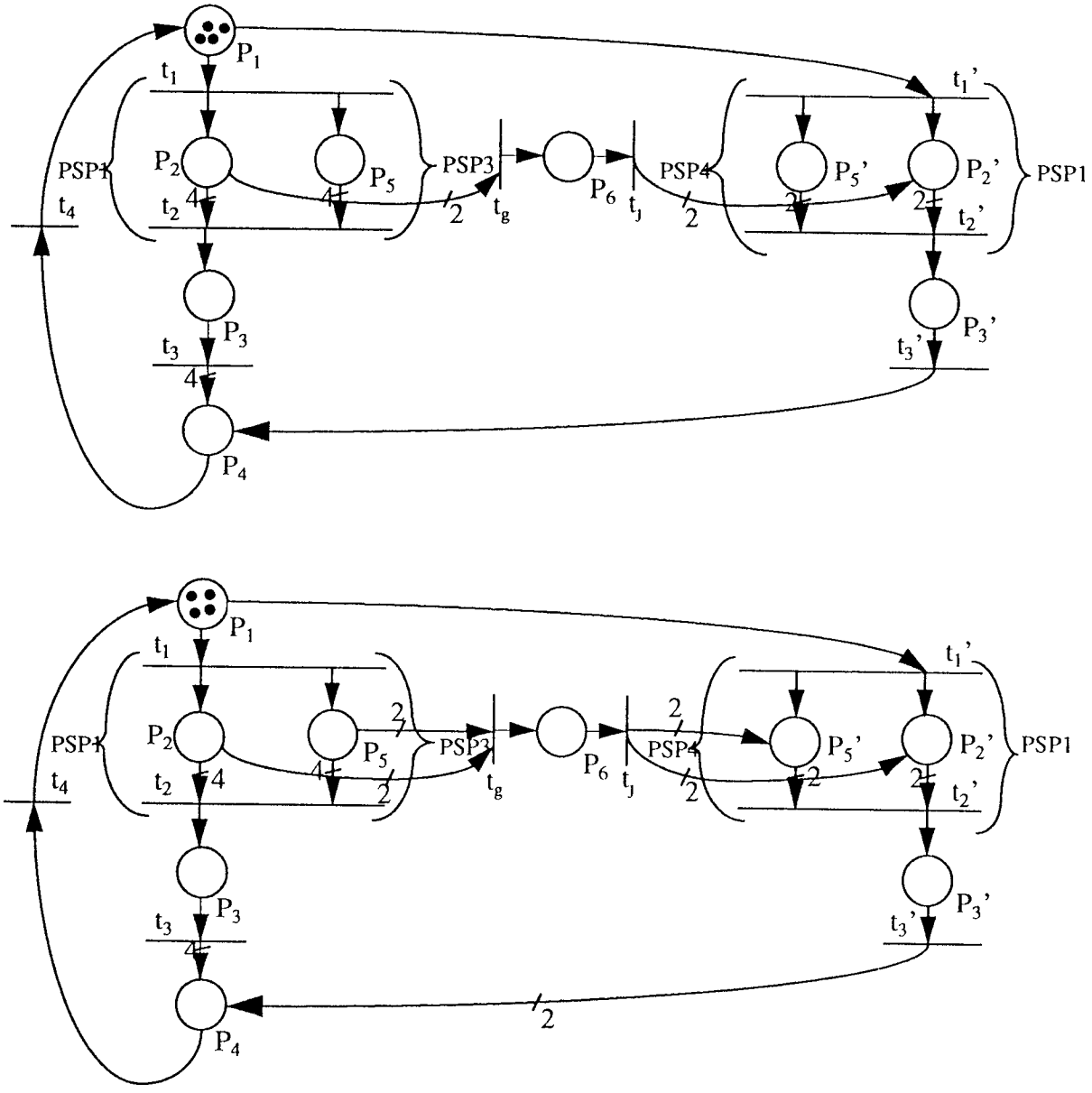


Fig 12. An example of the Arc-ratio rule ARR.1.b on top of PP.2 rule.

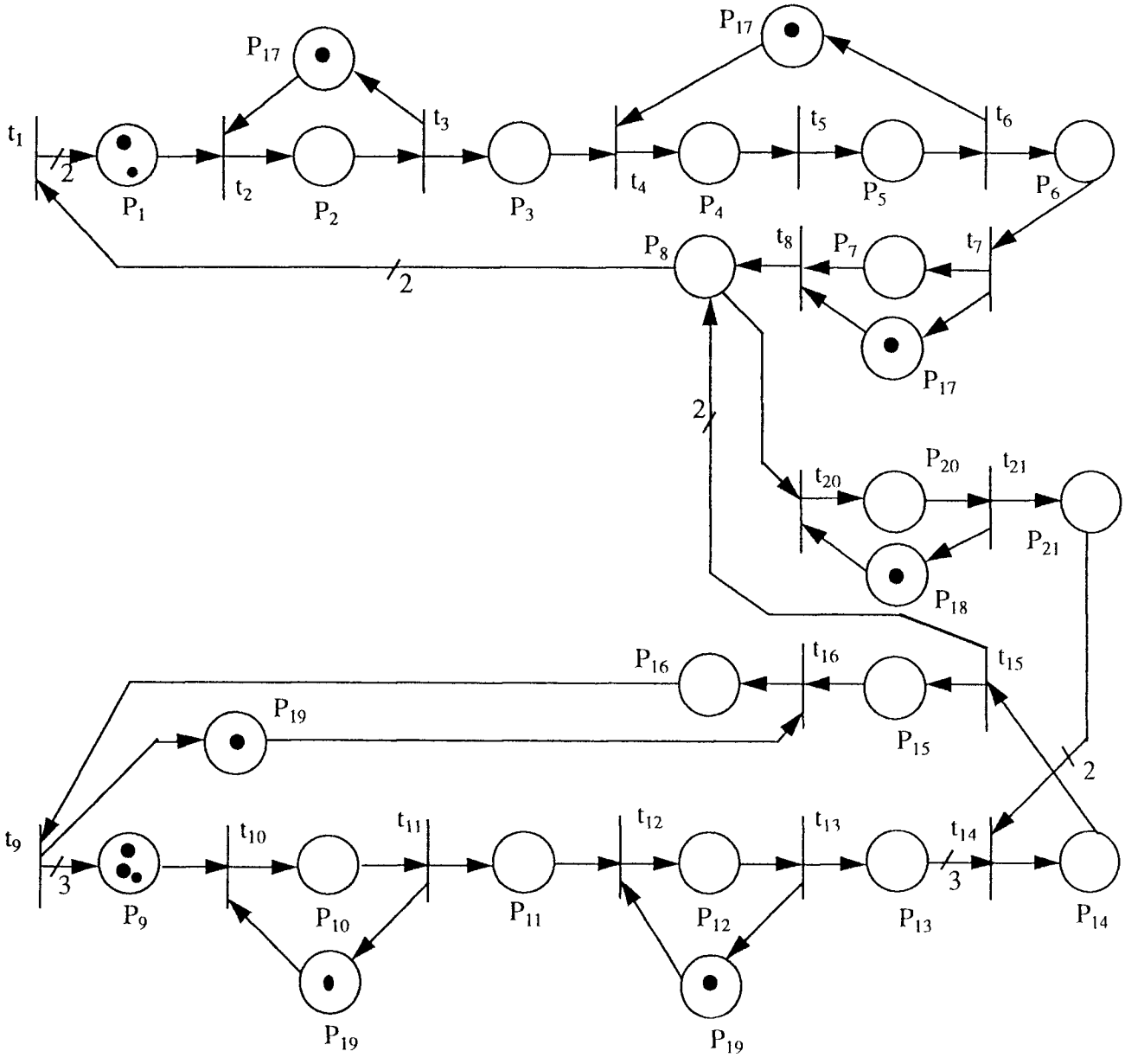
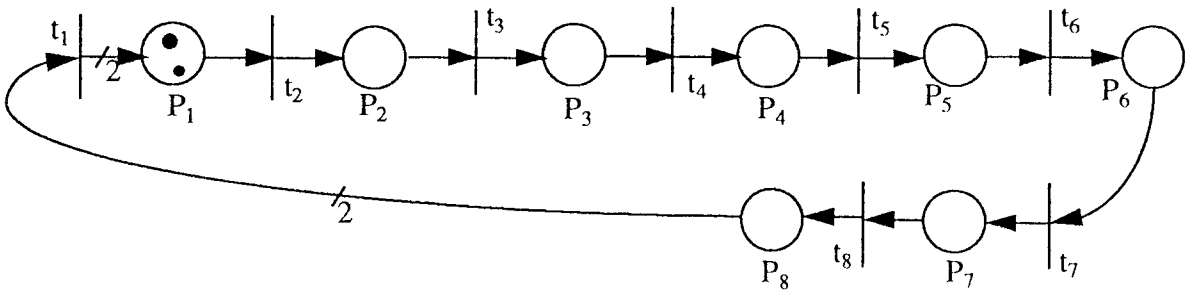


Figure 13 (a).The final net of a simple FMS



——— basic process

Fig 13(b) step 1: Basic process.

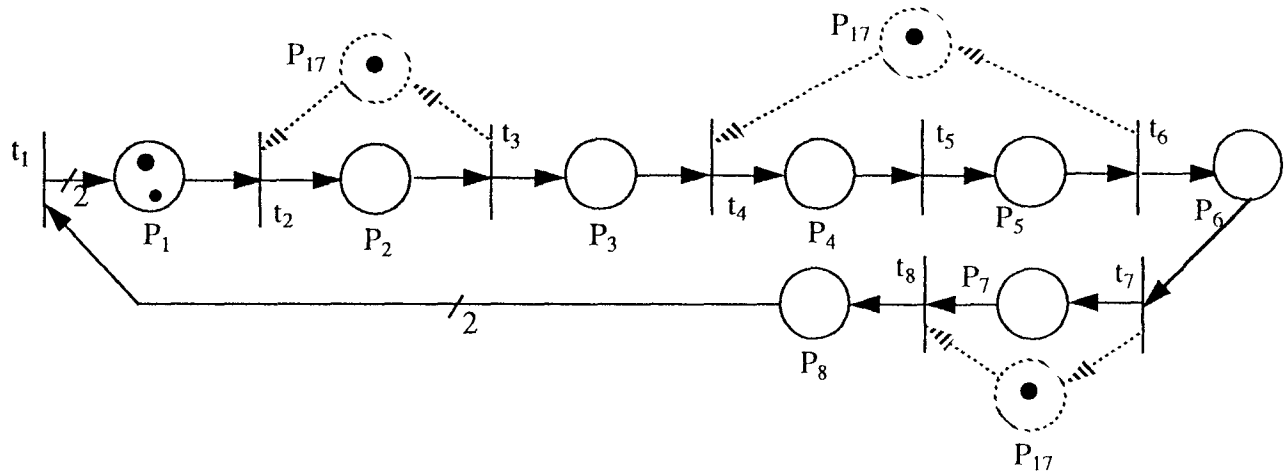


Fig 13(c) Step 2: Addition of TT-paths $(t_3P_{17}t_4)$, $(t_6P_{17}t_5)$ and $(t_7P_{17}P_7)$ using the TT.2 rule

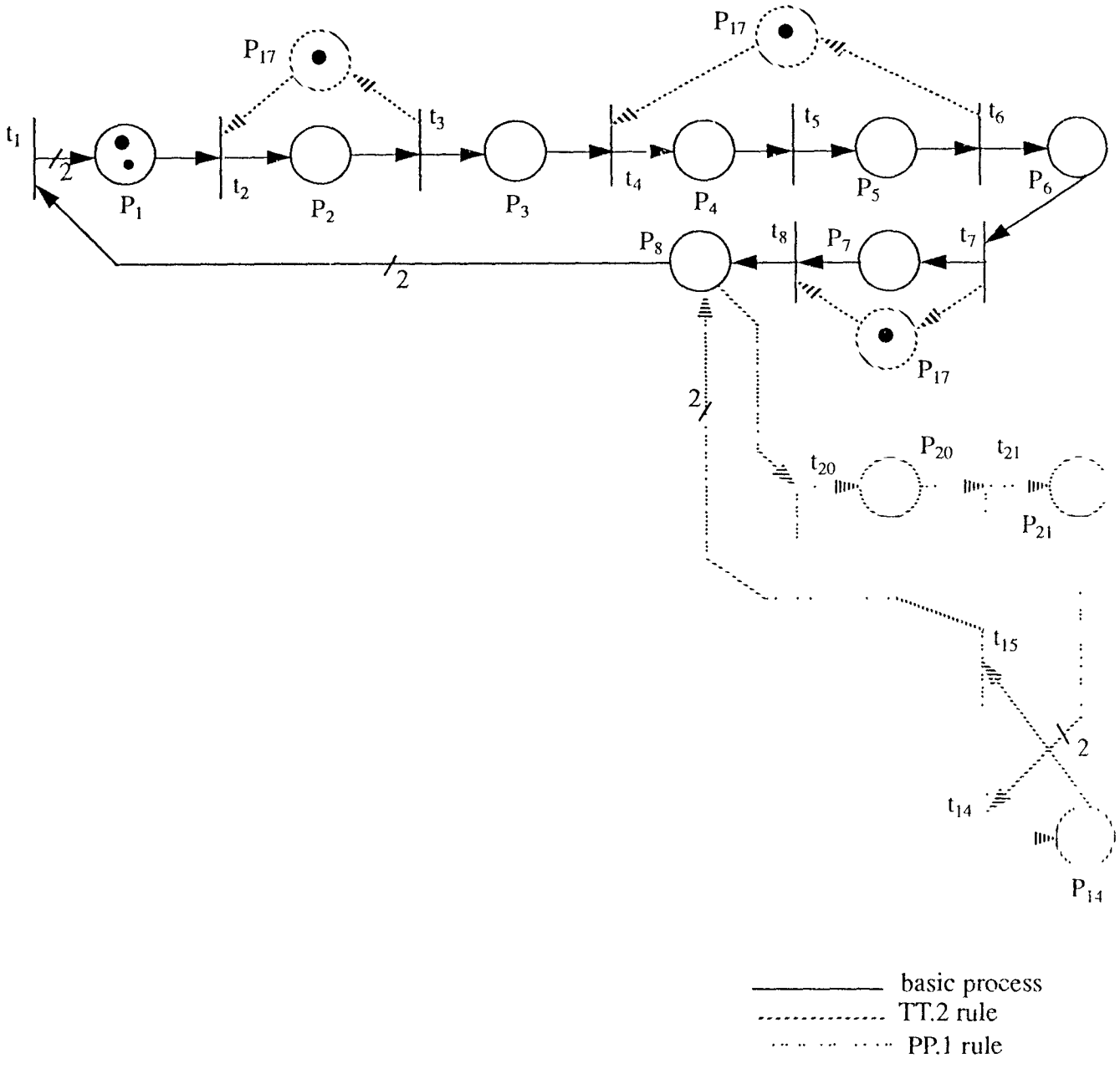


Fig 13 (d) Step 3: Addition of PP-path ($P_8t_{20}P_{20}\ t_{21}P_{21}t_{14}P_{14}t_{15}P_8$) using the PP.1 rule. $lmr_{88}=2/2$.

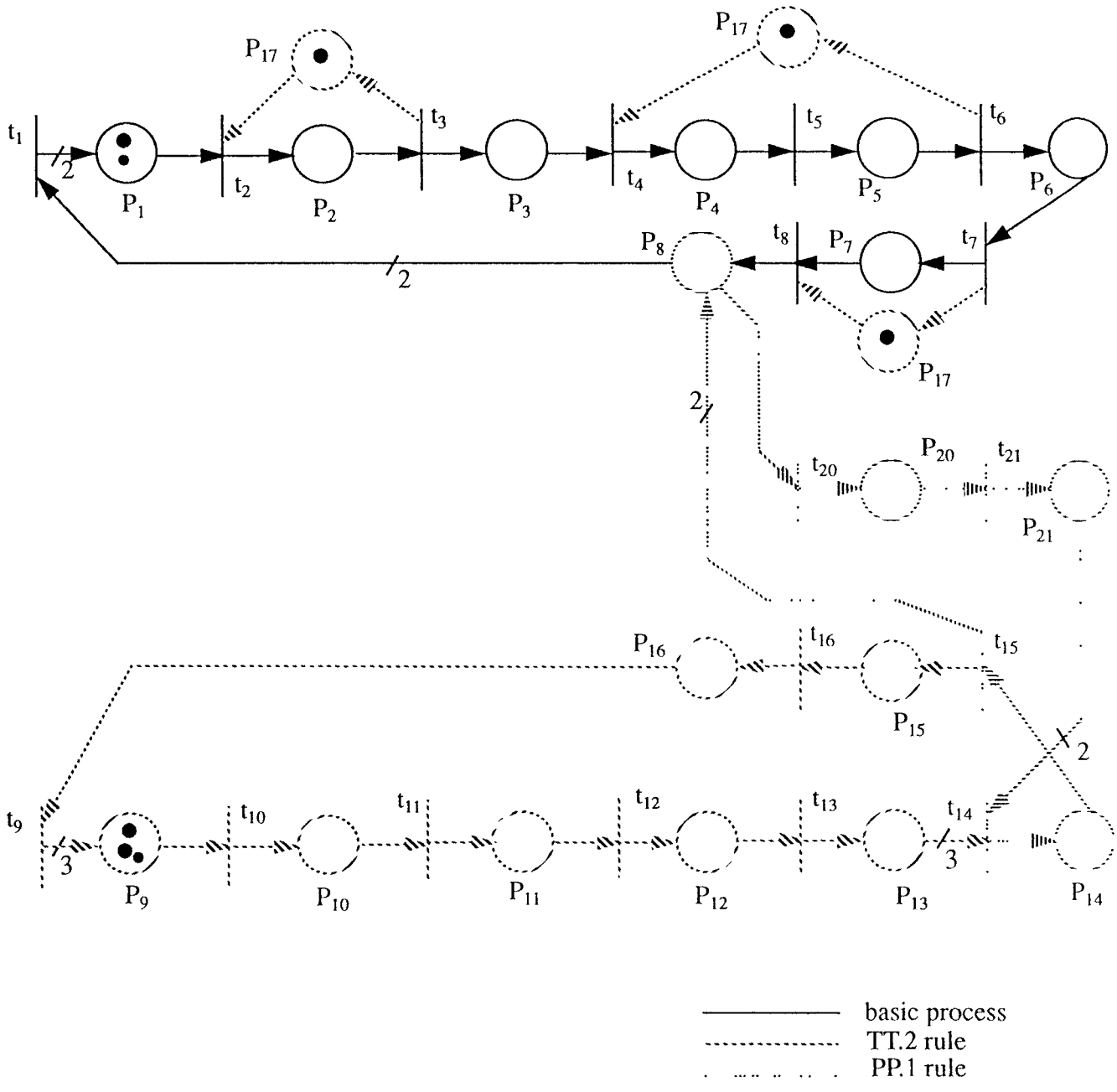


Fig 13 (e) Step 4: Addition of TT-paths :
($t_{15}P_{15}t_{16}P_{16}t_9P_9t_{10}P_{10}t_{11}P_{11}t_{12}P_{12}$) and
($t_{13}P_{13}t_{14}$) using the TT.2 rule.
lfr_{NP}=3/3 and P_9 has 3 tokens, enough to support t_{14} to fire once.

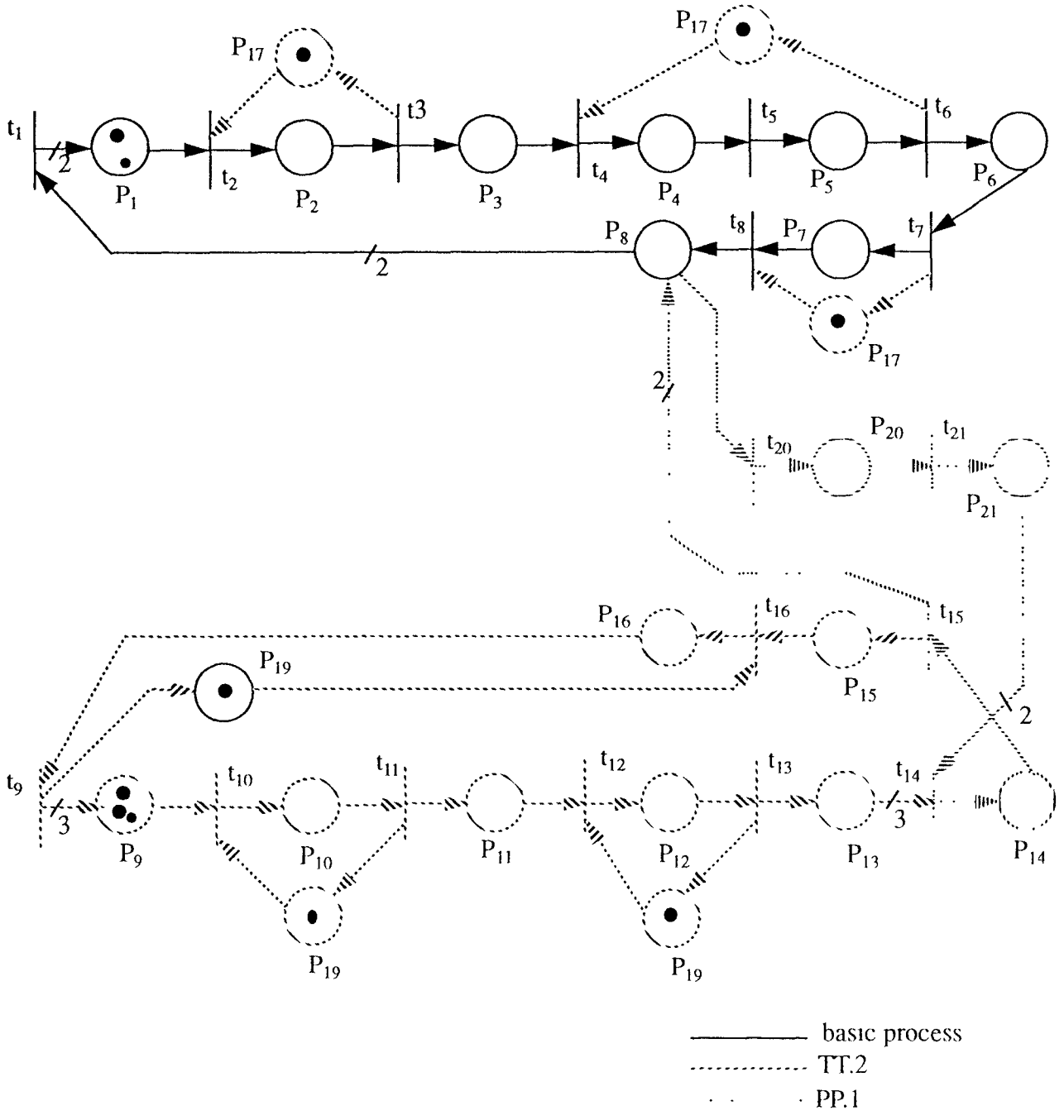


Fig 13(f). The final step. Addition of TT-paths. $(t_9P_{19}t_{16})$, $(t_{11}P_{19}t_{10})$ and $(t_{13}P_{19})_{12}$ using the TT-2 rule.

CONCLUSION

I have extended the knitting technique for ordinary Petri nets to generalized Petri nets by adding a few simple rules (Arc-Ratio rules) in addition to the connection rules. This rule also extends Koh's technique by allowing overlapping between two directed paths. Future work should be directed to apply these rules for reduction and for implementation as a computer-aided tool for designing concurrent systems.

BIBLIOGRAPHY

[BER 86] Berthelot, G., "Transformations and Decompositions of Nets," LNCS, Advances in Petri Nets , pp. 359-376, Part I, 1986.

[DAT 84] Datta, A. and Ghosh, S., "Synthesis of a Class of Deadlock-Free Petri Nets," Journal of ACM, pp. 486-506, Vol. 31, No. 3, 1984.

[KOH 90] Koh, I., and DiCesare, F., "Transformation Methods for Generalized Petri Nets and Their Applications to Flexible Manufacturing Systems," The 2nd International Conference on Computer Integrated Manufacturing, Troy, NY, MAY 1990, pp.364-371.

[MUR 89] Murata, T., "Petri Nets: properties, analysis and applications, " IEEE Proceedings, Vol. 77, No. 4, April 1989, pp. 541-580. .ip "[RAM 85]"

[RAM 85] Ramamoorthy, C.V., Dong, S.T., and Usuda, Y., "The Implementation of an Automated Protocol Synthesizer (APS) and Its Application to the X.21 Protocol," IEEE Trans. on Software Engineering, No. 9, September 1985, pp. 886-908.

[SUZ 83] Suzuki, I., and Murata, T., "A Method of Stepwise Refinement and Abstraction of Petri Nets," Journal of Computer and System Sciences 27, 1983, pp. 51-76.

[VAL 79] Valette, R., "Analysis of Petri Nets by Stepwise Refinement," Journal of Computer & System Sciences 18, 1979, pp. 35-46.

[VAI 90] Vaidyanathan, P.P., "Multirate Digital Filters, Filter Bands, Polyphase

Networks, and Approaches: A Tutorial", IEEE Proc., Jan. 1990, pp. 56-93,

[YAW 87] Yaw, Y., ''Analysis and Synthesis of Distributed Systems and Protocols," Ph.D. Dissertation, Dept. of EECS, U.C. Berkeley, 1987.

[YAW 88] Yaw, Y., "A Performance Simulation Using Time Petri Nets", Summer Computing Simulation Symposium, 88'.

[YAW 89] Yaw, Y., and Foun, F.L., ''The Algorithm of A Synthesis Technique for Concurrent Systems," 3rd International Petri Net Conf., Dec. 1989, pp. 266-276.

[CHA 91] D.Y. Chao., and D.T. Wang. Iteration Bounds of Multiple-Rate Data Flow Graphs For Concurrent Processing.