Searching Strict Minimal Siphons for SNC-Based Resource Allocation Systems

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

For a flexible manufacturing system (FMS) or Resource Allocation System (RAS) which can be decomposed into a number of synchronized choice nets interconnected by resource places, we propose to efficiently extract strict minimal siphons (SMS) in an incremental fashion rather than the traditional global approach. Only a subset of all SMS needs to be searched. The rest SMS can be found by adding and deleting common sets of places from existing ones.

Keywords: Petri nets, siphons, traps, FMS, algorithm, liveness, deadlock

1. INTRODUCTION

A siphon is a set of places that can become empty and is known to be the cause for the lack of liveness and reversibility in nets modeling the FMS resource allocation. Therefore, one supervisory control method seeks to enforce the liveness and reversibility by identifying all the minimal siphons and preventing them from becoming empty. If a siphon contains a trap, then tokens in it cannot leak out completely. A *strict minimal siphon* (*SMS*) is a minimum siphon that does not contain a trap. Once an SMS is found that can be emptied, output transitions of places in the siphon can never be fired. Hence the net is not live.

Li *et al.* [1] has established one of the best deadlock prevention techniques for FMS so far in terms of requiring the least of control arcs and nodes. Unlike others, control nodes and arcs are added for only elementary siphons. It adjusts the tokens in control places so that all redundant siphons are also controlled. As a result, it greatly reduces the number of control nodes and arcs. Their method suffers from the expensive computation of siphons since the number of siphons is theoretically exponential w.r.t the number of places.

Because deadlock occurs due to inappropriate resource sharing, SMS should contain places of resources. Yamauchi *et al*. [2] showed that the minimal siphons extraction problem (MSEP) covering a set Q of places for general Petri Nets is an NP-Complete problem. They also proposed algorithms to enumerate all minimal siphons containing Q.

Hence, efficient analysis techniques are limited to special classes of nets [3] based on local structures (input and output set of transitions or places) such as free choice nets (FC). This classification has nothing to do with siphons. FMS such as S^3PR (Systems of

Received June 8, 2005; revised August 23 & October 26, 2005; accepted November 9, 2005.
Communicated by Chin-Teng Lin.

Simple Sequential Processes) is not an FC; hence Kemper's polynomial algorithm [4] is not applicable.

For SNC, we showed [3] that if SMS exist, then the net is not live (irrespective of initial markings) and it is determined by a simple condition. Synchronized choice nets *covers well-behaved* (*live*, *bounded and reversible*) free choice nets *and yet is not included in* asymmetric choice nets [3].

An FMS model consists of a set of working processes (WP) competing for resources. A WP models a sequence of operations to manufacture a product. Circular wait for resources can bring the system into a deadlock where some working processes can never finish.

Ezpeleta *et al*. proposed a class of nets called Systems of Simple Sequential Processes with Resources (S^3PR) [5] where each WP is a state machine (SM) plus resource places. Their idea is to compute all SMS based on the approaches in [2, 6] with exponential time complexity of the given model and find the maximum number of tokens at each idle state followed by a control policy of adding control arcs and nodes with tokens. Most recent deadlock control approaches [7-9] extend Ezpeleta's work. Efficient methods to compute SMS are urgently needed.

Because only one resource is used in each job stage and the processes are modeled using state machines in S^3PR, its modeling power is limited. It cannot model iteration statements (loop) in each sequential process (SP) and the relationships of synchronization and communication in SP. At any state of a process, it cannot use multi-sets of resources. Our Synchronized Choice Net (SNC) model [3] removes this drawback.

We propose SNC-based FMS (called *System* of *Synchronized Choice Processes with Resources* (S^2CPR)) to enhance S^3PR (a subset of S^2CPR) by replacing the state machine (SM) with SNC. Each SNC with resources models a working process and its SMS can be determined in polynomial time. The rest SMS can be efficiently extracted in an incremental fashion rather than the traditional global approach. Only a subset of all SMS needs to be searched. The rest (called *compound siphons*) can be found by adding and deleting common sets of places from existing ones with search time significantly reduced.

Section 2 presents the basis to understand the paper. Section 3 presents the modular approach and shows that siphon can be synthesized by constructing handles upon a circuit. Section 4 computes all compound siphons respectively. Section 5 compares ours with other approaches. Finally section 6 concludes the paper. **For the sake of discussion continuity, all proofs are reported in Appendix 1. Examples are given for a simple S^3PR (Fig. 6, for first reading to readers new to the field) and a more complicated S^2CPR model (Fig. 7, for experienced readers and second reading).**

2. PRELIMINARIES

In this paper, we consider only strongly connected (SC) nets and we omit some basic definitions (available in [10, 11]) of Petri nets due to the space limitation.

Definition 1 A subnet $N_i = (P_i, T_i, F_i)$ of N is generated by $X = P_i \cup T_i$, if $F_i = F \cap (X \times X)$. It is an I–subnet (denoted by I) of N if $T_i = \bullet P_i$. A *strict minimal siphon* (*SMS*), denoted by S, is a minimal siphon that does not contain a trap. I_S is the I–subnet of an S.

Note $D = P(I_S)$; D is the set of places in I_S. A *siphon-trap invariant* (st-invariant) is both a siphon and trap.

We follow [3] for the definitions of *handles*, *bridges*, *AB-handles*, and *AB-bridges* where A and B can be T or P.

Definition 2 Let $N = (P, T, F)$ be a Petri net. $H_1 = [n_s n_1 n_2 \ldots n_k n_e]$ and $H_2 = [n_s n'_1 n'_2 \ldots n'_h n_e]$ are elementary directed paths, $n_i, n'_j \in P \cup T$, $i = 1, 2, \ldots, k$, $j = 1, 2, \ldots, h$. Each is called a handle in N if $n_i \neq n'_j$ $\forall i, j$ defined above; n_s and n_e are called the start and the end nodes of H_1 and H_2 respectively. Note that n_s and n_e may be identical. H_1 and H_2 are said to be *mutually* complementary. An elementary directed path $B = [n_a, n_b, \ldots, n_q]$ is a bridge from H_1 to H_2 if (1) $n_a \in H_1$, $n_q \in H_2$, $n_a \neq n_s$, $n_a \neq n_e$, $n_q \neq n_s$, $n_q \neq n_e$ and (2) $\forall n \in B$, if $n \neq n_a$, $n \neq n_q$, then $n \notin H_1$ and $n \notin H_2$. $p_1 \leftrightarrow p_2$ (p_1 and p_2 are mutually sequential) if p_1 and p_2 are on an elementary circuit. $n_1 \rightarrow n_2$ if $n_1 \leftrightarrow n_2$ and there is an elementary directed path from n_h to n_2 via n_1 where n_h is a reference node (initially marked) called a home place. The handle H to a subnet N' (similar to the handle of a tea pot) is an elementary directed path from n_s in N' to another node n_e in N'; any other node in H is not in N'.

In Fig. 1, $H_1 = [p_2\, t_4 p_4\, t_3]$ and $H_2 = [p_2\, t_2 p_3\, t_3]$, $n_s = p_2$, $n_e = t_3$. $B_{12} = [t_4 p_3]$ is a bridge from H_1 to H_2 and $B_{21} = [t_2 p_4]$ a bridge from H_2 to H_1.

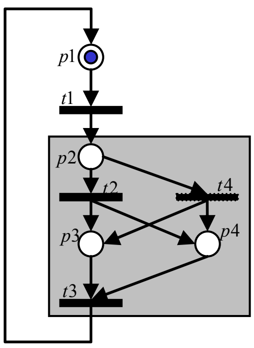

Fig. 1. An example of live & reversible SNC without inconsistent pair.

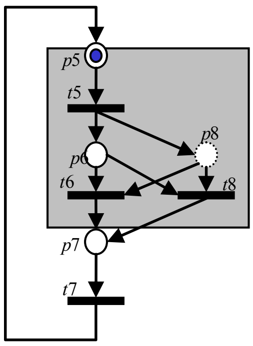

Fig. 2. Dual of the net in Fig. 1. The net is live & reversible without inconsistent pair.

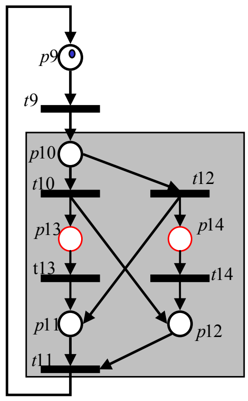

Fig. 3. Irreversible SNC with PT-inconsistent pair (p_{13}, p_{14}).

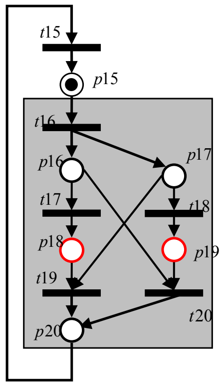

Fig. 4. Dual of the net in Fig. 9. The SNC is not live with TP-inconsistent pair (p_{18}, p_{19}).

Definition 3 H_1 and H_2 were defined in Definition 2.

1) Let $\Gamma_1 = [n_s n_1 n_2 \ldots n_k p_1]$, $\Gamma_2 = [n_s n'_1 n'_2 \ldots n'_h p_2]$ and $\Gamma_1 \cap \Gamma_2$ ($\Gamma_1 \cup \Gamma_2$) denotes the intersection (union) of two graphical objects Γ_1 and Γ_2. n_s is a start node of (p_1, p_2) if $\Gamma_1 \cap \Gamma_2 = \{n_s\}$ and the *nearest start node* of (p_1, p_2); *i.e.*, $n_s^{1,2} = n_s$ if Γ_1 does not contain other start nodes of (p_1, p_2). $n_e^{1,2}$ can be defined in a dual fashion. Let $\Gamma_3 = [p_1 n_s n_1 n_2 \ldots n_k n_e]$ and $\Gamma_4 = [p_2 n'_1 n'_2 \ldots n'_h n_e]$. n_e is an end node of (p_1, p_2) if $\Gamma_3 \cap \Gamma_4 = \{n_e\}$ the *nearest end node* of (p_1, p_2); *i.e.*, $n_e^{1,2} = n_e$ if Γ_3 does not contain other end nodes of (p_1, p_2). $N_s^{1,2}$ ($N_e^{1,2}$) is the set of all such $n_s^{1,2}$ ($n_e^{1,2}$).
2) Let $\Psi = H_1 \cup H_2$ denote the union of two graphical objects H_1 and H_2. H_1 (H_2) is a

prime handle to H_2 (H_1), if there are no bridges B between H_1 and H_2 and Ψ is defined to be a *first-order structure* (FOS).

3) If B_{12} (B_{21}) is the only bridge from H_1 to H_2 (H_2 to H_1), then $\varphi = H_1 \cup H_2 \cup B_{12} \cup B_{21}$ is defined to be a *second-order structure* (SOS) (see the shaded area in Figs. 1-4).
4) (p_1, p_2) is called a TP-inconsistent pair (TPIP) of places if $\exists n_s^{1,2} \in T$ and $\exists n_e^{1,2} \in P$. (p_1, p_2) is called a PT-inconsistent pair (PTIP) of places if $\exists n_s^{1,2} \in P$ and $\exists n_e^{1,2} \in T$.
5) Let ω be a first-order structure (FOS) (or SOS, handle, bridge, path), if its $n_s \in T$, $n_e \in P$, then ω is called a *TP-ω*. *PT-ω*, *TT-ω* and *PP-ω* can be defined similarly. If n_s and n_e are of the same type; *i.e.*, both are transitions or places, then ω is said to be *symmetrical*; otherwise it is *asymmetrical*.
6) A strongly connected net is SNC (Synchronized Choice net, denoted N^{cc}) if it satisfies the two requirements $R1$ and $R2$ where $R1$: ($R2$:) every PT- (TP-) handle to a certain circuit has a TP- (PT-) bridge from its complementary TP-handle to itself.

In Fig. 3, $N_s^{12,11} = \{t_{10}, t_{12}\}$ and $N_e^{12,11} = \{t_{11}\}$; p_{10} is not an $n_s^{12,11}$ because for each path from p_{10} to p_{11} or p_{12}, it contains other start nodes t_{10} or t_{12}. (p_{18}, p_{19}) in Fig. 4 is a TP-inconsistent pair because $n_s^{18,19} = t_{16} \in T$ and $n_e^{18,19} = p_{20} \in P$.

[$p_2\ t_2\ p_3$] and [$p_2\ t_4\ p_3$] in Fig. 1 are two *prime handles*; $n_s = p_2$ and $n_e = p_3$. Note that there are no bridges interconnecting them; hence, they together form a first-order structure. Since $n_s \in P$, $n_e \in P$, it is symmetrical.

Figs. 1-4 show examples of SNC where the shaded areas cover the structures involving $R1$ or $R2$. In Fig. 1, the only two PT-handles $H_1 = [p_2\ t_4 p_4\ t_3]$ and $H_2 = [p_2\ t_2 p_3\ t_3]$ start from the same place p_2 but they join at a transition t_3. To satisfy $R1$, there is a TP-bridge $B_{12} = [t_4 p_3]$ from H_1 to H_2 and a TP-bridge $B_{21} = [t_2 p_4]$ from H_2 to H_1. In Fig. 2, the only two TP-handles $H_1 = [t_5 p_6\ t_6\ p_7]$ and $H_2 = [t_5 p_8\ t_8 p_7]$ start from the same transition t_5 but they join at a place p_7. To satisfy $R2$, there is a PT-bridge $B_{12} = [p_6\ t_8]$ from H_1 to H_2 and a PT-bridge $B_{21} = [p_8\ t_6]$ from H_2 to H_1.

Note that any pair of places (excluding n_s and n_e) in an AFOS (asymmetrical first-order structure) is also inconsistent. This leads [3] to an integrated algorithm to construct an S-Matrix to detect SNC and liveness for an arbitrary net.

An SNC contains S if and only if it contains TP-inconsistent pairs of places [3]. It is interesting because it is bounded and the condition for liveness is so simple that there exists an integrated algorithm to check whether a net is an SNC and, if it is, whether it is live.

In Fig. 5, $\{p_{14}, p_{15}, p_{16}, p_{17}, p_{20}\}$ is an SMS. We can add TT-handle [$t_{15}\ p_{13}\ t_{16}$] and delete TT-handle [$t_{15}\ p_{15}\ t_{16}$] to get another S. Thus, for a single set of these two TT-handles, there are two SMS. There are two such sets in Fig. 5; hence there are totally 4 SMS. If there are k such sets, there will be 2^k SMS for just a single TP-inconsistent pair of places. Thus the number of SMS, even for SNC, grows exponentially. This implies TP-inconsistent pair of places provides a more concise way to characterize an SNC than SMS.

This triggers the idea of decomposition. If a net can be decomposed into a number of SNC components with no TP-inconsistent pair of places, then SMS occurs due to the merging of these components. We do not have to search them in each component greatly improving the search efficiency. Although it suffers from the efforts required for

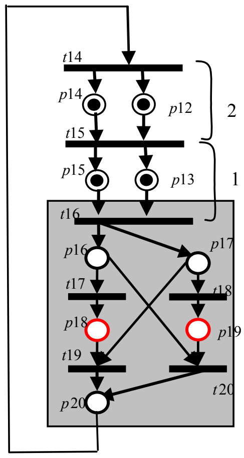

Fig. 5. $\{p_{14}, p_{15}, p_{16}, p_{17}, p_{20}\}$ is an SMS. We can add TT-handle $[t_{15}\ p_{13}\ t_{16}]$ and delete TT-handle $[t_{15}\ p_{15}\ t_{16}]$ to get an- other SMS.

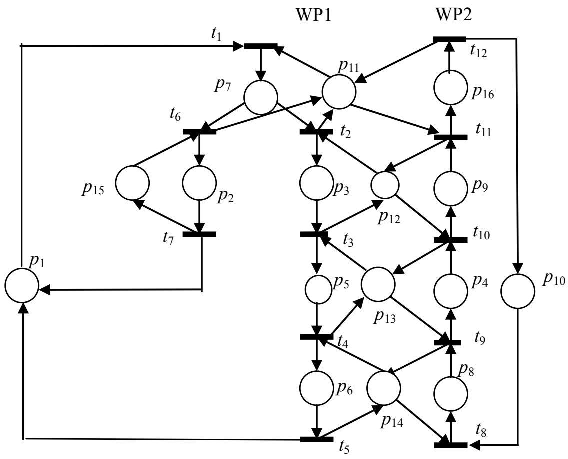

Fig. 6 [5]. A simple example of S^2CPR (A system of synchronized choice processes with resources), also an S^3PR.

decomposition, it is still better than the exponential time complexity as long as the time for decomposition is polynomial. Further, a WP can be easily identified.

Thus, the modular approach (see section 3) excels over all other approaches [2, 4, 6, 11] where all nodes and arcs in the net must be traced. We now apply this to SNC-based RAS defined below.

Definition 4 A *Synchronized Choice Process* (*SCP*) is a net $N = (P \cup \{p^0\}, T, F)$ where: (1) $P \neq \emptyset$, $p^0 \notin P$ (p^0 is called the process idle or initial or final state); (2) N is an SNC; and (3) every circuit of N contains the place p^0.

Note that SCP is different than the SP in [5] by removing the restriction of state machine. SCPR and S^2CPR below are defined similar to that for S^2PR (Simple Sequential Processes with Resources) and S^3PR.

Definition 5 A *Synchronized Choice Process with Resources* (*SCPR*), also called a working processes (WP), is a net $N = (P \cup \{p^0\} \cup R, T, F)$ so that (1) The subnet generated by $X = P \cup \{p^0\} \cup T$ is an SCP; (2) $R \neq \emptyset$ and $P \cup \{p^0\} \cap R = \emptyset$; (3) $\forall p \in P$, $\forall t \in \bullet p$, $\forall t' \in p\bullet$, $\exists r_p \in R$, $\bullet t \cap R = t'\bullet \cap R = \{r_p\}$; (4) The two following statements are verified: (a) $\forall r \in R$, $\bullet\bullet r \cap P = r\bullet\bullet \cap P \neq \emptyset$; (b) $\forall r \in R$, $\bullet r \cap r\bullet = \emptyset$. (5) $\bullet\bullet(p^0) \cap R = (p^0)\bullet\bullet \cap R = \emptyset$. $\forall p \in P \cup \{p^0\}$, p is called a *state place*. r is called a *resource place*. $H(r) = \bullet\bullet r \cap P$ denotes the set of holders of r (states that use r). $\boldsymbol{\rho(r) = \{r\} \cup H(r)}$**.** $\boldsymbol{r_1}$ **is a *nearby* of** $\boldsymbol{r_2}$ **if** $\boldsymbol{r_1 \in r_2\bullet\bullet}$**.**

Definition 6 A System of *SCPR*, *S^2CPR* is defined recursively as follows: (1) An SCPR is defined as an S^2CPR; (2) Let $N_i = (P_i \cup P_i^0 \cup R_i, T_i, F_i)$, $i \in \{1, 2\}$ be two S^2CPR so that $(P_1 \cup P_1^0) \cap (P_2 \cup P_2^0) = \emptyset$. $R_1 \cap R_2 = P_C$ $(\neq \emptyset)$ and $T_1 \cap T_2 = \emptyset$. The net $N = (P \cup \{p^0\} \cup R, T, F)$ resulting from the composition of N_1 and N_2 via P_C defined as follows: (1) $P = P_1 \cup P_2$; (2) $P^0 = P_1^0 \cup P_2^0$; (3) $R = R_1 \cup R_2$; (4) $T = T_1 \cup T_2$ and (5) $F = F_1 \cup F_2$ is also an S^2CPR.

Note that an S^2CPR differs from S^3PR only in that each SCP is replaced by an SM. The following lemma is easy to see.

Lemma 1 (1) $\rho(r)$ is both a trap and a siphon; *i.e.*, an st-invariant. (2) $\rho(R) = \rho(r_1) \cup \rho(r_2) \cup \rho(r_3) \cup \ldots \cup \rho(r_k)$ is an st-invariant. (3) Any minimal trap in $\rho(R)$ contains a $\rho(r_i)$, $1 \le i \le k$. (4) $\rho(r_i) \cap \rho(r_j) = \emptyset$, $\forall i, j, i \ne j, 1 \le i \le k, 1 \le j \le k$.

Examples of S^3PR and S^2CPR are shown in Figs. 6 and 7 with $\rho(p_{11}) = \{p_{16}, p_7\}$ and $\rho(R2) = \{p_2, p_4, p_8, p_{12}, p_{17}, p_{28}, R_2\}$ respectively.

3. MODULAR APPROACH AND SYNTHESIS OF SMS

Although SNC does not cover all kinds of resource sharing, it can serve as the backbone (*i.e.*, SCP) of a net. If any SCP has no TP-inconsistent pair of places, it has no SMS. Deadlock occurs when all resources in a circuit are used up since processes mutually waiting for them indefinitely. Thus, SMS must have something to do with circuits with resources as implied by the following lemma:

Lemma 2 [1] I_S is strongly connected (SC) and has an elementary circuit.

Upon the circuit, we construct handles to form the I_S of an SMS based on Lemma 3.

Lemma 3 (1) A subnet N' is the I of a minimal siphon *iff* each handle in N' is a PP- or TP- or *virtual* PT-handle (virtual means containing only two nodes) and there are none of PP-, TP-, and virtual PT-handles to N'. (2) S is an SMS *iff* there is a nonvirtual (more than two nodes) PT-handle to I.

Example: In Fig. 6, first find a circuit $c = [p_{13}\ t_4\ p_{14}\ t_9\ p_{13}]$. Second, add TP-handles $[t_4\ p_6\ t_5\ p_{14}]$ and $[t_9\ p_4\ t_{10}\ p_{13}]$ to get $S_1 = \{p_4, p_6, p_{13}, p_{14}\}$ with two nonvirtual PT-handles $[p_{14}\ t_8\ p_8\ t_9]$ and $[p_{13}\ t_3\ p_5\ t_4]$ to its c. Similarly $S_2 = \{p_5, p_9, p_{12}, p_{13}\}$ on $c = [p_{12}\ t_{10}\ p_{13}\ t_3\ p_{12}]$. However, the siphon $\{p_3, p_7, p_{11}, p_{12}, p_{16}\}$ on $c = [p_{11}\ t_{11}\ p_{12}\ t_2\ p_{11}]$ is not minimal since it contains $\rho(p_{11})$. Note that part ($[t_{11}\ p_{12}\ t_2]$) of the c become a TT-handle to $I(\rho(p_{11}))$. In Fig. 7, the siphon constructed based on $c = [p_{20}\ t_{19}\ p_{25}\ t_7\ p_{20}]$ is not minimal since $[t_{19}\ p_{25}\ t_7]$ is a TT-handle to $I(\rho(R1))$.

In Fig. 7, first find a circuit $c = [R2\ t_8\ M3\ t_{18}\ R2]$. Second add TP-handles $[t_{18}\ p_{16}\ t_{19}\ M3]$ and $[t_8\ p_{12}\ t_9\ R2]$; PP-handles $[R2\ t_{11}\ p_2\ t_{12}\ R2]$, $[R2\ t_{13}\ p_4\ t_{14}\ R2]$, and $[R2\ t_3\ p_8\ t_4\ R2]$ to get $S_{18} = \{p_2, p_4, p_8, p_{12}, p_{16}, R2, M3\}$ with a nonvirtual PT-handle $[R2\ t_{17} p_{17}\ t_{18}]$ to its c. Such a procedure to form SMS from a circuit is called *handle-construction* or synthesis.

Note that if there were PP-, TP-, and virtual PT-handles to N'; *i.e.*, N' was not maximal, then it would not be a siphon. If I_S contains only one resource place r, it cannot be strict since $\rho(r)$ is both a trap and a siphon. As an example, $\rho(M4) = \{p_{26}, p_{18}, p_{13}\}$.

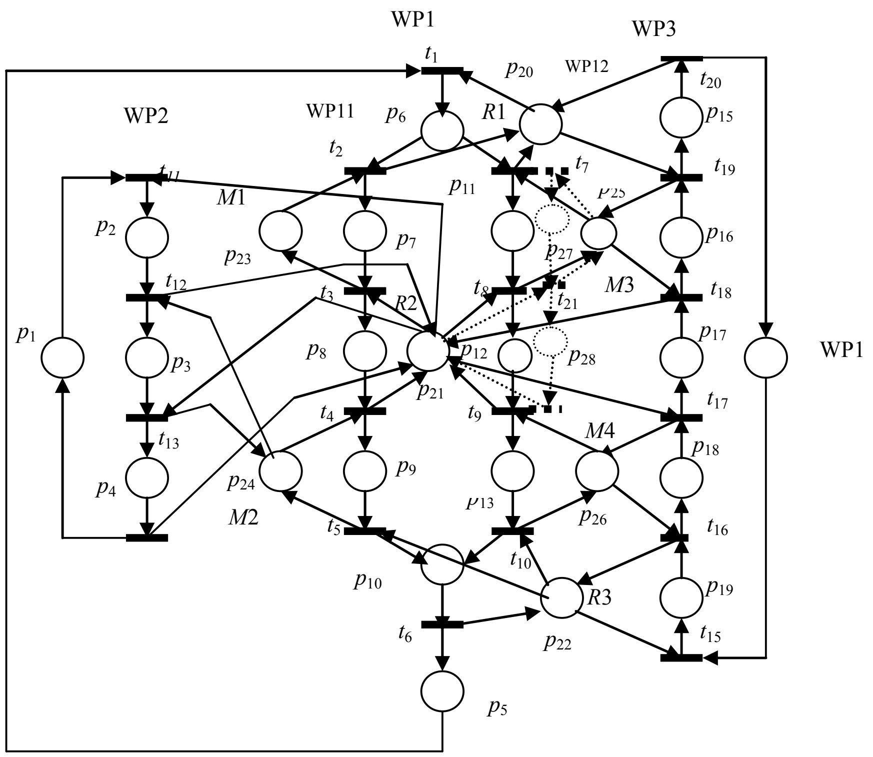

Fig. 7. An example of S^2CPR (A System of Synchronized Choices Processes with Resources). It differs from the well-known S^3PR example in [5] by adding the dashed path [$t_7\ p_{27}\ t_{21}\ p_{28}\ t_9$]. ($R1$, $R2$, $R3$) and ($M1$-$M4$) correspond to ($\{p_{20}, p_{21}, p_{22}\}$ and $\{p_{23}$- $p_{26}\}$) respectively.

The algorithm of finding all S is proposed as follows:

Algorithm 1 Search of *SMS*

Step 1: Add a new nearby r' to an already-traced – *preferably* last-traced – r. If no r' can be added, break. If $I(\rho(r'))$ contains a PT-handle, each strongly connected subnet of $I(\rho(r'))$ with no PT-handles in it is a new I_S.

Step 2: Find a new elementary circuit c' that contains r' and all r'' in c' have been traced. If "not found", go to Step 1.

Step 3: Add all PP- (with no new r) followed by all TP-handles and virtual PT-handles to c'. Denote the resulting net by I' (I-subnet) where the set of places $P(I')$ forms a new S^n.

Step 4: Delete all TT- and PT-handles in $I \cup I'$ where I is the I-subnet of any existing S^e that contains part of c' to form a new I''.

Step 5: Construct new $S = P(I'')$. If S contains a $\rho(r)$, then it is not minimal (by Lemma 1). Discard S, go to step 2 and repeat.

Step 6: Output all S.

Steps 2-4 are based on Lemmas 2 and 3. Note that $\rho(r) = \{r\} \cup H(r)$ can be computed in $O(|T|)$ time. Only r' (in c') in step 2 is new; the rest are old.

Thus, ***we synthesize minimal siphons by constructing a circuit followed by handles similar to the knitting technique.*** This is very **interesting** since both nets and siphons can be synthesized by first locating a circuit followed by adding handles.

Note that the I_S (S_{19} and S_{20} in Table 1) obtained in step 1 (the rest) contains only (more than) one resource place. Siphons obtained in steps 2 and 3 (step 4) are called basic (compound) siphons as shown in Table 1 (Table 2).

Table 1. Basic siphons.

Basic siphons	places	c_b	Type
S_1	$p_{10}, p_{18}, p_{22}, p_{26}$	$[p_{22}\ t_{10}\ p_{26}\ t_{16}\ p_{22}]$	α^4 (4th c_b)
S_4	$p_4, p_{10}, p_{17}, p_{21}, p_{22}, p_{24}, p_{26}$	$[p_{21}\ t_{17}\ p_{26}\ t_{16}\ p_{22}\ t_5\ p_{24}\ t_4\ p_{21}]$	α^5
S_{10}	$p_4, p_9, p_{12}, p_{17}, p_{21}, p_{24}$	$[p_{21}\ t_{13}\ p_{24}\ t_4 p_{21}]$	α^2
S'_{10}	$p_4, p_9, p_{17}, p_{21}, p_{24}, p_{28}$	$[p_{21}\ t_{13}\ p_{24}\ t_4 p_{21}]$	α^2
S_{16}	$p_2, p_4, p_8, p_{13}, p_{17}, p_{21}, p_{26}$	$[p_{21}\ t_{17}\ p_{26}\ t_9\ p_{21}]$	α^3
S_{17}	$p_2, p_4, p_8, p_{12}, p_{15}, p_{20}, p_{21}, p_{23}, p_{25}$	$[p_{21}\ t_3\ p_{23}\ t_2\ p_{20}\ t_{19}\ p_{25}\ t_{18}\ p_{21}]$	α^6
S'_{17}	$p_2, p_4, p_8, p_{15}, p_{20}, p_{21}, p_{23}, p_{25}, p_{28}$	$[p_{21}\ t_3\ p_{23}\ t_2\ p_{20}\ t_{19}\ p_{25}\ t_{18}\ p_{21}]$	α^6
S_{18}	$p_2, p_4, p_8, p_{12}, p_{16}, p_{21}, p_{25}$	$[p_{21}\ t_8\ p_{25}\ t_{18} p_{21}]$	α^1
S'_{18}	$p_2, p_4, p_8, p_{16}, p_{21}, p_{25}, p_{28}$	$[p_{21}\ t_8\ p_{25}\ t_{18} p_{21}]$	α^1
S_{19}	$p_2, p_4, p_8, p_{12}, p_{17}, p_{21}$	$[p_{21}\ t_8 p_{12}\ t_9 p_{21}]$	β
S_{20}	$p_2, p_4, p_8, p_{17}, p_{21}, p_{28}$	$[p_{21}\ t_{21} p_{28}\ t_9 p_{21}]$	β

Table 2. Compound siphons.

Compound siphons	Places
S_2	$p_4, p_{10}, p_{15}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}$
S_3	$p_4, p_{10}, p_{16}, p_{21}, p_{22}, p_{24}, p_{25}, p_{26}$
S_5	$p_4, p_9, p_{13}, p_{15}, p_{20}, p_{21}, p_{23}, p_{24}, p_{25}, p_{26}$
S_6	$p_4, p_9, p_{13}, p_{16}, p_{21}, p_{24}, p_{25}, p_{26}$
S_7	$p_4, p_9, p_{13}, p_{17}, p_{21}, p_{24}, p_{26}$
S_8	$p_4, p_9, p_{12}, p_{15}, p_{20}, p_{21}, p_{23}, p_{24}, p_{25}$
S'_8	$p_4, p_9, p_{15}, p_{20}, p_{21}, p_{23}, p_{24}, p_{25}, p_{28}$
S_9	$p_4, p_9, p_{12}, p_{16}, p_{21}, p_{24}, p_{25}$
S'_9	$p_4, p_9, p_{16}, p_{21}, p_{24}, p_{25}, p_{27}$
S_{11}	$p_2, p_4, p_8, p_{10}, p_{15}, p_{20}, p_{21}, p_{22}, p_{23}, p_{25}, p_{26}$
S_{12}	$p_2, p_4, p_8, p_{13}, p_{15}, p_{20}, p_{21}, p_{23}, p_{25}, p_{26}$
S_{13}	$p_2, p_4, p_8, p_{10}, p_{16}, p_{21}, p_{22}, p_{25}, p_{26}$
S_{14}	$p_2, p_4, p_8, p_{13}, p_{16}, p_{21}, p_{25}, p_{26}$
S_{15}	$p_2, p_4, p_8, p_{10}, p_{17}, p_{21}, p_{22}, p_{26}$

Based on Lemma 3, upon merging an r, we try to locate an elementary circuit c. If no c can be found, there is no S associated with r. Otherwise, we add a PP- or a TP- or a virtual PT-handle to c. Continue this process until no more such handles can be found. Deleting all TT-handles from the resulting subnet renders an I-subnet, wherein the set of places is an S. We then merge another nearby r and repeat the above process until all r have been merged. An r is more likely to form a circuit with a nearby r'. Thus, we merge r from bottom to top (corresponding to the direction in a WP from the final state to the idle state) or vice versa. This is more efficient than to search r in a random fashion.

For the example in Fig. 6, we first find the backbone; each separate component is an SNC free of TP-inconsistent pair of places. We then get $S_1 = \{p_4, p_6, p_{13}, p_{14}\}$ on circuit $c_1 = [p_{13}\ t_4\ p_{14}\ t_9\ p_{14}]$. Next we add p_{12}, a nearby of p_{13} and find $c_2 = [p_{12}\ t_{10}\ p_{13}\ t_3\ p_{12}]$. A new siphon $S_2 = \{p_5, p_9, p_{12}, p_{13}\}$ is created by synthesis or deleting $\{p_5, p_4\}$ (called CD in section 4) from $\rho(p_{12}) \cup \rho(p_{13})$. Now S_1 serves as an existing S^e in step 4. I_3 is obtained by deleting TT- handles $[t_3\, p_5\, t_4]$ and $[t_9\, p_4\, t_{10}]$ from $I_1 \cup I_2$. Alternatively, as will be detailed in section 4, S_1 contains p_{13} as S_2 does and we create S_3 by adding $\rho(p_{12})$ to S_1 and deleting the above CD similar to the creation of S_2 by adding $\rho(p_{12})$ to $\rho(p_{13})$ and deleting the above CD.

The example in Fig. 7 consists of three robots ($R1$, $R2$, $R3$) and four machines ($M1$ -$M4$). We first find the backbone; each separate component is an SNC free of TP-inconsistent pair of places. We then merge resource place $R1$; we cannot find a circuit for the first merge. For the second, we merge $M3$ with no success because it is not minimal since it contains $\rho(R1)$. We then merge $R2$ with $M3$ to find a circuit containing $R2$ and $M3$. Note that $I(\rho(R2))$, unlike others, is not an SM and it contains two I_S (S_{19} and S_{20} in Table 1).

Note that we need to consider a second circuit $[R1\ t_{19}\ M3\ t_{18}\ R2\ t_3\ M1\ t_2\, R1]$ containing $M1$, which is used exclusively for the middle process (WP1). We merge the rest in the order of $M2$, $M4$, and $R3$. Note the merging order is from top to bottom with the advantage of omitting step 3 in Algorithm 1 for constructing some new S'. This is because we can find S' based on existing ones in step 4 as will be shown in section 4.

Note that by Lemma 1.4, it takes $O(|P| + |R|)$ time to detect $\rho(r)$ in step 5 and $O(|F| + |T| + |P| + |R|)$ time to find a new circuit [17] where $|P|$ ($|R|$, $|F|$) is the total number of state places (resource places, arcs). Let n' and n be the total number of resource places and places respectively in the net. The set operation in steps 3&4 takes $O(n')$. There are totally $O(2^{n'})$ of new I. Hence the total time complexity is dominated by step 4 of the algorithm and hence is $O((|F| + |T| + |P| + |R|)2^{n'})$ better compared with $O((|F| + |T| + |P| + |R|)2^{n})$ in the global approach not exploiting the structural characteristics of S^2CPR.

Notice that it is still exponential and it cannot be polynomial since the number of SMS grows exponentially with the size of the net. Nevertheless, it opens a novel way to investigate the siphon extraction problems from a fresh point of view of handles and bridges.

There are two types of circuits c that may induce SMS. (1) All (at least two) places in c are resource places r similar to S^3PR. (2) c (called β-type c_b^{β}) is one circuit in $I(\rho(r))$ and contains only one r (none in S^3PR). Unlike S^3PR, such c can induce an SMS (*e.g.*, S_{19} in Table 1). Note that for comparison purpose, we have adopted the same subscripts as that of the S^3PR in [1] with only minor difference than that in Fig. 7. Some S are superscripted with ‘’’ to denote the change due to the difference.

Note that S'_{10} is obtained by deleting p_{12} from S_{10} and adding p_{28}. The I_S of S'_{10} and S_{10} have the same basic siphon circuit but different nonvirtual PT-handle containing p_{12} and p_{28} respectively. S'_{17} and S'_{18} can be obtained similarly.

It is easy to detect c_b^{β} since its $I(\rho(r))$ is not an SM and contains only one resource place. There are two types of circuits for (1): elementary (*e.g.*, [p_{13} t_4 p_{14} t_9 p_{14}] in Fig. 6; denoted by c_b^α) and compound (*i.e.*, multiple interconnected elementary circuits; denoted by c_p). The corresponding siphons are called *α-type basic siphons*, denoted by S_b^α and *compound* siphons (*e.g.*, S_3 in Fig. 6), denoted by S_p (see Tables 1 and 2) respectively.

For c_p, some PP-path to a c_b^1 is another c_b^2 corresponding to S_1 and S_2 respectively. Because $S_1 \cup S_2$ is another siphon S_a, we can obtain a new S_p from it by deleting some places (*e.g.*, $S_3 = S_1 \cup S_2 \backslash \{p_5, p_4\}$ in Fig. 6) which are on TT-handles of $I(S_a)$. In other words, compound siphons can be derived from basic siphons. It is interesting to see that if the S^2CPR in Fig. 7 is reduced to the S^3PR in [1], the basic and compound siphons correspond to elementary and redundant ones in [1] respectively.

For non-c_b^β, the I is an SM and the following lemma helps to locate a c.

Lemma 4 For a non-c_b^β, all places in the circuit must be resource places.

For instance, in Fig. 6, the siphon constructed on $c = [p_{12}\ t_2\ p_3\ t_3\ p_5\ t_4\ p_{13}\ t_9\ p_4\ t_{10}\ p_9\ t_{11}\ p_{12}]$ contains state places p_3, p_5, *etc.* and $\rho(p_{12})$ and is not minimal.

Thus for Type 1 circuit, we need only search I_S where all places in c are resource places; *i.e.*, c is a *resource circuit*. Note that c may appear in a single working process (WP). Any circuit (path, handle, subnet) with all places resources is called a *resource circuit* (*path*, *handle*, *subnet*).

There may be PP-handles to c_b which are also resource paths resulting in a new c_b. We do not have to consider TP-handles of resource paths for a different reason − they simply do not exist.

Lemma 5 Let H be a TP-handle or a nonvirtua PT-handle to c_b, then H is not a resource path.

Corollary 1: The resource subnet (Def. 8) of any I_S in an S^2CPR is a state machine (SM).

Upon a strongly connected (by Lemma 2) SM subnet N' (where all places are resource places), we can add nonresource TP- and PP-handles to form an I_S. Since $\rho(r)$ is a siphon for each $r \in N'$, so is the union of all such $\rho(r)$. Deleting nonvirtual PT- and TT-handles from the union forms the I-subnet of an SMS. The rest are PP- or TP-handles which are parts of circuits of some $I(\rho(r))$. Note that each circuit of an $I(\rho(r))$ contains exactly one state place. Note that if N' was non-SC, so would be the resulting I-subnet of SMS and it would violate Lemma 2.

To find all SMS, we have to locate all possible N'. For *paper-sized samples*, we can manually search N'. Upon which, it is easy to identify TP- and PP-handles visually. Hence we can manually construct basic siphons. It is better than resorting to a CAD tool for *paper-sized samples*. In addition to the need to learn the tool, it can be quite time-consuming to enter the structure and marking information as inputs. Scholars often desire to quickly find out SMS under investigation. Note that all techniques in literatures are computer based.

Let R be the set of their resource places in the net N, $T_u = \bullet R \cap R\bullet$ and N_u the net generated by R and T_u is a State Machine (SM). The following algorithm computes all SMS.

Algorithm 2 SMS computation
1. Find all strongly connected components (SCC) N" in N_u in linear time using the algorithm by Tarjan [12]. If not found, no SMS exists and exit.
2. For each N" (an SM), find all its sub-SCC. For each such sub-SCC, add all nonresource TP- and PP-handles to form an I_S. $P(I_S)$ is an SMS.

To the best of our knowledge, no efficient algorithms exist to find all sub-SCC (future work). Thus, we only need to find c_b for elementary (also basic) siphons. Since all places in a c_b are resources, we can remove all state places and their incident arcs in N and apply well-known algorithms [13] to search elementary directed circuits.

We now argue the correctness of Algorithm 1. Step 3 synthesizes a new siphon based on Lemma 3. Step 5 discards it if it is not minimal. Step 2 finds all possible c_b and step 4 finds all possible strongly connected resource subnets beyond a circuit. Thus, the algorithm considers all sub-SCC and finds all SMS.

4. COMPUTATION OF COMPOUND SIPHONS

The idea is based on the following:

Lemma 6 The union of two siphons $D_1 \cup D_2$ is another siphon.

For compound siphons, some PP'-path to a c_b^i is another c_b^j corresponding to S_i and S_j respectively. Because $S_i \cup S_j$ is another siphon S_a, we can obtain a new S from it by deleting some places which are on TT-handles of $I(S_a)$. In other words, compound siphons can be derived from basic siphons.

Table 3. New S generated based on the formula: $S = S_c \cup \rho(SD)\backslash CD$.

Note that $S_{10} = \rho(R2) \cup \rho(M2)\backslash CD$, $S_{16} = \rho(R2) \cup \rho(M4)\backslash CD$, $S_1 = \rho(R3) \cup \rho(M4)\backslash CD$.
No need to search these S; hence reducing the search time.

Seed (*SD*)	*Companion* S^c (*CP*)	*Common Deletions* (*CD*)	*New S*
$M2$	S_{18}	$\{p_2, p_3, p_8\}$	$S_9 = S_{18} \cup \rho(M2)\backslash CD$
	S'_{18}	$\{p_2, p_3, p_8\}$	$S'_9 = S'_{18} \cup \rho(M2)\backslash CD$
	S_{17}	$\{p_2, p_3, p_8\}$	$S_8 = S_{17} \cup \rho(M2)\backslash CD$
	S'_{17}	$\{p_2, p_3, p_8\}$	$S'_8 = S_{17} \cup \rho(M2)\backslash CD$
$M4$	S_{10}	$\{p_{18}, p_{12}\}$	$S_7 = S_{10} \cup \rho(M4)\backslash CD$
	S_{18}	$\{p_{18}, p_{12}\}$	$S_{14} = S_{18} \cup \rho(M4)\backslash CD$
	S_{17}	$\{p_{18}, p_{12}\}$	$S_{12} = S_{17} \cup \rho(M4)\backslash CD$
	S_9	$\{p_{18}, p_{12}\}$	$S_6 = S_9 \cup \rho(M4)\backslash CD$
	S_8	$\{p_{18}, p_{12}\}$	$S_5 = S_8 \cup \rho(M4)\backslash CD$
$R3$	S_{16}	$\{p_{13}, p_{19}\}$	$S_{15} = S_{16} \cup \rho(R3)\backslash CD$
	S_{14}	$\{p_{13}, p_{19}\}$	$S_{13} = S_{14} \cup \rho(R3)\backslash CD$
	S_{12}	$\{p_{13}, p_{19}\}$	$S_{11} = S_{12} \cup \rho(R3)\backslash CD$
	S_6	$\{p_{13}, p_{19}\}$	$S_3 = S_6 \cup \rho(R3)\backslash CD$
	S_5	$\{p_{13}, p_{19}\}$	$S_2 = S_5 \cup \rho(R3)\backslash CD$

In Table 3 we merge $M4$ to create S_{16} containing $R2$ and $M4$. We can then generate S_5-S_7, S_{12}, S_{14} from S_8-S_{10}, S_{17}, S_{18} respectively by adding the minimal siphon that contains $M4$, $\rho(M4)$ and deleting the set of places $P_4 = \{p_{12}, p_{18}\}$ – no need to find TT- and PT-paths in step 4 of Algorithm 1 again.

Note that all S_8-S_{10}, S_{17}, S_{18} contain $R2$ which is a nearby of $M4$; hence, all I of the union of $\rho(M4)$ and S_8-S_{10}, S_{17}, S_{18} share the same TT-handle [t_8 p_{12} t_9] and PT-handle [$M4$ t_{16} p_{18} t_{17}].

We call $M4$ a seed (*SD*), S_8 the Companion S (*CP*), and P_4 the Common Deletion (CD). New S ($= \rho(SD) \cup CP \backslash CD$) formed in this fashion are listed in Table 3. We explain below how to find CD for compound siphons.

Upon adding a new r, if there is a new circuit c containing the new r and an old r', we form a new SMS containing r and r' — called 2-*composite* since it contains only two r – by adding $\rho(r')$ to $\rho(r)$ and deleting CD which is the set of places inside all the non-virtual PT-handles to c. New compound siphons may be formed by deleting CD from the union of existing SMS and $\rho(r)$.

Three or more siphons can also composite a new interesting SMS as follows. It can be formed in a gradual fashion using the beginning 2-composites. For instance, we first create a 2-composite. After adding a new r, we may create a 3-composite by deleting CD from the union of the 2-composite with $\rho(r)$. After adding another new r', we may create a 4-composite by deleting CD from the union of the 3-composite with $\rho(r')$. Continue this process, we can create n-composite, where n = 5, 6, 7, *etc*.

5. COMPARISON WITH OTHER APPROACHES

Chu and Xie [14] proposed a linear programming approach to decide the deadlock-freeness of a Petri net that requires the examination of all minimal siphons. Unfortunately, it is well known that the total number of minimal siphons grows exponentially in the number of nodes.

One way to reduce the complexity of the linear programming approach is to find efficient algorithms for generating minimal siphons that do not contain traps without generating other siphons. All deadlock prevention approaches [7-9] consider only siphons containing resource places. References [6, 15] presented algorithms for generating all basis siphons. However, it is more efficient to employ minimal siphons (the number is usually less) to analyze liveness.

Reference [15] employed the sign incidence matrix in [6] to compute the set of minimal siphons containing a given resource place. However the siphons found may also be traps and the time complexity was not derived. If a minimal siphon contains a marked trap — *no indication as how to check this condition in* [6, 15] — then it will never become empty of tokens. The total number of traps grows exponentially with the size of the net. *Hence it takes exponential amount of time to find whether every minimal siphon contains a trap*, *while step 5 in Algorithm 1 checks it in linear time*.

We search only SMS. In addition, many SMS can be derived from existing ones in terms of formulas; thus the search time may be reduced much.

In summary, our contribution posts the following seven distinct features:

1. Novel – we propose a new model S^2CPR.
2. Synthetic – SMS can be synthesized by adding handles upon a circuit. This allows manual searching of SMS for paper-sized sample nets.
3. Coordinated – upon a new c_b, we build not only a new basic siphon, but also a set of new compound siphons in a coordinated fashion. This saves time much compared with the random approach since both the number and the size of a compound siphon may be much larger than that of basic siphons for medium to large net models.
4. Graphic – we, unlike others, show that SMS corresponds to graphic objects of all sub-SCC of the resource subnet N_u.
5. Efficient – we search only N_u rather the whole net as in the traditional global approach where all nodes and arcs in the net must be traced.
6. Incremental – we construct SMS in an incremental fashion rather than the traditional global and random approach. Instead of finding all circuits following by SMS construction, we start from a c_b and build a basic siphon. Next we find a handle – the traced net remains strongly connected – to the c_b and find all new c'_b, S_b, all c_p containing part of c'_b, and the associated S_p. Repeat this process until no more handles can be added. This process is similar to the synthesis of SNC using the knitting technique [16] based on the S-Matrix, which records structural relationships among subprocesses or handles. The approach has the advantage of adopting this S-Matrix to help us find new c_b and keep track of handles.
7. Strict – we search strict minimal siphons; the number of which is smaller than that of minimal (including strict and non-strict) siphons searched by most approaches.

6. CONCLUSION

We have proposed a new technique to extract SMS for SNC-based FMS which enhances the S^3PR by replacing the state machine with synchronized choice nets. Only a subset of all SMS needs to be searched; from which, we can then derive the rest of SMS.

Finally, it is very interesting that both nets and siphons can be synthesized by constructing handles upon a circuit!

REFERENCES

1. Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, Vol. 34, 2004, pp. 38-51.
2. M. Yamauchi and T. Watanabe, "Time complexity analysis of the minimal siphon extraction problem of Petri nets," *IEICE Transactions on Fundamentals*, Vol. E82-A, 1999, pp. 2558-2565.
3. D. Y. Chao and J. A. Nicdao, "Liveness for synchronized choice Petri nets," *Computer Journal*, Vol. 44, 2001, pp. 124-136.
4. P. Kemper, "Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net," M. A. Marsan, eds., *Application and Theory of Petri Nets*, LNCS 691, Springer Verlag, 1993, pp. 319-338.
5. J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention

policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, Vol. 11, 1995, pp. 173-184.
6. E. R. Boer and T. Murata, "Generating basis siphons and traps of Petri nets using the sign incidence matrix," *IEEE Transactions on Circuits and Systems*, *I–Fundamental Theory and Applications*, Vol. 41, 1994, pp. 266-271.
7. J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Transactions on Automatic Control*, Vol. 46, 2001, pp. 1572-1583.
8. Y. Huang, "Modeling, analysis, deadlock prevention and cell controller implementation for flexible manufacturing systems," Ph.D. Dissertation, Dept. of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan, 2001.
9. K. Barkaoui and I. B. Abdallah, "Deadlock avoidance in FMS based on structural theory of Petri nets," *IEEE Symposium on Emerging Technologies and Factory Automation*, Vol. 2, 1995, pp. 499-510.
10. D. Y. Chao, "Computation of elementary siphons in Petri nets for deadlock control," *Computer Journal*, Vol. 49, 2006, pp. 470-479.
11. D. Y. Chao, "An incremental approach to extract minimal bad siphons," *Journal of Information Science and Engineering*, Vol. 23, 2007, pp. 203-214.
12. R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, Vol. 1, 1972, pp. 146-160.
13. D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, Vol. 4, 1975, pp. 77-84.
14. F. Chu and X. L. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Transactions on Robotics and Automation*, Vol. 13, 1997, pp. 793-804.
15. F. Tricas and J. Ezpeleta, "Some results on siphon computation for deadlock prevention in resource allocation systems modeled with Petri nets," in *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation*, 2003, pp. 322-329.
16. D. Y. Chao and D. T. Wang, "Two theoretical and practical aspects of knitting technique – invariants and a new class of Petri net," *IEEE Transactions on Systems*, *Man*, *Cybernetics*, *Part B*, Vol. 27, 1997, pp. 962-977.
17. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press/MacGraw-Hill Book Company, 1992.

APPENDIX 1: PROOFS

Proof of Lemma 1: (1) $I(\rho(r))$ is a set of PP-circuits of the form $[r\ t\ p\ t'\ r]$. Let A and B be the sets of all such t and t' respectively. $\bullet r = B$, $r\bullet = A$, $\bullet H(r) = A$, $H(r)\bullet = B$. Thus, we have $\bullet\rho(r) = \bullet r \cup \bullet H(r) = A \cup B = \rho(r)\bullet = r\bullet \cup H(r)\bullet$. (2) & (3) Obvious. (4) It holds due to the requirement that each state uses only one resource. ❑

Proof of Lemma 3: (1) (→) Assume contrary and H must be non-virtual PT- or TT-handles. All places in H can be deleted from the siphon while the rest places still form a siphon violating the fact that S is minimal. (←) There is a circuit c in I_S' since it is strongly connected. $P(N')$ is a siphon ∵ $\forall p \in P(N')$, $\bullet p \in T(N')$ since $\bullet p$ is in c or in a

PP- or TP-handle to *N*'. For the same reason, $\forall t \in T(N')$, $t \in P(N')\bullet$; *i.e.*, $T(N') \subseteq P(N')\bullet$. (Note that $p\bullet$ may not be in $T(N')$ if $p\bullet$ is in a PT-handle to *N*'.) Thus $\bullet P(N') \subseteq P(N')\bullet$. It is minimal since if we remove a *p* in any TP- or PP-handle, then it is not a siphon. (2) ($\rightarrow$) Assume contrary. $\forall p \in S$, $\forall t \in p\bullet$, it is in I_S and $\exists p' \in S$, $t \in \bullet p'$. *S* is not an SMS − contradiction. ($\leftarrow$) Let *t* be an output transition of n_s of the nonvirtual PT-handle. $t \in S\bullet$, $t \notin \bullet S$. Hence *S* is an SMS. ❑

Proof of Lemma 4: Suppose there are state places in *c*, then $\exists t \in c$, $r \in c$, and $r \in \bullet t$ since the resource used in a state must be released when moving to the next state (Constraint (4-a) in Def. 5). Thus there is a virtual PT-handle [*r t*] and the *S* is not minimal because it contains another minimal bad siphon as a proper subset. ❑

Proof of Lemma 5: Assume contrary and the n_s (n_e) of *H* (recall Def. 9) has two output (input) resource places against the fact that any state place can use and release only one resource. ❑

Proof of Corollary 1: Lemma 5 indicates that all resource handles to a c_b must be PP-handles. c_b plus these PP-handles is a state machine (SM). ❑

Proof of Lemma 6: Obvious based on the definition of siphons. ❑

Daniel Yuh Chao (趙玉) received the Ph.D. degree from in 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. He has published 103 (including 33 journals) papers in the area of communication protocols, Petri nets, DQDB, networks, FMS, data flow graphs, and neural networks.