A Graphic-Algebraic Computation of Elementary Siphons of BS³PR

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

Unlike other techniques, Li et al. add control nodes and arcs for only elementary siphons, thus reducing the number of control nodes and arcs required for deadlock control in Petri net supervisors. Their method suffers from the expensive computation of all SMS (Strict Minimal Siphons). We propose a graphic-algebraic approach to compute elementary siphons without the knowledge of SMS. We show that each SMS corresponds to a strongly connected resource subnet (sub-SCC) whose characteristic T-vector \( \zeta \) can be computed as a linear sum of that of all resource places in the subnet. An SMS includes all resource places in the subnet plus all input operation places of transitions with positive components in \( \zeta \). We propose Algorithm 2 to find all sub-SCC. We prove that any sub-SCC \( N' \), containing an elementary resource circuit \( c \) as a proper subset and \( N' = N'' \cup c, N'' \cap c = \{r\} \), corresponds to a dependent siphon. Hence, elementary siphons are closely related to (and can be constructed from) elementary (called basic) circuits and in general, combinations of elementary circuits may contribute to elementary siphons. For a simple basic subclass of S³PR (called BS³PR), the set of elementary siphons is identical to that synthesized from elementary (basic) circuits. As a result, we simplify Algorithm 2 to find all elementary circuits. It is more efficient than traditional algorithms by terminating earlier upon detecting that the net is not a BS³PR.

Keywords: flexible manufacturing systems, deadlock control, Petri nets, siphons, elementary siphons, S³PR

1. INTRODUCTION

Liveness in S³PR (systems of simple sequential processes with resources) can be enforced by adding a control place to each strict minimal siphon (SMS). The method is simple but suffers from adding too many control places and arcs, leading to a much more complex PN than the uncontrolled one. In fact, the same amount of places as that of SMS are added in the target net and further, much more arcs are generally added, particularly for large-scale PN.

Iterative control methods in [1] find all SMS in each iteration step and add control places. Repeat it until no new SMS can be emptied. It becomes very difficult and complex even for a moderate-size model due to the fact that there are too many SMS.

The importance of elementary siphons in a Petri net has been increasingly recognized for the deadlock control purpose of discrete event systems including flexible manufacturing systems. However, according to the seminal work concerning elementary siphons by Li and Zhou [2], their computation is expensive since complete siphon enumeration of the Petri net is needed. It is well-known that the number of siphons grows...
quickly and in the worst case grows exponentially with the size of a net. Efficient and effective computation of elementary siphons is overwhelmingly needed from both a practical and theoretical point of view.

Motivated by the need mentioned above, we focus on the elementary siphon computation using a novel graphic-algebraic method as well as a few significant results related to the siphons in some class of Petri nets. The path followed in the paper to show the results is outlined as follows.

Section 2 presents the preliminaries followed by section 3 where we show that each SMS can be synthesized by constructing handles upon a circuit or a strongly connected component with all places resources (called sub-SCC). Section 4 presents Algorithm 2 to compute all sub-SCC. Section 5 defines elementary, dependent siphons and two characteristic $T$-vectors respectively. It shows that the characteristic $T$-vector $\zeta$ of a sub-SCC can be computed as a linear sum of that of all resource places in the subnet. An SMS includes all resource places in the subnet plus all input operation places of transitions with positive components in $\zeta$. We then present Algorithm 3 to compute an SMS based on the new characteristic $T$-vector. We prove in section 6 that any sub-SCC $N'$, containing an elementary resource circuit $c$ as a proper subset and $N' = N' \cup c, N' \cap c = \{r\}$, corresponds to a dependent siphon. Hence, elementary siphons are closely related to (and can be constructed from) elementary (called basic) circuits and in general, combinations of elementary circuits may contribute to elementary siphons. However, for a simple basic subclass of $S^3PR$ (called BS $S^3PR$), the set of elementary siphons is identical to that synthesized from elementary (basic) circuits. As a result, we simplify Algorithm 2 to find all elementary circuits (Algorithm 4). Section 7 concludes the paper.

2. PRELIMINARIES

In this paper, we consider strongly connected nets only.

Petri nets: A Petri net is a 4-tuple $PN = (P, T, F, M_0)$ where $P = \{p_1, p_2, \ldots, p_n\}$ is a set of places, $T = \{t_1, t_2, \ldots, t_m\}$ a set of transitions, with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$, $F: (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, \ldots\}$ is the flow relation and $M_0: P \rightarrow \{0, 1, 2, \ldots\}$ an initial marking vector whose $i$th component, $M_0(p_i)$, represents the number of tokens in place $p_i$. A node $x$ in $N = (P, T, F)$ is either a $p \in P$ or a $t \in T$. The post-set of node $x$ is $x\bullet = \{y \in P \cup T | F(x, y) > 0\}$, and its pre-set $\bullet x = \{y \in P \cup T | F(y, x) > 0\}$. A directed path $\Gamma = [n_1, n_2, \ldots, n_k], k \geq 1$, is a graphical object containing a sequence of nodes and the single arc between each two successive nodes in the sequence. A $P$-vector is a column vector $P \rightarrow Z$ indexed by $P$ and a $T$-vector is a column vector $T \rightarrow Z$ indexed by $T$. The incidence matrix of $N$ is a matrix $A: P \times T \rightarrow Z$ indexed by $[P]$ and $[T]$ such that $A(p, t) = -1$, if $p \in \bullet t \bullet; A(p, t) = 1$, if $p \in t \bullet \bullet t$; and otherwise $A(p, t) = 0$ for all $p \in P$ and $t \in T$, where $Z$ is the set of integers. $t_i$ is firable if each place $p_i$ in $\bullet t_i$ holds no less tokens than the weight $w_j = F(p_j, t_i)$. Firing $t_i$, under $M_0$ removes $w_j$ tokens from $p_i$ and deposits $w_k = F(t_i, p_k)$ tokens into each place $p_k$ in $\bullet t_i$; moving the system state from $M_0$ to $M_t$. Repeating this process, it reaches $M'$ by firing a sequence of transitions. $M'$ is said to be reachable from $M_0$, i.e., $M_0[\sigma > M']$. Ordinary Petri nets (OPN) are those for which $F: (P \times T) \cup (T \times P) \rightarrow \{0, 1\}$. An OPN is called a state machine (SM) if $\forall t \in T, [\bullet t] = [t \bullet t] = 1$. $N' = (P', T', F')$ is called a subnet of $N$ where $P' \subseteq P, T' \subseteq T$, and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. This subnet may include all resource places in the subnet plus all input operation places of transitions with positive components in $\zeta$. It shows that the characteristic vector $M_0(p_i)$ represents the number of tokens in place $p_i$. We then present Algorithm 3 to compute an SMS based on the new characteristic $T$-vector. We prove in section 6 that any sub-SCC $N'$, containing an elementary resource circuit $c$ as a proper subset and $N' = N' \cup c, N' \cap c = \{r\}$, corresponds to a dependent siphon. Hence, elementary siphons are closely related to (and can be constructed from) elementary (called basic) circuits and in general, combinations of elementary circuits may contribute to elementary siphons. However, for a simple basic subclass of $S^3PR$ (called BS $S^3PR$), the set of elementary siphons is identical to that synthesized from elementary (basic) circuits. As a result, we simplify Algorithm 2 to find all elementary circuits (Algorithm 4). Section 7 concludes the paper.
A net $N$ is strongly connected if for every node pair $(n_i, n_j)$, $n_i, n_j \in P \cup T$, there exists a directed path from $n_i$ to $n_j$. A strongly connected component (SCC) $N'$ of $N$ is a strongly connected subnet with no proper supersets. A sub-SCC of an $N'$ is a strongly connected subnet of $N'$.

All nets referred to in this paper will be OPN. Let $N = (P, T, F)$ be a net, $(N, M_0)$ be a marked net and $R(N, M_0)$ is the set of markings reachable from $M_0$. A transition $t \in T$ is live under $M_0$ iff $\forall M \in R(N, M_0), \exists M' \in R(N, M_0): M' \ni t \ni M >$ holds. A transition $t \in T$ is dead under $M_0$ iff $\exists M \in R(N, M_0), \forall p \in P, M(p) \leq k$, where $k$ is a positive integer. $\Gamma = [n_1, n_2, \ldots, n_k]$, $k \geq 1$, is an elementary directed path in $N$ if $\forall (i, j), 1 \leq i < j < k, n_i \neq n_j, \Gamma$ is (non) virtual if it contains only (more than) two nodes. $\Gamma$ is an elementary circuit $c$ in $N$ if $\forall(i, j), 1 \leq i \leq j < k, n_i = n_j$ implies that $i = 1$ and $j = k$, $\forall$ defined above, $|n_i, \ast| = |n_k, \ast| = 1, \ast \ni 1 < i < k, i.e., n_i$ is an interior node of $\Gamma$, then $\Gamma$ is called a simple path. A siphon (trap) $D(\tau)$ is a non-empty subset of places if $\bullet D \subseteq D \bullet (\bullet \subseteq \bullet \ast), i.e.,$ every transition having an output (input) place in $D(\tau)$ has an input (output) place in $D(\tau)$. A minimal siphon does not contain a siphon as a proper subset. A minimal siphon is called a strict minimal siphon (SMS), denoted by $S$, if it does not contain a trap. 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 $T_i = \bullet P_i$. $I_s$ is the I-subnet (the subnet derived from $(S, \bullet S)$ of an SMS $S$. Note that $S = P(I_s); S$ is the set of places in $I_s$.

The following definitions are from [2]. Refer to [3] for more details of the $S^3$PR model.

**Definition 1** [3] A simple sequential process ($S^3P$) 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 operation place; (2) $N$ is strongly connected state machine (SM) and (3) every circuit of $N$ contains the place $p^0$.

**Definition 2** [3] A simple sequential process with resources ($S^3PR$), also called a working processes (WP), is a net $N = (P \cup \{p^0\}, R, T, F)$ so that (1) the subnet generated by $X = P \cup \{p^0\} \cup R \cup T$ is an $S^3P$; (2) $R \neq \emptyset$ and $P \cup \{p^0\} \cap R = \emptyset$; (3) $\forall p \in P, \forall t \in \bullet p, \exists r \in R, \bullet r \ni t \ni r = \{r_p\}$; (4) The two following statements are verified: $\forall r \in R$, (a) $\bullet r \cap P = \emptyset \ni P \neq \emptyset$; (b) $\bullet r \cap R = \emptyset$. (5) $\bullet (p^0) \cap R = \emptyset$. $\forall p \in P \cup \{p^0\}, p$ is called an operation place. $\forall r \in R$, $r$ is called a resource place. $H(r) = \bullet r \cap P$ denotes the set of holders of $r$ (operation places that use $r$). $\rho(r) = \{r\} \cup H(r)$.

The above models the constraints as follows: Def. 2-(3) allows only one shared resource to be used at each operation place; Def. 2-(4.a) dictates that the resource used in an operation place be released when moving to the next operation place; Def. 2-(4.b) shows that two adjacent operation places cannot use the same resource and Def. 2-(5) ensures that initial and final operation place do not use resources. $S^3PR$ is an $S^3P$ which uses a unique resource at every non-idle operation place.

**Definition 3** [3] A system of $S^3PR$ ($S^3PS$) is defined recursively as follows: (1) An $S^3PS$ is defined as an $S^3PR$; (2) Let $N_i = (P_i \cup P_{i0} \cup P_{i1}, T_i, F_i), i \in \{1, 2\}$ be two $S^3PR$ so that $(P_1 \cup P_{10}) \cap (P_2 \cup P_{20}) = \emptyset$, $P_{21} \cap P_{22} = P_{2C} (\neq \emptyset)$ and $T_1 \cap T_2 = \emptyset$. The net $N = \ldots$
(P ∪ P^0 ∪ P_k, T, F) resulting from the composition of N_1 and N_2 via P_C (denoted by N_1 o N_2) defined as follows: (1) P = P_1 ∪ P_2; (2) P^0 = P_1^0 ∪ P_2^0; (3) P_k = P_{k1} ∪ P_{k2}; (4) T = T_1 ∪ T_2 and (5) F = F_1 ∪ F_2 is also an S^3PR. A directed path (circuit, subnet) Γ in N is called a resource path (circuit, subnet) if ∀p ∈ Γ, p ∈ P_k. An elementary resource circuit is a resource circuit that is also elementary.

The S^3PR example in Fig. 1 [2] consists of three robots (R1-R3) and four machines (M1-M4).

![Fig. 1. An example of systems of simple sequential processes with resources (S^3PR) [3].](image)

### 3. SYNTHESIS OF SMS

We construct an SMS based on the concept of handles. Roughly speaking, a “handle” is an alternate disjoint path between two nodes. A PT-handle starts with a place and ends with a transition while a TP-handle starts with a transition and ends with a place.

**Definition 4** [4] Let N = (P, T, F), H_1 = [n_1, n_2, ..., n_k] and H_2 = [n'_{1}, n'_{2}, ..., n'_{h}] are elementary directed paths, n_i, n'_j ∈ P ∪ T, i = 1, 2, ..., k, j = 1, 2, ..., h. Each is called a handle in N if n_i ≠ n'_j, ∀i, j defined above; n_t and n_e are called the terminal nodes or the start and the end nodes of H_1 and H_2 respectively. n_i and n'_j (1 ≤ i ≤ k) are called the interior nodes of H_1 and H_2 respectively. Note that n_t and n_e may be identical. An elementary directed path B = [n_{t1}, n_{t2}, ..., n_{te}] is a bridge from H_1 to H_2 if (1) n_{t1} ∈ H_1, n_e ∈ H_2, n_{t1} ≠ n_{t2}, n_{t2} ≠ n_{t3}, ..., n_{te} ≠ n_{t1} and (2) ∀n ∈ B, if n ≠ n_{t1}, n ≠ n_e, then n ∉ H_1 and n ∉ H_2. H_1 is a XY-handle where X and Y can be T or P. X is T(P) if n_t ∈ T(n_e ∈ P). Y is T(P) if n_e ∈ T(n_t ∈ P). H_1 is a resource handle if all places in H_1 are resource places. The handle H to
a subnet \( N' \) (similar to the handle of a tea pot) is an elementary directed path from \( n_i \) in \( N' \) to another node \( n_e \) in \( N' \); any other node in \( H \) is not in \( N' \). \( H \) is said to be a handle in \( N' = N' \cup H \). Let \( N' = N'\setminus H \). \( N' \) is the removal of graphical object \( H \) from another graphical object \( N' \). A PP-handle with \( n_i = n_e \) is called a PP-circuit.

In Fig. 1, TP-handle: \([t_{13} \ p_{16} \ t_{10} \ p_{25}]\) with \( n_i = t_{18}, n_e = p_{25} \); PP-handle: \([p_{21} \ t_{11} \ p_{2} \ t_{12} \ p_{21}]\) (also called a PP-circuit since \( n_i = n_e = p_{21} \)). \( H = [p_{21} \ t_{17}] \) is a virtual PT-handle since it contains only two nodes.

The basis for SMS synthesis comes from the following:

**Property 1** [4]  
1. The \( I \)-subnet (the subnet derived from (S, \( \bullet \)S)) of SMS S, \( I_S \), is strongly connected.  
2. A subnet \( N' \) is an \( I \)-subnet (see Def. 5) of a minimal siphon iff \( N' \) is maximal in the sense that each handle \( H \) in \( N' \) is a PP- or TP- or virtual PT-handle and there are none of PP-, TP-, and virtual PT-handles to \( N' \).  
3. \( P(N') \) is an SMS iff there is a nonvirtual PT-handle to \( N' \), which is a subnet of \( N' \) without any TP-handles.

**Definition 5**  
An elementary resource circuit is called a basic circuit, denoted by \( c_b \). The procedure to add handles to form \( I_S \) based on Property 1 is called handle-construction. The corresponding \( S = P(I_S) \) is called a basic siphon. The set of resource places in \( c_b \) is denoted by \( R_b \). An expanded \( c_b \) is the union of \( c_b \) and the set of all directed resource PP-handles (called PP-handles) of the form \([r_1 \ t' \ r_2] \), where \( r_1 \in c_b \) and \( r_2 \in c_b \).

**Example:** In Fig. 1, first find a circuit \( c_b = [p_{22} \ t_{10} \ p_{26} \ t_{10} \ p_{22}] \). Second add TP-handles \([t_{16} \ p_{18} \ t_{17} \ p_{26}] \) and \([t_{10} \ p_{10} \ t_{6} \ p_{22}] \) plus PP-handle \([p_{22} \ t_{3} \ p_{10}] \) to get \( I_{S_b} \) and \( S_1 = P(I_{S_b}) = \{p_{10}, p_{18}, p_{22}, p_{26}\} \) with a nonvirtual PT-handle \([p_{26} \ t_{6} \ p_{13} \ t_{10}] \) to \( c_b \).

**Property 2** [4]  
All places in a \( c_b \) must be resource places.

For instance, in Fig. 1, the nonminimal siphon constructed on \( c = [p_{25} \ t_{7} \ t_{11} \ t_{8} \ t_{12} \ t_{9} \ t_{13} \ t_{10} \ p_{26} \ t_{16} \ t_{18} \ t_{17} \ t_{18} \ t_{16} \ t_{19} \ t_{22}] \) contains operation places \( p_{11}, p_{12}, \) etc. and \( \rho(p_{25}) \). Property 2 helps to locate a \( c_b \). The following table shows all possible basic siphons and circuits in Fig. 1.

<table>
<thead>
<tr>
<th>Basic siphons ( S_i )</th>
<th>places</th>
<th>( c_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>( p_{10}, p_{18}, p_{22}, p_{26} )</td>
<td>( [p_{22} \ t_{10} \ p_{26} \ t_{16} \ p_{22}] )</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>( p_{4}, p_{10}, p_{17}, p_{21}, p_{22}, p_{24}, p_{26} )</td>
<td>( [p_{21} \ t_{17} \ p_{26} \ t_{16} \ p_{25} \ t_{4} \ p_{24} \ t_{4} \ p_{21}] )</td>
</tr>
<tr>
<td>( S_{10} )</td>
<td>( p_{4}, p_{8}, p_{12}, p_{17}, p_{21}, p_{24} )</td>
<td>( [p_{21} \ t_{17} \ p_{24} \ t_{4} \ p_{21}] )</td>
</tr>
<tr>
<td>( S_{16} )</td>
<td>( p_{2}, p_{4}, p_{8}, p_{13}, p_{17}, p_{21}, p_{26} )</td>
<td>( [p_{21} \ t_{17} \ p_{26} \ t_{9} \ p_{21}] )</td>
</tr>
<tr>
<td>( S_{17} )</td>
<td>( p_{2}, p_{4}, p_{8}, p_{12}, p_{15}, p_{20}, p_{21}, p_{23}, p_{25} )</td>
<td>( [p_{21} \ t_{17} \ p_{23} \ t_{25} \ p_{20} \ t_{19} \ p_{18} \ p_{21}] )</td>
</tr>
<tr>
<td>( S_{18} )</td>
<td>( p_{2}, p_{4}, p_{8}, p_{12}, p_{16}, p_{21}, p_{25} )</td>
<td>( [p_{21} \ t_{17} \ p_{26} \ t_{18} \ p_{21}] )</td>
</tr>
</tbody>
</table>

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.

**Property 3** [4]  
The resource subnet (Def. 3) of any \( I_S \) in an \( S^3PR \) is a state machine (SM).

Upon a strongly connected resource subnet \( N' \), 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
Deleting nonvirtual PT- and TT-handles from the I-subnet of the union forms that of an SMS. The rest are PP- or TP-handles which are parts of circuits of some \( R(\rho(r)) \). Notice that each circuit of an \( R(\rho(r)) \) contains exactly one state place.

To find all SMS, we have to locate all possible \( N^\prime \) in the next section. For large nets, however, computing a basic siphon may not be easy. Section 5 will present the concept of characteristic T-vector to compute basic siphons in an algebraic way.

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 \( T_u \). The following algorithm computes all SMS.

**Algorithm 1** SMS Computation

**Input:** \( N_u \). **Output:** the set of all SMS.

**Step 1:** Find all strongly connected components (SCC) \( N^\prime \) in \( N_u \) in linear time using the algorithm by Tarjan [5]. If not found, then exit since no SMS exists.

**Step 2:** For each \( N^\prime \) (an SM), find all its sub-SCC \( N_j \). For each such sub-SCC, add all nonresource TP- and PP-handles to form an \( IS, P(IS) \) is an SMS if it does not contain a \( \rho(r) \).

Since the number of SCC is exponential to the size of the net, the time complexity is at least exponential. To the best of our knowledge, no efficient algorithms exist to find all sub-SCC. In the next section, we will present one such algorithm (called Algorithm 2) where we will use the symbol “\( T^\prime \)” to indicate a new sub-SCC. We then present a new characteristic T-vector \( \zeta \) followed by Algorithm 3 to construct all SMS based on \( \zeta \).

Afterwards, we will adapt Algorithm 2 to find sub-SCC for all elementary siphons based on Theorem 1 which indicates that when the intersection of any two sub-SCC, \( N_j \cap N_{j2} = \{ r \} \), then the siphon constructed on \( N_j \cup N_{j2} \) is dependent (i.e., not elementary).

### 4. COMPUTATION OF ALL SUB-SCC

We follow an incremental approach (Algorithm 2). First (step 1), we find an elementary circle \( c([p_1 t_1 p_2 t_2 p_3 t_3 p_1]) \) in Fig. 2) which is a sub-SCC. The temporary set of sub-SCC \( \Lambda = \{ c \} \). Next (step 2), we find a handle \( H_1 \) to \( c \) and add two new sub-SCC to \( \Lambda \). We continue the process to create handles \( H_2, H_3, \ldots, H_k \) until all arcs in the net have been traced.

At each step of adding \( H_i \) (\( [t_1 p_6 t_4 p_3 t_3] \) in Fig. 2), we add two kinds of new sub-SCC. One (called \( \alpha \)-type), \( N_j = N_i \cup H_i \) (the whole net in Fig. 2), is the union of any current sub-SCC \( N_i \) in \( \Lambda \) and \( H_i \). \( N_i \) must be such that \( H_i \neq \emptyset \) and \( H_i \not\subseteq N_i \) (step 2.2; denote by \( B \) the set of new \( T^\prime \) so obtained) so as to make \( N_j \) strongly connected (by Property 2) and results in a new larger set of resource places \( R(N_j) \supset R(N_i) \).

**Algorithm 2** Computation of sub-SCC of an \( N_u \) for all SMS

**Input:** \( N_u \). **Output:** \( \Lambda \), the set of all sub-SCC.

**Step 1:** Select a \( c \) of \( N_u \). Set \( \Lambda = \{ c \} \).

**Step 2:** Repeat

2.1 Add a handle \( H_i \) to \( c \). If \( H_i \) contains only two places, continue (to avoid duplicates since it does not add any new place) and add another \( H_i \) to \( c \).

2.2 Set \( B = \emptyset \). ∀ \( A \) \in \( \Lambda \), such that (s.t.) \( n(H_i) \in A \) and \( n(H_i) \subseteq A \).
\textbf{Definition 6}  Let $\Lambda_6 \in \Lambda$, $H_i$ a handle to $\Lambda_6$, $N_j \equiv \Lambda_6 \cup H_i$, $H'_j$ is a bridge or a handle in $N_j$ such that $N'_j \equiv (N_j/H'_j)$ is SC and $\Lambda \cup H'_j$ is not.

In summary, we select $N_j$ based on the (called unique) principle that the net obtained from the deletion of $H$ from any $\alpha$-type (\textbf{\beta}-type) $\Gamma$ in $B(C)$ remains to be (is not) SC.

In Figs. 2 (a) and (b), each has 6 sub-SCC $\Gamma_1 \sim \Gamma_6$. Upon the addition of $H_2$, $\Lambda = \{\Gamma_1, \Gamma_2, \Gamma_3\}$. Steps 2.2.1 and 2.2.2, produce $B = \{\Gamma_4, \Gamma_5\}$ and $C = \{\Gamma_6\}$ respectively. We propose Algorithm 2 to find all sub-SCC.

The following lemma helps to find $H'_j$. 

\begin{figure}
\centering
\begin{subfigure}{0.3\textwidth}
    \centering
    \includegraphics[width=\textwidth]{fig2a.png}
    \caption{(a)}
\end{subfigure}
\hspace{0.5cm}
\begin{subfigure}{0.3\textwidth}
    \centering
    \includegraphics[width=\textwidth]{fig2b.png}
    \caption{(b)}
\end{subfigure}
\hspace{0.5cm}
\begin{subfigure}{0.3\textwidth}
    \centering
    \includegraphics[width=\textwidth]{fig2c.png}
    \caption{(c) solid, $H_i$; dashed, $H'_i$; bold.}
\end{subfigure}
\caption{Three kinds of $H'_j ([p_2 t_4]$ in (a), $[t_4 p_3]$ in (b), and $[p_2 t_2 p_5]$ in (c). $\Gamma_1 = c$, $\Gamma_2 = c \cup H_1$, $\Gamma_3 = (\Gamma_1 \cup H_1) \cup H_1$, $\Gamma_4 = \Gamma_5 \cup H_2$, $\Gamma_5 = \Gamma_3 \cup H_2$, and $\Gamma_6 = \Gamma_2 \cup H'_2$; bold. $H_2$: dashed.}
\end{figure}

The second (\textbf{\beta}-type), $N'_j = (N_j/H'_j) = ([p_1 t_1 p_2 t_3 p_3 p_1])$ in Fig. 2 (c), where $N_j$ is a sub-SCC found earlier at step 2.2.1, $H'_j ([p_2 t_2 p_3]$ in Fig. 2 (c)) is a bridge $B'$ or a $H'_i$ (complementary handle of $H_i$) in $N_j$ but neither a handle nor a bridge in $\Lambda_6$ (step 2.2.2; denote by $C$ the set of new $\Gamma$ so obtained). If it was a handle or a bridge (e.g., in the presence of other handles $H'_1$ from $p_3$ to $t_4$ in Fig. 2 (a) or from $t_3$ to $p_4$ in Fig. 2 (b)), then deleting $H'_1$ from $N_j$ resulted in another $N'_i$ in $\Lambda$. $N'_j \equiv N'_i \cup H_i$ is in $B$ and was already found in step 2.2.1. Examples are shown in Figs. 2 (a) and (b) respectively. $H'_i ([p_2 t_2 p_3]$ in Fig. 2 (c)) is a $H'_i$ if $n_i(H'_i)$ and $n_4(H'_i)$ coincide with $n_i(H_i)$ and $n_4(H_i)$ respectively.

Note the above $H'_i$ are all simple paths on $H'_i$ with terminal nodes coincident with those of $H'_i$. In general, they remain to be simple paths on $H'_i$, while their terminal nodes may no longer be coincident as shown in the following formal definition.
Lemma 1  Let $\Lambda_k \in \Lambda$, $H_i$ a handle to $\Lambda_k$, $H_i'$ a bridge or a handle in $N_j = \Lambda_k \cup H_i$ and $N_j' = (N_j \setminus H_i')$. $N_j'$ is SC and $\Lambda_k \setminus H_i'$ is not, iff it meets the following conditions: (1) $\forall n \in H_i'$, $n \neq n_k(H_i')$, n is an interior node of $H_i'$, $|n\bullet| = |n\bullet\ast| = 1$; i.e., $H_i'$ is a simple path. (2) $H_i' \subseteq H_i'$; i.e., $H_i'$ is on a certain $H_i'$'. (3) $|H_i'| = 1$ in $\Lambda_k$; i.e., (only one $H_i'$). (4) When $n_k(H_i') \neq n_k(H_i)$ (i.e., $n_k(H_i) \neq n_k(H_i)$), $|n\bullet| > 0$, and $|n\bullet\ast| > 0$ ($|n\bullet\ast| > 0$, and $|n\bullet| > 0$ in $\Lambda_k \setminus H_i'$, and (5) there exists a reverse directed path from $n_k(H_i')$ to $n_k(H_i)$). Note that if $n_k(H_i') = n_k(H_i)$, $n_k(H_i)$ or $n_k(H_i')$), the reverse directed path is artificially denoted by $[(n_k(H_i) \setminus (n_k(H_i')))]$.

Proof: $(-\rightarrow)$ (1) It holds since after the deletion of $H_i'$ from $\Lambda_k$, $N_j'$ is not SC. (2) Assume contrary and $N_j'$ and $\Lambda_k \setminus H_i'$ are either both SC or both not SC – contradiction. (3) Assume contrary and $\Lambda_k \setminus H_i'$ is SC – contradiction. (4) It holds since after the deletion of $H_i'$ from $\Lambda_k$, $N_j'$ is not SC. (5) Assume contrary, then after the deletion of $H_i'$ from $N_j'$, $\Lambda_k \setminus H_i'$ is no longer SC – contradiction. ($\rightarrow\leftarrow$) Since $|H_i'| = 1$ in $\Lambda_k$ and $H_i'$ is on $H_i'$', after the deletion of $H_i'$ from $\Lambda_k$ it (i.e., $\Lambda_k \setminus H_i'$) is not SC. Only paths of the form $[n_1 \ldots n_k(H_i')][n_k(H_i') \ldots n_2] = \Gamma, H_i' \Gamma_2$ may make $n_1$ and $n_2$ disconnected after deleting $H_i'$. Under Condition 5, $\Gamma, \Gamma', H_i', H_i', \Gamma_2 (\Gamma, \Gamma', H_i', \Gamma_2, \Gamma_2)$ is also a path in $\Lambda_k (N_j')$ where $\Gamma, (\Gamma', \Gamma)$ is a reverse path from $n_k(H_i')$ (n_k(H_i')) to $n_k(H_i') (n_k(H_i'))$. Hence, $N_j'$ is SC.

Based on this lemma, we propose the following procedures:

```
find_all_Hi_from_n_k(H)_side(\Lambda_k, H_i)
{
    Repeat
    1. Set $n = n_k(H_i)$;
    2. Find a simple path $\Gamma$ which starts from $n$ (i.e., $n(\Gamma) = n$) and ends at $n(\Gamma)$;
    3. If a reverse path from $n_k(\Gamma)$ to $n_k(H_i)$ does not exist, then break and exit; // if Condition 5 is violated, exit the procedure.
    4. $\Gamma$ is a $H_i'$; set $n = n(\Gamma)$; continue;
    until $n_k(\Gamma) = n_k(H_i)$;
}
```

```
find_all_Hi_from_n_k(H)_side(\Lambda_k, H_i)
{
    1. if $n = n_k(H_i)$
    break and exit; // we have reached $n_k(H_i)$ and no more simple paths
    2. Repeat
    2.1 Set $n = n_k(H_i)$;
    2.2 Find a simple path $\Gamma$ which ends at $n$ (i.e., $n(\Gamma) = n$) and starts from $n(\Gamma)$;
    2.3 If a reverse path from $n_k(H_i)$ to $n_k(\Gamma)$ does not exist, then
        break and exit; // if Condition 5 is violated, exit the procedure.
    2.4 $\Gamma$ is an $H_i'$; set $n = n(\Gamma)$; continue;
    until $n_k(\Gamma) = n_k(H_i)$;
}
```

```
find_all_Hi(\Lambda_k, H_i)
{
    2. $\forall \Lambda_k \in B$ that contains only one $H_i'$; // For each $\Lambda_k$ in $B$.
    }
```
Lemma 2  The procedure \texttt{find\_all\_Hi}[\Lambda_k, H_i] obtains all $H_i^j$ given $\Lambda_k$ and $H_i$.

\textbf{Proof:} Omitted due to space limitation.

Lemma 3  \begin{enumerate}[(1)]  \item $\Lambda \cap B = \Lambda \cap C = B \cap C = \Phi$. \item Steps 2.2.1 and 2.2.2 in Algorithm 2 produce all new sub-SCC that contains $H_i$.  \end{enumerate}

\textbf{Proof:} (1) At the beginning of step 2 of each iteration, $\Lambda$ contains all sub-SCC obtained in previous iterations. $\forall N_j \in B, N_j = \Lambda \cup H_i$ as in step 2.2.1 where $\Lambda_k$ is SC, while $\forall N'_j \in C$, such that $N'_j = \Lambda_k \cup H_i$ is not SC. Thus, $N_j \notin C$ and we have $B \cap C = \Phi$. $\forall N_j \in B \cup C, H_i \subset N_j, \forall \Lambda \in \Lambda_k$, hence $\Lambda \cap (B \cup C) = \Lambda \cap B = \Lambda \cap C = \Phi$. \item Steps 2.2.1 and 2.2.2 in Algorithm 2 produce new sub-SCC that contains $H_i$ since $\forall \Lambda_k \in \Lambda$, it does not contain $H_i$, while those in $B \cup C$ do. \item Step 2.2.1 (2.2.2) creates such new sub-SCC as the union of all sub-SCC (all non-sub-SCC) and $H_i$. There is no other new sub-SCC. Hence it creates all new sub-SCC.

In the first step, $|\Lambda| = 1$. Let $X_i = |\Lambda|$ at the end of $i$th step. We have in the worst case $X_i = X_{i+1} + 2X_{i+1} = 3X_{i+1} = 3^{i+1}$. Hence, the total number of sub-SCC $\theta$ is $\sum_{i=1}^{K} 3^{i-1} = (3^K - 1)/2 = O(3^K)$. In the worst case, $K = O(|T| + |P| + |F|) = O(|N_u|)$, the sum of the number of transitions, places and arcs. Thus, $\theta = O(3^K)$ which far exceeds that ($O(|N_u|)$) to search $c$ and all handles. The time complexity to compute each new sub-SCC is $O(|N_u|)$. The time complexity is dominated by the verification of Condition 5 in Lemma 1. It takes $O(|\Lambda_u|)$ to find a reverse path for each $H_i^j$ and there are $O(|\Lambda_u|)$ $H_i^j$. Hence it takes $O(|\Lambda_u|^2)$ time for $\Lambda_u$ during one iteration. There are $O(|N_u|)$ iterations and $O(|\Lambda_u|) = O(|N_u|)$ in the worst case. Thus each $\Lambda_u$ takes $O(|N_u|^2)$ time and the total worst case time complexity is $O(|N_u|^3)$. Thus, even though the worst case time complexity is exponential, it is linear to $\theta$ and is polynomial to the size of the net if $\theta$ is. To our best knowledge, this is the first result available in the literature.

Proposition 1  Algorithm 2 produces all sub-SCC and no duplicates.

\textbf{Proof:} Lemma 3 guarantees that upon each new $H_i$, it produces all sub-SCC with no duplicates for the subnet (an SCC) that has been traced. At the end of Algorithm 2, the net has been completely traced and all its sub-SCC have also been created with no duplicates.

Next section will present an algebraic technique to compute siphons based on the sub-SCC obtained. To compute elementary siphons, we need not find all sub-SCC. Instead, we will adapt Algorithm 2 to search only elementary circuits as shown in Algo-
We will show that elementary siphons are closely related to (and can be constructed from) elementary circuits and in general, combinations of elementary circuits may contribute to elementary siphons. However, for a simple basic subclass of S'PR (called BS'PR), the set of elementary siphons is identical to that synthesized from elementary circuits. As a result, we propose Algorithm 4 to find all elementary circuits.

### 5. ELEMENTARY SIPHONS AND CHARACTERISTIC T-VECTORS

This section defines elementary, dependent siphons and characteristic T-vectors. In this and next section, unless otherwise mentioned, all \( H \) referred are part of an \( H(\rho(r)) \), \( r \in R \).

**Definition 7** [2] Let \( \Omega \subseteq P \) be a subset of places of \( N \). P-vector \( \lambda_\Omega \) is called the characteristic P-vector of \( \Omega \) iff \( \forall p \in \Omega, \lambda_\Omega(p) = 1 \); otherwise \( \lambda_\Omega(p) = 0 \).

**Definition 8** [2] Let \( \Omega \subseteq P \) be a subset of places of \( N \), and \( \lambda_\Omega \) the characteristic P-vector of \( \Omega \). \( \eta \) is called the characteristic T-vector of \( \Omega \), if \( \eta = \lambda_\Omega \bullet A \), where \( A \) is the incidence matrix.

**Definition 9** [2] \( \forall S_0 \in \Pi \) (the set of all \( S \)), if \( \not\exists S_1, S_2, \ldots, S_n \in \Pi \ (\forall i \in \{1, 2, \ldots, n\}, S_0 \not\equiv S_i) \) such that \( \eta_0 = a_1 \times \eta_1 + a_2 \times \eta_2 + \cdots + a_n \times \eta_n \) holds, where \( a_1, a_2, \ldots, a_n \in \mathbb{R} \), the set of all real numbers, then \( S_0 \) is called an elementary siphon of net \( N \). \( \Pi_\mathbb{R} \) denotes the set of all \( S_0 \).

**Definition 10** [7] Let \( S_0 \in \Pi \Pi_\mathbb{R} \) be a siphon in a net \( N \) and \( S_1, S_2, \ldots, S_n \) be its elementary siphons. \( S_0 \) is called a dependent siphon w.r.t. \( S_1, S_2, \ldots, S_n \), and if \( \eta_0 = a_1 \times \eta_1 + a_2 \times \eta_2 + \cdots + a_n \times \eta_n \) holds, where \( a_1, a_2, \ldots, a_n \in \mathbb{R} \), the set of all real numbers. \( \eta_0 \) is said to touch \( \eta_1, \eta_2, \ldots, \eta_n \). \( S_0 \) is also called a strongly dependent siphon if \( \forall i \in \{1, 2, \ldots, n\}, a_i > 0 \). \( S_0 \) is also called a weakly dependent siphon if \( \exists i, j \in \{1, 2, \ldots, n\}, a_i > 0 \) and \( a_j < 0 \).

The following two tables show all elementary siphons and their \( \eta \) for the net in Fig. 1 respectively.

<table>
<thead>
<tr>
<th>Elementary SMS</th>
<th>( \eta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>( [-t_0 + t_{10} - t_{15} + t_{18}] )</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_1 + t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_3 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_5 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_6 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_7 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
<tr>
<td>( S_8 )</td>
<td>( [-t_1 + t_5 - t_9 + t_{10} - t_{11} - t_3 - t_{13} + t_{15} + t_{17}] )</td>
</tr>
</tbody>
</table>

Similarly, we propose \( \zeta_\mathbb{R} \) based on the set of resource places in \( S \).

**Definition 11** Let \( R \subseteq P \) be a subset of resource places of \( N \), and \( \lambda_\mathbb{R} \) the characteristic P-vector of \( R \). \( \zeta_\mathbb{R} \) is called the characteristic T-vector for \( R \), if \( \zeta_\mathbb{R} = \lambda_\mathbb{R} \bullet A \).
Lemma 4 \[ \zeta_R = \sum_{r \in R} \zeta \] where \( R \) is the set of all resource places in a resource subnet.

**Proof:** It holds by Def. 8.

We need to compute SMS from \( \zeta \) given a sub-SCC \( N_c \). The following lemma is helpful.

Lemma 5 For a sub-SCC \( N_c \) with \( \zeta \) the characteristic \( T \)-vector of the SMS \( S \), we have
\[ (1) \forall t \in N_c, \zeta(t) = 0. \quad (2) \forall t \in \bullet R, R = P(N_c), t \notin T(N_c), \text{if } \zeta(t) = 1, \text{then } t \text{ is on a non-resource a TP- or PP-handle } H \text{ to } N_c \text{ where } H \text{ is part of an } I(\rho(r)), r \in P(N_c). \quad (3) \exists \ = R \cup \{p \} \ p \in \bullet t, \zeta(t) = 1 \].

**Proof:** (1) Since \( N_c \) is strongly connected and a state machine, every transition \( t \) in \( N_c \) has exactly one input and one output place, we have \( \zeta(t) = 0 \). (2) Since \( t \notin T(N_c) \), \( t \) must be on a handle \( H \) to \( N_c \). \( \zeta(t) = 1 \) implies \( t \) must be an input transition of \( r \in R \), where \( H \) is part of an \( I(\rho(r)) \). Because the end node of \( H \) is a place, it must be a TP- or PP-handle by Property 1. (3) Applying the handle-construction procedure on \( N_c \), the \( I \)-subnet \( I \) contains \( N_c \) and all possible \( H \) in (2). Thus \( S = P(I) \cup \{p \} p \in \bullet t, \zeta(t) = 1 \). 

This lemma computes \( S \) given \( N_c \) if \( S \) is minimal. In some case, the computed \( P(I) \) may not be so if it contains a \( \rho(r), r \in P(N_c) \). We now have

<table>
<thead>
<tr>
<th>Algorithm 3</th>
<th>SMS Computation given an ( N_c ) based on ( \zeta )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>( N_c )</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td>( S ), the corresponding SMS if ( S ) is minimal.</td>
</tr>
<tr>
<td><strong>Step 1:</strong></td>
<td>( S = P(N_c) ).</td>
</tr>
<tr>
<td><strong>Step 2:</strong></td>
<td>( \zeta_R = \sum_{r \in R} \zeta ) where ( R = P(N_c) ).</td>
</tr>
<tr>
<td><strong>Step 3:</strong></td>
<td>Set ( S = S \cup {p } p \in \bullet t, \zeta(t) = 1 ).</td>
</tr>
<tr>
<td><strong>Step 4:</strong></td>
<td>If ( \exists r \in R, \rho(r) \subseteq S ), then ( S ) is not minimal; break and exit.</td>
</tr>
</tbody>
</table>

The time complexity for Algorithm 3 have been derived in [8] to be \( O(|P| + |R|) \). Note that \( N_c \) can be computed using Algorithm 2. It is unclear how to select \( N_c \) to compute elementary siphons. In the next section, we propose one such algorithm for a specific subclass of \( S^3PR \), called BS\(^3\)PR (basic \( S^3\)PR).

### 6. COMPUTATION OF SUB-SCC FOR ALL ELEMENTARY SIPHONS

This section proves (based on Lemma 6) that any sub-SCC \( N' \), containing an elementary resource circuit \( c \) as a proper subset and \( N' = N' \cup c, N' \cap c = \{r\} \), corresponds to a dependent siphon. That is, \( N' \) corresponds to a dependent siphon if \( N' = N' \cup c, N' \cap c = \{r\} \). We will then modify Algorithm 2 slightly to find \( N' \) for all elementary siphons.

In the sequel, all lemmas and theorems apply to \( S^3PR \) only and we omit their referral to save space. Also \( \Gamma_1 \) and \( \Gamma_2 \) are two resource directed paths, \( \eta_1, \eta_2 \) and \( \eta_3 \) are the characteristic \( T \)-vectors of \( \Gamma_1, \Gamma_2, \) and \( \Gamma_3 \) respectively, and \( \Gamma_3 = \Gamma_1 \cup \Gamma_2 \).

Lemma 6 For an \( S^3PR \), \( \eta_3 = \eta_1 + \eta_2 \iff \exists r \in R, \Gamma_1 \cap \Gamma_2 = \{r\} \).

**Proof:** There are 4 cases for \( t \in c = [t_1 t_2 p_1 t_3 r_1] \) of \( \rho(r_1) \). (a) \( t(t_2 \text{ and } t_3) \) is on a PP-circuit
c' to \( \Gamma_1 \cup \Gamma_2 \). \( \eta(t) = \eta(t) + \eta(t) = 0 + 0 \). (b) \( t(t_2 \text{ and } t_3) \) is on a TT-handle to \( \Gamma_1 \cup \Gamma_2 \). Part of \( c' \) becomes a TT-handle \([t_2, p, t_1] \). \( \eta(t_2) = 0 = \eta(t_2) + \eta(t_2) = -1 + 1, \eta(t_3) = \eta(t_3) + \eta(t_3) = 0 + 1, \eta(t_3) = 0 = \eta(t_3) + \eta(t_3) = 0 + 0 \). (c) \( t(t_2 \text{ and } t_3) \) is on a TP-handle to \( \Gamma_1 \cup \Gamma_2 \). \( \eta(t_2) = 1 = \eta(t_2) + \eta(t_2) = -1 + 0, \eta(t_3) = \eta(t_3) + \eta(t_3) = \eta(t_3) + \eta(t_3) = 0 \).

**Theorem 1** For an S'PR where \( \Gamma_1, \Gamma_2, \text{ and } \Gamma_3 \) are sub-SCC \( N_{j_1}, N_{j_2}, \text{ and } N_{j_3} \) respectively, \( \eta_1 = \eta_1 + \eta_2 \) iff \( \exists r \in R, N_{j_1} \cap N_{j_2} = \{r\} \).

**Proof:** It follows by setting \( \Gamma_1, \Gamma_2, \text{ and } \Gamma_3 \) to be \( N_{j_1}, N_{j_2}, \text{ and } N_{j_3} \) respectively in Lemma 6.

**Corollary 1** For an S'PR where \( \Gamma_1, \Gamma_2, \text{ and } \Gamma_3 \) are \( c_{b_1}, c_{b_2}, \text{ and } c_{b_3} \) respectively, \( \eta_1 = \eta_1 + \eta_2 \) iff \( \exists r \in R, c_{b_1} \cap c_{b_2} = \{r\} \).

**Proof:** It follows by setting \( N_{j_1}, N_{j_2}, \text{ and } N_{j_3} \) to be \( c_{b_1}, c_{b_2}, \text{ and } c_{b_3} \) respectively in Theorem 1.

Thus, if two basic circuits \( c_{b_1} \) and \( c_{b_2} \) intersect at only \( r \), then the siphon constructed on \( c_{b_3} = c_{b_1} \cup c_{b_2} \) is a dependent siphon since \( \eta_1 = \eta_1 + \eta_2 \). Conversely, if \( \eta_2 = \eta_1 + \eta_2 \), then \( c_{b_1} \) and \( c_{b_2} \) intersect at only at \( r \).

**Theorem 2** If \( H_i \) is a PP-circuit \( c \) as in step 2.2 of Algorithm 4, then \( N_j = A_k \cup H_i \) corresponds to a dependent siphon, where \( A_k \in A \) and \( A_k \cap c = \{r\} \).

**Proof:** The proof is similar to that for Theorem 1 where \( A_k \) is a circuit.

**Algorithm 4** Computation of elementary circuits for all elementary siphons

**Input:** \( N_e \)  
**Output:** \( A \), the set of all elementary circuits.

**Step 1:** Select a \( c \) of \( N_e \). Set \( \Lambda = \{c\} \).

**Step 2:** Repeat

1. Add a non-P-P’-handle \( H_i \) to \( \Lambda \).
2. Set \( \Lambda' = \Phi \cup \Lambda_k \in \Lambda, \) such that (s.t.) \( n_i(H_i) \in \Lambda_k \) and \( n_i(H_i) \in \Lambda_k \).
3. Set \( \Lambda' = (\Lambda, H_i) \cup H_i, C = \Lambda' \cup \Lambda' \) (For each \( \Lambda_k \), delete \( H_i \) and add \( H_i \) to form \( \Lambda' \) ). Add \( \Lambda' \) to \( C \).
4. Add new PP’-handles \( H_i \) to \( c' \) and find all new circuits \( c'' \) that contains \( H_i \). \( C = \Lambda \cup \{c'' \} \) (a PP’-circuit to the avoid containment test) so that \( \Lambda \rightarrow \Lambda' \) (If \( \Lambda \) is reduced to \( \Lambda' \)). If \( c' \cap N_e = \{r\} \) (If \( c' \) is not PP-circuit to the net that has been traced), declare that it is not a BS’PR and exit.
5. \( \Lambda \) (add \( C \) to \( \Lambda \)).

**Step 3:** Output the final \( \Lambda \).

Thus, when the newly added handle is a \( c_{b_3} \), then the siphon constructed on the new sub-SCC \( N_j \) is dependent. At this \( i \)-th step, all new siphons constructed are dependent.

However, \( \eta_3 \neq \eta_1 + \eta_2 \) does not imply that \( D_i \) is elementary since \( \eta_3 \) may be the linear combination of the characteristic T-vectors of more than 2 siphons. Thus, even though \( c_{b_1} \cap c_{b_2} \) is more than a single resource place, we may still construct a dependent siphon from \( c_{b_1} \cup c_{b_2} \). An example is shown in Fig. 3 with four emptiable minimal siphons \( S_1 = \)
In a BS₃PR, the set of basic and elementary siphons are identical. Based on this theorem, we can compute all elementary siphons by searching all elementary resource circuits in the BS₃PR. A number of algorithms [5, 6] exist for the searching of all elementary circuits.

Note that if |cₛ| is polynomial with respect to the size of the net, then it takes polynomial time to search for all elementary circuits cₛ using Johnson’s algorithm [6]. By Theorem 1 in [7], the number of elementary siphons (or rank of matrix η), is no greater...
than \( \min\{|P| + |R| + |P^\emptyset|, |T|\} \), so is \(|c_b|\). Thus, \(|c_b| \leq \min\{|P| + |R| + |P^\emptyset|, |T|\} \). For each \( c_b \), it takes \( O(|P| + |R|) \) \( O((|P| + |R|)|c_b|) \) for all time to compute an SMS using Algorithm 3. Thus, the total time complexity for Algorithm 4 to compute all elementary siphons is \( O((|P| + 2|R| + |T|)|c_b|)(= O((|P| + 2|R| + |T|)\min\{|P| + |R| + |P^\emptyset|, |T|\})) \) which is polynomial.

However, we have to take extra time to check whether it is a \( \text{BS}^3\text{PR} \) based on the conditions in Def. 12. It is more efficient to check the conditions while finding new \( c_b \) since there is no need to continue the search as soon as we find that it is not a \( \text{BS}^3\text{PR} \).

Recall that \( \forall c_{b1} \in C_b, c_{b2} \in C_b \), such that \( c_{b1} \cap c_{b2} \neq \Phi \), if neither \( R_{b1} \subseteq R_{b2} \) nor \( R_{b2} \subseteq R_{b1} \) (called the containment test), then \( c_{b1} \cap c_{b2} = \{r\}, r \in P_R \). Thus, after locating the first \( c_{b1} \), all new handles must be a circuit, otherwise, \( c_{b1} \cap c_{b2} \) is a path rather than a single node. Based on this, we modify Algorithm 1 to find \( c_b \) (\( \Lambda \) is a set of \( c_b \) rather than sub-SCC) rather than sub-SCC as in Algorithm 2. In reality, it may be that \( R_{b1} \subseteq R_{b2} \) or \( R_{b2} \subseteq R_{b1} \) and as a preprocessing, we remove all PP’-handles so that the remaining subnet remains to be SC (strongly connected) but without PP’-handles and thus neither \( R_{b1} \subseteq R_{b2} \) nor \( R_{b2} \subseteq R_{b1} \).

Afterwards, we start by locating a \( c_{b1} \) followed by adding a handle \( H^r \) to \( c_{b1} \). If \( H^r \) is not a circuit, we stop and exit declaring a nonBS3PR. Otherwise, we continue adding handles \( H^r \) and stop similarly when the \( H^r \) is not a circuit.

In Algorithm 4, we avoid the preprocessing step by a trick that first adds all new PP’-handles followed by deleting all PP’-handles to the \( c^\prime \) that contains \( H^r \). This reduces \( \Lambda \) to \( \Lambda^\prime \) and it is not a BS3PR, if \( H^r \) is not PP-circuit. We no longer need to do the containment test.

The time complexity to find circuit \( c^\prime \) in step 2.3 is \( O(|\Lambda|) \) \((|\Lambda| = |P| + |P^\emptyset| + |R| + |T| + |F|) \). Hence the worst total time complexity is \( O(|\Lambda|\Lambda) \). Thus, even though the worst case time complexity is exponential, it is linear to \(|\Lambda|\) and is polynomial to the size of the net if \(|\Lambda|\) is.

**Theorem 4** Algorithm 4 produces all elementary circuits of a BS3PR.

**Proof:** The proof is similar to that for all sub-SCC. A new elementary circuit \( c^\prime \) must include \( H_i \) and part of an old \( \Lambda_i \) in \( \Lambda \). In order to be a circuit, the complementary handle, \( H_i^c \), must be deleted from \( \Lambda_i \cup H_i^c \); i.e., \( c^\prime = (\Lambda_i \cup H_i^c) \cup H_i \) as in step 2.3 of the algorithm. Thus, it produces all elementary circuits. By deleting all PP’-circuits to \( c^\prime \) at step 2.4 in each iteration, we remove the possibility that \( c^\prime \cap c^\prime = c^\phi, \forall c^\phi \) in the reduced \( \Lambda^\prime \). By the definition of BS3PR, \( c^\prime \cap c^\phi = \Phi \) or \( = \{r\}, r \in R \); this is checked at step 2.4 and if it is not a BS3PR, the algorithm terminates at step 2.4 of a certain iteration.

Thus, after producing all elementary circuits of a BS3PR using Algorithm 4, we can apply Algorithm 3 to compute all basic or elementary siphons. We then can follow the algorithm in [4] to produce all dependent siphons.

**7. CONCLUSION**

We have proposed a new \( T \)-characteristic vector \( \zeta \) to compute elementary siphons for BS3PR (basic systems of simple sequential processes with resources) in a graphic-
algebra fashion without the prior knowledge of all SMS. In a future paper, we will generalize the results to arbitrary S'PR. We will derive the condition under which a strongly connected resource subnet (other than an elementary circuit) corresponds to an elementary siphon. Subsequently, we will modify Algorithm 2 slightly to find all strongly connected resource subnets that correspond to elementary siphons.

REFERENCES


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.