http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (4): 856-862, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.856.862

A Polynomial-time Decomposition Algorithm for Petri Nets Based on Indexes of Transitions

Qingtian Zeng
College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China

Abstract: Similar to the decomposition approach for Petri nets based on the indexes of places, decomposition based on the indexes of transitions is also convenient to analyze dynamic properties of structure-complex Petri nets. This study proposes an algorithm for the decomposition approach based on indexes of transitions and analyzes the complexity of the given algorithm. The main data structures required and four key functions contained in the decomposition algorithm are addressed firstly. It is proved that the proposed decomposition algorithm is a polynomial-time algorithm.

Key words: Petri net, index of transition, decomposition, polynomial-time algorithm

INTRODUCTION

As models for physical systems, Petri nets are well suited to describe and analyze systems with concurrency, synchronization and conflicts (Murata, 1989; Zeng, 2004; Zeng and Duan, 2007; Wang and Zeng, 2008). However, with the increase of the node number in a Petri net, its structure will be more complex so it is difficult to analyze the properties of the net system. Traditionally, in order to overcome this difficulty, some solutions including decomposition, reduction, composition and net operation are introduced by many researchers. Decomposition of Petri nets is one of very useful methods for property analysis of structure-complex systems. Kwang *et al*. (1987) gave several generalized reduction methods of Petri nets. In Koriem (1999), two analytical decomposition techniques are proposed for computing the transient state space solution of large Stochastic Petri Net (SPN) models of Multistage Interconnection Networks (MINs) and Hierarchical Interconnection Networks (HINs). A large scale SPN model is partitioned into smaller submodels. The submodels are compressed and combined to calculate the entire net. In Esparza (1994), many reduction rules are introduced that make it possible to reduce all and only live and bounded Free Choice Petri nets to a circuit containing one place and one transition. The reduction algorithm is shown to require polynomial time in the size of the system. In Zeng (2007), two decomposition methods are proposed for structure-complex Petri nets based on the indexes of places and transitions, respectively. The decomposition methods proposed in Zeng (2007) are very useful for property analysis of structure-complex Petri nets since the structure of the decomposition net is well-formed by Zeng (2007, 2008) and Cui *et al*. (2011). The language and process relations are analyzed during the decomposition and a method is proposed to present the process of a structure-complex Petri net (Zeng, 2008), respectively. However, we only presented the decomposition method for a Petri net based on the indexes of places or transitions in the previous research results (Zeng, 2007, 2008; Cui *et al*., 2011).

To find a decomposition algorithm especially a polynomial-time decomposition algorithm for a Petri net is also as important as the decomposition method. Unlike the traditional work on Petri net decomposition (Kwang *et al*., 1987; Koriem, 1999; Esparza, 1994; Zeng, 2007), this study addresses a polynomial-time algorithm for Petri net decomposition. Recently, we presented a polynomial-time algorithm was presented for the decomposition approach based on indexes of places (Zeng *et al*., 2008). This study gives a decomposition algorithm for a Petri net based on indexes of transitions and analyzes the complexity of the given algorithm. The main data structures and key functions contained in the algorithm will be addressed in detail. It is proved that the proposed decomposition algorithm is a polynomial-time algorithm. It provides a useful computation method for the decomposition of structure-complex Petri nets.

DECOMPOSITION OF A PETRI NETS BASED ON THE INDEXES OF TRANSITIONS

Some of the essential terminology and notations related to this study are defined first. To save space, more concepts of Petri nets can be seen (Murata, 1989; Zeng

2004; Zeng and Duan, 2007; Wang and Zeng, 2008). Convenient to define, we assume that the Petri nets discussed in this study are finite and connected and they are not T-Nets (Murata, 1989; Zeng, 2004; Zeng and Duan, 2007; Wang and Zeng, 2008).

Definition 1: Let $\Sigma = (S, T; F, M_0)$ be a Petri net, a function $f: T \rightarrow \{1, 2, ..., k\}$ is said to be an index function defined on the transition set if:

$$\forall t_1, t_2 \in T, (t_1^{\bullet} \cap t_2^{\bullet} \neq \phi) \vee ({}^{\bullet}t_1 \cap {}^{\bullet}t_2 \neq \phi) \rightarrow f(t_1) \neq f(t_2)$$

f (t) is named as the index of the transition t (Zeng, 2007).

Definition 2: Let $\Sigma = (S, T; F, M_0)$ be a Petri net, $f: T \rightarrow \{1, 2, ..., k\}$ be the index function on the transitions of Σ. The Petri net $\Sigma_i = (S_i, T_i; F_i, M_{0i})$ $(i \in \{1, 2, ..., k\})$ is said to be the decomposition net of Σ based on the index function f if Σ_i satisfies the following conditions (Zeng, 2007).

$$T_i = \{t \in T \mid f(t) = i\} \quad (1)$$

$$S_i = \{s \in S \mid \exists t \in T_i, s \in {}^{\bullet}t \cup t^{\bullet}\} \quad (2)$$

$$F_i = \{(S_i \times T_i) \cup (T_i \times S_i)\} \cap F \quad (3)$$

$$M_{0i} = \Gamma_{S \rightarrow S_i} M_0 \quad (4)$$

where, $\Gamma_{S \rightarrow S_i} M_0$ is the projection of M_0 such that:

$$\forall s \in S_i, \Gamma_{S \rightarrow S_i} M_0(s) = M_0(s)$$

Simply, Σ_i is named as the index decomposition net of Σ.

Definition 3: A Petri net $\Sigma = (S, T; F, M_0)$ is a T-Net iff $\forall s \in S: |{}^{\bullet}s| \leq 1$ and $|s^{\bullet}| \leq 1$ (Zeng, 2007).

Theorem 1: Let $\Sigma = (S_i, T_i; F_i, M_{0i})$ $(I \in \{1, 2, ..., k\})$ be the decomposition net based on index of transition of a Petri net $\Sigma_i = (S, T; F, M_0)$, then Σ_i is a T-net (Zeng, 2007).

Proof: In the decomposition nets $\Sigma_i = (S_i, T_i; F_i, M_{0i})$ $(i \in \{1, 2, ..., k\})$, we assume that there is at least one net is not a T-net. Without loss of generality, let $\Sigma_j = (S_j, T_j; F_j, M_{0j})$ $(1 \leq j \leq k)$ be not a T-net. Because $\Sigma_j = (S_j, T_j; F_j, M_{0j})$ is not a T-net, there is at least one place $s \in S_j$ such that $|{\bullet}s|>1$ or $|s{\bullet}|>1$ according to the definition of T-net.

- In the case $|{\bullet}s|>1$, it means that the place s has at least two input transitions. Without loss of generality, we assume that the input transitions of s are t_1 and t_2, thus:

$$t_1^{\bullet} \cap t_2^{\bullet} = \{s\} \neq \phi$$

- According to Definition 1, $f(t_1) \neq f(t_2)$. According to Definition 2 t_1 and t_2 should be decomposed into two subnet, so t_1 and t_2 can not be the input transitions of s in $\Sigma_j = (S_j, T_j; F_j, M_{0j})$. Thus, the assumption is not correct
- In the case $|s{\bullet}|>1$, the conflict can also be obtained using the similar proof process. Therefore, in the decomposition nets $(\Sigma_j = (S_j, T_j; F_j, M_{0j})$ $(i \in \{1, 2, ..., k\})$ there is no any net is not a T-net. The theorem is proved

The Petri net shown in Fig. 1 is the model for the problem of the dining philosophers. We use the decomposition method of Definition 2 to decompose Σ.

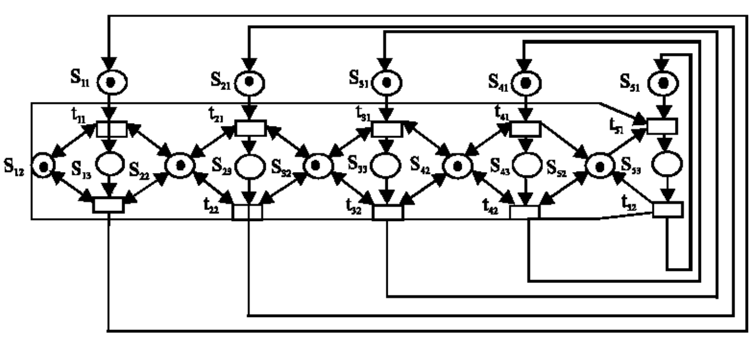

Fig. 1: The Petri net model for the problem of the dining philosophers

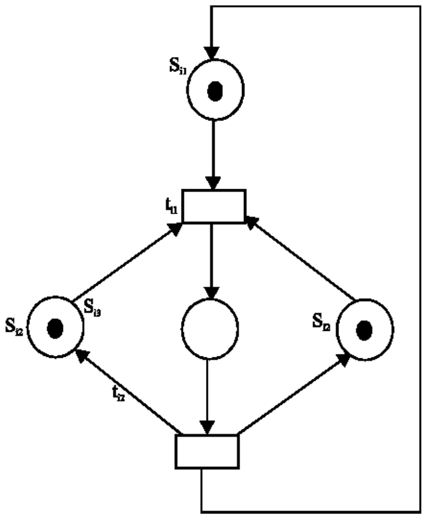

Fig. 2: The decomposition subnets of the Petri net shown in Fig. 1

A function f is first defined on the place set such that:

$$f(t_{11}) = f(t_{12}) = 1,\ f(t_{21}) = f(t_{22}) = 2,$$
$$f(t_{31}) = f(t_{32}) = 3,\ f(t_{41}) = f(t_{42}) = 4,$$
$$f(t_{51}) = f(t_{52}) = 5$$

It can prove that f satisfies all the conditions in Definition 1. Based on the method of Definition 2, five index decomposition net systems Σ_1, Σ_2, Σ_3, Σ_4 and Σ_5 are obtained. Each Σ_i(i = 1, 2, 3, 4, 5) is shown in Fig. 2. The semantic of the decomposition approach is clear. We can see that each Σ_i(i = 1, 2, 3, 4, 5) shown in Fig. 2 is the model for one philosopher.

More discussion about the decomposition approach based on the indexes of transitions can be seen by Zeng (2007). Based on the results presented by Zeng (2007), to find a polynomial-time algorithm for the decomposition approach is the future work to be addressed. In the following sections, we present a polynomial-time algorithm for the decomposition approach. Firstly, the main data structures and key functions contained in the algorithm are addressed first.

MAIN DATA STRUCTURES

Firstly, the main data structures to store the components of a Petri net are presented, including its flow relation, the input and output set of each transition, each place and its tokens.

Store of flow relation: Because any Petri net can be determined by its input matrix and output matrix (Murata, 1989), we use output matrix:

$$A^+ = [a^+_{ij}]_{n\times m}$$

and input matrix:

$$A^- = [a^-_{ij}]_{n\times m}$$

to represent the structure of a Petri net, where,

$$a^+_{ij} = \begin{cases} 1 & \text{if } \ (t_i, s_j) \in F \\ 0 & \text{otherwise} \end{cases} \text{ and } a^-_{ij} = \begin{cases} 1 & \text{if } (s_j, t_i) \in F \\ 0 & \text{otherwise} \end{cases}$$

$$A^+ = [a^+_{ij}]_{n\times m} \text{ and } A^- = [a^-_{ij}]_{n\times m}$$

can be stored by a two-dimension array, respectively.

Store of the input and output set of each place: The input set and output set of each place are stored by a one-dimension array with |T|+1 length, respectively. An Array as represents the input set of a place s, while bs represents the output set of s, which are shown as followings. If a place t_i belongs to as (or bs), the (i+1)th position in as (or bs) will be set as 1, otherwise be set as 0.

as	...	...	...

bs	...	...	...

Store of a set X_k (k = 1, 2...): The transitions with same indexes will be put into a set X_k (k = 1, 2...) and X_k (k = 1, 2...) is stored by a one-dimension array with |T| length. If the (i)th position in X_k is set as 1, it means t_i belongs to X_k(k = 1, 2...). Otherwise, the corresponding position will be set as 0.

t_1 t_2 ...t_n

		...	

Store of index of each transition: The index of each transition is stored by a one-dimension array with |T| length, P_k (k = 1, 2...). If the index of t_i is l, the corresponding position of t_i in Q will be set as *l.* in P_k (k = 1, 2...).

t_1	t_2	...	t_m
		...	

Store of tokens: A one-dimension array with |S| length, Q will be used to store all the tokens in the places. If s_i contains l tokens, the corresponding position of s_i in Q will be set as *l*.

S_1	S_2	...	S_n
		...	

Store of a set Y_u (u = 1, 2...): Another set Y_u (u = 1, 2...) are used to store the transitions of each decomposition net. Y_u (u = 1, 2...) is also represented by a one-dimension array with |T| length. If the (i)th position in Y_u is set as 1, it means t_i belongs to Y_u (u = 1, 2...). Otherwise, the corresponding position will be set as 0.

t_1	t_2	...	t_m
		...	

ALGORITHM DESIGN

The main point in the decomposition algorithm is to obtain the index of each transition. According to the index of each transition, the decomposition net can be obtained transaction with same index. At the intialization step of the algorithm, we put all the transition s into set X_1. For each transition in X_1, denoted by t, let:

$$\lambda[t]=|\{t^{'}\,|\,\exists p\in P,\{t,t^{'}\}\subseteq {}^{\bullet}p \text{ or } \text{ or } \{t,t^{'}\}\subseteq p^{\bullet}\}|$$

In the following step, we select a transition t in X_l (l = 1, 2, ...) such that $\lambda[t]\neq 0$ and move t from X_l to X_{l+1}. If there is a transition moved out, the value $\lambda[t]$ for each transition in X_l will tions with same index. At the initialization step of the be updated. If $\lambda[t]$ for each transition in X_l is not equal to 0, the selecting and moving operations will be repeated on X_l; Otherwise, repeat the selecting and moving operations on X_{l+1}. After the selecting and moving operations on completed on all sets, the transitions in one set can be assigned with one same index.

Key functions: Firstly, four key functions contained in the decomposition algorithm are presented which are Mark (X_k), Move (X_k, y), Divide (X_k) and Outface (Y_u).

- **Mark(X_k) //Obtain λ[p] for each place in X_k**
  INPUT: X_k
  OUTPUT: λ[p] for each transition in X_k

{
Step 0: for each $y\in X_k$, $\lambda[y]\leftarrow 0$. Let $i\leftarrow 1$.
Step 1: If $i>|S|$, halt.
Step 2: For each s_i, if there exists $x\in X_k$ and $x\neq y$ such that $y\in\bullet s_i$ and $x\in\bullet s_i$ DO $\lambda[y]\leftarrow\lambda[y]+1$
Step 3: If there exists $x\in X_k$ and $x\neq y$ such that:

$${}^{\bullet}x\cap{}^{\bullet}y\neq\varnothing \text{ and } {}^{\bullet}x\cap{}^{\bullet}y\neq\varnothing$$

go to Step 5
Step 4: For each s_i, if there exists $x\in X_k$ and $x\neq y$ such that $y\in\bullet s$ and $x\in\bullet s_i$, DO$\lambda[y]\leftarrow\lambda[y]+1$
Step 5: let $i\leftarrow i+1$, goto Step 1.
}

- **Move(X_k, y) // Move transition *y* from X_k to X_{k+1}**
  INPUT: X_k and transition y
  OUTPUT: set X_k and X_{k+1}

{
Put y into X_{k+1};
Search y in X_k and delete y from X_k.
}

- **Divide (X_k) //Divide X_k into, Y_j, Y_{j+1}, Y_{j+2}...**
  INPUT: X_k
  OUTPUT: sets Y_j, Y_{j+1}, Y_{j+2}...

```
{
    While (there exist transitions in X_k)
      Do
      {
Move the first transition into Y_j;
for each transition x∈Y_i
for each transition y∈X_i
for i = 1 to |S|
if (x∈•s_i and y∈•s_i) or (x∈•s_i and y∈•s_i), then move y into y_j
end for
end for
end for
j = j+1;
      }
}
```

- **Outface (Y_j) //Output the outface subnet of transitions in Y_j**
  INPUT: Y_j
  OUTPUT: outface subnet N_j

```
{for each transition y∈X_i
for i = 1 to |S|
if there is an edge connecting s_i and y
then s_i and y are connected
end for
end for}
```

A polynomial-time decomposition algorithm: Now we present a polynomial-time decomposition algorithm for Petri nets based on indexes of transitions.

INPUT: a Petri net $\Sigma = (N, M) = (S, T; F, M_0)$
OUTPUT: Decomposition subnets of Σ

```
Step1:   // To obtain the presets and postset of each place
         for i = 1 to |S|
         for j = 1 to |T|
         if there is an edge from t_j to s_j then t_j∈•s_i
         if there is an edge from s_i to t_j then t_j∈s•_i
         end for
         end for
Step 2:  Store the markings of Σ in M
Step 3:  Put all transitions of Σ into X_1
Step 4:  Mark (X_1)
Step 5:  // obtain the index of each transition
         for k = 1 to |T|
         for j = 1 to |T|
         select y∈X_k such that λ[y] is not 0;
         Move (X_k, y);
         Mark (X_k) //updateλ[y]
         If λ[y] of each transition in X_k is 0,
         then break; //quit and execute next loop;
         end for
         Mark (X_{k+1})
         If each λ[y] of transition in X_{k+1} is 0, then break;
         end for
Step 6:  // Divide X_1, X_2, ... into Y_1, Y_2, ...
         for i = 1 to k
         Divide (X_k)
         end for
         Step 7: //Output each subnet of Σ
         for j = 1 to |T|
         Outface (Y_j)
         end for
Step 8:  // Output markings of each subnet
         for i = 1 to |T|
         for j = 1 to |S|
         if s_i is in subnet N_j, then add M[k] tokens to s_i
         end for
         end for
```

COMPLEXITY ANALYSIS OF THE ALGORITHM

Firstly, we analyze the complexity of four key functions required by the decomposition algorithm.

- **Step 1:** In function Mark (X_k), each if loop executes finite number of judgments and assignments, so the time complexity is only related to the layers of loops. There are three for loops, so the time complexity of the full function is O (m^2n), where n = |S| and m = |T| and the same to the followings
- **Step 2:** In function Move (X_k, y), the worst case of step Search y in X_k and delete y from X_k is to go through the whole set X_k, so the time complexity of this function is only O (m)
- **Step 3:** In function Divide (X_k), from the number of layers of the for loops, it can be determined that the time complexity of this function is also O (m^2n)
- **Step 4:** In function Outface (Y_j), each if loop executes finite number of judgments and assignments, so its time complexity is O(nm)

In the main algorithm, because the inner for loop executes finite number of judgments and assignments, the time complexity of the first step is O (nm). In step 2, there are n times for assignments, so the time complexity of Step 2 is O (n).In step 3, the time complexity is O (m).The time complexity of step 4 is actually same to that of the function Mark (X_k), so it is O (m^2n). In step 5, the step select $y \in X_k$ such that $\lambda[y]$ is not 0 is actually searching a place whose index is non-zero, so the worst time complexity of this step is O (m). The if loop is actually to go through the whole set X_k, so the time complexity of the inner for loop is O (m^2n). There are two outside layers of "for" loops, so the time complexity of step 5 is O (m^4n). In step 6, the inner function is Divide (X_k) and the time complexity of this step is O (m^3n). The time complexity of step 7 is mainly determined by the "for" loop and the function Outface (Y_j), so the time complexity of this step is O (m^2n). In step 8, there are n times for search and assignments, so the time complexity of this step is O (m^2n).

According to the time complexity of each step in the main algorithm, the time complexity of the whole algorithm is O ($nm + n + m + m^2n + m^4n + m^3n + m^2n + m^2n$) = O ($m^4n$). Therefore, the algorithm proposed in this study is a polynomial-time decomposition algorithm.

EXAMPLE

We take the Petri net for the problem of the dining philosophers in Fig. 1 as an example to show the implementation process of the algorithm proposed in the study.

Step 1: Assignment the input and output set of each place.

$$
\begin{aligned}
&{}^\bullet p_{11}=\{t_{12}\}, {}^\bullet p_{12}=\{t_{12},t_{52}\}, {}^\bullet p_{13}=\{t_{11}\}, p_{11}{}^\bullet=\{t_{11}\}, p_{12}{}^\bullet=\{t_{11},t_{51}\} \\
&p_{13}{}^\bullet=\{t_{12}\}, {}^\bullet p_{21}=\{t_{22}\}, {}^\bullet p_{22}=\{t_{12},t_{22}\}, {}^\bullet p_{23}=\{t_{21}\}, p_{21}{}^\bullet=\{t_{21}\} \\
&p_{22}{}^\bullet=\{t_{11},t_{21}\}, p_{23}{}^\bullet=\{t_{22}\}, {}^\bullet p_{31}=\{t_{32}\}, {}^\bullet p_{32}=\{t_{22},t_{32}\}, {}^\bullet p_{33}=\{t_{31}\} \\
&p_{31}{}^\bullet=\{t_{31}\}, p_{32}{}^\bullet=\{t_{21},t_{31}\}, p_{33}{}^\bullet=\{t_{32}\}, {}^\bullet p_{41}=\{t_{42}\}, {}^\bullet p_{42}=\{t_{32},t_{42}\} \\
&{}^\bullet p_{43}=\{t_{41}\}, p_{41}{}^\bullet=\{t_{41}\}, p_{42}{}^\bullet=\{t_{31},t_{41}\}, p_{43}{}^\bullet=\{t_{42}\}, {}^\bullet p_{51}=\{t_{52}\} \\
&{}^\bullet p_{52}=\{t_{42},t_{52}\}, {}^\bullet p_{53}=\{t_{51}\}, p_{51}{}^\bullet=\{t_{51}\}, p_{52}{}^\bullet=\{t_{41},t_{51}\}, p_{53}{}^\bullet=\{t_{52}\}
\end{aligned}
$$

Step 2: Put the number of tokens of each place to set M, so we get the set M = {p_{11} (1), p_{12} (1), p_{13} (0); p_{21} (1), p_{22} (1),p_{23}(0); p_{31} (1), p_{32} (1), p_{33} (0); p_{41} (1), p_{42} (1), p_{43} (0); p_{51} (1), p_{52} (1), p_{53} (0)}.

Step 3: Put all transitions of Σ into X_1, then we get X_1 = {t_{11}, t_{21}; t_{21}, t_{22}; t_{31}, t_{32}; t_{41}, t_{42} ;t_{51}, t_{52}}

Step 4: Assign mark to each place in X_1. Let t (k)represent that the mark of transition t is k, so we can obtain P_1 = {t_{11} (2), t_{12} (2); t_{21} (2), t_{22} (2); t_{31} (2), t_{32} (2); t_{41} (2), t_{42}(2); t_{51} (2), t_{52} (2)}.

Step 5: Decompose the net. Choose one transition t_{21} from X_1 whose mark is non-zero and move it to X_2, so X_2 = {t_{21}}. Update the mark of each transition in $X_1$and store them in P_1, then, P_1 = {t_{11} (1), t_{12} (2); t_{22} (2); t_{31} (1), t_{32} (2); t_{41} (2), t_{42} (2); t_{51} (2), t_{52} (2)}; Continue selecting transitions from X_1 and moving it to X_2. Without loss of generalization, transition t_{22} whose mark is non-zero is chosen and moved to X_2, so X_2 = {t_{21}, t_{22}}. Update the mark of each transition in X_1, so P_1 = {t_{11} (1), t_{12} (1); t_{31} (1); t_{41} (2); t_{51} (2), t_{52} (2)}. Select transition t_{31}whose mark is non-zero and move it to X_2, so X_2 = {t_{21}, t_{22}, t_{31}}. Update the mark of each transition in X_1, P_1 = {t_{11} (1), t_{12} (1); t_{32} (1); t_{41} (1); t_{51} (2), t_{52} (2)}. Select transition t_{32} and move it to X_2, so X_2 = {t_{21}, t_{22}, t_{31}, t_{32}}. Update the mark of each transition in X_1 again, P_1 = {t_{11} (1), t_{12} (1); t_{31}(1), t_{32} (2); t_{41} (2), t_{42}(2); t_{51}(2), t_{52} (2)}, so X_2 = {t_{21}, t_{22}; t_{31}, t_{32}; t_{41}}. Update the mark of each transition in X_1, P_1={t_{11} (1), t_{12} (1); t_{42} (1); t_{51} (1), t_{52} (2)}. Select transition t_{42} and move it to X_2 and X_2 = {t_{21}, t_{22}; t_{31}, t_{32}; t_{41}, t_{42}}. Update the mark of each transition in X_1, P_1={t_{11} (1), t_{12} (1); t_{51} (1), t_{52} (1)}. Select transition t_{51} and move it to X_2 and X_2 = {t_{21}, t_{22}; t_{31}, t_{32}; t_{41}, t_{42}, t_{51}}. Update the mark of each transition in X_1 and P_1 = {t_{11} (0), t_{12}(1); t_{51}(1)}. Select transition t_{52} and move it to t_{52} and X_2 = {t_{21}, t_{22}; t_{31}, t_{32}; t_{41}, t_{42}; t_{51}, t_{52}}. Update the mark of each transition in X_{51}, P_1 ={t_{11} (0), t_{12} (0)} and the selecting and moving operations on P_1 have been finished. Next, repeat the selecting and moving operations on X_2. At last, we can get X_1 = {t_{11}, t_{12}}; X_2 = {t_{21}, t_{22}}; X_3 = {t_{31}, t_{32}}; X_4 = {t_{41}, t_{42}}; X_5 ={ t_{51}, t_{52}}.

Step 6: To obtain connected subnets. Y_1 = {t_{11}, t_{12}}; Y_2 = {t_{21}, t_{22}}; Y_3 = {t_{31}, t_{32}}; Y_4 = {t_{41}, t_{42}}; X_5 ={ t_{51}, t_{52}}.

Step 7: Output the outface subnet. Firstly, we output the outface subnet of Y_1. For transition t_{11}, if there is an edge which connects t_{11} and s_i, then there will be an edge connecting t_{11} and s_i. Repeat the processing on t_{12}, then we get the subnet of Σ_1. Using the same method, we can get the subnets of Σ_1, Σ_2, Σ_3, Σ_4 and Σ_5.

Step 8: Output the tokens of each place in each subnet. Note that the number of tokens of each place of the original net has already been stored in the set M.

The output result of the whole algorithm is shown in Fig. 2.

CONCLUSION

In order to analyze properties of structure-complex Petri nets, the decomposition for a Petri net based on the indexes of transitions is a very convenient and useful approach. This study proposes a polynomial-time for the decomposition approach. The main data structures required and four key functions contained in the decomposition algorithm are addressed in details. It is proved that the proposed decomposition algorithm is a polynomial-time algorithm which provides methods for property analysis of structure-complex Petri nets.

ACKNOWLEDGMENTS

This study is supported partly by the NSFC under Grant No.60603090, 90718011 and 50875158; Science and Technology Development Fund of Shandong Province of China (2010GSF10811); the Excellent Young Scientist Foundation of Shandong Province of China under Grant No. BS2009DX004; the Special Fund for Fast Sharing of Science Study in Net Era by CSTD (20093718110008); the Open Project of Computer Architecture Lab of ICT, CAS, (No. ICT-ARCH200807); the Research Foundation of Qingdao under Grant No.10-3-3-32-nsh; Foundation for Distinguished Young Scholars of SDUST and the Taishan Scholar Program of Shandong Province.

REFERENCES

Cui, T., Q. Zeng and D. Zhang, 2011. Recognition algorithm design and complex analysis for languages of S-nets. Inform. Technol. J., 10: 106-112.

Esparza, J., 1994. Reduction and synthesis of live and bounded free choice petri nets. Inform. Comput., 114: 50-87.

Koriem, S.M., 1999. Fast and simple decomposition techniques for the reliability analysis of interconnection networks. J. Syst. Software, 45: 155-171.

Kwang, H.L., J. Favrel and P. Baptiste, 1987. Generalized Petri net reduction method. IEEE Trans. Syst. Man Cybernetics, 17: 297-303.

Murata, T., 1989. Petri nets: Properties, analysis and applications. Proc. IEEE, 77: 541-580.

Wang, H.Q. and Q.T. Zeng, 2008. Modeling and analysis for workflow constrained by resources and non-determined time: An approach based on petri nets. IEEE Trans. Syst. Man Cybernetics Part A: Syst. Hum., 38: 802-817.

Zeng, Q.T., 2004. Behavior descriptions of structure-complex Petri nets based on synchronous composition. J. Software, 15: 327-337.

Zeng, Q., 2007. Two symmetrical decomposition methods for structure-complex Petri net and their applications. Proceedings of 8th ACIS International Conference on Software Engineering, July 30-Aug. 1, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp: 1101-1106.

Zeng, Q.T. and H. Duan, 2007. Behavior description for complex flexible manufacturing system based on decomposition of Petri net. Int. J. Comput. Syst. Sci. Eng., 22: 359-363.

Zeng, Q., 2008. A construction method for the process expression of petri net based on decomposition. Inform. Technol. J., 7: 420-429.

Zeng, Q., X. Hu, J. Zhu and H. Duan, 2008. A polynomial-time decomposition algorithm for a petri net based on indexes of places. J. Applied Sci., 8: 4668-4673.