Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 14 (19): 2246-2257, 2014
ISSN 1812-5654 / DOI: 10.3923/jas.2014.2246.2257

Service Compositon Based on Enhanced Logic Petri Nets

YuHui Ning, YuYue Du and ShuXia Yu
Shandong University of Science and Technology, Qingdao, 266590, China

Abstract: With the development of information technology, the quantity of web services in internet has increased rapidly. The time complexity of service composition becomes higher. To solve this problem, a new method of service composition is proposed based on Enhanced Logic Petri nets (ELPNs) in this study. The main innovation is the construction of a composition library. The experiment in this study shows the time complexity of service composition is decreased. Firstly, web services and service compostion are modeled by ELPNs. Then, the reachability of ELPNs is analyzed. All cases of service composition are obtained based on the ELPNs model of service composition. A composition library is constructed. The method of service composition is proposed based on the composition library. Moreover, some theorems are given, such as enabled conditions of transitions, marking computing. Some algorithms are introduced, such as reachable markings, service composition and so on. Finally, the validity and advantages of proposed methods are illustrated by experiments and comparative analysis.

Key words: Web service, service composition, enhanced logic Petri nets, reachablility analysis, composition library

INTRODUCTION

With the development of information technology, the web service based on XML (Extensible Markup Language) has developed rapidly (D'Mello and Ananthanarayana, 2010). Web service is the computer program which is marked by the URL (Uniform Resource Locator). Web service can be used distributely on the cross-platform system. Because of the above advantages of web service, the quantity of services has increased rapidly (Liu *et al*., 2007). Meanwhile, the demands for web services are also increased. Moreover, there are many demands which can not be satisfied by a single web service. Thus, there is the problem that how to compose single web service to satisfy the user requirement. To solve this problem, the researches on service composition (Wang, 2011) were studied by scholars.

There have been already a lot of achievements on service composition. In order to minimize the duration of service compositon, the concepts of service clusters (Sheng *et al*., 2009) has been introduced. The services which are similar on inputs and outputs should be grouped when service composition (Deng and Du, 2012). In order to unify the rules of service composition, the concept of semantic was introduced into web service composition (Tao, 2012). The information representation can be unified base on the semantic vocabulary tree (Lei and Duan, 2009). Moreover, the types of service composition were given, such as simple service composition, labeled composition, parallel composition and hybrid composition (Deng and Du, 2013).

From the above researches, the main method for service composition is service discovery. When the demand is given by the user, the system would discover the web services from service clusters and compose them to the user (Xie, 2011). Howere, the quantity of services has increased rapidly. Web service providers have given a large number of services in the internet. Although, clustering services can minimize the cardinal number for service discovery, the total time complexity is high for service composition.

To solve the above problem, a new method of service composition is proposed based on Enhanced Logic Petri Nets (ELPNs) in this study. The innovation of this study has three aspects. Firstly, the composition library is constructed. All cases of service composition can be found from composition library. When the demand is given by the user, the system can found the case of service composition which can satisfy the user's demand from composition library and not have to find web services from service clusters. The experiment in this study virified the time complexity of service composition is decreased. Secondly, the web services and service composition are modeled by ELPNs. Logic Petri Nets (LPNs) are the mathematical tool and suitable to describe batch processing function and passing value

Corresponding Author: Yu Yue Du, Shandong University of Science and Technology, Qingdao, 266590, China
Tel: 86 532 86058031 Fax: 86 532 86057622

indeterminacy in cooperative systems (Du and Jiang, 2002; Du *et al*., 2011). The inputs, outputs and other aspects of web service have the characteristics of batch processing function and passing value indeterminacy. So, the web service can be modeled properly by LPNs (Du *et al*., 2008, 2009). Howere, it's difficult to use LPNs to decscribe the case that the output and input places of a transition are all restricted by a logic expression at the same time. Thus, the ELPNs is introduced and defined in this study. The web services and service composition are modeled by ELPNs conveniently. Thirdly, the reachability of ELPNs is analyzed. Then, all cases of service composition are obtained from the ELPNs model of service composition. Moreover, the definitions of web serive and user' demand are given. Some theorems are proposed, such as enabled conditions of transitions, marking computing. Some algorithms are introduced, such as reachable markings, service composition and so on. The validity and advantages of proposed methods are illustrated by some experiments and their comparative analysis.

ELPNs MODEL OF WEB SERVICE

The definitions of web service and ELPNs are showed in this section. The ELPNs model of web service is also introduced. A brief introduction of Petri Nets (PNs) (Hu *et al*., 2010a, b) is given as follows:

Definition 1: N = (P, T, F) is a net, where:

- P is a finite set of places
- T is a finite set of transitions with $P\cup T \neq \varnothing$ and $P\cap T = \varnothing$
- $F\subseteq(P\times T)\cup(T\times P)$ is a set of arcs
- $dom(F)\cup cod(F) = P\cup T$, where:

$$dom(F) = \{x\in P\cup T|\exists y\in P\cup T: (x, y)\in F\}$$

$$cod(F) = \{x\in P\cup T|\exists y\in P\cup T: (y, x)\in F\}$$

Definition 2: $x\in P\cup T$ is a node in N = (P, T, F). ${}^{\bullet}x = \{y|(y, x)\in F\}$ denotes a pre-set of x and $x^{\bullet} = \{y|(x, y)\in F\}$ denotes a post-set of x. If $X\subseteq P\cup T$, the pre-set and post-set of X are represented by ${}^{\bullet}X = \cup_{x\in X}{}^{\bullet}x$ and $X^{\bullet} = \cup_{x\in X}x^{\bullet}$, respectively.

Definition 3: PN = (P, T, F, M_0) is a marked PN, where:

- N = (P, T, F) is a net
- M: $P\rightarrow Z$ is a marking function, where M_0 is the initial marking. Z = {0, 1, 2,...} is a natural number set
- Transition firing rules:
  - t is enabled at M if $\forall p\in{}^{\bullet}t$: $M(p)\geq 1$, represented by M[t>
  - If t is enabled, it can fire and a new marking M' is generated from M, represented by M[t>M', where:

$$M'(p) = \begin{cases} M(p)+1 & \text{if } p\in t^{\bullet}-{}^{\bullet}t \\ M(p)-1 & \text{if } p\in{}^{\bullet}t-t^{\bullet} \\ M(p) & \text{else} \end{cases}$$

In order to model web service conveniently, the definition of ELPNs is introduced as follows:

Definition 4: Let LN = (P, T, F, L). ELPN = (LN, M) be an enchanced logic Petri nets, where:

- P is a finite set of places
- T is a finite set of transitions, $T\cup P \neq \varnothing$, $P\cap T = \varnothing$, $\forall t\in T$: ${}^{\bullet}t\cap t^{\bullet} = \varnothing$. $\forall t\in T$, the input places of t are restricted by a logic input expression $f_I(t)$ and the output places of t are restricted by a logic output expression $f_O(t)$
- $F\subseteq(P\times T)\cup(T\times P)$ is a finite set of directed arcs
- L is a mapping set from the transitions to the logic input and output expressions, i.e., $\forall t\in T$, $L(t) = <f_I(t), f_O(t)>$
- M: $P\rightarrow\{0, 1\}$ is a marking function, where $\forall p\in P$, M(p) is the number of tokens in p
- The logic operator of logic expressions in ELPN is only "$\wedge$"
- Transition firing rules: $\forall t\in T$, $f_I(t)|_M$ = .T., where .T. denotes the logic value 'true'. .F. denotes the logic value 'false'. If M[t>M', then $\forall p\in{}^{\bullet}t$: M'(p) = 0; $\forall p\in t^{\bullet}$ must satisfy $f_O(t)|_{M'}$ = .T. and $\forall p\notin{}^{\bullet}t\cup t^{\bullet}$: M'(p) = M(p)

Form definition 4, the differences between ELPN and LPN is that the transitions in ELPN are restricted by the logic input expressions and output expressions at the same time. The other aspect is that the logic operator of logic expressions in ELPN is only "$\wedge$". The definition of web service is proposed as follows:

Definition 5: Let webservice = (Identity, Inputs, Outputs, Relations) be a web service, where:

- Identity denotes the unique mark number of the web service
- Inputs = {$Input_1$, $Input_2$,⋯, $Input_j$} is a finite set which contains the input parameter of the web service

- Outputs = {$Output_1$, $Output_2$,⋯, $Output_k$} is a finite set which contains the output parameter of the web service
- Relations = {$Relation_1$, $Relation_2$,⋯, $Relation_l$} is a finite set which contains the logic relations between inputs and outputs of the web service
- The format of the item of relations is <input logic expression, output logic expression> = <($Input_w \wedge Input_r \wedge \ldots \wedge Input_t$), ($Output_h \wedge Output_q \wedge \ldots \wedge Output_p$)>. It means that if the inputs of webservice are $Input_w$, $Input_r$, ⋯ and $Input_t$, the outputs of webservice are $Output_h$, $Output_q$, ⋯ and $Output_p$

The algorithm for modeling web service by ELPNs is introduced as follows:

Algorithm 1: Modeling web service by ELPNs

Input: Web Service $Webservice_m$ = (Identity, Inputs, Outputs, Relations)

Output: The ELPNs model of Webservice

Step 1: Create and clear an ELPN $ELPN_a$ = (P, T, F, L, M_0). Create a Web service Wservice = $Webservice_m$. Create a variable x = 1

Step 2: Traverse Wservice.Relations

2.1: Suppose the current item of Relations is $Relation_i$. Create a new ELPN $ELPN_b$ = (P, T, F, L, M_0), where T = {t}, $|{}^{\bullet}t|$ = |Wservice. Inputs|, $|t^{\bullet}|$ = |Wservice.Outputs|. t is labeled by Wservice.Identity"+"x, i.e., $t_{Identity+x}$. x = x+1

2.1.1: Traverse the place set ${}^{\bullet}t$

2.1.1.1: Suppose the current item of ${}^{\bullet}t$ is $p_z \in P$. Then traverse Wservice.Inputs

2.1.1.1.1: Suppose the current item is $Input_y$. The label of the place which labeled by p_z is replaced by $Input_y$. And delete $Input_y$ from Wservice.Inputs.Return

2.2.1: Traverse the place set $t^{\bullet}$

2.2.1.1: Suppose the current item of $t^{\bullet}$ is $p_n \in P$. Then traverse Wservice.Outputs

2.2.1.1.1: Suppose the current item is $Output_x$. The label of the place which labeled by p_n is replaced by $Output_x$. And delete $Output_x$ from Wservice.Outputs. Return

2.2: Set $L(t) \leq f_I(t)$, $f_O(t) \geq Relation_i$

2.3: Put $ELPN_b$.P to $ELPN_a$.P. Put $ELPN_b$.T to $ELPN_a$.T. Put $ELPN_b$.F to $ELPN_a$.F. Put $ELPN_b$.L to $ELPN_a$.L. Set $ELPN_a.M_0$ to be zero

Step 3: Output $ELPN_a$

Algorithom 1 presents a method for modeling web services by ELPNs. An example is given as follows:

Example 1: Let $webservice_1$ = ($Identity_1$, $Inputs_1$, $Outputs_1$, $Relations_1$) to be a web service, where $Identity_1$ = 1, $Inputs_1$ = {a, b, c, d}, $Outputs_1$ = {e, f, g, h}, $Relations_1$ = {<(a), (e)>, <(b), (f)>, <($c \wedge d$), ($g \wedge h$)>}. According to algorithm 1, the ELPNs model of $webservice_1$ is showed in Fig. 1.

From Fig. 1, the web service $webservice_1$ is represented by ELPNs all-sidedly. The service composition will be modeled based on ELPNs in the next sections.

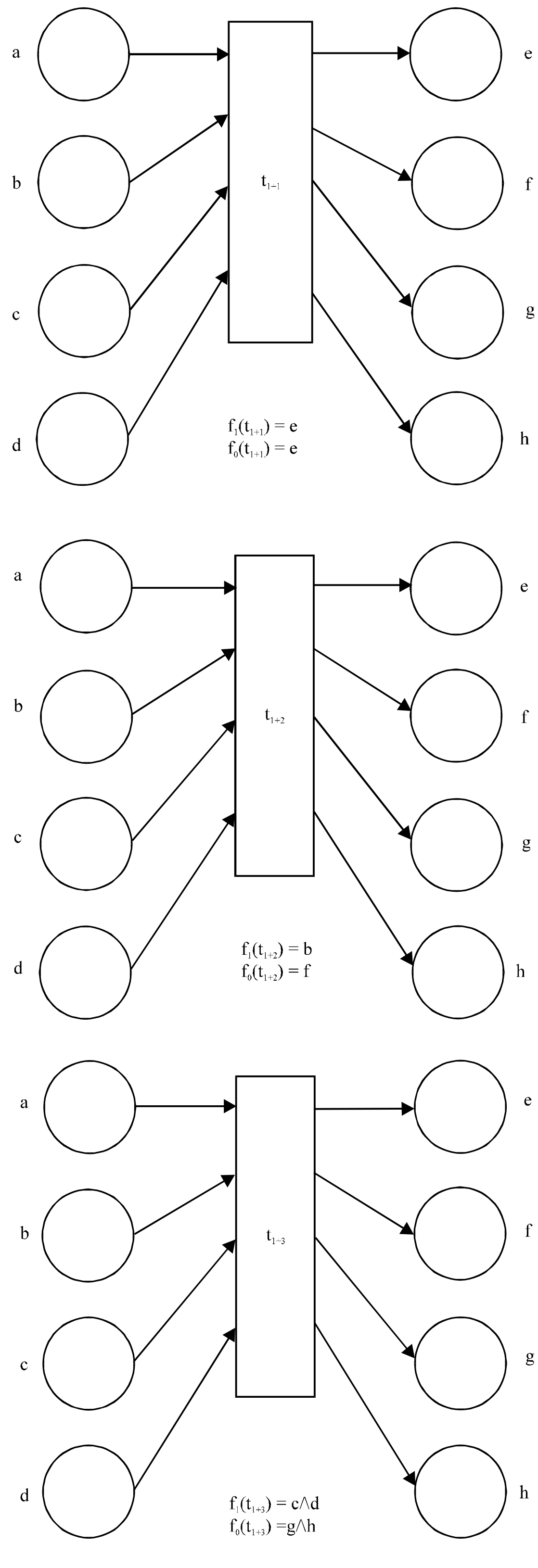

Fig. 1: ELPNs model of $webservice_1$

ELPNs MODEL OF SERVICE COMPOSITION

The ELPNs model of service composition is given in this section. Firstly, the definition and operational rules of operator "≡" is given as follows:

Definition 6: Let $ELPN_j = (P, T, F, L, M_0)$ is an ELPN. p_1 and p_2 are the labels of two places in P. Then, the rules of operator "≡" is shown as follows:

$$p_1 \equiv p_2 = \begin{cases} 1, \text{ if the string "} p_1 \text{"} \\ \text{is the same as "} p_2 \text{"} \\ 0, \text{ Else} \end{cases}$$

Example 2: $ELPN_j = (P, T, F, L, M_0)$ is showed in Fig. 2. From Fig. 2, the Name and Grade are labels of two places in P. From definition 6, Name ≡ Grade = 0.

The algorithm for searching the label of a place in ELPNs from the input or output logic expressions of transitions is given as follows:

Algorithm 2: Searching the label from logic expression

Input: The label of a place in ELPNs p_i and the input or output logic expression of a transition t $f(t) = (a_1 \wedge a_2 \wedge \ldots \wedge a_j)$

Output: The result

Step 1: Create a label set C = Ø and a constant x = 0

Step 2: From the logic expression $f(t) = (a_1 \wedge a_2 \wedge \ldots \wedge a_j)$. Put $a_1, a_2, \cdots, a_j$ to the label set C

Step 3: Traverse the label set C

3.1: Suppose the current item of C is C_k. If the result of $C_k \equiv p_i$ is 1, then x = 1. Return

Step 4: If x is 1, then output the result that "have found". Else, output the result that "not found"

The algorithm for modeling service composition by ELPNs is introduced as follows:

Algorithm 3: Modeling service composition by ELPNs

Input: The web service set Q = {$webservice_1$, $webservice_2, \cdots,$ $webservice_m$}

Output: The ELPNs model of service composition

Step 1: Create and clear an ELPN $ELPN_a = (P, T, F, L, M_0)$

Step 2: Traverse web service set Q

2.1: Suppose the current item of Q is $webservice_e$ = (Identity, Inputs, Outputs, Relations). $Webservice_e$ can be modeled by ELPNs according to algorithm 1 and the ELPNs model named $ELPN_b$. Then, Put $ELPN_b$.P to $ELPN_a$.P. Put $ELPN_b$.T to $ELPN_a$.T. Put $ELPN_b$.F to $ELPN_a$.F. Put $ELPN_b$.L to $ELPN_a$.L. Set $ELPN_a.M_0$ to be zero

Step 3: Traverse $ELPN_a$.T

3.1: Suppose the current item of T is t_n. Traverse the place set ${}^\bullet t_n \cup t_n^\bullet$

3.2: Suppose the current item of P is labeled by p_t. From algorithm 2, search p_t from the input and output logic expressions of the transition t_n. If the result is "not found", then delete the place p_t and delete the arcs connected to p_t

Step 4: Traverse $ELPN_a$.T

4.1: Suppose the current item of T is t_m. Traverse the place set ${}^\bullet t_m \cup t_m^\bullet$

4.1.1: Suppose the current item of P is labeled by p_r. Traverse the place set ${}^\bullet t_m \cup t_m^\bullet$ and except p_r

4.1.1.1: Suppose the current item of P is p_z. If $p_z \equiv p_r = 1$, then delete the place p_z. And the arcs connected to p_z are changed to connect p_r

4.1.1.2: Traverse $ELPN_a$.T and except t_m

4.1.1.2.1: Suppose the current item of T is t_s. Traverse the place set ${}^\bullet t_s \cup t_s^\bullet$

4.1.1.2.1.1: Suppose the current item of P is labeled by p_v. If $p_v \equiv p_r = 1$, then delete the place p_v. And the arcs connected to p_v are changed to connect p_r

Step 5: Output $ELPN_a$

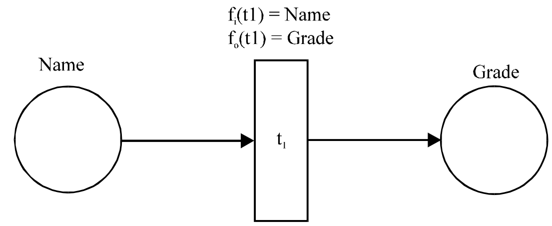

Fig. 2: An ELPNs $ELPN_j$

Algorithom 2 presents a method for modeling services composition by ELPNs. An example is given as follows:

Example 3: There are the web service set Q = {$webservice_1$, $webservice_2$}, where $webservice_1$ is the same as the web service in example 1. $Webservice_2$ = ($Identity_2$, $Inputs_2$, $Outputs_2$, $Relations_2$), where $Identity_2 = 2$, $Inputs_2$ = {a, j, d}, $Outputs_2$ = {e, i, h}, $Relations_1$ = {<(a), (e)>, <(j∧d), (i∧h)>}. According to algorithm 3, the ELPNs model of service composition with $webservice_1$ and $webservice_2$ is showed in Fig. 3.

The ELPNs model after the 3rd step of algorithm 3 is showed in Fig. 3.

The ELPNs model after the 5th step of algorithm 3 is showed in Fig. 4.

From Fig. 4, the service composition between $webservice_1$ and $webservice_2$ is modeled by ELPNs.

REACHABILITY ANALYSIS OF ELPNs

Enabled transition: The enabled conditions of transitions are given here.

From definition 4, the transition of an ELPN is restricted by the logic input and output expressions. For $\forall t \in T$, if $f_I(t)|_M$ = **.T.**, then M[t>. The definition of enabling label set is defined below:

Definition 7: For ELPN = (P, T, F, L, M_0) and $t_i \in T$. $f_I(t_i) = (a_1 \wedge a_2 \wedge \ldots \wedge a_j)$ is the logic input expression of t_i. $V_i = (a_1, a_2, \cdots, a_j)^T$ denotes the enabling label set of t_i. And $V = \{V_i | t_i \in T, i \in \{1, 2, \ldots, |T|\}\}$ denotes the enabing label set of T.

The definition and operational rules of operator Θ is given as follows:

Definition 8: Let $ELPN_m = (P, T, F, L, M_0)$ be an ELPN and $M \in R(M_0)$. V is the enabing label set of T. $t_i \in T$ and $V_i = (a_1, a_2, \cdots, a_j)^T$. The rules of operator "Θ" is shown as follows:

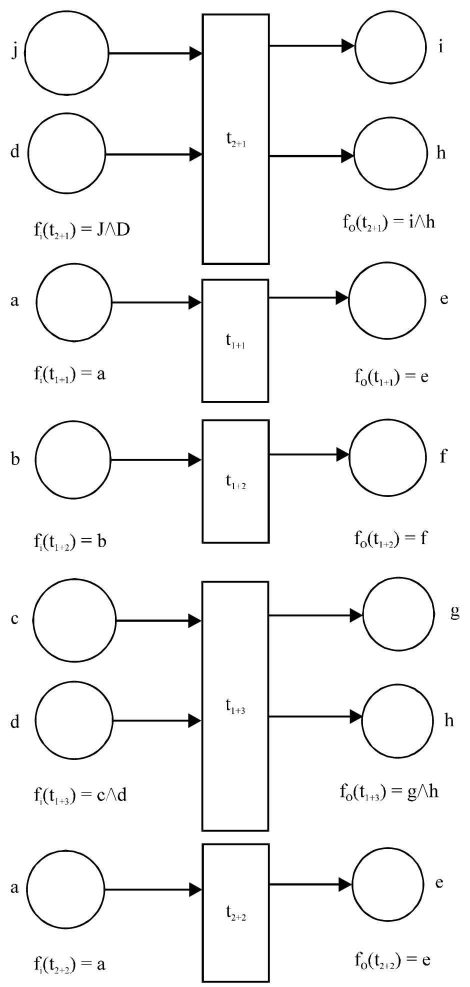

Fig. 3: ELPNs model after the 3rd step of algorithm 3

$$M\Theta V_i = \begin{cases} 1, \text{ if } M(a_1) = M(a_2) = \cdots = M(a_j) = 1 \\ 0, \text{ Else} \end{cases}$$

The enabled conditions of transitions are obtained from theorem 1.

Theorem 1: For ELPN = (P, T, F, L, M_0) and M$\in$R(M_0). V is the enabing label set of T. $\forall t_i \in T$, t_i is enabled at M if and only if $M\Theta V_i = 1$.

Proof: [Necessity] For $\forall t_i \in T$, since V is the enabing label set of T and V_i is the enabing label set of t_i. Suppose $V_i = (a_1, a_2, \cdots, a_j)^T$. Thus, the input expression of t_i is $f_I(t_i) = (a_1 \wedge a_2 \wedge \ldots \wedge a_j)$. Suppose t_i is enabled at M, i.e., M[t_i>. From definition 4, since M[t_i>, thus, the logic value of the logic input expression of t_i is .T. at the Marking M. i.e. $f_I(t_i)|_M$ = .T.. So, $M(a_1) = M(a_2) = \cdots = M(a_j) = 1$. i.e., $M\Theta V_i$.

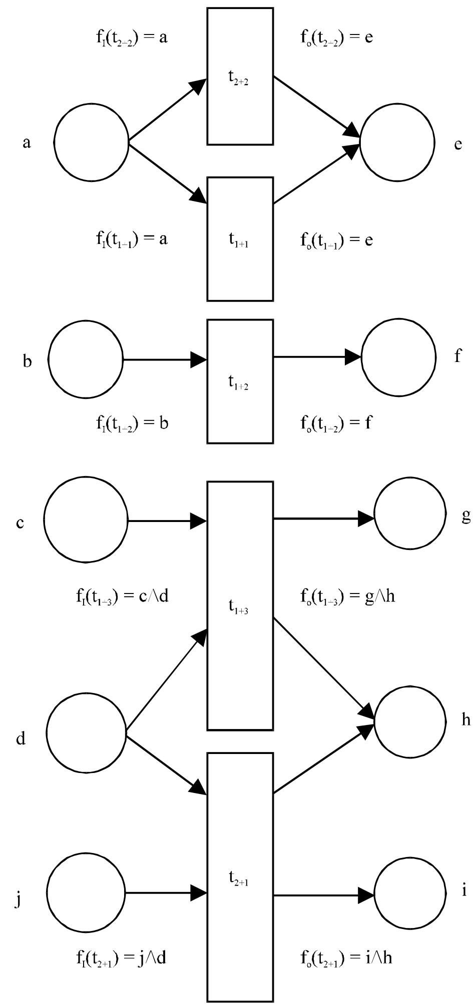

Fig. 4: ELPNs model after the 5th step of algorithm 3

[Sufficiency] For $\forall t_i \in T$, since V_i is the enabing label set of t_i. Suppose $V_i = (a_1, a_2, \cdots, a_j)^T$. Thus, the input expression of t_i is $f_I(t_i) = (a_1 \wedge a_2 \wedge \ldots \wedge a_j)$. Since, $M\Theta V_i = 1$, thus, $M(a_1) = M(a_2) = \cdots = M(a_j) = 1$. So, the logic value of the logic input expression of t_i is .T. at the Marking M. i.e., $f_I(t_i)|_M$ = .T.. From definition 4, since M[t_i>.

From theorem 1, the enabled conditions of transitions are given and the enabled transitions at the current marking can be obtained. The theorem for computing marking after an enabled transition fired is introduced in the next section.

Marking computing: The method for computing marking after an enabled transition fired is given in this section. The definition of output label set is defined below.

Definition 9: For ELPN = (P, T, F, L, M_0) and $t_i \in T$. $f_O(t_i) = (b_1 \wedge b_2 \wedge \ldots \wedge b_j)$ is the logic output expression of t_i

$B_i = (b_1, b_2, \cdots, b_j)^T$ denotes the output label set of t_i. And $B = \{B_i | t_i \in T, i \in \{1,2, \ldots, |T|\}\}$ denotes the output label set of T.

The marking computing theorem is introduced as follows:

Theorem 2: For ELPN = (P, T, F, L, M_0), where $P = \{p_1, p_2, \cdots, p_m\}$ and $M \in R(M_0)$. V is the enabing label set of T. B is the output label set of T. $\forall t_i \in T$, If $M[t_i>M'$, then $M' = (M'(p_1), M'(p_2), \cdots, M'(p_m))^T$, where for $j \in \{1, 2, \cdots, m\}$:

$$M'(p_j) = \begin{cases} 0, & p_j \in V_i \\ 1, & p_j \in B_i \\ M(p_j), & \text{Else} \end{cases} \quad (1)$$

Proof: For $\forall t_i \in T$, since, V is the enabing label set of T and V_i is the enabing label set of t_i. From definition 7, the item in $^\bullet t_i$ is the same as the item in V_i. Since, $M[t_i>M'$. From definition 4, $\forall p \in {}^\bullet t_i$: $M'(p) = 0$. Thus, $\forall p \in V_i$: $M'(p) = 0$. For $\forall t_i \in T$, Since, B is the output label set of T and B_i is the output label set of t_i. From definition 9, the item in $t_i^\bullet$ is the same as the item in B_i. Since, $M[t_i>M'$. From definition 4, $\forall p \in t^\bullet$ must satisfy $f_O(t)|_{M'} = .T.$. Since, the logic operator of logic expressions in ELPN is only "$\wedge$". Thus, $\forall p \in B_i$: $M'(p) = 1$. Since, $M[t_i>M'$. From Definition 4, $\forall p \notin {}^\bullet t_i \cup t_i^\bullet$: $M'(p) = M(p)$, thus, $\forall p \notin V_i \cup B_i$: $M'(p) = M(p)$.

Theorem 2 presents a method for computing marking when a enabled transition fired. All reachable markings can be obtained in the next section.

Reachability analysis: The method for obtaining all reachable markings is introduced in this section. The algorithm for computing reachable markings of an ELPN is showed as follows:

Algorithm 4: Computing reachable markings

Input: ELPN = (P, T, F, L, M_0)
Output: All reachable markings of ELPN
Step 1: Create four constants q = 0, w = 1, e = 2, i = 3. Create two variables x = y = 0. Create and clear a three-dimensional array A[n][i] and n = +∞
Step 2: The initial marking M_0 is set to the current marking M
Step 3: Traverse ELPN.T
3.1: Suppose the current item of T is t_i. From theorem 1, if t_i is enabled. Suppose $M[t_i>M'$, then, M' would be obtained from theorem 2
3.2: Put M to A[x][q]. Put M' to A[x][w]. Put t_i to A[x][e]
3.3: For y from 0 to x-1
3.3.1: if A[x][w] is equal to M', then return. Else, x = x+1
3.2.2: M' is set to the current marking M and back to step 3
Step 4: Output A[n][i]

Algorithm 4 presents a method for obtaining all reachable markings of an ELPN. From algorithm 4, the three-dimensional array A[n][i] contains all reachability information of an ELPN.

Example 4: Suppose the current marking M of ELPN set Σ in Fig. 4 is {M(a) = M(f) = M(c) = M(d) = M(j) = 1, M(e) = M(b) = M(g) = M(h) = M(i) = 0}. From theorem 1, the transitions t_{1+1}, t_{2+2} are enabled at M. From theorem 2, when the transition t_{1+1} is fired, the marking M of ELPN set Σ in Fig. 4 is {M(e) = M(b) = M(c) = M(h) = M(i) = 1, M(a) = M(f) = M(g) = M(d) = M(j) = 0}. From algorithm 4, A[0][0] = {M(a) = M(f) = M(c) = M(d) = M(j) = 1, M(e) = M(b) = M(g) = M(h) = M(i) = 0}, A[0][1] = {M(e) = M(f) = M(c) = M(d) = M(j) = 1, M(a) = M(b) = M(g) = M(h) = M(i) = 0}, A[0][2] = t_{1+1}.

CONSTRUCTION OF COMPOSITION LIBRARY

The method for constructing composition library is introduced in this section. The definition of composition library is given as follows:

Definition 10: Scomlry = (C_ELPNs, C_Relations) is a composition library, where:

- C_ELPNs is the ELPNs model of servcie composition
- C_Relations = {$Relation_1, Relation_2, \cdots, Relation_i$} is a finite set which contains the mapping relations of inputs, outputs, transitions in C_ELPNs
- The format of the item of C_Relations is <inputs of C_ELPNs, outputs of C_ELPNs, transitions of C_ELPNs> = <($Input_a, Input_b, \cdots, Input_c$), ($Output_d, Output_e, \cdots, Output_f$), ($t_g, t_h, \cdots, t_i$)>. It means that if there are the inputs of service composition are $Input_a$, $Input_b$, ⋯ and $Input_c$, the outputs of service composition would be $Output_d$, $Output_e$, ⋯ and $Output_f$, after the transitions t_g, t_h, ⋯, t_i fired successively

The definition and operational rules of operator "⊗" is given as follows:

Definition 11: Let $ELPN_m$ = (P, T, F, L, M_0) be the ELPNs model of service composition, where $P = \{p_1, p_2, \cdots, p_m\}$ and $M \in R(M_0)$. Q is the label set of the places in $ELPN_m$.P. The rules of operator "⊗" is shown as follows:

$$M \otimes Q = M, \text{ where, For } j \in \{1, 2, \cdots, m\}$$

$$\begin{cases} M(p_j) = 1, & p_j \in Q \\ M(p_j) = 0, & \text{Else} \end{cases}$$

The function for combination is given as follows:

Function 1: COMB(s, n, m, top, W, queue, array)

Input: The viriables s, n, m. The label set W. One-dimensional array queue and array

Output: The label set W

Step 1: Create a viriable i. If s>n, then return

Step 2: If top is equal to m, then create a new label sub-set of W named as U = Ø

2.1: For i = 0 to m

2.1.1: Put queue[i] to U

2.2: Put U to W and return

Step 3: Set queue[top] = array[s]. top = top+1. Excecute the function W = COMB(s+1, n, m, top, W, queue, array). top = top-1. Excecute the function W = COMB(s+1, n, m, top, W, queue, array)

Step 4: Output the label set W

The algorithm for constructing initial marking set is given as follows:

Algorithm 5: Construction initial marking set

Input: The ELPNs model of service composition $ELPN_i$ = (P, T, F, L, M_0)

Output: The initial marking set Q

Step 1: Create the initial marking set Q = Ø. Create the label set W = Ø. Create viriables l = m = n = top = 0. Create an one-dimensional array queue[n] = {0}

Step 2: Traverse $ELPN_i$.T

2.1: Suppose the current item of T is t_j. Traverse the place set $^{\bullet}t_j$

2.2: Suppose the current item of $^{\bullet}t_j$ is p_k. Put the label of p_k to W

Step 3: Create an one-dimensional array array[o] and o = |W|

3.1: Traverse W

3.1.1: Suppose the current itme of W is p_b, then, array[l] = p_b. l = l+1

Step 4: Set W = Ø. Set n = o. For m = 0 to o

4.1: Excecute the function W = COMB(0, n, m, top, W, queue, array)

Step 5: Traverse the sub-set of W

5.1: Suppose the current item of W is U_k. Create a new marking of $ELPN_i$ named M_c. Put $M_c \otimes U_k = M_c$ to Q

Step 6: Output the initial marking set Q

The algorithm for constructing composition library is introduced as follows:

Algorithm 6: Construction composition library

Input: The ELPNs model of service composition $ELPN_i$ = (P, T, F, L, M_0)

Output: The composition library Scomlry = (C_ELPNs, C_Relations)

Step 1: Create four constants q = 0, w = 1, e = 2, i = 3. Create two marking set $Q_1 = Q_2$ = Ø. Create variables k = j = r = v = f = 0. Create and clear an one-dimensional array Z[u] and u = +∞

Step 2: From algorithm 5, the initial marking set of $ELPN_i$ can be obtained. Suppose the initial marking set of $ELPN_i$ is Q

2.1: Traverse Q

2.1.1: Suppose the current item of Q is M_c. Set $ELPN_i.M_0 = M_c$. From algorithm 4, all reachable markings of $ELPN_i$ at M_0 can be obtained. Suppose the three-dimensional array A[n][i] obtained from algorithm 4 contains all reachability information of $ELPN_i$ at M_0

2.1.2: For r = 0 to n-1. For v = 0 to 1

2.1.2.1: Traverse Q_1

2.1.2.1.1: Suppose the current item of Q_1 is M_s, if M_s is equal to A[r][v], then f = 1 and return

2.1.2.2: If f is equal to 0, then put A[r][v] to Q_1 and Q_2, set Z[j] = A[r][v] and j = j+1. Else, f = 0

2.1.3: j = j-1. Create an adjacent matrix matrix[j][j] and set all items in matrix[j][j] to +∞. Create a path matrix path[j][j] and set all items in path[j][j] to -1. Copy A[n][i] to B[n][i]. For r = 0 to n-1. For v = 0 to 1. For f = 0 to j-1

2.1.3.1: If B[r][v] is equal to Z[f], then replace B[r][v] with f

2.1.3.2: For r = 0 to n-1. Set matrix[B[r][q]][B[r][w]] = 1. Set path[B[r][q]][B[r][w]] = B[r][w]

2.1.4: For r = 0 to j-1. For v = 0 to j-1. For f = 0 to j-1

2.1.4.1: If (matrix[r][v]>matrix[r][f]+matrix[f][v]). Then, matrix[r][v] = matrix[r][f]+matrix[f][v] and path[r][v] = path[r][f]

2.1.5: Create a composition library $Scomlry_j$ = (C_ELPNs, C_Relations). Set $Scomlry_j$.C_ELPNs to $ELPN_i$ = (P, T, F, L, M_0). Set $Scomlry_j$.C_Relations to Ø

2.1.6: Traverse Q_1

2.1.6.1: Suppose the current item of Q_1 is M_s. For r = 0 to j-1. If Z[r] is equal to M_s, then return r. Traverse Q_2

2.1.6.1.1: Suppose the current item of Q_2 is M_k. For v = 0 to j-1. If Z[v] is equal to M_k, then return v

2.1.6.1.2: If matrix[r][v] is not +∞, then create a new item of $Scomlry_j$. C_Relations R_t and R_t = <$inputs_t$, $outputs_t$, $transitions_t$>. Set $inputs_t$ to M_s. Set $outputs_t$ to M_k. Create a transition set Q_3 = Ø. f = path[r][v]

2.1.6.1.3: If f≠v, then, for k = 0 to n-1

2.1.6.1.3.1: If A[k][q] is equal to Z[r] and A[k][w] is equal to Z[f], then, put A[k][e] to Q_3. Set r to f. Set f to path[f][v] and back to 7.1.3

2.1.6.1.4: Set $transitions_t = Q_3$

Step 3: Traverse C_Relations

3.1: Suppose the current item of C_Relations is R_t and R_t = <$inputs_t$, $outputs_t$, $transitions_t$>. Traverse C_Relations

3.1.1: Suppose the current item of C_Relations is R_x and R_x = <$inputs_x$, $outputs_x$, $transitions_x$>. If the item of $inputs_t$ is the same as $inputs_x$, the item of $outputs_t$ is the same as $outputs_x$ and the item of $transitions_t$ is the same as $transitions_x$, then, delete R_x from C_Relations

Step 4: Output the composition library $Scomlry_j$ = (C_ELPNs, C_Relations)

Algorithm 6 presents a method for constructing composition library.

Example 5: The ELPNs model of service composition $ELPN_i$ = (P, T, F, L, M_0) is the same as the ElPN in Fig. 4. From algorithm 5, the label set is {{a}, {b}, {c}, {d}, {j}, {a, b}, {a, c}, {a, d}, {a, j}, {b, c}, {b, d}, {b, j}, {c, d}, {c, j}, {d, j}, {a, b, c}, {a, b, d}, {a, b, j}, {a, c, d}, {a, c, j}, {a, d, j}, {b, c, d}, {b, c, j}, {b, d, j}, {c, d, j}, {a, b, c, d}, {a, b, c, j}, {a, b, d, j}, {a, c, d, j}, {b, c, d, j}, {a, b, c, d, j}}. From algorithm 6, the composition library can be constructed as Scomlry = (C_ELPNs, C_Relations), where C_ELPNs = $ELPN_i$ and C_Relations = {<{a}, {e}, t_{1+1}>, <{b}, {f}, t_{1+2}>, <{c,d}, {g,h}, t_{1+3}>, <{j,d}, {i,h}, t_{2+1}>}.

SERVICE COMPOSITION ORIENTED TO USER'S DEMANDS

The method for service composition oriented to user's demands is introduced in this section. The definition of user's demand is given as follows:

Definition 12: Udemand = (Id, Uinputs, Uoutputs) is an user's demand, where:

- Id denotes the unique mark number of the user's demand
- Uinputs = {$Input_1$, $Input_2$,⋯, $Input_j$} is a finite set which contains the input parameters of the user's demand
- Uinputs = {$Output_1$, $Output_2$, ⋯, $Output_k$} is a finite set which contains the output parameters of the user's demand

- The user's demand can be satisfied, if the outputs can be given by the service composition system based on the user's inputs
- The output and input parameters of user's demand is consistent with the input and output parameters of web services

The theorem for service composition oriented to user's demands is introduced as follows:

Theorem 3: $Udemand_i$ = (Id, Uinputs, Uoutputs) is an user's demand. $ELPN_j$ = (P, T, F, L, M_0) is the ELPNs model of service composition and $M_a \in R(M_0)$. The user's demand can be satisfied if and only if $(M_a \otimes Uinputs) \in R(M_0 \otimes Uoutputs)$.

Proof: [Necessity] From algorithm 1 and 3, the $Udemand_i$. Uinputs and $Udemand_i$.Uoutputs are the label sets of the places in $ELPN_j$.P. Thus, from definition 11, $M_a \otimes$Uinputs denotes the marking of $ELPN_j$ which is consistent with the user's input parameters. $M_0 \otimes$Uoutputs denotes the marking of $ELPN_j$ which is consistent with the user's output parameters. Since, the user's demand can be satisfied. From definition 12, the outputs can be given by the service composition system based on the user's inputs. Thus, the marking $M_a \otimes$Uinputs of $ELPN_j$ can be reached from the marking $M_0 \otimes$Uoutputs of $ELPN_j$ i.e., $(M_a \otimes Uinputs) \in R(M_0 \otimes Uoutputs)$.

[Sufficiency] Since, $(M_a \otimes Uinputs) \in R(M_0 \otimes Uoutputs)$, thus, the marking $M_a \otimes$Uinputs of $ELPN_j$ can be reached from the marking $M_0 \otimes$Uoutputs of $ELPN_j$. From definition 11, $M_a \otimes$Uinputs denotes the marking of $ELPN_j$ which is consistent with the user's input parameters. $M_0 \otimes$U outputs denotes the marking of $ELPN_j$ which is consistent with the user's output parameters. Thus, the outputs of the user's demand can be given by the service composition system based on the user's inputs. From definition 12, the user's demand can be satisfied.

Definition 13: Rtable = (Records) is the composition result for user's demand, where:

- Records = $\{record_1, record_2, \cdots, record_k\}$ is a finite set
- The format of the item of Records is <Rid, Rinputs, Routputs, Sid>, where:
  - Rid denotes the unique mark number of the item and created in ascending order successively
  - Rinputs denotes the input parameters
  - Routputs denotes the output parameters
  - Sid denotes the unique mark number of web service

Example 6: There is a composition result for user's demand $Rtable_i$ = ($Records_i$). Suppose $Records_i$ = {$record_1$, $record_2$}, where $record_1$≤1, {a}, {b}, 3> and $record_2$≤2, {b}, {c}, 5>. It means that the user's demand is input a and output c. The demand can be satisfied by the work flow that user inputs a and b can be obtained using the web service which Identity = 3. Then, the user inputs b and c can be obtained using the web service which Identity = 5.

The algorithm for service composition oriented to user's demands is introduced as follows:

Algorithm 7: Service composition oriented to user's demands

Input: The user's demand $Udemand_i$ = (Id, Uinputs, Uoutputs) and the composition library $Scomlry_j$ = (C_ELPNs, C_Relations)

Output: The composition result Rtable = (Records)

Step 1: Create the composition result $Rtable_k$ = (Records) = Ø. Create a variable x = 0

Step 2: Traverse $Scomlry_j$.C_Relations

2.1: Suppose the current item of C_Relations is R_k and $R_k = <inputs_k, outputs_k, transitions_k>$. If the items of $Udemand_i$.Uinputs are the same as the items of $R_k.inputs_k$ and the items of $Udemand_i$.Uoutputs are the same as the items of $R_k.outputs_k$, then, return $R_k.transitions_k$

Step 3: Suppose $transitions_k = \{t_1, t_2, \cdots, t_m\}$. Traverse $transitions_k$

3.1: Suppose the current item of $transitions_k$ is t_b. Suppose the C_ELPNs. $L(t_b) \le f_I(t_b), f_O(t_b)>$, where $f_I(t_b) = (a_1 \wedge a_2 \wedge \ldots \wedge a_e)$ and $f_O(t_b) = (c_1 \wedge c_2 \wedge \ldots \wedge c_w)$

3.2: Create a new item of $Rtable_k$.Records $record_u$≤Rid, Rinputs, Routputs, Sid>. Set $record_u$. Rid = x and x = x+1. Set $record_u$. Rinputs = $\{a_1, a_2, \cdots, a_e\}$ according to the logic input expression of t_b. Set $record_u$.Routputs = $\{b_1, b_2, \cdots, b_w\}$ according to the logic output expression of t_b. Set $record_u$.Sid = b according to the label of the transition t_b

Step 4: Output the composition result $Rtable_k$ = (Records)

EXPERIMENT AND COMPARISON

In order to verify the validity and advantages of proposed methods in this study, the experiment and comparison are given in this section. Because of no standard software and test data, one hundred services are defined according to definition 5. Although, the services defined in this study is not real, there are no effect on the result of the experiment.

Construction of web services: From definition 5, one hundred services are defined. For simplicity, the inputs and outputs of web services are defined as letters. The first six items of service set are showed in Table 1.

Table 1: First six web services

Identity	Inputs	Outputs	Relations
1	aeo	xae, u	<(aeo), (xae∧u)>
2	bea, ewr	cet, y	<(bea∧ewr), (cet∧y)>
3	cxa, hej, io	crw, ki	<(cxa∧hej∧io), (crw∧ki)>
4	dnm, ik	q	<(dnm∧ik), (q)>
5	evb, jlm	j, p	<(evb∧jlm), (j∧p)>
6	fie, kzx	fg	<(fie∧kzx), (fg)>

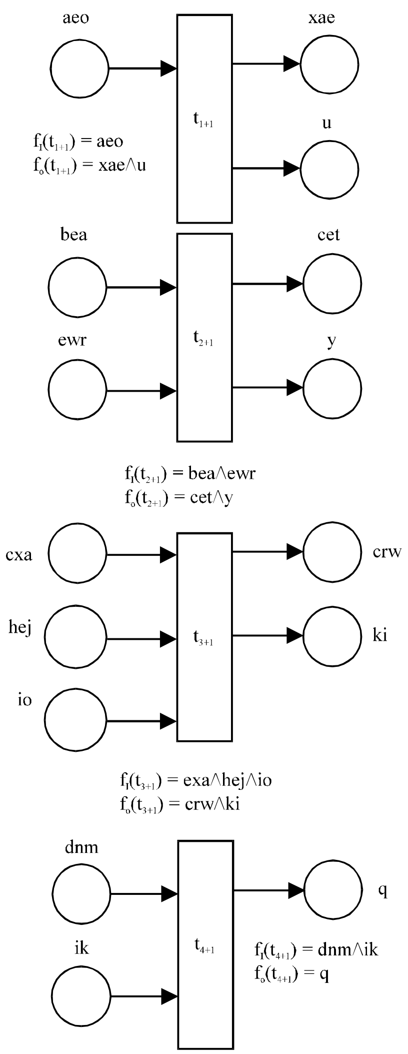

Fig. 5: First four ELPNs model of services

Model web services by ELPNs: From algorithm 1, the web services defined in this section can be model by ELPNs. The first four ELPNs models of services is showed in Fig. 5.

Model service composition by ELPNs: From algorithm 3, the services composition oriented to the services defined in this section can be model by ELPNs. The part of ELPNs model of service composition is showed in Fig. 6.

Construction of composition library: From algorithm 6, the composition library based on ELPNs model of service composition can be constructed as Scomlry = (C_ELPNs, C_Relations). C_ELPNs is equal to the ELPNs model of service composition and the first six items of C_Relations are shown below:

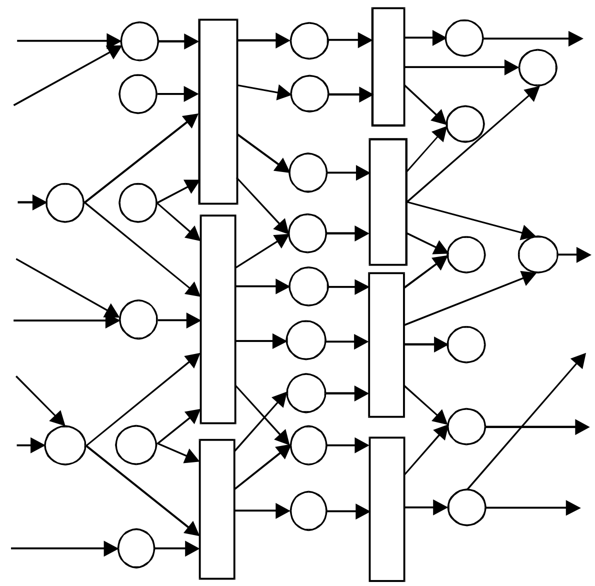

Fig. 6: Part of ELPNs model of service composition

Table 2: First three items of user's demands

Identity	Uinputs	Uoutputs
1	aeo	fg
2	cxa, hej, io	pts, tts
3	dnm, ik	q
4	c	d
5	vse	bet
6	wrt	xt

C_Relations

$$< \{aeo\}, \{xae, u\}, t_{1+1} >$$
$$< \{bea, ewr\}, \{cet, y\}, t_{2+1} >$$
$$< \{cxa, hej, io\}, \{crw, ki\}, t_{3+1} >$$
$$< \{dnm, ik\}, \{q\}, t_{4+1} >$$
$$< \{evb, jlm\}, \{j, p\}, t_{5+1} >$$
$$< \{fie, kzx\}, \{fg\}, t_{6+1} >$$

Construction of user's demands: From definition 12, fifty user's demands are defined. The outputs and inputs of user's demand are consistent with which in web services. The first three items of user's demands are showed in Table 2.

Service composition oriented to user's demands: From algorithm 7, all composition result can be obtained. The first six composition results binded to user are showed in Table 3.

Comparison: There are a lot of achievements on service composition. The innovation and superiority of this study have three aspects. Firstly, the composition library is constructed before service discovery. Secondly, the

Table 3: First six composition results

Demand.Id	Rtable.Records
1	<1, {aeo}, {xae, twe}, 18>, <2, {xae}, {kzx}, 93>, <3, {twe, {fie}, 47}>, <4, {fie, kzx}, {fg, 6}>
2	<1, {cxa, hej, io}, {crw, ki}, 3>, <2, {crw, ki}, {pts, tts}, 58>
3	<1, {dnm, ik}, {q}, 4>,
4	<1, {c}, {d}, 84>
5	Ø
6	<1, {wrt}, {aeo}, 77>, <2, {aeo}, {xae, twe}, 18>, <3, {twe}, {fie}, 47>, <4, {fie}, {xt}, 12>

ELPNs is introduced and defined for the first time. The methods for analyzing reachability of ELPNs are given. At last, the service composition is modeled by ELPNs. The comparison analysis is showed as follows:

- The comparison analysis with the time complexity for service composition oriented to user's demand is shown as follows:
  - Service composition oriented to user's demand from the proposed method in this study

Suppose the service composition library is constructed and named as $Scomlry_j$ = (C_ELPNs, C_Relations). Suppose | C_Relations| = n. From algorithm 7, if the item in C_Relations satisfied to user's demand is not found, the time complexity for service composition is O(n). Else, if the item in C_Relations satisfied to user's demand can be found and named as $R_k = <inputs_k, outputs_k, transitions_k>$. Suppose $|transitions_k|$ = m, thus, the time complexity for service composition is O(n+m):

  - Service composition oriented to user's demand from web service set Q (Sheng *et al*., 2009; Nayak and Lee, 2007)

Suppose |Q| = n, If the user's demand is segmented to m atomic demands completely. Obviously, the time complexity for service composition is O(n×m).

From above analysis the time complexity of service combination using the method proposed in this study is decreased:

- The comparison analysis with the optimality of service composition oriented to user's demand is showed as follows:
  - The result of service composition generated by the propose method in this study

From algorithm 5 and 6, all reachable markings of ELPNs model of service composition can be obtained. In order to construct the optimal composition library, the adjacent matrix is constructed. Every path from the inputs to outputs of user's demand is obtained through the classic shortest path first algorithm. For example, from Table 3, for the second user's demand, the service composition is that user inputs cxa, hej, io and crw, ki can be obtained using the web service which Identity = 3. Then, the user inputs crw, ki and pts, tts can be obtained using the web service which Identity = 58:

  - The result of service composition generated without the pretreatment for the shortest path (Lian and Zheng, 2011; Liu *et al*., 2010)

For the user's demand, if the demand is not satisfied by a single web service, the demand should be segmented to several sub-demands. Obviously, it is hard to segment user's demand optimally without the pretreatment of the shortest path computation. For example, from Table 3, the second user's demand is $Udemand_2$ = (2, {cxa, hej, io}, {pts, tts}). From the service composition oriented to $Udemand_2$ in Table 3, the optimal segment of $Udemand_2$ is $Udemand_{2\text{-}1}$ = (2-1, {cxa, hej, io}, {crw, ki}), $Udemand_{2\text{-}2}$ = (2-2, {crw, ki}, {pts, tts}). Howere, the segment of $Udemand_2$ would be not the same as the above case.

So, the method proposed in this study is superior on the optimality of service composition:

- The comparison analysis between ELPNs and LPNs with modeling web service is shown as follows:

Model web service by ELPNs: From definition 5, the web service has three aspects inputs, outputs and the relations between input and output parameters. From definition 4, the transitions in ELPNs are restricted by logic input and output expressions at the same time. Thus, From algorithm 1, the web service can be model by ELPNs conveniently. For example, Let $webservice_k = (Identity_k, Inputs_k, Outputs_k, Relations_k)$ to be a web service, where $Identity_k = k$, $Inputs_k = \{a, b, c, d\}$, $Outputs_k = \{e, f, g, h\}$, $Relations_k = \{<(c\wedge d), (g\wedge h)>\}$. According to algorithm 1, the ELPNs model of $webservice_k$ is showed in Fig. 7.

Model web service by LPNs: From the definition of LPNs, there are three kinds of transitions in LPNs. T_D denotes the traditional transitions. T_I denotes the transitions restricted by the logic input expressions (Du and Guo, 2009). T_O denotes the transitions restricted by the logic output expressions. The LPNs model of $webservice_k$ is showed in Fig. 8.

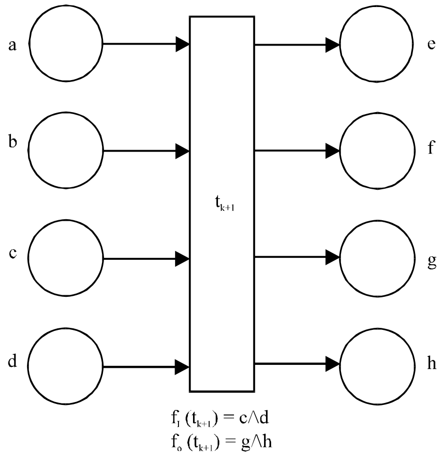

Fig. 7: ELPNs model of webservice$_k$

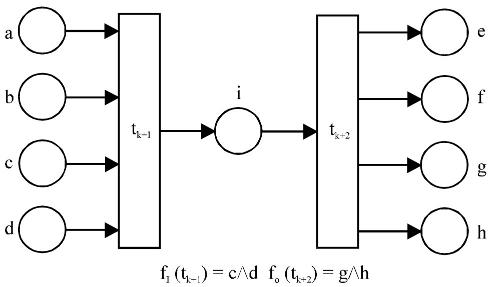

Fig. 8: LPNs model of webservice$_k$

From Fig. 7 and 8, obviously, the transition and place in Fig. 8 is more than that in Fig. 7. Thus, it is easier for modeling web service by ELPNs than by LPNs.

From the above analysis, the validity, superiority and effectiveness of the proposed method are illustrated.

CONCLUSION

A new method for service composition is introduced in this study. In order to model web service conveniently, an Enhanced Logic Petric nets is defined in this study firstly. The comparison has already verified that ELPNs is the abstract and extension of LPNs and high-level PNs. The enabled conditions of transitions and marking computation theorem of ELPNs are given. The reachability of ELPNs is analyzed. The ELPNs models of web service and service composition are given. All reachable markings of ELPNs model of service composition are obtained. The service composition library is constructed based on ELPNs model of service composition. Every path in composition library from the inputs to outputs of user's demand is obtained through the classic shortest path first algorithm. The method for service composition oriented to user's demand is introduced. From the simulation experiments and comparisons, the time complexity of service composition oriented to user's demand is decreased and the result binding to user is optimal.

Further study will be the properties analysis of ELPNs, including fairness, reversibility and the construction of reachable marking gragh. Moreover, service discovery, service binding and service substitution will be studied.

ACKNOWLEDGMENTS

This study is supported by the National Basic Research Program of China under Grant No. 2010CB328101; the National Natural Science Foundation of China under Grant No. 61170078 and 61173042; the Doctoral Program of Higher Education of the Specialized Research Fund of China under Grant No. 20113718110004; Basic Research Program of Qingdao City of China under Grant No. 13-1-4-116-jch and the SDUST Research Fund of China under Grant No. 2011KYTD102.

REFERENCES

D'Mello, D.A. and V.S. Ananthanarayana, 2010. Dynamic selection mechanism for quality of service aware web services. Enterp. Inform. Syst., 4: 23-60.

Deng, S.Y. and Y.Y. Du, 2012. Logic Petri net based model for web service cluster. J. Comput. Appl., 32: 2328-2332.

Deng, S.Y. and Y.Y. Du, 2013. Web service composition approach based on service cluster and Qos. J. Comput. Appl., 33: 2167-2170.

Du, Y.Y. and C.J. Jiang, 2002. Formal representation and analysis of batch stock trading systems by logical Petri net workflows. Proceedings of the 4th International Conference on Formal Engineering Methods, October 21-25, 2002, Shanghai, China, pp: 221-225.

Du, Y.Y., C.J. Jiang and M.C. Zhou, 2008. A petri nets based correctness analysis of internet stock trading systems. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev., 38: 93-99.

Du, Y.Y., C.J. Jiang and M. Zhou, 2009. A petri net-based model for verification of obligations and accountability in cooperative systems. IEEE Trans. Syst. Man Cybern. A, 39: 299-308.

Du, Y.Y. and B.Q. Guo, 2009. Logic petri nets and equivalency. Inform. Technol. J., 8: 95-100.

Du, Y.Y., L. Qi and M.C. Zhou, 2011. A vector matching method for analysing logic Petri nets. Enterp. Inform. Syst., 5: 449-468.

Hu, H., M.C. Zhou and Z. Li, 2010a. Low-cost and high-performance supervision in ratio-enforced automated manufacturing systems using timed Petri nets. IEEE Trans. Autom. Sci. Eng., 7: 933-944.

Hu, H., M.C. Zhou and Z. Li, 2010b. Algebraic synthesis of timed supervisor for automated manufacturing systems using Petri nets. IEEE Trans. Autom. Sci. Eng., 7: 549-557.

Lei, L.H. and Z.H. Duan, 2009. Modeling and verifying composite web services based on semantic annotated Petri nets. J. Front. Comput. Sci. Technol., 3: 173-187.

Lian, C.S. and C. Zheng, 2011. Web service discovery algorithm based on comprehensive ontology similarity computation. Comput. Appl. Software, 28: 273-276.

Liu, S.L., Y.X. Liu, F. Zhang and G.F. Tang, 2007. A dynamic web services selection algorithm with Qos global optimal in web services composition. J. Software, 18: 646-656.

Liu, G.J., C.J. Jiang and M.C. Zhou, 2010. Two simple deadlock prevention policies for S3PR based on key resource/operation-place pairs. IEEE Trans. Autom. Sci. Eng., 7: 945-957.

Nayak, R. and B. Lee, 2007. Web service discovery with additional semantics and clustering. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, November 2-5, 2007, Fremont, CA., pp: 555-558.

Sheng, Q.Z., B. Benatallah, Z. Maamar and A.H.H. Ngu, 2009. Configurable composition and adaptive provisioning of web services. IEEE Trans. Serv. Comput., 2: 34-49.

Tao, S.G., 2012. Research on ontology based semantic web services discovery technology. South-Central university for nationalities, China.

Wang, S., 2011. Semantic-based Web service compostion on model research and implementation. Hunan University, China.

Xie, L.L., 2011. Ontology-based semantic web services cluster and discovery. Tianjin University, China.