Service Composition 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 composition 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, reachability 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 distributively 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 composition, 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). However, 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 verified 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). However, it's difficult to use LPNs to describe 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 service 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 $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs
- $\text{dom}(F), \text{cod}(F) = P \cup T$, where:
  \[
  \text{dom}(F) = \{ x \in P \cup T | \exists y \in P \cup T : (x, y) \in F \} \\
  \text{cod}(F) = \{ x \in P \cup T | \exists y \in P \cup T : (y, x) \in F \}
  \]

**Definition 2:** $x \in P \cup T$ is a node in $N = (P, T, F)$, $x = \{ y(y, x) \in F \}$ denotes a pre-set of $x$ and $x' = \{ y(x, y) \in F \}$ denotes a post-set of $x$. If $X \subseteq P \cup T$, the pre-set and post-set of $X$ are represented by $X' = \bigcup_{x \in X} x$ and $X' = \bigcup_{x \in X} x'$, 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 $\forall p \in T: M(p) \geq 1$, represented by $M[t>\cdot$.
  - If $t$ is enabled, it can fire and a new marking $M'$ is generated from $M$, represented by $M[t>\cdot$, where:
  \[
  M'(p) = \begin{cases} 
  M(p) + 1 & \text{if } p \in t' - t \\
  M(p) - 1 & \text{if } p \in t - t' \\
  M(p) & \text{else}
  \end{cases}
  \]

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 enhanced logic Petri nets, where:

- $P$ is a finite set of places
- $T$ is a finite set of transitions, $T \cup P \neq \emptyset$, $P \cap T = \emptyset$
- $\forall t \in T$, $t^\prime - t = \emptyset$, $\forall t \in T$, the inputs 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 set of directed arcs
- $L$ is a mapping set from the transitions to the logic input and output expressions, i.e., $\forall t \in T, L(t) = \langle f_i(t), f_o(t) \rangle$
- $M: P \rightarrow \{0, 1\}$ 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 "\&"
- Transition firing rules: $\forall t \in T, f_o(t)|_{M} = T, \forall t \in T$, denotes the logic value 'true'. $F$ denotes the logic value 'false'. If $M[t>\cdot$, then $\forall p \in T: M'(p) = 0$; $\forall p \in T$, $M'(p) = M(p)$

From 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 "\&". The definition of web service is proposed as follows:

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

- Identity denotes the unique mark number of the web service
- Inputs = $\{ \text{Input}_1, \text{Input}_2, ..., \text{Input}_n \}$ is a finite set which contains the input parameter of the web service
Outputs = \{Output_1, Output_2, ..., Output_n\} is a finite set which contains the output parameter of the web service.

Relations = \{Relation_1, Relation_2, ..., Relation_m\} 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 \langle input logic expression, output logic expression \rangle = \langle Input_1 \land \ldots \land Input_n, Output_1 \land \ldots \land Output_m \rangle \rangle. It means that if the inputs of web service are Input_1, Input_2, ..., Input_n, and Input_j, the outputs of web service are Output_1, Output_2, ..., and Output_m.

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

**Algorithm 1: Modeling web service by ELPNs**

**Input:** Web Service WebService = (Identity, Inputs, Outputs, Relations)

**Output:** The ELPNs model of Web service

**Step 1:** Create and clear an ELPN ELPN_i = (P, T, F, I, M), Create a Web service WebService = WebService_i. Create a variable x = 1

**Step 2:** Traverse WebService.Relations

2.1: Suppose the current item of Relations is Relation. Create a new ELPN ELPN_i = (P, T, F, I, M), where T = \{\}, \|T\| = |WebService.Inputs|, \|F\| = |WebService.Outputs|. \(t_i\) is labeled by WebService.Identity \(\forall i\), i.e., \(L_{t_i} = x = 1\)

2.1.1: Traverse the place set \(T\)

2.1.1.1: Suppose the current item of \(T\) is \(p_i \in P\). Then traverse WebService.Inputs

2.1.1.1.1: Suppose the current item is Input_j. The label of the place which labeled by \(p_i\) is replaced by \(Input_j\). And delete \(Input_j\) from WebService.Inputs. Return

2.1.1.2: Traverse the place set \(T\)

2.1.1.2.1: Suppose the current item of \(T\) is \(p_i \in P\). Then traverse WebService.Outputs

2.1.1.2.1.1: Suppose the current item is Output_k. The label of the place which labeled by \(p_i\) is replaced by \(Output_k\). And delete \(Output_k\) from WebService.Outputs. Return

2.2: Set \(I(0) = I(0) \land F(0) \lor Relation\)

2.3: Put ELPN_P to ELPN_P. Put ELPN_T to ELPN_T. Put ELPN_F to ELPN_F. Put ELPN_L to ELPN_L. Set ELPN_M to be zero

**Step 3:** Output ELPN_i

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

**Example 1:** Let web service = (Identity, Inputs, Outputs, Relations,) be a web service, where Identity = 1, Inputs = \{a, b, c, d\}, Outputs = \{e, f, g, h\}, Relations = \{<a(b), <(c(d), <(e(f), <(h)\rangle\rangle. According to algorithm 1, the ELPNs model of web service, is showed in Fig. 1.

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

Fig. 1: ELPNs model of web service.
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 = (P, T, F, L, M_0) be an ELPN. _p_ and _p_ are the labels of two places in P. Then, the rules of operator "=" is shown as follows:

\[
\begin{align*}
\text{if } p_i = p_j & : \text{if the string } "p_i" \\
& \text{is the same as } "p_j" \\
\text{else } & 0.
\end{align*}
\]

**Example 2:** ELPN = (P, T, F, L, M_0) is shown 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_ and the input or output logic expression of a transition (R) = (α/α′/α″... / α).

**Output:** The result

**Step 1:** Create a label set _C_ = 0 and a constant _x_ = 0

**Step 2:** From the logic expression R = (α/α′/α″... / α), put α, α′, α″, α... to the label set C

**Step 3:** Traverse the label set C

3.1: Suppose the current item of _C_ is _C_. If the result of _C_ = _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, webservice2, ..., webservice_n}

**Output:** The ELPNs model of service composition

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

**Step 2:** Traverse web service set Q

2.1: Suppose the current item of _Q_ is webservice, = (Identity, Inputs, Outputs, Relations). Webservice, can be modeled by ELPNs according to algorithm 1 and the ELPNs model named ELPN. Then, put ELPN_P to ELPN_Q. Put ELPN_ELPN_P to ELPN_Q. Put ELPN_ELPN_F to ELPN_Q. Put ELPN_ELPN_L to ELPN_Q. Set ELPN_M to be zero

**Step 3:** Traverse ELPN_Q, T

3.1: Suppose the current item of _T_ is _t_. Traverse the place set "t_0, t_1, ..".

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

**Step 4:** Traverse ELPN_T

4.1: Suppose the current item of _T_ is _t_. Traverse the place set "t_0, t_1, .."

4.1.1: Suppose the current item of _P_ is labeled by _p_. Traverse the place set "t_0, t_1, .." and except _p_.

4.1.1.1: Suppose the current item of _P_ is labeled by _p_. If _p_ = _p_, then delete the place _p_ and the arcs connected to _p_.

4.1.1.2: Traverse ELPN_T and except _t_.

4.1.1.2.1: Suppose the current item of _T_ is _t_. Traverse the place set "t_0, t_1, .."

4.1.1.2.1.1: Suppose the current item of _P_ is labeled by _p_. If _p_ = _p_, then delete the place _p_ and the arcs connected to _p_.

**Step 5:** Output ELPN

**Fig. 2:** An ELPNs ELPN_Q

Algorithm 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, webservice2, webservice3}, where webservice, is the same as the web service in example 1. Webservice2 = (Identity, Inputs, Outputs, Relations), where Identity = 2, Inputs = {a, b, c}, Outputs = {e, i, h}, Relations = {<a(a), <b(b), <c(c)}, (i/h>l)>. According to algorithm 3, the ELPNs model of service composition with webservice, and webservice2, is shown in Fig. 3.

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

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

From Fig. 4, the service composition between webservice, and webservice2, 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 _t_ ∈ _T_ if _f(t)_ = _T_., then M[/>. The definition of enabling label set is defined below:

**Definition 7:** For _ELPN = (P, T, F, L, M_0)_ and _t_ ∈ _T_. _f(t)_ = (α/α′/α″... / α) is the logic input expression of _t_. _V_ = (α_1, α_2, ..., α_n) is the enabling label set of _t_. And _V_ = {V | _t_ ∈ _T_, _V_ ∈ {1, 2, ..., |_T|}} denotes the enabling label set of _T_.

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

**Definition 8:** Let _ELPN_n = (P, T, F, L, M_0) be an ELPN and M ∈ R(M_0). _V_ is the enabling label set of _T_. _t_ ∈ _T_ and _V_ = (α_1, α_2, ..., α_n) is the enabling label set of _T_. The rules of operator "Θ" is shown as follows:
Fig. 3: ELPNs model after the 3rd step of algorithm 3

![Diagram](image3.png)

**Theorem 1:** For ELPN = (P, T, F, L, M₀) and Mk(M₀). V is the enabling label set of T. ∀t ∈ T, tᵢ is enabled at M if and only if MθVᵢ = 1

**Proof:** [Necessity] For ∀t ∈ T, since Vᵢ is the enabling label set of T and Vᵢ is the enabling label set of tᵢ. Suppose Vᵢ = (a₁, aᵢ, ..., aᵢ). Thus, the input expression of tᵢ is f(tᵢ) = (aᵢ / a₁ ∧ a₂ ∧ ... ∧ aᵢ). Suppose tᵢ is enabled at M, i.e., M[V]. From definition 4, since M[V], thus, the logic value of the logic input expression of tᵢ is Tᵢ at the Marking M. i.e., f(tᵢ)[₀] = Tᵢ. So, M(a₁) = M(aᵢ) = ... = M(aᵢ) = 1. i.e., MθVᵢ = 1

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

![Diagram](image4.png)

[Sufficiency] For ∀t ∈ T, since Vᵢ is the enabling label set of tᵢ. Suppose Vᵢ = (a₁, aᵢ, ..., aᵢ). Thus, the input expression of tᵢ is f(tᵢ) = (aᵢ / a₁ ∧ a₂ ∧ ... ∧ aᵢ). Since, MθVᵢ = 1, thus, M(a₁) = M(aᵢ) = ... = M(aᵢ) = 1. So, the logic value of the logic input expression of tᵢ is Tᵢ at the Marking M. i.e., f(tᵢ)[₀] = Tᵢ. From definition 4, since M[V]

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, M₀) and t ∈ T. f(t) = (b₁ ∧ b₂ ∧ ... ∧ bᵢ) is the logic output expression of tᵢ
The marking computing theorem is introduced as follows:

**Theorem 2:** For $\text{ELPN} = (P, T, F, L, M_0)$, where $P = \{p_1, p_2, \ldots, p_n\}$ and $\text{MeR}(M_0)$, $V$ is the enabling label set of $T$. $B$ is the output label set of $T$. $\forall t \in T$, if $M[t \triangleright M']$ then $M' = (M'(p_1), M'(p_2), \ldots, M'(p_n))^T$, where for $j \in \{1, 2, \ldots, m\}$:

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

**Proof:** For $\forall t \in T$, since $V$ is the enabling label set of $T$ and $V_i$ is the enabling label set of $t$. From definition 7, the item in $t$ is the same as the item in $V$. Since, $M[t \triangleright M']$. From definition 4, $\forall p \in T$: $M'(p) = 0$. Thus, $\forall p \in V_i$: $M'(p) = 0$. For $\forall t \in T$, since $B$ is the output label set of $T$ and $B_i$ is the output label set of $t$. From definition 9, the item in $t$ is the same as the item in $B$. Since, $M[t \triangleright M']$. From definition 4, $\forall p \in T$: $M'(p) = 1$. Since, $M[t \triangleright M']$. From definition 4, $\forall p \in V_i$: $M'(p) = M(p)$, thus, $\forall p \in V_i$ or $B$: $M'(p) = M(p)$.

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 shown 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 $a = 0$, $w = 1$, $e = 2$, $i = 3$. Create two variables $x = y = 0$. Create and clear a three-dimensional array $A[n][i]$ and $n = i + 1$.

**Step 2:** The initial marking $M_0$ is set to the current marking $M$

**Step 3:** Traverse ELPN.

3.1: Suppose the current item of $T$ is $t_i$. From theorem 1, if $t_i$ is enabled. Suppose $M[t \triangleright M']$, then $M'$ would be obtained from theorem 2


3.3: For $y$ from 0 to $y+1$

3.3.1: if $A[x][w]$ is equal to $M'$, then return, Else, $x = x+1$

3.3.2: $M'$ 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 $\Sigma$ 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 $t_1, t_2$ are enabled at $M$. From theorem 2, when the transition $t_1$ is fired, the marking $M$ of ELPN set $\Sigma$ in Fig. 4 is $M(a) = M(b) = M(c) = M(d) = M(j) = 1$, $M(e) = M(f) = M(g) = M(h) = M(i) = 0$. From algorithm 4, $A[0][0] = \{M(a) = M(f) = M(c) = M(d) = M(j) = 1, M(e) = M(b) = M(g) = M(h) = M(i) = 0\}$, $A[0][1] = \{M(e) = M(f) = M(c) = M(d) = M(j) = 1, M(a) = M(b) = M(g) = M(h) = M(i) = 0\}$, $A[0][2] = t_1, t_2$.

**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:** $\text{Scomlry} = (C_{ELPNs}, C_{Relations})$ is a composition library, where:

- $C_{ELPNs}$ is the ELPNs model of service composition
- $C_{Relations} = \{\text{Relation}_1, \text{Relation}_2, \ldots, \text{Relation}_n\}$ 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 $C_{ELPNs}, outputs \text{ of } C_{ELPNs}, transitions \text{ of } C_{ELPNs}> = \langle Input_{in}, Input_{out}, \text{ Output}_{in}, \text{ Output}_{out}, (t_{in}, t_{out}, \ldots, t_{out}) \rangle$. It means that if there are the inputs of service composition are $Input_{in}$, $Input_{out}$ and $Input_{in}$, the outputs of service composition would be $Output_{in}$, $Output_{out}$ and $Output_{out}$ after the transitions $t_{in}, t_{out}, \ldots, t_{out}$ fired successively.

The definition and operational rules of operator “$\circ$” is given as follows:

**Definition 11:** Let $\text{ELPN}_n = (P, T, F, L, M_0)$ be the ELPNs model of service composition, where $P = \{p_1, p_2, \ldots, p_n\}$ and $\text{MeR}(M_0)$, $Q$ is the label set of the places in $\text{ELPN}_n.P$. The rules of operator “$\circ$” is shown as follows:

$$M \circ Q = M, \text{ where, for } j \in \{1, 2, \ldots, m\}$$

$$M(p_j) = 1, p_j \in Q$$

$$M(p_j) = 0, \text{ Else}$$

The function for combination is given as follows:
Function 1: COMB(k, n, m, top, W, queue, array)

Input: The variables s, a, n, m. The label set W. One-dimensional array queue and array.
Output: The label set W
Step 1: Create a variable i. If i = n, then return.
Step 2: If top is equal to m, then create a new label sub-set of W named as U = 0
2.1.1: For i = 0 to m
2.1.1.1: Put queue[i] to U
2.2.2: Put U to W and return
Step 3: Set queue[n] = array[n], top = top + 1. Execute the function W = COMB(s+1, n, m, top, W, queue, array, top = top + 1. Execute 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 = (P, T, F, L, M,
Output: The initial marking set Q
Step 1: Create the initial marking set Q = 0. Create the label set W = 0. Create variables l = n = n = top = 0. Create an one-dimensional array queue[n] = [0]
Step 2: Traverse ELPN, T
2.1: Suppose the current item of T is t ∈ T. Traverse the place set t → 2.2: Suppose the current item of a = p. Put the label of p ∈ W
Step 3: Create an one-dimensional array array[n] and a = |W|
3.1: Traverse W
3.1.1: Suppose the current item of W is p, then array[p] = 1 = l+1
Step 4: Set W = 0. Set n = 0. For m = 0 to 0
4.1: Execute 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. Create a new marking of ELPN, named M, Put M, M0 = U, 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 = (P, T, F, L, M,
Output: The composition library Scomly = (C, ELPNs, C_Relations)
Step 1: Create four constants q = 0, w = 1, e = 2, i = 3. Create two marking set Q = Q0 = 0. Create variables k = j = r = v = f = 0. Create and clear an one-dimensional array Z[a] and u = e+a
2.1: Traverse Q
2.1.1: Suppose the current item of Q is M. Set ELPN, M0 = M0. From algorithm 4, all reachable markings of ELPN, at M0 can be obtained. Suppose the three-dimensional array A[n][i] obtained from algorithm 4 contains all reachability information of ELPN, at M0
2.1.2: For r = 0 to m - 1. For v = 0 to 1
2.1.2.1: Traverse Q
2.1.2.1.1: Suppose the current item of Q is M. If M is equal to A[n][i], then f = 1 and return
2.1.2.2: If f = 0, then put A[n][i] to Q and Q0, set Z[i] = A[n][i] and j = j + 1. Else, f = 0
2.1.3: Create an adjacent matrix matrix[j][j] and set all items in matrix[j][j] to 1. Create a path matrix path[i][j] and set all items in path[i][j] to 1. Copy A[n][i] to B[i][0]. For r = 0 to n. For v = 0 to 1. For f = 0 to 1
2.1.3.1: If B[i][j] is equal to Z[i], then replace B[i][j] with f
2.1.3.2: For f = 0 to m. Set matrix[B[i][j][B[i][j]]] = 1. Set path[B[i][j][B[i][j]]] = B[i][j]
2.1.4: For r = 0 to j. For v = 0 to i. For f = 0 to j
2.1.4.1: If (matrix[i][j][matrix[i][j] = matrix[j][j] = matrix[i][j] or path[i][j] = path[i][j]
2.1.5: Create a composition library Scomly = (C, ELPNs, C_Relations). Set Scomly, C_LPNs to ELPN = (P, T, F, L, M0). Set Scomly, C_Relations to 0
2.1.6: Traverse Q
2.1.6.1: Suppose the current item of Q is M. If r = 0 to j. If Z[i] is equal to M, then return. Traverse Q
2.1.6.1.1: Suppose the current item of Q is M. If r = 0 to j. If Z[i] is equal to M, then return. Traverse Q
2.1.6.1.2: If matrix[i][j] is not ≠, then create a new item of Scomly, C_Relations and R = <inputs, outputs, transitions>. Set inputs, to P. Set outputs, to P. Create a transition set Q0 = 0. f = path[i][j]
2.1.6.1.3: If (r + 1 to n - 1, then for k = 0 to n - 1
2.1.6.1.3.1: If A[k][b] is equal to Z[i] and A[b][b] is equal to Z[i], then, put A[k][b] to Q. Set r to f. Set to path[i][j] and back to 2.1.6.1.3
2.1.6.1.4: Set transitions, to Q
2.2: Traverse C_Relations
3.1: Suppose the current item of C_Relations is R. If r = 0 to j, then create a new item of Scomly, C_Relations and R = <inputs, outputs, transitions>. Set inputs, outputs, to R. Traverse C_Relations
3.1.1: Suppose the current item of C_Relations is R. If r = 0 to j, then create a new item of Scomly, C_Relations and R = <inputs, outputs, transitions>. If the item of inputs, is the same as inputs, the item of outputs, is the same as outputs, and the item of transitions, is the same as transitions, then, delete R from C_Relations
2.2: Traverse C_Relations
Step 4: Output the composition library Scomly = (C, ELPNs, C_Relations)

Algorithm 6 presents a method for constructing composition library.

Example 5: The ELPNs model of service composition ELPN = (P, T, F, L, M0) is the same as the ELPN in Fig. 4. From algorithm 5, the label set is \{a, b, c, d, i, j, a, b, l, a, c, j, a, d, j, b, e, b, d, a, b, j, b, c, a, c, d, c, d, a, d, l, b, c, d, b, c, j, b, d, j, c, d, j, a, b, c, j, a, b, d, j, c, a, c, d, b, c, d, j\}. From algorithm 6, the composition library can be constructed as Scomly = (C, ELPNs, C_Relations), where C_LPNs = ELPN, and C_Relations = \{<a, e, t_{1,2}, <b, f, t_{2,3}, <c, d, g,h, t_{3,4}, <d, e, t_{4,1}, \ldots}\}

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, Inputs, Outputs) is an user’s demand, where:

- Id denotes the unique mark number of the user’s demand
- Inputs = \{Input1, Input2, ..., Inputi\} is a finite set which contains the input parameters of the user’s demand
- Outputs = \{Output1, Output2, ..., Outputn\} is a finite set which contains the output parameters of the user’s demand

2252
• 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:** $J_{\text{Input}} = (I_{\text{Input}}, U_{\text{Input}}, O_{\text{Output}})$ is an user’s demand. ELPN = $(P, T, F, L, M_n)$ is the ELPNs model of service composition and $M_\text{Input} \in \text{R}(M_n)$. The user's demand can be satisfied if and only if $(M_\text{Input} \in \text{R}(M_n)) \land (O_{\text{Output}} \in \text{R}(M_n))$.

**Proof:** [Necessity] From algorithm 1 and 3, the $J_{\text{Input}}$, $U_{\text{Input}}$, and $O_{\text{Output}}$ are the label sets of the places in ELPN, $P$. Thus, from definition 11, $M_\text{Input}$ denotes the marking of ELPN, which is consistent with the user's input parameters. $M_\text{Output}$ denotes the marking of ELPN, 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 $M_\text{Output}$ of ELPN can be reached from the marking $M_\text{Input}$ of ELPN, i.e., $(M_\text{Input} \in \text{R}(M_n)) \land (O_{\text{Output}} \in \text{R}(M_n))$.

[Sufficiency] Since, $(M_\text{Input} \in \text{R}(M_n)) \land (O_{\text{Output}} \in \text{R}(M_n))$, thus, the marking $M_\text{Input}$ of ELPN, can be reached from the marking $M_\text{Output}$ of ELPN. From definition 11, $M_\text{Input}$ denotes the marking of ELPN, which is consistent with the user's input parameters. $M_\text{Output}$ denotes the marking of ELPN, 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:** $R_{\text{table}} = (\text{Records})$ is the composition result for user’s demand, where:

- Records = {record, record2, ..., recordn} is a finite set
- The format of the item of Records is <Rid, Rinput, Routput, Sid>, where:
  - Rid denotes the unique mark number of the item and created in ascending order successively
  - Rinput denotes the input parameters
  - Routput denotes the output parameters
  - Sid denotes the unique mark number of web service

**Example 6:** There is a composition result for user's demand $R_{\text{table}} = (\text{Records})$. Suppose $\text{Records} = \{\text{record}, \text{record2}, \ldots, \text{recordn}\}$, where $\text{record}[\leq 1, \{a, b\}, 3] > \text{record}[2, \{b, c\}, 5]$. 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 2:** Service composition oriented to user's demands

Input: The user's demand $J_{\text{Input}} = (I_{\text{Input}}, U_{\text{Input}}, O_{\text{Output}})$ and the composition library $\text{Scen} = (C_{\text{ELPN}}, C_{\text{Relations}})$

Output: The composition result $R_{\text{table}} = (\text{Records})$

Step 1: Create the composition result $R_{\text{table}} = (\text{Records}) = \emptyset$. Create a variable $x = 0$

Step 2: Traverse $\text{Scen} = (C_{\text{ELPN}}, C_{\text{Relations}})$

2.1: Suppose the current item of $C_{\text{Relations}}$ is $R_x$ and $R_y = \{\text{inputs}, \text{outputs}, \text{transitions}\}$. If the items of $J_{\text{Input}}$, $U_{\text{Input}}$, and $O_{\text{Output}}$ are the same as the items of $R_x$, then return $R_x$, otherwise continue the cycle to traverse $R_x$.

Step 3: Suppose $\text{transitions} = \{t_1, t_2, \ldots, t_x\}$. Traverse transitions $t_x$

3.1: Suppose the current item of transitions is $t_x$. Suppose the $C_{\text{ELPN}}$. $\text{Input}(t_x) = \{\text{inputs}, \text{outputs}, \text{transitions}\}$, where $\text{Input}(t_x) = \{I_x, O_x, T_x\}$ and $T_x = \{t_1, t_2, \ldots, t_x\}$

3.2: Create a new item of $R_{\text{table}}$. Record = <Rid, Rinput, Routput, Sid>. Set record.

Set record. $R_{\text{Input}} = \{a, b, \ldots, a_n\}$ according to the logic input expression of $t_x$. Set record. $R_{\text{Output}} = \{b, c, \ldots, b_n\}$ according to the logic output expression of $t_x$. Set record. $R_{\text{Id}} = b$ according to the label of the transition $t_x$.

Step 4: Output the composition result $R_{\text{table}} = (\text{Records})$

**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>
<thead>
<tr>
<th>Identity</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>neo</td>
<td>xae, u</td>
<td>&lt;&lt;(aeo), (xae\text{u}&gt;)</td>
</tr>
<tr>
<td>2</td>
<td>beq, cwr</td>
<td>cet, y</td>
<td>&lt;(beq\text{cwr}, (cet\text{y})&gt;</td>
</tr>
<tr>
<td>3</td>
<td>cno, hje, io</td>
<td>cwr, iko</td>
<td>&lt;&lt;(cno\text{hje\text{io}}, (cwr\text{iko})&gt;</td>
</tr>
<tr>
<td>4</td>
<td>dnm, fkr</td>
<td>q</td>
<td>&lt;(dnm\text{fkr}, (\text{q})&gt;</td>
</tr>
<tr>
<td>5</td>
<td>evk, jkm</td>
<td>j, p</td>
<td>&lt;(evk\text{jk}\text{m}, (\text{j\text{p}}&gt;)</td>
</tr>
<tr>
<td>6</td>
<td>fte, kxz</td>
<td>fko</td>
<td>&lt;(fte\text{kxz}, (\text{fko})&gt;</td>
</tr>
</tbody>
</table>
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 Scmrlry - (C_ELPNs, C_Relations). C_ELPNs is equal to the ELPNs model of service composition and the first six items of C_Relations are showed below:

<table>
<thead>
<tr>
<th>Identity</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aeo</td>
<td>fg</td>
</tr>
<tr>
<td>2</td>
<td>csa, hej, io</td>
<td>pts, tts</td>
</tr>
<tr>
<td>3</td>
<td>dmm, ik</td>
<td>q</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>vse</td>
<td>bet</td>
</tr>
<tr>
<td>6</td>
<td>wrt</td>
<td>xt</td>
</tr>
</tbody>
</table>

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,
Table 3: First six composition results

<table>
<thead>
<tr>
<th>Demand Id</th>
<th>Stable Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;1, {seo}, {xce, txe}, 18&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;2, {xae}, {kxe}, 93&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;3, {twe}, {fle}, 47&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;4, {tie}, {kek}, {fle}, 6&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;1, {exe, he}, {io}, {crw}, {ki}, 3&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;2, {crw}, {ki}, {pts}, {tis}, 58&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;1, {dhr}, {il}, {q}, 4&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;1, {c}, {d}, 82&gt;</td>
</tr>
<tr>
<td>5</td>
<td>&lt;6, {twe}, {fle}, 47&gt;</td>
</tr>
<tr>
<td>6</td>
<td>&lt;1, {wtc}, {aceo}, 77&gt;</td>
</tr>
</tbody>
</table>

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 \( \text{ScomLib}_j = (C\_\text{ELPNs}, C\_\text{Relations}) \). Suppose \( |C\_\text{Relations}| = n \). From algorithm 7, if the item in \( C\_\text{Relations} \) is satisfied to user’s demand is not found, the time complexity for service composition is \( O(n) \). Else, if the item in \( C\_\text{Relations} \) satisfied to user’s demand can be found and named as \( R_x = \langle \text{inputs}_x, \text{outputs}_x, \text{transitions}_x \rangle \). Suppose \( |\text{transitions}_x| = m \), 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 exa, he, io and crw, ki can be obtained using the web service which Identity = 3. Then, the user inputs crw, ki and pts, tis 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 \( U\text{demand}_2 = (2, \{\text{exa}, \text{he}, \text{io}\}, \{\text{pts}, \text{tis}\}) \). From the service composition oriented to \( U\text{demand}_2 \) in Table 3, the optimal segment of \( U\text{demand}_2 \) is \( U\text{demand}_{21} = (2-1, \{\text{exa}, \text{he}, \text{io}\}, \{\text{crw}, \text{ki}\}) \), \( U\text{demand}_{22} = (2-2, \{\text{crw}, \text{ki}\}, \{\text{pts}, \text{tis}\}) \). However, the segment of \( U\text{demand}_2 \) 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 \( \text{webService}_x = (\text{Identity}_x, \text{Inputs}_x, \text{Outputs}_x, \text{Relations}_x) \) to be a web service, where \( \text{Identity}_x = k, \text{Inputs}_x = \{a, b, c, d\}, \text{Outputs}_x = \{e, f, g, h\}, \text{Relations}_x = \{<\text{c}\land\text{d}, \text{g}\lor\text{h}>\} \). According to algorithm 1, the ELPNs model of webService\(_x \) is showed in Fig. 7.

**Model web service by LPNs**: From the definition of LPNs, there are three kinds of transitions in LPNs. \( T_0 \) denotes the traditional transitions. \( T_L \) denotes the transitions restricted by the logic input expressions (Du and Guo, 2009). \( T_0 \) denotes the transitions restricted by the logic output expressions. The LPNs model of webService\(_x \) is showed in Fig. 8.
user's demand is introduced. From the simulation experiments and comparisons, the time complexity of service composition orientated 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 graph. 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. 20113118110004; 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


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