International Journal of Security and Its Applications
Vol. 10, No. 6 (2016) pp.249-260
http://dx.doi.org/10.14257/ijsia.2016.10.6.24

Research on Probabilistic Model based Services Composition Compatibility Focusing on Structural Behaviors Modeling

Honghao Gao[1,2], Yucong Duan[3] and Yonghua Zhu[2]

1. Computing Center, Shanghai University, 200444 Shanghai, P.R. China
2. School of Computer Engineering and Science, Shanghai University, 200444 Shanghai, P.R. China
3. College of Information Science and Technology, Hainan University, 570100 Haikou, P.R. China
Email: gaohonghao@shu.edu.cn

Abstract

With the increasing of Web services on Internet, services composition requests considering not only the static information compatibility, such as interface grammar and semantic, but also the structural behavior compatibility, i.e., control flow and communication protocol. However, due to the uncertainty of Internet, application failures occurred in service-based software will cause kinds of economic losses. This stochastic phenomenon has been impacted on the structural behavior which makes services composition should take the probabilistic compatibility into account. The probabilistic compatibility needs to compute the degree of compatibility for the effective performance evaluation, which has been an important issue during services collaboration. In this paper, the probabilistic interface automaton is proposed to formalize service behaviors. Then, the synchronized product model is used to the qualitative checking for verifying whether they can interact with each other or not. Third, the probabilistic compatibilities of composite service and component service are discussed for the quantitative computing. Our method provides a reference to generate the correctness and reliable services composition.

***Keywords**: Services Composition, Probabilistic Compatibility, Structural Behaviors Modeling, Quantitative Computing*

1. Introduction

Web service publishes functions through its public interfaces, which can be invoked by cooperative partners. It is considered as a promising approach to overcome the problem of language and platform dependency to reuse third-party application [1, 2]. Thus, modern enterprises have been employing services to implement their business processes to crossing the boundary of E-commerce.

In general, the single service is hard to meet complex requirements due to the diversity and complexity of user requirements [3]. The aim of services composition is to select multiple individual services and assemble them together to create new or additional value-added services [4]. To remain competitive, the services composition is an efficient way to reduce product cost and enable enterprise to seize the business opportunity. However, with the increasing of services, there are a large number of available services offering similar functionalities. To integrate services effectively and make them worked seamlessly, how to ensure the compatibility among component services selected to business collaboration is a challenging topic in the research field of services computing.

The compatibility is not only dependent on the service itself but also correlated to other services. Most existing methods research one solution that is compatible in some given criteria, such as interface compatibility and behavior compatibility. The interface compatibility checks the static semantic by matching the correctness of number, sequences, and types of parameters in each operation from provide interfaces and require interfaces [5]. The behavior compatibility verifies the business logics of service process focusing on inner state transition and event flow. However, due to the heterogeneous, open and collaborative natures of Internet, any failure may immediately result in service interrupts, which will seriously impact on the reliability of service-oriented software. Once the non-functional requirements are considered, these methods behave quite inefficiently because it lacks relevant solutions to support the probabilistic compatibility. The services composition with optimal performance should be selected to user. Thus, there is a need to study the probabilistic compatibility.

In this paper, the probabilistic model based compatibility is proposed to improve the services composition. First, Web service is formalized into probabilistic model which describes probabilistic behaviors. Then, the synchronized product is introduced to generate the synthesis model for services composition, based on which the composition compatibility is checked. Third, the compatibility computing method is proposed to analyze service interactions, which will calculate the degree of compatibility for composite service and component service. Three compatibility computing scenarios are discussed, mainly the complete compatibility path, miraculous path, and dead lock path. The probability value of compatibility considered as a metrics makes sure the composition result is trustworthy. Eventually, the quantitative computing provides compatibility value to the guidance of services composition, by which kinds of candidate services can be selected to further applications.

The rest of this paper is organized as follows. Section 2 introduces formal models and synchronized product used in the remainder of the text. Section 3 gives the compatibility computing method according to shared actions between component services. Section 4 reviews related works. Section 5 concludes this research and discusses future directions.

2. The Probabilistic Model for Services Composition

Service will display stochastic behavior and failures may happen frequently. Thus, invoking services may be unsuccessful in random phenomenon due to the uncertainty of Internet. Even if services are compatible to each other, they will display with different probability. The services composition also appears probabilistic behaviors. However, users like reliable composite service. Thus, the probabilistic compatibility requires taking stochastic behaviors into account to estimate the services composition.

To this point, the probabilistic model is employed to formalize service behaviors, which considers both service interface and probability factors. The interaction criterion for services composition depends on exchanged messages between service interfaces.

2.1. Service Modeling

Definition 1 (Probabilistic Interface Model for Web Service, PIM). The probabilistic interface model for Web service is derived from the Finite State Machine (FSM) and Service Interface View (SIV). The PIM model is a tuple $M=(S, s_0, s_e, \Sigma, \delta, P, L)$, where,

(1) S is a finite set of states;

(2) $s_0 \in S$ is the starting state, and $s_e \in S$ is the ending state;

(3) Σ is a finite set of actions which can be triggered through interfaces;

(4) $\delta \subseteq S \times \Sigma \times S$ is a finite set of transition relations called as service processes;

(5) $P : \delta \rightarrow [0,1]$ is probabilistic relation function to the corresponding transition;

(6) $L : S \rightarrow 2^{AP}$ is a mapping function that labels each state with the power set of atomic propositions *true* in that state. *AP* is a set of atomic propositions.

In graphical description, circle node represents state, and arrow line represents transition relation. The interactions between models are worked through interface to exchange message. A transition denotes action responses for one or more states. For $\forall$ s$\in$S, there is $\exists s' \in$S, which satisfies $\exists a \in \Sigma \bullet (s,a,s') \in \delta$. Each service interaction triggered by interface makes state been changed with a certain probability of *p*, otherwise, the corresponding state will remain unchanged with a probability of 1-*p*. Therefore, invoking service with different parameters at interface may output different results with a certain probability. It is called as probabilistic behavior.

In implementation level, the OWL-S specification is used as the modeling source, which describes service behaviors, consisting of control structure (If-then-else, Repeat, While, Sequence) and interface dependencies (Input, Output, Precondition, Effects). Thus, transforming OWL-S into probabilistic interface model contributes to formalize the business logics and its interfaces. The δ contains transition relations, by which *P* gives a set of all discrete probability distributions over *S*. For given action $a \in \Sigma$, the set of discrete probability distributions is $\{p_i \mid s \in S,\ a \in \Sigma,\ s' \in S,\ (s,a,s') \in \delta \bullet p_i \in P((s_i,a,s_j))\}$where $\Sigma(p_i)=1$.

2.2. Case Study

As Figure 1 shown, the example involves services composition, where Agent Service (AU) interacts with Shopping Service (SS) to achieve Online Shopping business logics via interface actions, mainly, PO (PurchaseOrder), DEL (Delivery), REF (Refusal), CP (CashPay), and BTP (BankTransferPay). The service processes work as follows: AU first sends orders to SS by action PO. If SS has enough tickets in stock, it will inform AU messages by action DEL. Second, AU will choose the payment method, including the cash pay and bank transfer pay by actions CP and BTP, respectively. Otherwise, SS notifies AU to terminate interactions by action REF.

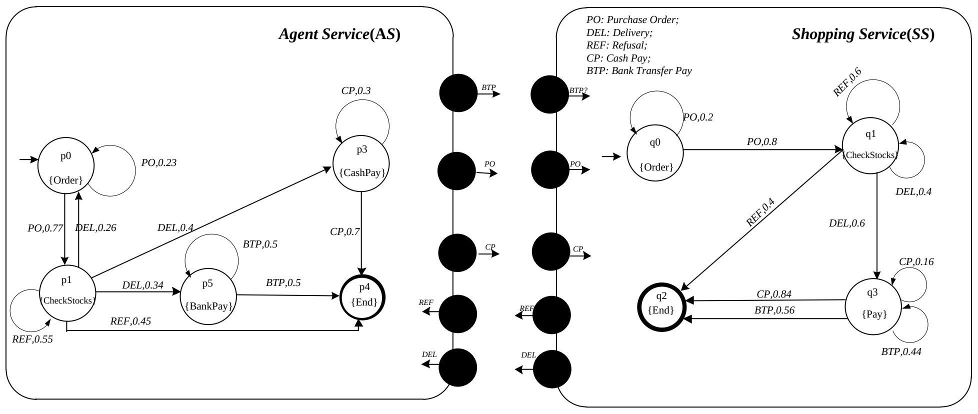


Figure 1. An Example of Services Composition for Online Shopping System

According to Definition 1, the probabilistic interface model for Agent Service (AU) $M^{AU}=(S, s_0, s_e, \Sigma, \delta, P, L)$ is specified as follows,

(1) S ={Order, CheckStocks, CashPay, BankPay, End};

(2) s_0 ={Order}, s_e ={End};

(3) Σ ={PO,DEL,REF,CP,BTP};

(4)δ={(Order,PO,Order),(Order,PO,CheckStocks),(CheckStocks,DEL,Order),(CheckStocks,DEL,CashPay),(CheckStocks,DEL,BankPay),(CheckStocks,REL,CheckStocks),(C heckStocks,REL,End),(CashPay,CP,CashPay),(CashPay,CP,End),(BankPay,BTP,BankPay),(BankPay,BTP,End)};

(5)*P*={(Order,PO,Order,0.77),(Order,PO,CheckStocks,0.23),(CheckStocks,DEL,Order,0.26),(CheckStocks,DEL,CashPay,0.4),(CheckStocks,DEL,BankPay,0.34),(CheckStocks,REL,CheckStocks,0.55),(CheckStocks,REL,End,0.45),(CashPay,CP,CashPay,0.3),(CashPay,CP,End,0.7), (BankPay,BTP,BankPay,0.5),(BankPay,BTP,End,0.5)};

(6)*L*(Order)={order},*L*(CheckStocks)={checkStocks},*L*(CashPay)={cashPay},*L*(BankPay)={bankPay}, *L*(End)={end}.

According to Definition 1, the probabilistic interface model for Shopping Service (SS) $M^{SS}=(S, s_0, s_e, \Sigma, \delta, P, L)$ is specified as follows,

(1) S ={Order, CheckStocks, Pay, End};

(2) s_0 ={Order} , s_e ={End};

(3) Σ={PO,DEL,REF,CP,BTP};

(4)δ={(Order,PO,Order),(Order,PO,CheckStocks),(CheckStocks,DEL,CheckStocks),(CheckStocks,REL,CheckStocks),(CheckStocks,DEL,Pay),(Pay,CP,Pay),(Pay,BTP,Pay),(CheckStocks,REL,End), (Pay,CP,End), (Pay,BTP,End)};

(5)*P*={(Order,PO,Order,0.2),(Order,PO,CheckStocks,0.8),(CheckStocks,DEL,CheckStocks,0.4),(CheckStocks,REL,CheckStocks,0.6),(CheckStocks,DEL,Pay,0.6),(CheckStocks,REL,End,0.4), (Pay,CP,Pay,0.16), (Pay,BTP,Pay,0.44), (Pay, CP, End,0.84), (Pay,BTP,End,0.56)};

(6)*L*(Order)={order},*L*(CheckStocks)={checkStocks},*L*(Pay)={cashPay,bankPay}, *L*(End)={end}.

The interface actions via shared channels make services been compatible. For example, PO is sent by AS and is received by SS, in which states of their service process will be changed. As Table 1 shown, there are 3 complete paths from the starting state to ending state. Under action sequence <PO,REL>, corresponding path sequences <(Order,PO,CheckStates), (Order,REL,End)>, <(Order,PO,CheckStates), (Order, REL,End)> are extracted from AS model and SS model, respectively. In consequence, these two sequences have different probability values.

Table 1. The Complete Path between Service Interactions

ID	Action Sequences	Paths in AS Model	Paths in SS Model
1	<PO,REL>	<Order,CheckStates,End>	<Order,CheckStates,End>
2	<PO,DEL,BTP>	<Order,CheckStates,CashPay,End>	<Order,CheckStates,Pay,End>
3	<PO,DEL,CP>	<Order,CheckStates,BankPay,End>	<Order,CheckStates,Pay,End>

For probabilistic behaviors, we consider Markov Chain to compute the degree of compatibility. Assume that x_i is a random event occurring at the *i-th* period and $p(x_i)$ represents the probability that a random event x_i occurs at this time. The probability $p(x_i)$ is only related to the former event $p(x_{i-1})$. Thus, for each k>=1, the probability is $p(x_k|x_1,x_2,\ldots,x_{k-1})= p(x_k|x_{k-1})$. Calculating the joint probability of transition relations δ is reduced to get each probabilistic relation function $p(t)=P((s,a,s'))$, $t\in\delta$. Thus, the probability value of a path built by transitions is computed as follow:

$$p(t_0,t_1, \ldots,t_{k-1})= p(t_0)\, p(t_1 |t_0)\, p(t_2 |t_1)\ldots p(t_k |t_{k-1}) \qquad (1)$$

Based on formula (1), the composite service compatibility is proposed to handle the complete path constructed by the composition results.

Definition 2 (The Path Compatibility). Let $\gamma1=<t_1,t_2,t_3,\ldots,t_n>$ and $\gamma2=<t'_1,t'_2,t'_3,\ldots,t'_n>$ be complete path sequences from probabilistic interface model M_1 and M_2, where $0<i\leq n$, $t'_i\in\delta_1$ and $t_i\in\delta_2$. They are selected to services composition according to the compatibility degree.

The compatibility degree $CP^T(\gamma1, \gamma2)$ for two paths is computed as follow,

$$CP^T(\gamma1,\gamma2)=\prod_{i\in n\bullet t_i\in\gamma1\wedge t_i'\in\gamma2} p(t_i,t_i') \tag{2}$$

Formula (2) is the multiplicative computing, where $p(t_i,t_i')$ is defined as

$$p(t_i,t_i')=\begin{cases} P(t_i)*P(t_i') & Shared(t,t')\neq\varnothing \\ 0 & otherwise \end{cases} \tag{3}$$

Furthermore, Let $T1$ and $T2$ be sets of transition sequences under shared interface actions. The compatibility degree $CP^M(T1, T2)$ for services composition is computed as follow,

$$CP^M(T1,T2)=\sum_{\gamma1\in T1,\gamma2\in T1} CP^T(\gamma1,\gamma2) \tag{4}$$

The formula (4) is the accumulation computing, that the composition compatibility $CP^T(T1, T2)$ is reduced to calculate the multiplicative computing $CP^M(\gamma1, \gamma2)$, where paths $\gamma1\in T1$ and $\gamma2\in T2$ are generated by shared action using the determining function $Shared(t,t')$. If $Shared(t,t')=0$, the paths $\gamma1$ and $\gamma2$ are incompatible.

2.3. The Synchronized Product for Models Composition

We transform the problem of services composition into the models composition problem. The composition is represented in the form of a synchronized product. Based on the formalizations, the composite service compatibility is proposed to check synchronized product model in a qualitative way.

It first introduces the incompatible interactions. Although more paths can be extracted from PIM model for possible interactions, they will make composition into a dead lock. For example, under action sequence <PO, REF>, paths <Order, CheckStates, CheckStates> in AS Model and <Order,CheckStates,End> in SS Model are possible composition behaviors. However, if they are used to services composition, the state of SS model is terminated, while the state of AS model is active. Thus, these paths should be removed. Our goal of synchronized product is to give the generalized model for services composition, by which all interactions between services can be triggered from their starting states to ending states.

Definition 3 (The Synchronized Product for PIMs). Let PIM_1, PIM_2 be two probabilistic interface models. Their synchronized product is also a probabilistic interface model, $PIM=PIM_1 \| PIM_2=(S, s_0, s_e, \Sigma, P, \delta, L)$, that is,

1) $S=\{(s_1, s_2) | (s_1, s_2)\in S_1\times S_2\}$;

2) $s_0=(s^1_0, s^2_0)$, $s_e=(s^1_e, s^2_e)$;

3) $\Sigma=\Sigma_1\cap\Sigma_2$;

4) $L((s_1, s_2))=L_1(s_1)\cup L_2(s_2)$, and $AP=AP_1\cup AP_2$;

5) $((s_1,s_2),a,(s'_1,s'_2))\in\delta$, where for $\forall a\in\Sigma_1\cap\Sigma_2$, it should follow rule $(s_1,a,s'_1)\in\delta_1\wedge(s_2,a,s'_2)\in\delta_2$ that $s_1\in\delta_1$, $s_2\in\delta_1$, $s'_1\in\delta_2$, and $s'_2\in\delta_2$.

6) $P((s_1,s_2),a,(s'_1,s'_2))= P((s_1,a,s'_1))*P((s_2,a,s'_2))$

The formal framework of synchronized product provides base formalizations to models composition. The result is still a probabilistic model that $P((s_1,s_2),a,(s'_1,s'_2))$ according to $P((s_1,a,s'_1))*P((s_2,a,s'_2))$ value. The condition for $((s_1,s_2),a,(s'_1,s'_2))\in\delta$ is action a which should be active in all transitions $(s_1,a,s'_1)\in\delta_1$ and $(s_2,a,s'_2)\in\delta_2$. However, the transition of synchronized product will emerge dead lock states. The transition relations set δ should be,

$$\{((s_1,s_2),a,(s'_1,s'_2))|((s_1,s_2),a,(s'_1,s'_2))\in\delta\wedge(s'_1\neq s^1_e\wedge s'_2\neq s^2_e)\vee(s'_1=s^1_e\wedge s'_2=s^2_e)\} \tag{5}$$

The formula (5) means that both s'_1 and s'_2 are ending states or both s'_1 and s'_2 are not ending states. Otherwise, it will encounter a dead lock.

The Definition 3 indicates that If $\text{PIM}_1 \parallel \text{PIM}_2=\varnothing$ then services are totally incompatible. Based on this judgment, services composition has the ability to provide value-added services when $\text{PIM}_1 \parallel \text{PIM}_2\neq\varnothing$.

Definition 4 (Action Sequences). Let PIM=$(S, s_0, \Sigma, P, \delta, L)$ be probabilistic model. The Act(PIM)=$\{<a_0\ a_1,\ldots, a_n>|\exists s_i,s_{i+1}'\in S,\exists a_i\in\Sigma,i<n\bullet \bigwedge_{i=0}^{n-1} (s_i,a_i,s_{i+1}')\in\delta\wedge s_n=s_e\}$ is a set of action sequences.

Definition 5 (The Compatibility Checking). Let two PIM models be used to services composition. The compatibility checking of $\text{PIM}_1 \parallel \text{PIM}_2\neq\varnothing$ is equivalent to find whether Act (PIM) is empty or not.

$\text{PIM}_1 \parallel \text{PIM}_2\neq\varnothing$

$\cong$

$$Act(\text{PIM}_1 \parallel \text{PIM}_2)=\{<a_0, a_1,\ldots, a_n>|\exists s_i,s_{i+1}'\in S,\exists a_i\in\Sigma \bullet \bigwedge_{i=0}^{n-1} (s_i,a_i,s_{i+1}')\in\delta\wedge s_n=s_e\} \neq \varnothing \tag{6}$$

If $\text{PIM}_1 \parallel \text{PIM}_2\neq\varnothing$ is proved, the compatibility formula CP^T (T1, T2) is used to compute the degree of compatibility of a composite service. It means that there is at least one message exchange sequence, which makes two services stopped at their ending states.

3. The Probabilistic Compatibility Computing

In this section, we will give the component service compatibility computing method. First, if $\text{PIM}_1 \parallel \text{PIM}_2\neq\varnothing$, The reachable state set is defined to the compatibility checking. After that, the compatibility of component service is computed by focusing on probabilistic behaviors are correlated to composition.

However, the service composition may be more complex. Some participant services may be unavailable when the composition is running. The participant with best degree of compatibility can be selected to assemble in a composition, achieving the optimal composite service. In order to show the method for component service compatibility in probabilistic behaviors, the action of synchronized product is taken into account.

Suppose that we have a synchronized product model $M^s=(S^s, s^s_0, \Sigma^s, P^s, \delta^s, L^s)$ and a path sequence $<t_1,t_2,t_3,\ldots,t_n>$. Then, for a component model $M=(S, s_0, \Sigma, P, \delta, L)$, the reachable state set according to interaction is defined as

$$Q_i=IntSeq(Q_{i-1},\bigcap_{k=1}^{n} Put(act_k,i-1)\ )\ where\ act_k\in Act(M^s) \tag{7}$$

The Q_i is a reachable state set which is determined by the former states in Q_{i-1} and shared actions in *Put* (act_k,*i*-1). The function *Put* (act_k,*i*-1) selects (*i-1*)-*th* element of the action sequence $act_k \in Act$ (M^s). In general, $Q_1=IntSeq(s_0, a_0)$ which is computed from the starting state and first action a_0. The $\cap$ calculation generates all (*i-1*)-*th* from action sequences of synchronized product. The *IntSeq* function gives the next reachable states of Q_{i-1} in component model *M*. If $Q_i=\varnothing$, the component service compatibility is 0.

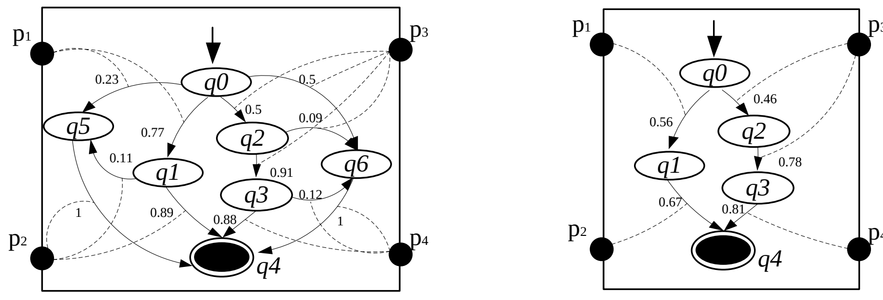
When more than one service provides the same function, it needs a method to evaluate the compatibility of component service in a quantitative way. Path concepts for the component service compatibility are defined, mainly the complete compatibility path, miraculous path and dead lock path. These paths contribute to the degree of compatibility for component service. Then, it selects the one with the highest degree of compatibility to be assembled.

Definition 6 (Processible Actions). Let PIM=(S, s_0, Σ, P, δ, L) be probabilistic model. A state $s\in S$ can handle action $a\in\Sigma$ if there is a transition from s by action a. The $Processible(s)=\{a|\exists s'\in S\bullet(s,a,s')\in\delta\}$ is a set of all actions to force state s to be changed.

Definition 7 (The Complete Compatibility Path). For all paths generated from action sequences of synchronized product model, the reachable state set for component service should be not empty *Act* ($PIM_1 \parallel PIM_2$)$\neq\varnothing$. The component service with complete compatibility path is denoted as $Path^c$, which is satisfied that,

$$\mathrm{Path}^c \cong \forall act\in Act(M^s)\bullet(\forall a\in Put(act,i-1),\exists s\in Q_{i-1}\bullet IntSeq(Q_{i-1},a)\neq\varnothing\wedge a\in Processible(s)) \quad (8)$$

The complete compatibility path of component service contains the ending state, which makes service interaction stopped in a safety way.



a) Component Service b) Synchronized Product Model

Figure 2. An Example of The Complete Compatibility Path

As Figure 2 shown, there are two complete compatibility paths in a) part. For each action sequence in {<p1, p2>, <p3, p3, p4>}, there has corresponding paths from the starting state to ending state. For action sequence <p1, p2>, the reachable states $Q_0=\{q0\}$,$Q_1=\{q1\}$,$Q_2=\{q4\}$. For action sequence <p3, p3, p4>, the reachable states $Q_0=\{q0\}$,$Q_1=\{q2\}$,$Q_2=\{q3\}$, $Q_2=\{q4\}$.

The degree of component service compatibility for $Path^c$ is

$$CP^C(M,M^s)=\sum_{\gamma|=Act(M)\wedge\gamma|=Path^c}\ \prod_{t\in\{(s,a,s)|\forall a\in\gamma,\exists a\bullet(s,a,s)\in\delta\}} P(t) \quad (9)$$

The symbol |= is a satisfaction relation. The parameter $\gamma|=Act\ (M)\wedge\gamma|=Path^{c}$ defines that the action sequence should construct complete compatibility paths.

Definition 8 (The Miraculous Path). For all paths generated from action sequences of synchronized product model, the reachable state set $Q^{act}{}_{1}$ is empty. The component service with miraculous path is denoted as $Path^{M}$, which is satisfied that,

$$Path^{M}\ \cong \forall act\in Act(M^{s})\bullet\ Q^{act}{}_{1}=\varnothing \tag{10}$$

The degree of component service compatibility for $Path^{M}$ is

$$CP^{M}(M,M^{s})=0 \tag{11}$$

It means that there failure occurs in the initialization state, which make service unusable. Considering that the candidate services may be used to dynamic replacement, the miraculous path should be avoided.

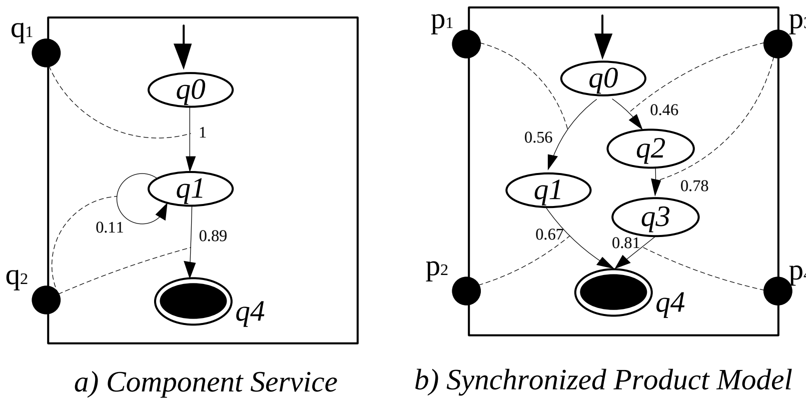


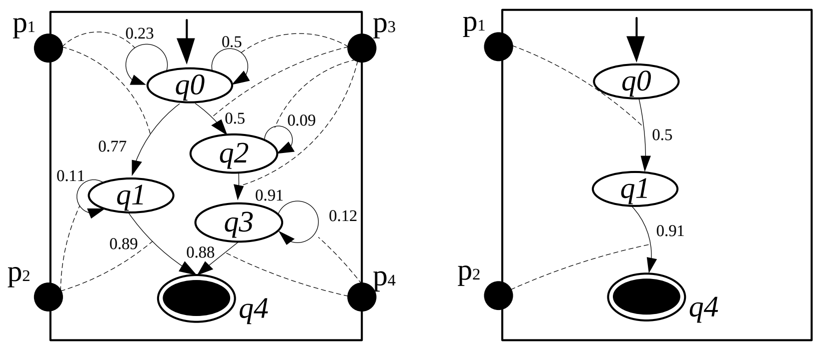
Figure 3. An Example of the Miraculous Path

As figure 3 shown, there is no compatibility path in a) part. This case may happened when updating service business logics. Thus, although synchronized product model is not empty, no corresponding interaction behavior can be found from initial state of component service.

Definition 9 (The Dead Lock Path). For $\forall act\in Act(M^{s})$, $1<f<n$, the reachable state set $Q^{act}{}_{f}\neq\varnothing$. Then the next reachable state set $Q^{t}{}_{f+1}=IntSeq(Q_{f},Put(t,f))=\varnothing$. The component service with dead lock paths is denoted as $Path^{DL}$, which is satisfied that,

$$Path^{DL}\ \cong \exists a\in Put\ (act,f),\forall s\in Q_{f}\bullet a\notin Processible(s) \tag{12}$$

In this case, there is no more state can be forced to its next state by *f-th* in $act\in Act\ (M^{s})$. However, the current state is active without stopping. Hence, the dead lock path is not useful for compatibility computing.



a) Component Service *b) Synchronized Product Model*

Figure 4. An Example of the Dead Lock Path

As Figure 4 shown, it shows an example of dead lock path. For action sequence in {<p1, p2>}, there has corresponding paths from starting state to ending state. In a) part, Q_0={q0}, Q_1={q0,q1}, and Q_2={q4}. However, two paths <(q0,p1,q1),(q1,p2,q4)> and <(q0,p1,q0)> are generated. For the former, it is a complete compatibility path. For the latter, it happen dead lock at state q0 which cannot be changed under action p2. Thus, in spite of they are syntactically and semantically compatible in interfaces and behaviors, actions enabled by improper temporal operations will make the composite service entered into a dead lock.

The degree of component service compatibility for $Path^{DL}$ is

$$CP^{DL}(M,M^s)=\sum_{\gamma|=Act(M)\wedge\gamma|=Path^{DL}} p$$

$$=\begin{cases} p=\prod\limits_{t\in\{(s,a,s)|\forall a\in\gamma,\exists a\bullet(s,a,s)\in\delta\}} P(t) & \gamma\downarrow s_e\neq\varnothing \\ 0 & otherwise \end{cases} \tag{13}$$

In formula (13), the parameter $\gamma|=Act\ (M)\wedge\gamma|=Path^{DL}$ defines that component service has dead lock paths. The projection symbol $\downarrow$ is used to check whether the path sequence contains the ending state. If the path does not contain any ending state, the compatibility value is 0. Otherwise, the compatibility value is computed by the multiplication value of each probabilistic relation function *P* (*t*) in path sequence.

4. Related Works

Web service is widely adopted in the businesses and the scientific community. How to select an optimal composite service with a high compatibility from candidate services plays a major role in the fields of service compositions. Many approaches for the compatibility of services composition had been published in literatures, including Petri-Net [6-8], Process Algebra [9,10], and Automata [11-13]. We give a review on the major techniques and researches that are most closely related to our works.

Martens *et. al.,* [14] adopted Petri Net to model BEPL and used reliability property to verify the compatibility of service composition. It aims to check whether the local process can be worked well with global process. They also developed an automated analysis tool WOMBAT4WS. Hamadi *et. al.,* [15] mapped service operations and service input/output to the Transition and Place of Petri Net. Then the property consisting of Liveness, Boundedness, and deadlock is used to check the compatibility of composition behavior. However, the space complexity of Petri Net is high, and the property verification is a NP problem.

Benatallah *et. al.,* [16] used the simulation equivalence to check the compatibility and replaceability for services composition. Ali *et. al.,* [17] employed the asymmetric simulation relations to verify the new replaced service can simulate the behavior of previous service. Chae *et. al.,* [18] proposed a compatibility concept from simulation view, by which each behavior of trace should be reserved in its partner. Mario and Gianluigi [19] presented a compatibility checking method for service composition, which is based on the protocol theory. Brogi *et. al.,* [20] introduced CCS to Web service choreography WSCI for formal modeling, which analyzed the compatibility of service composition. However, the process algebra is so abstract that most of users call for the graphical interface and operations supporting.

Shi *et. al.,* [21] employed automata to formalize interaction of service compositions, by which the compatibility was checked. Lee *et. al.,* [22] used interface automata to check the compatibility of interaction, which describes the access order for other services to invoke current service, and gives the access order for current service to invoke other services. Fu *et. al.,* [11] analyzed and verified the BPEL based service composition, where LTL was used to describe the compatibility property. Foster *et. al.,* [23] transformed service composition into FSM model, and then analyzed the compatibility of interaction of services composition in their tool LTSA. However, these methods lack of non-functional checking, which has been attracted by scientific researchers, such as time constraint and random probability.

5. Conclusion

Services composition requires mechanisms to ensure component services compatible with each other. The probabilistic compatibility plays a critical role in composite service. This paper proposes a probabilistic model based services composition compatibility method. First, the probabilistic model is used to formalize service behaviors. Second, the synchronized product is introduced to the compatibility checking. Third, the component service compatibility computing is discussed. The former checks the compatibility of composite service in a qualitative way, while the later computes the degree of compatibility of component service in a quantitative way.

However, we only consider structural behaviors modeling of services composition. Therefore, in the future, our work will focus on timed constraints for more complex functional and non-functional descriptions. We will design algorithms and turn our method into concrete business applications. Furthermore, the probabilistic model checking tools will be integrated for formal verification purpose to effectively generate the credible composite service.

Acknowledgement

This paper is supported by National Natural Science Foundation of China (NFSC) under Grant No.61572306, 61502294, Natural Science Foundation of Shanghai under Grant No.15ZR1415200, Natural Science Foundation of Hainan under Grant No. 20156234, CERNET IPv6 Innovation Project under Grant No.NGII20150609, 2015 Film Summit Science Foundation of Shanghai University, and Foundation of Science and Technology Commission of Shanghai Municipality under Grant No. 14590500500.

References

[1] Papazoglou MP and Georgakopoulos D., "Service-oriented computing", Communication ACM, vol. 46, no. 10, **(2003)**, pp. 25-28.

[2] Z. Wu, S. Deng, Y. Li and J. Wu, "Computing compatibility in dynamic service composition", Knowledge and Information Systems, vol. 19, **(2009)**, pp. 107-129.

[3] W. Tan, Y. Fan and MC. Zhou, "A Petri net-based method for compatibility analysis and composition of web services in business process execution language", IEEE Transactions on Automation Science & Engineering, vol. 6, no.1, **(2009)**, pp. 94-106.

[4] M. P. Papazog Lou, P. Traverso and S. Dustdar, "Service-oriented computing: A research roadmap", International Journal on Cooperative Information Systems, vol. 17, no. 2, **(2008)**, pp. 223-255.

[5] M. Brahim and B. Athman, "A multilevel composability model for semantic Web services", IEEE Transactions on Knowledge & Data Engineering, vol. 17, no. 7, **(2006)**, pp. 954-968.

[6] A. Martens, "On compatibility of Web services", Petri Net Newsletter, vol. 65, **(2003)**, pp. 12-20.

[7] Verbeek HMW and van der Aalst WMP, "Analyzing BPEL processes using Petri Nets", The International workshop on applications of Petri Nets to coordination, workflow and business process management, **(2005)**, pp. 59-78.

[8] Y. Yang, Q. Tan, Y. Xiao, J. Yu and F. Liu, "Exploiting hierarchical CP-nets to increase reliability of web services workflow", The International symposium on applications and the Internet, **(2006)**, pp. 116-122.

[9] L. Bordeaux and G. Salaün, "Using process algebra for Web services: early results and perspectives", The VLDB Workshop on Technologies for E-Services, **(2004)**, pp. 54-68.

[10] G. Salaün, L. Bordeaux and M. Schaerf, "The International Conference on Web Services", **(2006)**, pp. 116-128.

[11] X. Fu, T. Bultan and J. Su, "Analysis of interacting BPEL Web services", The International Conference on World Wide Web, **(2004)**, pp. 621-630.

[12] R. Hull and J. Su, "Tools for composite Web services: a short overview", SIGMOD Record, vol. 34, no. 2, **(2005)**, pp. 86-95.

[13] A. Brogi, C. Canal and E. Pimentel, "Formalizing Web service choreography", The International workshop on Web services and formal methods, **(2004)**, pp. 576-581.

[14] A. Martens, S. Moser, A. Gerhardt and K. Funk, "Analyzing Compatibility of BPEL Processes", The International Conference on Internet and Web Applications, **(2006)**, pp. 147-147.

[15] H. Rachid and B. Boualem, "A Petri Net-based Model for Web Service Composition", The Fourteenth Australasian database conference on database technologies, **(2003)**, pp. 191-200.

[16] B. Benatallah, F. Casati and F. Toumani, "Representing, Analysing and Managing Web Service Protocols", Journal of Data and Knowledge Engineering, **(2006)**, vol. 58, no. 3, pp. 327-357.

[17] A. Ait-Bachir, M. Dumas and M-C. Fauvet, "BESERIAL: Behavioural Service Interface Analyser", The 6th International Conference on Business Process Management, **(2008)**, pp. 374-377.

[18] H. S. Chao, J.-S. Lee and JH. Bae, "An Approach to Checking Behavioral Compatibility between Web Services", International Journal of Software Engineering and Knowledge Engineering, vol. 18, no. 2, **(2008)**, pp. 223-241.

[19] M. Bravetti and G. Zavattaro, "Towards a Unifying Theory for Choreography Conformance and Contract Compliance", The 6th Symposium on Software Composition, **(2007)**, pp. 34-50.

[20] A. Brogi, C. Canal, E. Pimentel and A. Vallecillo, "Formalizing Web Service Choreographies", Journal of Electronic Notes in Theoretical Computer Science, vol. 105, **(2004)**, pp. 73-94.

[21] Y. Shi, L. Zhang, F. Liu, L. Lin and B. Shi, "Compatibility Analysis of Web Services", The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, **(2005)**, pp. 483-486.

[22] E. A. Lee and Y. Xiong, "System-level types for component-based design", The First International Workshop on Embedded Software, **(2001)**, pp. 237-253.

[23] H. Foster, S. Uchitel, J. Magee and J. Kramer, "Compatibility Verification for Web Service Choreography", The IEEE International Conference on Web Services, **(2004)**, pp. 738-741.

Authors

Honghao Gao, received the Ph.D degree in computer application technology from the School of Computer Engineering and Science of Shanghai University, Shanghai, P. R. China, in 2012. His research interests include Web service and model checking.

Yucong Duan, received the Ph.D degree in Software Engineering from Institute of Software, Chinese Academy of Sciences, P. R. China in 2006. He is currently a full professor and vice director of Computer Science department in Hainan University, P. R. China. His research interests include software engineering, service computing, cloud computing, and big data. He is a member of IEEE, ACM and CCF (China Computer Federation).

Yonghua Zhu, received the Ph.D degree from the School of Communication & Information Engineering of Shanghai University, Shanghai, China, in 2006. His research interests include formal method, high performance computing, communication and information engineering.