International Journal of u- and e- Service, Science and Technology
Vol.9, No. 10 (2016), pp.423-434
http://dx.doi.org/10.14257/ijunesst.2016.9.10.38

Research on Service Process Verification of Probabilistic Spanning Tree for Dynamic Composition

Honghao Gao and Ying Li

College of Computer Science and Technology, Zhejiang University, 310058 Hangzhou, P.R. China
gaohonghao@zju.edu.cn

Abstract

Web services have been widely employed in E-commerce system. It is a flexible way to use third-party applications for complex business logics implementation, which supports data transparent transmission via public interfaces. However, the single service with a fixed function has limitations to the diversity and complexity of user requirements, especially the interoperability of applications interactions. To address the challenge, the capability of dynamic services composition aims to select services and assign them to business process. However, there are kinds of possible composition plans to satisfy requested functions. It is a great importance to evaluate these plans to guarantee the solution of services composition is correctness and acceptable. In this paper, it proposes a service process verification approach to probabilistic spanning tree for dynamic composition. First, the probabilistic spanning tree is introduced to describe services composition plans according to the interface operations decomposition of given service requirements. Second, the service process model is generated from probabilistic spanning tree based on different pruning strategies, where the formal verification is performed to check each solution against quantitative properties related to the functional and non-functional requirements. Our method is feasibility to dynamic services composition since the quantitative verification guarantees the performance of composition solution.

Keywords*: Services Composition, Probabilistic Spanning Tree, Interface Operations Decomposition, Quantitative Verification*

1. Introduction

With the development of Internet, complex applications call for integrating a set of Web services and coordinating interactions to support E-business [1]. The dynamic composition is considered as a large granularity service with business processes that receive different inputs and sends different output achieving requested business logics [2]. Today, enterprises and organizations have extended their cooperation and explored appropriate solutions via services composition to promote core business activities among customers, suppliers and partners. However, services composed from heterogeneous systems may be encountered different composition problems, such as the functional and non-functional requirement. The major problem is how to select and manage proper services to work with each other, which has been a challenge in service oriented computing. Thus, it needs an efficient method to verify the service process of dynamic composition.

Given a set of candidate services, the services composition aims to return different solutions to user. There are two main methods to services composition, including Orchestration and Choreography [3]. The orchestration approach should predefine workflows and dynamically assign services. However, the fixed workflow is not flexible to implement user requirements. When one component service is invalid, the composite service will be failure. It behaves quite inefficiently because of the searching efficiency.

ISSN: 2005-4246 IJUNESST

In the contrary, the choreography approach selects services according to user requirements decomposition step by step, which is based on services aggregation pattern without defining workflows. However, it may lead to space explosion since it has multiple composition plans to fulfill tasks of workflow. To this end, we are motived to check each composition plan by quantitative verification using model checking [45]. Only the property satisfied services composition can be considered as a solution to user.

In general, a Web service consumes inputs and produces outputs to user. Thus, the interface operation decomposition is introduced to find solutions to dynamic services composition, by which each solution starts at least one of given input and ends at least one of given output. The services selection of services composition is reduced to the interface operation searching to generate required outputs. Unfortunately, due to natures of open, interconnection and changeful, Internet is a non-deterministic environment where services may be impacted by random phenomenon [6-22]. Different composition strategies make the problem more complicated since they have different service processes with various reliabilities caused by network quality over time. To this problem, the probabilistic spanning tree is used to describe the probabilistic behaviors of possible business processes. Using probabilistic spanning tree can achieve benefits as follows,

1) Feasibility. Each part of probabilistic spanning tree may be a solution to user's query. Thus, if one failed component makes current services composition disabled, more possible composition plans generated from probabilistic spanning tree continues to implement target business logics. It is feasibility to services composition because the interface operation decomposition describes all composition plans.

2) Probability Description. Users are paying more attention to the global reliability of composite services, as there may be some available services with the same functionality, displaying different service quality of service. In cases, although a composition result satisfies the input and output of service requirement, it does not satisfy the reliability requirement. Due to the probabilistic characteristic of spanning tree, it describes probabilistic behaviors by which we can perform quantitative verifications to select an optimal composition.

Although probabilistic spanning tree provides solutions to services composition, different solution will display different performance. For example, the optimal composite service can be found conveniently and efficiently, such as the smallest service number and the maximum probability. To this propose, different services compositions are generated from spanning tree based on pruning criteria to remove redundant, which can be verified by probabilistic model checking to judge whether the solution is correctness and acceptable. We focus on check the property of dynamic composition both at the functional and non-functional level of requirements.

The contributions of this paper includes: 1) The probabilistic spanning tree is used to specify all services composition plans with probabilistic behaviors. The composition strategy is based on interface operations decomposition and service searching. 2) The corresponding strategies are proposed to generate service process, which uses pruning criteria to optimize services composition. 3) The service process model is verified in a quantitative way by applying formal verification, which outputs a valuable reference for user to choose.

The rest of this paper is organized as follows. Section II introduces our motivation. Section III discusses probabilistic spanning tree model for describing services composition. Section IV discusses how to construct service process model from probabilistic spanning tree, and uses probabilistic model checking to perform formal verification. Section V reviews related works. Section VI concludes this research and discusses future directions.

2. Motivation

The example is about Travel Approval Management Service (TAMS) e-commerce system. If application is failed, the dynamic services composition is requested to handle the service failure and support original business logic functions. However, we cannot understand service processes in detail. Thus, dynamic services composition should satisfy the input and output requirements of TAMS.

The inputs of TAMS are travel address *TraAdd* and travel date *TraDa*. The output of TAMS is approval result with two Boolean values *BoResult* and *Valid*. The details of all available services are presented in Table 1, where Input column shows service inputs and Output column shows service outputs.

Table 1. The Services Set for Composition

ID	*Service*	*Input*	*Output*
1	*S1*	*TraAdd*	*AppalLevel1*
2	*S2*	*TraAdd*	*Visa*
3	*S3*	*AppalLevel1*	*AppalLevel2*
4	*S4*	*AppalLevel2,Visa*	*BoResult*
5	*S5*	*TraDa*	*VeriDa*
6	*S6*	*TraDa*	*HasPas*
7	*S7*	*VeriDa,HasPas*	*Valid*
8	*S8*	*Visa, VeriDa*	*Valid*
9	*S9*	*AppalLevel1*	*AppalLevel2,Aduit*
10	*S10*	*CheckLog*	*HasPas*

The tasks for services composition are worked under interface operations decomposition starting from Input={*TraAdd*, *TraDa*}. Each task will stop at a service which outputs requested data belonged to Output={*BoResult*, *Valid*}. We use '~' to denote the temporal innovation relation between two services. For example, $ws_2 \sim ws_3$ indicates that ws_2 invokes ws_3 because output data of ws_2 can be used as input data of service ws_3.

As one solution, services *S*1, *S*2, *S*4, *S*5, *S*6, *S*7 and *S*9 are integrated as a composite service where *s* is a starting symbol, *e* is an ending symbol, symbols ⊗ is asynchronous concurrent, and ⊕ is synchronous concurrent.

$$s \sim (S6 \oplus S5 \sim S7) \otimes (S2 \sim S4) \otimes (S1 \sim S9 \sim S4) \sim e \quad (1)$$

In this example, three paths of service process are merged into a composite service. For path 1), service *S*1 receives input *TraAdd* and sends output *AppalLevel1*. After that, output data of *S*1 is considered as input data of next service. Service *S*9 receives input *AppalLevel1* and sends output *AppalLevel2*. For path 2), service *S*2 receives input *TraAdd* and sends output *Visa*. Then, service *S*4 receives input *AppalLevel2* and sends output *BoResult*. For path 3), service *S*5 and *S*6 receives input data *TraDa* and sends output data *VeriDa* and *HasPas*, respectively. Then, service *S*7 receives input *VeriDa* and *HasPas* and sends output *Vaild.* Finally, services *S*4 and *S*7 return *Vaild and BoResult* as final outputs.

Besides above service processes, there are more complex services composition plans. However, it is difficult to decide which solution is the best services composition. For these reasons, services composition should take quantitative evaluation to recommend a suitable solution to user. In our research, there are two problems of service process verification to be addressed. The first problem is to generate plans of composite services, which consists of different paths. The second problem is to consider the probability of service reliability, which may have great effect on performance of composite services.

3. Probabilistic Spanning Tree Based on Interface Operations Decomposition

In this section, we first define key concepts used in services composition. Then, the probabilistic spanning tree is introduced to service interface operations decomposition, which provides multiple composition plans as solutions.

3.1. Formal Model of Web Service

From the user viewpoint, a Web service has a set of interface operations to deal with input and return output.

Definition 1(Web Service). A Web service is defined as a tuple $ws=(I,O,D)$, where,
1) $I=\{i_1,i_2, i_3,.., i_n\}$ is a set of input parameters.
2) $O=\{o_1,o_2, o_3,.., o_n\}$ is a set of output parameters.
3) D is the domain concept, which shows a special classification.

The domain concept is a knowledge which describes the service set with similar functionality. In practice, Ontology tree is used as service classification, where each node corresponds to a domain concept [7]. The domain concept of each service is mapped to a node of Ontology tree for services classification.

Using Web service aims at maximizing software reuse, which fulfills specified tasks with public interfaces. However, user only knows the input and output features without considering their implementation details. In general, the service function is displayed with the input and output behavior.

Definition 2(Service Function). The Web service function is defined as $SF=(ws, f, p)$
1) $f: ws.I \rightarrow ws.O$ shows the operation between input and output.
2) p: $ws \rightarrow [0, 1]$ denotes the reliability of Web service in the form of probability value.

The service function SF divides Web service into different interface operations. The operation f defines a rule between input data and output data, by which different inputs will lead to different outputs. The reliability p is probability value shows the random phenomenon of service failures impacted by unstable Internet.

Users prefer to describe the target service with input and output since they are not familiar with service process. In order to tackle the problem of dynamic composition, services should be selected as candidates according to services composition requirements.

Definition 3(Services Composition Requirements). The services composition requirement requested by user is defined as a tuple $scr=(CI,CO,CD)$, where
1) $CI=\{i_1,i_2, i_3,.., i_n\}$ is a set of inputs requested by user.
2) $CO=\{o_1,o_2, o_3,.., o_n\}$ is a set of outputs requested by user.
3) CD is the domain concept, which represents an expect application scenarios.

All candidate services should be belonged to same domain to supports services composition. In implementation, services labeled with same domain concept to services composition requirements $scr.CD$ will be selected to build the candidate service set.

Definition 4(Candidate Service Set). The candidate service set is $CS=\{ws_i\}_{i=0}^{k}$ where,
1) $\forall ws \in CS \bullet ws.D = scr.CD$
2) $\forall i \in scr.CI, \exists ws \in CS \bullet \exists j \in ws.I \wedge i=j$
3) $\forall n \in scr.CO, \exists ws \in CS \bullet \exists m \in ws.O \wedge n=m$

If the candidate service set is empty $CS=\varnothing$, then services composition is infeasible. To provide fast interface retrieval, an inverted index is built based on input data. Definition 4 indicates that each input and output of services composition requirements should be guaranteed.

3.2. Service Process of Probabilistic Spanning Tree

Given candidate services *CS* and services composition requirements *scr*=(*CI*,*CO*,*CD*), the interface operation decomposition aims to iteratively select services according to their input and output.

Definition 5(The Operation Sequence). The operation sequence is computed based on interface operation decomposition at the beginning, which is defined as follow,

$$OS=\{<i,o_1,o_2,\ldots,o_{n-1},o>|\forall i\in scr.CI,\exists o\in scr.CO\bullet f_1(i)=o_1\wedge f_2(o_1)=o_2\wedge f_3(o_2)=o_3\wedge\ldots\wedge f_n(o_{n-1})=o\} \quad (1)$$

In operation sequence, i is the initial input data and o is the terminal output data. Each o_i has dual roles that the output data of precursor service ws_{i-1} can be used as input data of current services ws_i. Thus, service sequence is a path of service process, that

$$ss=<s, ws_1, ws_2, \ldots, ws_n, e> \quad (2)$$

Each service sequence includes one input and output of services composition requirements. For each o_i in *OS* to operation sequence, formula (2) means ws_i receives input o_{i-1} and sends output o_i that $f_i(o_{i-1})=o_i$. A virtual node s will be mapped to ws_1 with an input of services composition requirements, and ws_n will be mapped to another virtual node e with an output of services composition requirements.

Definition 6(The Service Sequence Set**).** The service sequence is reduced to compute the following set.

$$SS=\{(s,ws_1,ws_2,\ldots, ws_n,e)|\ \forall os\in OS,\forall k,\exists(ws_k,f_k,p_k)\ \bullet f_k(o_{k-1})=o_k\ \} \quad (3)$$

Definition 6 shows that the service sequence set consists of all possible composition paths of service process. We use probabilistic spanning tree to give the formal model of service sequences. It contributes to get solutions to service composition from the service sequence set.

Definition 7(Probabilistic Spanning Tree of Dynamic Composition). According to service sequence set *SS*, the probabilistic spanning tree is defined as tuple *PST*=(v_s, v_e, *V*, O, *P*, *E*), where

1) v_s is the starting vertex, which represents the user requested input.

2) v_e is the ending vertex, which represents the user requested output.

3) *V* is the set of service vertexes, where each vertex corresponds to the functionality of service sequence.

4) *O* is the set of outputs of candidate services in service vertex.

5) $E\subseteq V\times O\times V$ is the set of edges. For each vertex, outputs of ingoing vertex must be satisfied by at least one of outgoing vertexes' inputs.

6) P: $E\rightarrow[0,1]$ is the probabilistic relation function assigned a value to an edge ranged from 0 to 1;

The vertexes is generated form services of service sequence *SS*. The edge corresponds to temporal innovation relation. The probability for each edge is defined as follow.

1) If a vertex has two branches, then each edge's probability is

$$P((v,v'))=\frac{p(v')}{\sum_{\forall v''\in V\bullet(v,v'')\in E}p(v'')}=\frac{p(ws')}{\sum_{\forall ws''\in V\bullet(ws,ws'')\in E}p(ws'')} \quad (4)$$

2) If a vertex has one branches which is not a ending vertex, then the edge' probability is

$$P((v,v')) = p(v') = p(ws) \tag{5}$$

3) The probability for the ending vertex is

$$P((v,v_e)) = 1 \tag{6}$$

We use the example in motivation section to show the building process of probabilistic spanning tree. Suppose that services *S*1 has a probability of 0.9, *S*2 has a probability of 0.92, *S*3 has a probability of 0.86, *S*4 has a probability of 0.93, *S* 5 has a probability of 0.93, *S*6 has a probability of 0.89, *S*7 has a probability of 0.87, *S*8 has a probability of 0.88, *S*9 has a probability of 0.8, and *S*10 has a probability of 0.89. The probabilistic spanning tree for motivation section is as Figure 1 shown.

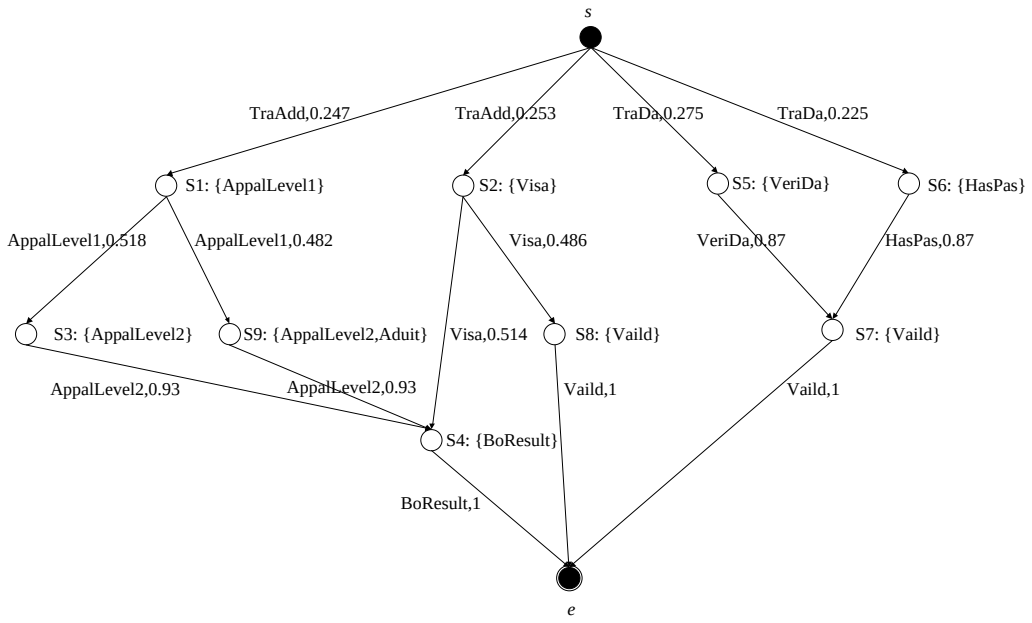

Figure 1. The Probabilistic Spanning Tree of Services Composition

In Figure 1, the initial input is Input={*TraAdd*, *TraDa*}. For the first layer of probabilistic spanning tree, there are four services {*S*1, *S*2, *S*5, *S*6} selected to continue with Output= {*AppalLevel1*, *Visa*, *VeriDa*, *HasPas*}. The vertex form v_s to *S*1 is an edge (v_s,*S*1) with a probability of 0.247. The vertex form v_s to *S*2 is an edge (v_s,*S*2) with a probability of 0.253. The vertex form v_s to *S*5 is an edge (v_s,*S*5) with a probability of 0.275. The vertex form v_s to *S*6 is an edge (v_s,*S*6) with a probability of 0.225. Then the second layer of probabilistic spanning tree uses output of the first layer as input to select their possible subsequent interaction services. For each round, if services can use invoker service's output, they will be regarded as next round services. However, for vertexes {S3, S9}, their probability is depended on follow-up service's probability since each vertex only has one edge. For the ending vertexes {S4, S8, S7}, their probability is fixed as 1.

4. Service Process Verification based on Probabilistic Model Checking

The services composition generated from combining existing services to provide functions with newly-added value requires corresponding mechanisms to ensure that component services are compatible with each other. We are motived to verify the composition plan in a quantitative way, mainly the reliability of probabilistic behaviors. It may have significant impact on the overall composite service.

According to services composition requirements *scr*=(*CI*,*CO*,*CD*), there are kinds of possible composition plans generated from probabilistic spanning tree. Each composition plan is first formalized into probabilistic automaton as service process model.

Definition 8(Service Process Model for Services Composition). The service process model is probabilistic automaton, which is defined as $SPM=(S, s_0, s_e, \Sigma, P, L)$, that,

(1) S is a finite set of services;

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

(3) $\delta \subseteq S \times S$ is a finite set of transition relations;

(4) $P : \delta \rightarrow [0,1]$ is the probabilistic relation function assigned a value to a transition ranged from 0 to 1;

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

For each solution of service process, it should be transformed into service process model. *AP* contains atomic propositions and interface data. The starting service s_0 is mapped to the starting vertex v_s. The ending service s_e is mapped to the ending vertex v_s. In Table 2, the algorithm analyzes each edge of probabilistic spanning tree and build service process model, where the label function is transformed based on output of service function to label atomic propositions of a state.

Table 2. Service Process Model Generation Algorithm

***Algorithm**: Service Process Model Generation Algorithm SPMGA()*
***Input**: PST=(v_s,v_e,V,O,P,E) and SCR=(CI,CO,CD)*
***Output**: Service Process Model SPM*

```
Function SPMGA (PST,SCR)
  s0←vs, and se←ve
  For each e=(v,o,v') in E
     S ←v
     Σ←(v,v')
     P(v,v')←P(e)
     L(v') ←f(v')
  EndEach
End Function
```

The probabilistic spanning tree describes all possible composition plans, which may have redundant paths of service process. Thus, the service process model should contain the minimum set covering the input and output of services composition requirements. An algorithm for pruning model is developed based on depth-first-search (DFS) to generate composition plans, which cuts a part of probabilistic spanning tree based on different pruning principles. We define following pruning principles for service process model.

Definition 9(Pruning Principles for Service Process Model). The pruning principles used to generate service process model from probabilistic spanning tree includes the Smallest Service Number, the Shortest Service Edge, the Minimum Input and the Minimum Output.

1) Smallest Service Number (SSN). For each input of services composition requirements, the SSN algorithm is to find a path with the smallest service number. After that, it ranks each path with input and output for paths integration covering services composition requirements. Finally, it merges these paths of service process into a composite service.

2) Shortest Service Edge (SSE). For each input of services composition requirements, the SSE algorithm is to find a path with the shortest service edge. After that, it ranks each path with input and output for paths integration covering services composition requirements. Finally, it merges these paths of service process into a composite service.

3) Minimum Input (MI). In probabilistic spanning tree, if each input of services composition requirements has more than two corresponding transitions initiating from starting service, then the MI algorithm is to randomly select one transition as a fixed path and cut other transitions using forward method. The forward algorithm iteratively cuts each path which is related to delete transitions. During deleting transition, the algorithm should guarantee the output of services composition requirements.

4) Minimum Output (MO). In probabilistic spanning tree, if each output of services composition requirements has more than two corresponding transitions finishing at ending service, then the MO algorithm is to randomly select one transition as a fixed path and cut other transitions using backtrack method. The algorithm iteratively cuts each path which is related to deleted transitions. During deleting transitions, the algorithm should guarantee the input of services composition requirements.

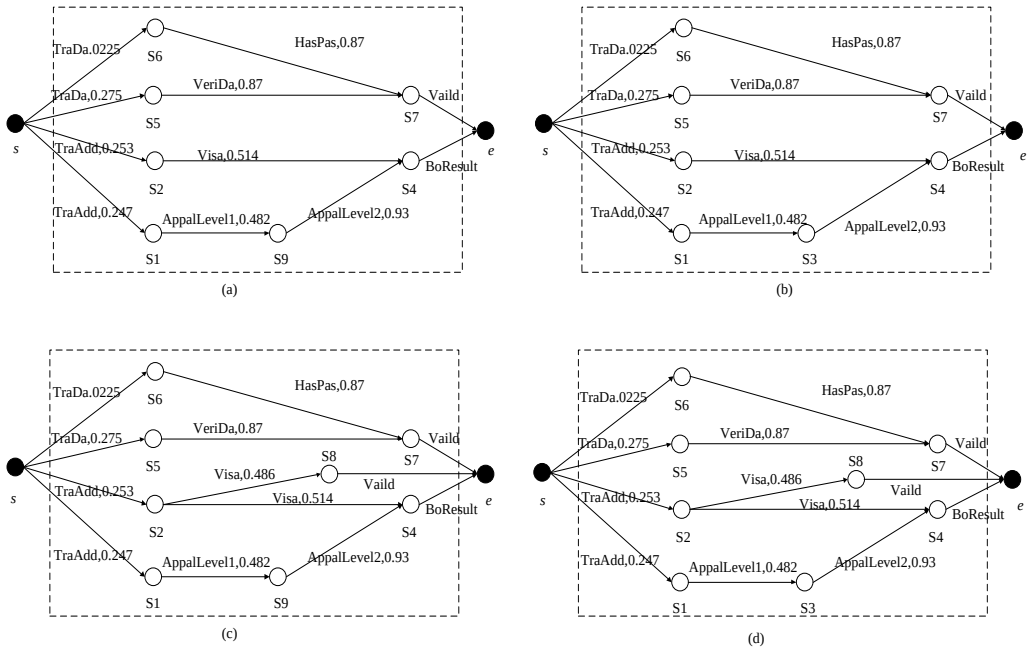

Figure 2. The Service Process of Services Composition

For example, in Figure 2, four models are generated from Figure 1. For a service, at least one of its inputs must be provided by some services in the previous step. How to evaluate these service processes is the core problem of formal verification. To characterize the personalized requirements of different users, it should take into account both the functional and non-functional requirements of end users. The verification property is in the form of Probabilistic Computation Tree Logic (PCTL), which can specify quantitative properties [8].

Definition 10(Probabilistic Computation Tree Logic). The PCTL syntax is defined as follows:

$$\varphi ::= true \mid false \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \neg \varphi \mid P_{\sim p}[X\varphi] \mid P_{\sim p}[\varphi \mathbf{U} \varphi] \mid P_{\sim p}[\varphi \mathbf{U}^{\leq k} \varphi] \mid P_{\sim p}[\mathbf{F}\varphi] \mid P_{\sim p}[\mathbf{G}\varphi]$$

where $\sim \in \{<, \leq, >, \geq\}$, $0<p<1$ is a probability bound or threshold, a is an atomic proposition, and $k \in$ □ is denote as time steps. Operator **P** is the probability operator. Operators **X**, **F**, **G** and **U** are all state quantifiers, meaning 'ne**X**t state', **'F**uture state', **'G**lobal state', and **'U**ntil', respectively.

Definition 11(The Verification Property Generation)**.** According to services composition requirements, we define PCTL based properties for formal verification.

1) The reachability property is generated as follow, that,

$$\bigcup_{\forall o \in scr.CO} P_{\sim p}(Fo) \tag{7}$$

The reachability property requires each output of services composition requirements should be covered in a future state with a probability value satisfied to the specified threshold value.

2) The data order property is generated as follow, that,

$$\bigcup_{\forall o \in scr.CO, \forall i \in scr.CI} P_{\sim p}(\neg F(o \rightarrow Fi)) \tag{8}$$

The data order property should prevent that output data is returned before input data with a probability value satisfied to the specified threshold value.

3) The data lost property is generated as follow, that,

$$\bigcup_{\forall o \in scr.CO, \forall i \in scr.CI} P_{\sim p}(\neg i \cup o) \tag{9}$$

The data lost property means that the output data is returned without using any input data with a probability value satisfied to the specified threshold value.

4) The data redundancy property is generated as follow, that,

$$\bigcup_{\forall ap \notin AP} P_{\sim p}(F(ap)) \tag{10}$$

The data redundancy property indicates that a data is redundant to service composition with a probability value satisfied to the specified threshold value. However, there is no service to use that data.

Model checker will receive model and property to perform automatic verification, such as PRISM [8]. When verifying service process model, all properties should be verified. For the reachability property and the data order property, if one of properties is proved to be false, the service process model for services composition is not acceptable. For the data lost property and the data redundancy property, if one of properties is proved to be true, the service process model for services composition is not acceptable. Otherwise, the service process model for services composition is selected and recommended to user as a solution.

5. Related Works

To the best of our knowledge, researchers have concentrated on services composition. Services composition improves the flexibility of service software, by which business processes does not be constructed in advance. It is a promising solution to resource sharing and application integration. We give a review on the major techniques and researches that are most closely related to our works.

Many literatures of dynamic services composition research AI planning problem. Remli [9] presented the experience gained on semantic services composition technique

which is applied to bioinformatics domain. They executed the composite service by treating composition as planning problem using Hierarchical Task Network (HTN) planning system based on Simple Hierarchical Order Planner 2 (SHOP2). Huang [10] presented an exception handling framework for services composition. The framework provided a model for Web service with exceptions specification. A holistic planning paradigm was introduced to increase the expressiveness of AI planning domain. Zhu [11] proposed a multi-purpose model to specify multiple goals. Then, they developed planning techniques to achieve automated service composition and derived workflows for the system which can deal with multiple goals.

Many literatures of dynamic services composition are related to user requirements. Li [12] considered user-centric application scenarios, and proposed PASS, a novel approach to personalized automated service composition. With PASS, both the hard-constraints represented by user's initial state, and the soft-constraints represented by user preferences could be satisfied in the process of automated service composition. Chattopadhyay [13] presented an approximate QoS-aware service composition mechanism which balanced the computational complexity of service composition. The Heuristic based approaches was proposed to compose services optimally. Takahashi [14] proposed a new composition approach in which a developer directly expressed his or her intentions as constraints via metadata, and then the system searched for optimal composition methods based on the constraints. Tan [15] proposed a fully automated approach to synthesize the response time requirement of component services, in the form of a constraint on the local response times, which guarantees the global response time requirement.

Many literatures of dynamic services composition focus on searching algorithm. Mohr [16] presented an algorithm that automatically composes services without making such assumptions. They employed a backward search algorithm to discover candidates until a solution was found, which starts from an empty composition. Available services were determined during the search process. Fki [17] proposed a solution that enables a certain flexibility and adaptability without dealing with composition from scratch at runtime. It supports generating automatically a composition schema at the abstract service level. The design-time components of solution were OWL-S abstract services, which were used within the composition process for deriving different composition possibilities. It reduces the difficulty of tasks to discover and select the actual Web services.

Many literatures of dynamic services composition design system architecture and framework. Siriweera [18] proposed a novel architecture to automate data analytics process using Nested Automatic Service Composition (NASC) and CRoss Industry Standard Platform for Data Mining (CRISP-DM). The CRISP-DM will be mapped with Big Data analytical process and NASC will automate the process of CRISP-DM in an intelligent and innovative way. Pathathai [19] presented a framework for services composition and execution based on the integration of different domains. It first provided an automated composition of services based on Fluent Calculus. Then, the plan was transformed into a BPMN (Business Process Modeling Notation) model in order to be executed by any BPMN engine. Kuehne [20] presented the Automatic Web service composition Architecture (AWSCA) to support a fair comparison for service composition. Qi [21] summarized existing services composition approaches, and then presented a service classification management mechanism to organize web services more efficiently and accurately. It included two main parts: service management subsystem and service provision subsystem.

Many literatures of dynamic services composition concern on non-determinism planning problems. Mohr [22] showed how non-determinism in automated service composition can be reduced. They introduced context rules to derive semantic knowledge from output values of services. These rules enabled us to replace nondeterministic composition operations by less nondeterministic or even completely deterministic ones. Groba [23] presented a novel service composition protocol that allocates and invokes

service providers opportunistically to minimize the impact of topology changes and to reduce failure. Automated model checking verified the protocol deadlock, by which it terminated in a valid end state after having allocated the correct number of service providers for all required sub-services. Lee [24] proposed a novel service type to improve functional flexibility of service composition. It possessed nondeterministic service interface beyond concrete and abstract services used by current service composition methods.

Different from above existing works, our contribution is to apply probabilistic model checking to verify quantitative properties against service process model. We propose a probabilistic spanning tree to describe the possible composition plans specified with different service probabilities. The service process is generated according to probabilistic automaton for formal verification. Then, each dynamic composition plan is checked based on probabilistic model checking to decide whether it is a suitable solution to user or not.

6. Conclusion

Web service technologies have been widely used by the businesses and scientific community. It exposes its functionality by the mean of public interface consisting of several operations. However, complex user requirements call for services composition which selects and organizes a set of services to work with each other. It is an important task to make sure that the services composition conforms to their functional and non-functional requirements.

In this paper, we propose an approach to service process verification of probabilistic spanning tree for dynamic composition. First, Web service is formalized with input and output, by which the interface operations decomposition is used to build probabilistic spanning tree. Second, the service process model is generated from probabilistic spanning tree which considers different pruning strategies to extract the minimum covering set, including the smallest service number, the shortest service edge, the minimum input, and the minimum output. After that, the probabilistic model checking is employed to verify the service process model by using quantitative properties.

Time constrains of service interactions affect the overall functionalities of services composition, such as response time. The satisfaction of the response time of composite service should be guaranteed. To this propose, we will research how to handle timed probabilistic properties and take more consideration of users' preferences to support personalized compositions in the future.

Acknowledgment

This paper is supported by Postdoctoral Science Foundation of Zhejiang Province of China under Grant No.BSH1502119.

References

[1] 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.
[2] Y. Yin and S. G. Deng, "Analysing and determining substitutability of different granularity Web services", International Journal of Computer Mathematics, vol. 90, no. 11, **(2013)**, pp. 2201-2220.
[3] S. Dustdar and W. Schreiner, "A survey on Web services composition", International Journal of Web and Grid Services (IJWGS), vol. 1, no. 1, **(2005)**, pp. 1-30.
[4] S. Narayanan and S. A. McIlraith, "Simulation, Verification and Automated Composition of Web Services", The 11th international conference on World Wide Web (WWW), **(2002)**, pp. 77-88.
[5] Y. Feng, A. Veeramani, R. Kanagasabai and S. Rho, "Automatic Service Composition via Model Checking. The 2011 IEEE Asia-Pacific Services Computing Conference (APSCC), **(2011)**, pp.477-482.
[6] Lixing Li, Zhi Jin, Ge Li, Liwei Zheng and Qiang Wei, "Modeling and Analyzing the Reliability and Cost of Service Composition in the IoT: A Probabilistic Approach", The 2012 IEEE 19th International Conference on Web Services (ICWS), **(2012)**, pp. 584-591.

[7] S. K. Malik, N. Prakash and S. A. M. Rizvi, "Ontology Merging Using Prompt Plug-In of Protégé in Semantic Web", The 2010 International Conference on Computational Intelligence and Communication Networks (CICN), **(2010)**, pp. 476- 481.
[8] A. Hinton, M. Kwiatkowska, G. Norman and D. Parker, "PRISM: A Tool for Automatic Verification of Probabilistic Systems", The 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), **(2006)**, pp. 441-444.
[9] M. A. B. Remli and S. B. Deris, "Automated biological pathway knowledge retrieval based on semantic web services composition and AI Planning", The 2012 International Conference on Information Retrieval & Knowledge Management (CAMP), **(2012)**, pp. 281-284.
[10] J. Huang, W. Zhu, F.Bastani, I. L. Yen and J. Fu, "Automated Exception Handling in Service Composition Using Holistic Planning", The 2012 IEEE 15th International Conference on Computational Science and Engineering (CSE), **(2012)**, pp. 251- 258.
[11] W. Zhu, J. Huang, F. B. Bastani and I. L. Yen, "Multi-purpose Planning for Practical Web Service Composition Problems", 2013 Fifth International Conference on Service Science and Innovation (ICSSI), **(2013)**, pp. 131-138
[12] Y. Li, J. P. Huai, H. Sun, T. Deng and H. Guo, "PASS: An Approach to Personalized Automated Service Composition", IEEE International Conference on Services Computing (SCC), **(2008)**, pp. 283-290.
[13] S. Chattopadhyay, A. Banerjee and N. Banerjee, "A Scalable and Approximate Mechanism for Web Service Composition", The 2015 IEEE International Conference on Web Services (ICWS), **(2015)**, pp. 9-16.
[14] R. Takahashi, F. Ishikawa, K. Tei and Y. Fukazawa, "Intention-Based Automated Composition Approach for Coordination Protocol", The 2013 IEEE 20th International Conference on Web Services (ICWS), **(2013)**, pp. 260-267.
[15] T. H. Tan, E. Andre, J. Sun, Y. Liu, J. S. Dong and M. Chen, "Dynamic synthesis of local time requirement for service composition", The 2013 35th International Conference on Software Engineering (ICSE), **(2013)**, pp. 542-551.
[16] F. Mohr, A. Jungmann and H. K. Büning, "Automated Online Service Composition", The 2015 IEEE International Conference on Services Computing (SCC), **(2015)**, pp. 57-64.
[17] E. Fki, S.Tazi and M. Jmaiel, "A Semantic Driven Approach for an Automated Composition Based on Abstract Services", The 2015 IEEE 12th International Conference on e-Business Engineering (ICEBE), **(2015)**, pp. 141-146.
[18] T. H. A. S. Siriweera, I. Paik, B. T. G. S. Kumara and K. R. C. Koswatta, "Intelligent Big Data Analysis Architecture Based on Automatic Service Composition", The 2015 IEEE International Congress on Big Data, **(2015)**, pp. 276-280.
[19] P. N. Lumpoon, M. C. Fauvet and A. Lbath, "Toward a framework for automated service composition and execution, The 2014 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), **(2014)**, pp. 1-8.
[20] B. T. Kuehne, R. Helena, C. Santana, V. Linnemanna and M. J. Santana, "Performance evaluation of an automatic web service composition architecture", The 2013 International Conference on High Performance Computing and Simulation (HPCS), **(2013)**, pp. 123-130.
[21] S. Qi, X. Tang and D. Chen, "An Automated Web Services Composition System Based on Service Classification and AI Planning", The 2012 Second International Conference on Cloud and Green Computing (CGC), **(2012)**, pp. 537-540.
[22] F. Mohr, T. Lettmann and H. K. Büning, "Reducing Nondeterminism in Automated Service Composition", The 2013 IEEE 6th International Conference on Service-Oriented Computing and Applications, **(2013)**, pp. 154 -161.
[23] C. Groba and S. Clarke, "Opportunistic Service Composition in Dynamic Ad Hoc Environments", IEEE Transactions on Services Computing, vol. 7, no. 4, **(2014)**, pp. 642 -653.
[24] C. H. Lee, S. Y. Hwang and I. L. Yen, "Service Composition with Functional Flexibility Using Nondeterministic Service Interface", The 2013 IEEE 10th International Conference on e-Business Engineering (ICEBE), **(2013)**, pp. 435-440.

Author

Honghao Gao, is a post-doctoral researcher of College of Computer Science and Technology of Zhejiang University. His research interests include Service Computing and Software Engineering.

