Model Checking for Asynchronous Web Service Composition Based on XYZ/ADL

Guangquan Zhang[1,2], Huijuan Shi[1], Mei Rong[3], and Haojun Di[1]

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
[2] State Key Laboratory of Computer Science, Chinese Academy of Science, Beijing, China
[3] Shenzhen Tourism College, Jinan University, Shenzhen, China
gqzhang@suda.edu.cn

Abstract. Concerned with Web service composition, this paper proposes a model checking method of verifying asynchronous communication behaviors and timed properties. Firstly, analyzing Web service composition from software architecture, the interactive behaviors and timed properties are described by XYZ/ADL based on temporal logic language. Secondly, timed asynchronous communication model (TACM) which accords with the specification of model checker UPPAAL is proposed. Finally, based on the transition from XYZ/RE communication commands to TACM, the correctness of asynchronous communication behaviors of the service composition system can be verified by UPPAAL.

Keywords: Web service composition, XYZ/ADL, asynchronous communication, timed properties, model checking.

1 Introduction

Service Oriented Architecture (SOA), which supports reusability, loose coupling, interoperability, platform independent, is an innovative architecture paradigm. Web service technology, a widely used distributed computing technology, has become main implement way of SOA. With the fast development of e-business, single Web service couldn't meet the complicated demand, which causes service composition to be a research hotspot of software service field.

Web service composition implements big service function through the communication and coordination among small Web services. The communication includes synchronous and asynchronous communication. The existing works [1-3] almost are based on synchronous communication. However, the nature of distributed systems and particularly Web services are asynchronous, which makes these approaches restrictive in real application scenarios. To overcome such limitations, some works tried to consider the asynchronous communication. Ref. [4] compared the difference and similarity between synchronous and asynchronous communication of Web service in detail, and presented up-bottom and bottom-up asynchronous interaction model of it. Ref. [5] researched asynchronous communication using recall, and proposed a reliability interaction pattern of Web service. Ref. [6] discussed choreography, orchestration and session of Web service by asynchronous π-calculus. However, these works didn't consider timed properties when analyzing the asynchronous communication. Additionally, we verify timed properties of composite Web service in Ref. [7], but it didn't support the asynchronous communication.

To resolve above problems, we propose a model checking method to verify the interactive behavior and timed properties of Web service. Firstly, use XYZ/ADL based on temporal logic language to precisely characterize the interactive behaviors and timed properties of Web service. Then propose a Timed Asynchronous Communication Model (TACM), which can be verified directly using model checker UPPAAL. By translating the XYZ/RE communication commands to TACM, the related properties of Web service can be verified by UPPAAL. Compared to other approaches, it need not convert the system model to the specification of model checker, which simplifies the verification process.

2 Web Service Composition Based on XYZ/ADL

Compared Web service composition and Software Architecture (SA), we can find that they have some similarities. Web service can be considered as component. The interactive rules of Web services correspond to connector. The whole layout of Web service composition can be regarded as configuration. These relations provide support and foundation for researching Web service composition from a higher level. It is convenient to realize formal verification using XYZ/ADL, an architecture description language (ADL) based on temporal logic, to describe Web service composition.

2.1 XYZ/E

XYZ/E is an executable temporal logic language, in which basic unit is conditional element. There are two forms:

$$\mathtt{LB=y} \wedge \mathtt{R} \Rightarrow \mathtt{\$Ov=e} \wedge \mathtt{\$OLB=z} \tag{1}$$

$$\mathtt{LB=y} \wedge \mathtt{R} \Rightarrow \mathtt{@(Q} \wedge \mathtt{LB=z)} \tag{2}$$

Where, R and Q are first-order logic formula, R represents the condition part, Q and $Ov=e are action parts, LB is control variable, y and z are the entry and exit label respectively, => represents logic implication. In formula (1), conditional element defines the transition relation of the adjacent states. The symbol @ in formula (2) can be next operator $O or final operator < >, they represent the abstract specification of a program.

2.2 XYZ/ADL and XYZ/RE

XYZ/ADL, supports the concept of component, connector, configuration in SA, is an ADL extended from XYZ/E. It can describe system from formal specification to executable program under unified logic framework. A completed XYZ/ADL description is shown as follows.

```
%COMPONENT COM==[
%PORT P:Record (%CHN A:DATATYPE1; %CHN B: DATATYPE2)
```

```
  %PROPERTY== […]
  %COMPUTATION== […]]
  %CONNECTOR CON== [
  %ROLE SourceData==OUT(DT1,v1);
  %ROLE  SinkData==IN(DT2,v2) ;
  %GLUE== […]]
  %ATTACHMENTS==[ComIns.Port#ConIns.Role;ComIns.Port##Por
t;…]
```

Where, the internal specification of component uses a XYZ/E unit to represent the behavior of different layer of Web service. If involves timed restraint it can be described by XYZ/RE. Extended the temporal operator $O, < >, [ ], $U, $W in XYZ/E in order to make them capable to express the real time lower limit(l) and upper limit(u), so we get real time XYZ/E, namely XYZ/RE [8]. XYZ/RE conditional element has tow basic forms:

```
  LB=L0∧R⇒$O{l,u}(Q∧LB=L1)
  LB=L0∧R⇒@{l,u}(Q∧LB= L1)
```

3 Asynchronous Communication of Web Services

Asynchronous communication is important for building robust Web services [4]. This section presents a survey of related work on analyzing and modeling the asynchronous Web service.

3.1 Analysis of Asynchronous Communication

Web services interact by exchanging messages, which includes sending and receiving message. They are denoted by !m and ?m respectively. The real Web service composition often involves timed properties, which are crucial properties of service interaction [9]. When modeling Web services, we use the standard timed automata clocks to capture the timed properties. The values of these clocks increase with the passing of time. Transitions are labeled by timed constraints, called guards, and resets of clocks. Figure 5 shows an example of timed asynchronous services interaction. To assure the correct interaction between asynchronous services, each service is equipped with a queue to store the incoming message. We assume the queue is unbounded and messages can be consumed in any arbitrary order.

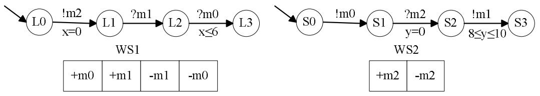

Fig. 1. A simple example of timed asynchronous service

Let us not consider the timed constraint at first. The service WS1 starts by sending message m2, which is stored (indicated by '+') in the queue of WS2. On the other hand, WS2 can send message m0, which is added in the queue of WS1. The service WS1 remains blocked because the message m1 can't be available. Then WS2 consumes (indicated by '-') message m2 and send message m1. After that, WS1 can consume message m1 and m2. Consequently, WS1 and WS2 finish interaction successfully and both queues become empty. If consider the timed constraints, the communication process is shown as follows. WS1 sends message m2 and reset the clock (x=0).WS2 reset the clock (y=0) after consuming message m2. Then WS2 sends message m1 after 8 and within 10 units of time ($8 \le y \le 10$). Finally, WS1 must consume message m0 within 6 units of time. However, m0 can only be consumed after consuming m1, i.e., after 8 units of time. Obviously, WS1 represents a timed conflict at state L2 and can't transfer to final state L3.

3.2 Timed Asynchronous Communication Model

To model the asynchronous Web service, we first abstract clocks and messages to two variables of distinct types respectively. Initially, the values of the variables equal to zero and the message queues are empty. The clock variables are continuous. Their values increase automatically with the passing of time and can be reset to zero. The message variables are discrete and their values can only equal to 0 or 1.When send message, this message is added into the queue and the value of associated variable is set to 1.When receive message, first check if this message be in the queue or the value of related variable equals to 1. If the result is true, consume this message and set the value of corresponding variable to 0. According to above analysis, Web service asynchronous communication model is defined as follows:

Definition 1. (Timed Asynchronous Communication Model) A TACM is a tuple (S, s_0, F, C, M, A, T) such that S is a set of states, s_0 is initial state ($s_0 \in S$), F is the set of final states ($F \subseteq S$), C is the set of clock variables, M is the set of message variables, $A{:}M \rightarrow \{?,!\}$ is a labeled function, it assigns an action of receive (?) or send (!). $T \subseteq S \times g(C, M) \times u(C, M) \times S$ is a set of state transition. A transition from state s to state s', denoted (s, g, u, s'), will be triggered if the guard condition g is satisfied, and it will execute action u to update the values of clock and message variables.

The set of constraints over C and M, denoted g(C, M), is defined as follows:

$g(C,M)= \text{true} \mid x \sim v \mid y == 1 \mid \psi 1 \wedge \psi 2$, where $\sim \in \{\le,<,==,!=,>,\ge\}, x \in C$, $y \in M, \psi 1, \psi 2 \in g(C,M)$, $v \in \mathbb{R}_{\ge 0}$ is a positive real constant.

The set of actions which update C and M, denoted u(C, M), is defined as follows:

$u(C,M) = x{:=}n \mid y{:=}0 \mid y{:=}1 \mid \varphi 1 \wedge \varphi 2$, where $x \in C$, $y \in M$, n is a constant, $\varphi 1, \varphi 2 \in g(C,M)$.

If A (m) =? then the constraints over M in g(C,M) is m==1, the updating action over M in u(C,M) is m:=0; If A (m) =! then there is no constraints over M, the updating action over M in u(C,M) is m:=1.

TACM is a behavioral model, which can precisely characterize the asynchronous interactive behaviors of Web service and related timed restraints. The elements describing behavior consist of finite states, conditions of triggering transition and actions executed during transition, which completely correspond to the elements of XYZ/E conditional element. Additionally, TACM accords with the specification of model checker UPPAL.

4 Model Checking of Asynchronous Web Service Composition

Model checking [10] is a method for formally verifying finite-state concurrent system. Specifications about system are expressed as temporal logic formula, and test automatically the given model of system whether meets the specifications. At present, a lot of distinctive model checking tools have been used widely, such as SPIN, NuSMV, UPPAAL and so on. This paper selects UPPAAL to verify Web service composition. The main reasons are that UPPAAL can verify the timed properties of Web service and TACM satisfies the input model of UPPAAL.

The specific verification process of asynchronous Web service composition based on XYZ/ADL is shown as follows. Firstly, use XYZ/ADL to describe the Web service composition. Then translate the XYZ/RE communication commands in XYZ/ADL to TACM. Finally, express the specification with a CTL formula, and put both TACM and the specification into UPPAAL to verify the relative properties.

In the XYZ/ADL description of Web service composition, the interactive behaviors are described by XYZ/E unit and the related timed properties are represented by XYZ/RE. Web services interact by exchanging messages. Obviously, the mostly used statements are input and output sentences, namely the communication commands. Therefore, we only realize the mapping from communication commands to TACM. The table1 shows the mapping rule.

Table 1. Mapping rule from communication commands to TACM

	XYZ/RE	**TACM**
Input sentence	LB=L1 $\wedge R_1 \wedge$ ch?$\Rightarrow$\$O$\{l,u\}$($Q_1 \wedge$ \$OLB=L2)	$L1 \xrightarrow{R_1, l \le x \le u, ch==1 \quad Q_1, ch:=0} L2$
Output sentence	LB=L3 $\wedge R_2 \wedge$ ch!$\Rightarrow$\$O$\{l,u\}$($Q_2 \wedge$ \$OLB=L4)	$L3 \xrightarrow{R_2, l \le x \le u \quad Q_2, ch:=1} L4$

5 Case Analysis

Let us present a composite service BuyBook to illustrate our approach. It involves three Web services: Customer, BookShop and Bank. The interactive process and timed constraints are briefly summarized as follows. Customer sends a *searchBook* request to BookShop. After searching, if there is stock BookShop returns *bookID* and *bookPrice* successively, or returns *outStock* with in 5 units of time. Customer receives the message *bookPrice* at first, sends Bank a request *balanceInquire* within 2 units of time to search the balance of account, and then receives *bookID*. Bank returns the message *balance* within 3-5 units of time. If the balance is enough Customer sends a message *payInfo* to Bank, or sends *cancel* to BookShop within 5-10 units of time. After paying the bill, Bank sends a message *payConfirm* within 3 units of time to inform BookShop that Customer has paid the bill. Finally, BookShop sends *invoice* to Customer within 4 units of time. This is an example of timed asynchronous communication. The XYZ/ADL description codes are shown as follows.

```
%COMPONENT Customer==[
LB=S0∧!searchBook⇒$OLB=S1;
LB=S1∧?outStock⇒$OLB=S2;
LB=S1∧?bookPrice⇒$O(x=0∧LB=S3);
LB=S3⇒$O{0,2}(!balanceInquire∧LB=S4);
LB=S4∧?bookID⇒$OLB=S5;
LB=S5∧?balance⇒$O(x=0∧LB=S6);
LB=S6∧!cancel⇒$O{5,10}(LB=S7);
LB=S6∧!payInfo⇒$O{5,10}(x=0∧LB=S8);
LB=S8∧?invoice⇒$OLB=S9;]
%COMPONENT BookShop==[
LB=L0∧?searchBook⇒$O(y=0∧LB=L1);
LB=L1∧!outStock⇒$O{0,5}(LB=L2);
LB=L1∧!bookID⇒$O{0,5}(LB=L3);
LB=L3∧!bookPrice⇒$OLB=L4;
LB=L4∧?cancel⇒$OLB=L5;
LB=L4∧?payConfirm⇒$O(y=0∧LB=L6);
LB=L6⇒$O{0,4}(!invoice∧LB=L7);]
%COMPONENT Bank==[
LB=M0∧?balanceInquire⇒$O(z=0∧LB=M1);
LB=M1⇒$O{3,5}(!balance∧LB=M2);
LB=M2∧?payInfo⇒$O(z=0∧LB=M3);
LB=M3⇒$O{0,3}(!payConfirm∧LB=M4);]
```

According to the mapping rule from XYZ/RE communication commands to TACM, we can obtain the TACM of Customer, BookShop and Bank. Figure 6 shows the models, where S2, S7 and S9 in Customer are final states. L2, L5 and L7 are final states of BookShop. In Bank, M2 and M4 are final states. These final states are all labeled in green.

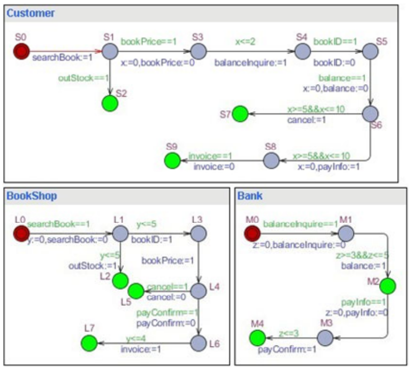

Fig. 2. TACM of Customer, BookShop and Bank

We use model checker UPPAAL to mainly verify deadlock, safety properties and liveness properties.

(1) Deadlock indicates that Web service stay at some state, which doesn't satisfy transition condition, and can't continue to interact. In TACM, no deadlock is equivalent to check if all the services reach their final states. In other words, when the services reach their final states, all the values of the message variables must be equal to zero. It is specified as the following CTL formulas:

E<>(Customer.S2 and BookShop.L2) or (Customer.S7 and BookShop.L5 and Bank.M2) or (Customer.S9 and BookShop.L7 and Bank.M4) and (*searchBook*==0 and *bookPrice*==0 and *balanceInquire*==0 and *bookID*==0 and *balance*==0 and *payInfo*==0 and *invoice*==0 and *cancel*==0 and *outStock*==0 and *payConfirm*==0).

(2) Safety properties are on the form: "something bad will never happen". In this example, the user requests "Customer receive the message *invoice* without paying the bill" never happen. It is specified as the following CTL formulas:

A[] not (Customer.S7 and Customer.S9).

(3) Liveness properties are on the form: "something will eventually happen". In this example, the user requests "Customer receive *invoice* within 10 units of time after paying the bill" eventually happen. It is specified as the following CTL formulas:

A<>Customer.S8 imply Customer.S9 and x<10.

6 Conclusion

Web service composition is one of the research hotspots in Service Oriented Computing (SOC). Asynchronous communication is an important feature of information exchange in Web service composition. However, the current works seldom consider timed properties and asynchronous communication. Therefore, we propose a timed asynchronous communication model TACM, which can precisely describe the timed properties and asynchronous communication behaviors. Additionally, TACM can be directly put into UPPAAL to verify the related properties. On the other hand, we analyze Web service composition from software architecture, which makes it possible to control the layout of the whole system. As future work, we will improve TACM to apply to more asynchronous communication scenarios. What's more, we will consider how to capture and handle all kinds of exceptions occurring in the interactive process of Web service.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China under Grant No.60973149; the Natural Science Foundation of Jiangsu Province, China; the Open Funds of State Key Laboratory of Computer Science of Chinese Academy of Science under Grant No.SYSKF0908; the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant No.08KJB520010.

References

1. Pistore, M., Roveri, M., Busetta, P.: Requirements-driven verification of Web services. In: WSFM 2004, pp. 95–108 (2004)
2. Lei, L.H., Duan, Z.H.: An extended deterministic finite automata based method for the verification of composite Web services. Journal of Software 18(12), 2980–2990 (2007) (in Chinese)

3. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based verification of Web service compositions. In: IEEE ASE 2003, pp. 152–161 (2003)
4. Fu, X., Bultan, T., Su, J.: Synchronizability of conversations among Web services. IEEE Transactions on Software Engineering 31(12), 1042–1055 (2005)
5. Pallickara, S., Fox, G., Pallickara, S.L.: An analysis of reliable delivery specifications for Web services. In: 10th International Conference on Information Technology: Coding and Computing, pp. 360–365 (2005)
6. Vieira, H.T., Caires, L., Seco, J.C.: The conversation calculus: A model of service oriented computation. In: 17th European Symposium on Programming, pp. 269–283 (2008)
7. Zhang, G.Q., Rong, M., He, Y.L., Zhu, X.Y., Yan, R.J.: A refinement checking method of Web service composition. In: IEEE SOSE 2010, pp. 103–106 (2010)
8. Tang, Z.S., et al.: Temporal logic programming and software engineering. Science Press, Beijing (2002) (in Chinese)
9. Guermouche, N., Godart, C.: Timed model checking based approach for Web services analysis. In: IEEE ICWS 2009, pp. 213–221 (2009)
10. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)