Compatibility Verification of Web Service Composition Based on Pi-calculus

Zhichun JIA[1,a*] and Xing XING[1,2,b]

[1] College of Information Science and Technology, Bohai University, Jinzhou 121013, China

[2] School of Astronautics, Harbin Institute of Technology, Harbin 150001, China

[a] zhichun.jia@bhu.edu.cn, [b] xingxing@bhu.edu.cn

Keywords: Behavior Compatibility, Interactive Behavior, Pi-calculus, Web Service Composition.

Abstract. With the rapid development of web services, service computing is evolving into an important computing paradigm. One of the challenges in this evolution is how to ensure the correction of service composition. To verify the compatibility of composite service, we use pi-calculus to model the composite service process and present the related concepts and theorem of composition compatibility. Moreover, we propose an automate method to generate the process expression for increasing the verification efficiency.

Introduction

As a popular distributed computing paradigm, service-oriented computing makes software applications even more attractive due to it provides a lot of benefits like scalability, loose-coupled and on demand business applications in heterogeneous environments. However, along with these benefits, web service also raises some concerns especially how to ensure the correct running of service applications [1]. A critical challenge is how to build high-reliable composite service applications in the large-scale and complex computing environment.

The main standard to verify the correctness and usability of web service composition is that the composite service applications can run correctly and achieve the users' goals successfully [2]. Therefore, it is critical to analyze and verify web service composition before the implementation of composite service applications. At present, there exits three formal methods for verifying the web service composition: Petri nets, finite automation and process algebra [3]. The behavior theory of pi-calculus provides a good theoretical basis for the verification of web service composition [4]. So, we use pi-calculus to model and verify the behavior compatibility of web service composition in this paper.

The main goal of the compatibility verification for web service composition is to verify their interaction logic in a modeling manner for ensuring that their interactions do not lead to any undesired outcome and violate any business process constraint [5]. Many earlier studies focused on the correctness verification of internal business logic without considering the dynamic property of service behavior [6]. Deng in [4] gave the interface view, behavior view and service view to explain the service behavior compatibility and presented a verification method of behavior compatibility. The literature [7] applied pi-calculus to model web service and formalized the interactive behavior between services. However, this method did not consider the verification of multi-services interaction (three or more services). The literature [8] dealt with the change of composite service by using pi-calculus to present the business process execution language. However, the method proposed in [8] described the behavior variables and control flows in a static manner and did not explain the dynamic interaction process between services. The literature [1] represented and verified the dynamic interaction process between composite services by pi-calculus, but it only considered the message exchange in two services. In contrast with them, we focus on the behavior interaction between multi-services. In our verification method, a service could have two or more outputs and inputs at the same time. Moreover, our method can automatically generate the process expression and the message sequence of web service.

Service Behavior and Behavior Compatibility

Service Behavior.

Definition 1 (message body) A message body is 4-tuple $mb = (gate, type, mes, num)$, where:

(1) S^n is a set of web services which communicate with each other, and s_i is a service in S^n;

(2) *gate* means a channel of the service s_i, *type* means the type of the channel, *accept* denotes receive channel and *send* denotes send channel;

(3) *mes* means the messages sent by the channel *gate*, *num* means the number of messages in *mes*.

Definition 2 (state transition cell) A state transition cell is 3-tuple $st = (t_f, t_a, C, MB)$, where:

(1) t_f means the state of the service s_i before transition, t_a means the state of the service s_i after transition, *C* means the set of the transition conditions, if there is no any conditions, $C = \phi$;

(2) *MB* means the set of message bodies, and the messages in these message bodies are sent or received by the service s_i.

Definition 3 (service behavior) A service behavior is 3-tuple $sb = (t_b, t_e, ST)$, where:

(1) t_b means the beginning state of the service s_i, t_e means the ending state of the service s_i;

(2) *ST* means the set of all state transition cells in the service s_i.

Definition 4 (transition sequence) A transition sequence of the service s_i in the set of S^n $TS_i =< st_1, st_2, \ldots, st_k)$, where:

(1) $\forall st_j \in TS_i, \exists st_j \in ST_i$ and $1 \le j \le k$, $st_1 = (t_{f1}, t_{a1}, C_1, MB_1)$ and $t_{f1} = t_{bi}$;

(2) ST_i is the set of all state transition cells in $sb_i = (t_{bi}, t_{ei}, ST_i)$ and sb_i is the service behavior of the service s_i, if $st_j = (t_{fj}, t_{aj}, C_j, MB_j)$ and $st_{j+1} = (t_{f(j+1)}, t_{a(j+1)}, C_{j+1}, MB_{j+1})$ in TS_i are two consecutive state transition cells, and $1 \le j \le k$, then $t_{aj} = t_{f(j+1)}$.

Definition 5 (entire transition sequence) Let a transition sequence of the service s_i in S^n $TS_i =< st_1, st_2, \ldots, st_k >$ and $st_k = (t_{fk}, t_{ak}, C_k, MB_k)$, if t_{ak} is the end-state of s_i, then TS_i is an entire transition sequence.

Behavior Compatibility.

Web service behavior describes the transition of service state and transmission of service message. Web service compatibility is also called the correctness of service interaction protocol, which is used to validate the interaction process between services. Take Fig. 1 for example, the service B sends the message mes1 to the services A and C by the gate G1. A receives mes1 by G2 and C receives mes1 by G3. At the same time, the states of three services change from 1 to 2.When the states of three services all are 2, C sends mes2 to B by G4 and its state changes from 2 to ending state 3. If the condition is y (Con=y), B sends mes5 to A by G9. Otherwise B sends mes4 to A by G8. After that, the state of B becomes the ending state 4. When the service A receives the message mes4 by G10 or mes5 by G11, its state also becomes the ending state 4. At the moment, the interaction of three services is over. According to the Fig 1, the interactive behaviors between the services are able to transform into the state transition and message transmission.

Definition 6 (coupled message) A coupled message is 2-tuple $cm = (MB_s, MB_a)$, where:

(1) *MB* is a set of all message bodies in S^n, $MB_s = \{mb_{s1}, mb_{s2}, \ldots, mb_{sn}\}$, $MB_s \subseteq MB$ and $MB_s \neq \phi$, $MB_a = \{mb_{a1}, mb_{a2}, \ldots, mb_{am}\}$ and $MB_a \subseteq MB$;

(2) if $mb_{si} = (gate_{si}, type_{si}, mes_{si}, num_{si}) \wedge 1 \le i \le n \wedge mb_{si} \in MB_s$, then $type_{si} = send$;

(3) if $mb_{aj} = (gate_{aj}, type_{aj}, mes_{aj}, num_{aj}) \wedge 1 \le j \le m \wedge mb_{aj} \in MB_a$, then $type_{aj} = accept$;

(4) if $mb_{aj} \in MB_a$, then $\forall mb_{si} \in MB_s \Rightarrow mes_{aj} = mes_{si} \wedge gate_{aj} = gate_{si}$.

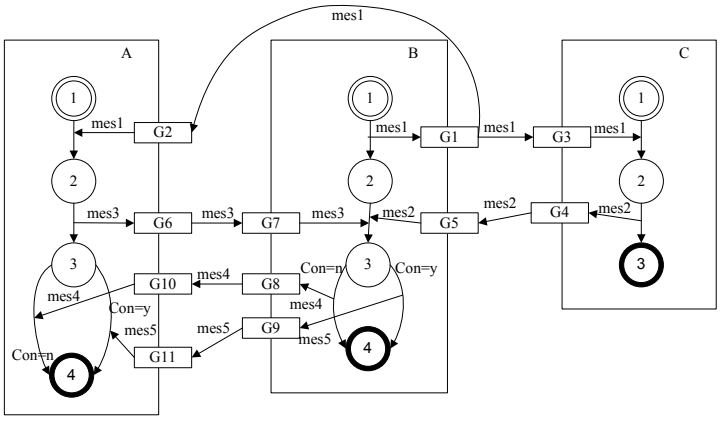

Figure 1. Example of the interactive behaviors between the services

Definition 7 (effective coupled message) Let a coupled message in S^n $cm = (MB_s, MB_a)$, if the number of messages in MB_s equals with the number in MB_a, then *cm* is an effective coupled message.

Definition 8 (message sequence) Let *TSet* is the set of the entire transition sequences in S^n and *F* is a set of state transition cells in *Tset*. $\forall st = (t_f, t_a, C, MB) \in F$ and t_f is the beginning state of a service. Find an effective coupled message from *F* and put the set of messages Mes_1 in it into the sequence *MS*; then $F = \phi$, delete *st* from *TSet*, and put the next state transition cell for every service in S^n into *F*; in the same way, find an effective coupled message from *F* and put the set of messages Mes_2 in it into the sequence *MS* in order. Repeat the process until there doesn't exist the effective coupled message in *F*. Finally, $MS =< Mes_1, Mes_2, \ldots >$ is a message sequence.

Definition 9 (effective message sequence) Let *TSet* is the set of all entire transition sequences in S^n and $MS =< Mes_1, Mes_2, \ldots >$ is a message sequence from *TSet*; if $TSet = \phi$, then *MS* is an effective message sequence.

Definition 10 (compatibility) If there exist the effective message sequence in S^n, then *n* services in S^n are compatible.

Automate Verification of Compatibility

Pi-calculus Description of Web services.

Take Fig. 1 for example, three processes of the services A, B and C are described as follows:

(1) $P_1 = G2(mes1).G6(mes3).(G10(mes4) + G11(mes5))$

(2) $P_2 = !\overline{G1}(mes1).(G5(mes2) \mid G7(mes3)).(\overline{G8}(mes4) + \overline{G9}(mes5))$

(3) $P_3 = G1(mes1).\overline{G4}(mes2)$

The composite process *P* of three services is described as follows:

$$P = P_1 \mid P_2 \mid P_3 = G2(mes1).G6(mes3).(G10(mes4) + G11(mes5))$$
$$\mid !\overline{G1}(mes1).(G5(mes2) \mid G7(mes3)).(\overline{G8}(mes4) + \overline{G9}(mes5)) \mid G1(mes1).\overline{G4}(mes2)$$

Compatibility Verification.

Theorem 1 (compatibility decision theorem) Let $s_1, s_2, \ldots, s_n$ are n services in S^n and $P_1, P_2, \ldots, P_n$ are the process expressions of these services; iff $P_1 \mid P_2 \mid \ldots \mid P_n \Rightarrow 0$, $s_1, s_2, \ldots, s_n$ are compatible.

Proof: (1) According to Definition 10, if $s_1, s_2, \ldots, s_n$ are compatible, then there exist at least an effective message sequence to be able to change the states of n services from beginning state to

ending state. At the same time, $P_1, P_2, \ldots, P_n$ communicate with each other by the given message sequence and these processes finally evolve into n null processes. $P_1 \mid P_2 \mid \ldots \mid P_n \Rightarrow 0$ holds. (2) Assume $P_1 \mid P_2 \mid \ldots \mid P_n \Rightarrow 0$, it explains the process expression $P_1 \mid P_2 \mid \ldots \mid P_n$ could evolve into the null process by internal interaction. During the interaction, the process generates a message sequence. And the sates of $s_1, s_2, \ldots, s_n$ change from beginning state to ending state by the message sequence. Therefore, the services $s_1, s_2, \ldots, s_n$ are compatible.

According to the Theorem 1, the evolving procedure of *P* by the message sequence $M_1 = < \{mes1\}, \{mes2, mes3\}, \{mes4\} >$ is as follows:

$$
\begin{aligned}
P &= P_1 \mid P_2 \mid P_3 \\
&= G2(mes1).G6(mes3).(G10(mes4) + G11(mes5)) \\
&\quad \mid !\overline{G1}(mes1).(G5(mes2) \mid G7(mes3)).(\overline{G8}(mes4) + \overline{G9}(mes5)) \mid G1(mes1).\overline{G4}(mes2) \\
&\xrightarrow{\{mes1\}} \overline{G6}(mes3).(G10(mes4) + G11(mes5)) \\
&\qquad \mid (G5(mes2) \mid G7(mes3)).(\overline{G8}(mes4) + \overline{G9}(mes5)) \mid \overline{G4}(mes2) \\
&\xrightarrow{\{mes2, mes3\}} (G10(mes4) + G11(mes5)) \mid (\overline{G8}(mes4) + \overline{G9}(mes5)) \mid 0 \\
&\xrightarrow{\{mes4\}} 0 \mid 0 \mid 0
\end{aligned}
$$

The above evolving procedure makes the process *P* become a null process. It proofs the services A, B and C are compatible.

Automate Verification Method.

To the actual web service composition, the interactive process is more complicated than our example in this paper. It is hard to manually convert composite service process into pi-calculus process expression. Therefore, we present service forest generation algorithm and service tree generation algorithm to automatically generate the pi-calculus process expression.

Algorithm 1 Service Forest Generation Algorithm
Input: the set of service behavior $SB = \{sb_i \mid 1 \le i \le n\}$
Output: service forest *SF*, process expression *pe*
Begin
01: for *i*=1:*m* /* *m* is the number of services */
02: $t = sb_i.t_b$; $expr = ''$; $parent = new()$; $expr = STree(t, parent, sb_i.ST, expr)$; $SF(i) = parent$;
03: $expr = substring(expr, 2, expr.length)$; $pe = pe + '|' + expr$;
04: end for
05: $pe = substring(pe, 2, pe.length)$;
06: return (*SF*, *pe*);
End

The Algorithm 1 coverts the service behavior sequence of each service into the service tree and generate a process expression. In this expression, we use "|" to separate these services. In this algorithm, we invoke the Algorithm 2. The Algorithm 2 is used to generate a service tree for each service. In Algorithm 2, |*ST*| denotes the number of state transition cells in *ST*; *pst*() returns the process expression of the given state transition cell.

Algorithm 2 service tree generation *STree*(*bt*, *parent*, *ST*, *expr*)
Input: the set of state transition cell *ST*, beginning state *bt*, root node *parent*
Output: root node *parent*, process expression *expr*
Begin
01: $k = 0$; $expr = expr + '.('$; $n = |ST|$;
02: for $i = 1:n$
03: $st = ST(i)$;
04: if $bt == st.t_f \sqrt{a^2 + b^2}$
05: $parent \to child(k) = st$; $parent \to child(k).data = pst(st)$; $ST = ST - st$;

06: if $k == 0$ then $expr = expr + pst(st)$;
07: if $k > 0 \& st.C == \phi$ then $expr = expr + '|' + pst(st)$;
08: if $k > 0 \& st.C != \phi$ then $expr = expr + '+' + pst(st)$;
09: $expr = STree(st.t_a, parent \rightarrow child(k), ST, expr)$; k++;
10: end if
11: end for
12: $expr = expr + ')'$;
13: return *expr*;
End

Summary

In this paper, we propose the related definitions by using the pi-calculus to model the web service composition. And we present a compatibility decision theorem and proof it is correct. Moreover, we apply an example to verify the behavior compatibility of service composition. Finally, we use our process expression generation algorithms and pi-calculus tool to carry out our method.

Acknowledgement

This research was financially supported by the National Science Foundation No. 61503036 and the Scientific Research Fund of Liaoning Provincial Education Department, China No. L2015007.

References

[1] B. Yun, Formal modeling of trust web service composition using pi-calculus, TELKOMNIKA: Indonesian Journal of Electrical Engineering. 11 (2013) 4385-4392.

[2] R. Mateescu and G. Salaün, PIC2LNT: model transformation for model checking an applied pi-calculus, Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013), Rome, Italy, 2013.

[3] D. Sangiorgi and D. Walker, The pi-calculus: a theory of mobile processes: Cambridge University Press, 2003.

[4] S. Deng, Research on automatic service composition and formal verification (Ph.D), School of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, 2007.

[5] Y. Du, W. Tan, and M. Zhou, Timed compatibility analysis of web service composition: a modular approach based on petri nets, IEEE Transactions on Automation Science and Engineering. 11 (2014) 594-606.

[6] X. Fu, T. Bultan, and J. Su, Synchronizability of conversations among web services, IEEE Transactions on Software Engineering. 31 (2005) 1042-1055.

[7] J. Liu, J. Wang, K. He, X. Li, and F. Liu, Using pi-calculus to model web service interaction, Journal of Computational Information Systems. 9 (2013) 1759-1767.

[8] P. Marwaha, H. Banati, and P. Bedi, Formalizing BPEL-TC through Π-calculus, International Journal of Web & Semantic Technology. 4 (2013) 11-21.