Behavior adaptation based on projection in Web services composition

ZHANG Chi
School of Software and Communication Engineering, Jiangxi University of Finance and Economics
Nanchang, China, 330013
E-mail: zhangchi_1972@163.com

Abstract—Decentralized behavioral adaptation of Web service composition is proposed. Firstly, Web services behavioral was simulated by Labeled Transition System (LTS) and was projected to different connectors thus Web service behavior is orthogonalized, and then system compatibility was judged by checking if there was deadlock in synchronous product of different pairs of behavioral projections over connectors. Secondly, for the mismatch types adaptation countermeasures were analyzed then the mapping relationship of actions was expressed as synchronous vectors and the sequential relationship of vectors was defined as adaptation contract. Lastly, adaptation algorithm was designed to generate adaptation protocol automatically ensuring the mismatch Web services can correct interaction under the mediation of adaptor and the validity of the proposed method was verified by an application system.

Keywords—Web Service; Projection; Compatibility; Adaptation contract; Adaptor protocol

I. INTRODUCTION

Service oriented computing (SoC) provides a new computing paradigm [1] for solving distributed application integration by compositing the reusable Web services. But because the function provided by single web service is relatively simple, for the realization of complex business logic, we must to carry on the assemblage of the web services. The key problem of service composition lies in how to ensure the consistency and validity of the interactive services, services participating in service composition required to not only meet the input/output syntax layer requirements of single method but also satisfy the sequential relationship of operations invocation, namely behavioral protocol compatible. however, usually Web services are provided by different software vendors so that the difference of operations name, operations order, operations granularity in Web service interfaces are almost inevitable due to different use perspectives. Incompatibility among Web services caused by these differences during the assembly process is almost inevitable too. Whether the incompatible assembly can be adaptable and if adaptable how to generate the adaptor protocol are important problems that must be solved in Web services assembly practice.

In view of the above problems, this paper proposes decentralized method to coordinate the differences of web services interactive protocol and gives the algorithm to generate adaptor protocol automatically, thereby reduce complexity and improve assembly flexibility.

The remainder of the paper is organized as follows: after this introduction Section 2 introduces the related works. Section 3 formally introduces Web services model, services behavior projection and the case study we use throughout this paper. Section 4 analyzes different mismatch situations and corresponding adaptation strategies; defines the synchronous vectors according to the messages receiving and sending and defines the sequential relationship of synchronous vectors as adaptation contract then designs the algorithm to work out the mismatch interaction behavior between two Web service projections. Section 5 ends the paper with some concluding remarks.

II. RELATED WORKS

The adaptation methods are classified as restrictive methods [2,3] and generative methods[4,13]. The former one encourages deleting the path that causes incomatibility from components or services behavior protocol when composing. Automatically generating adaptors without the intervention of the human is one of the significant advantages of restrictive methods. However, deleting path will result in the loss of some functions of components or services. Generative methods mediate the incompatible behaviors through saving, reordering events without modifying services’ behaviors. But they usually require the intervention of manual work.

It is also classified as method based on finite state automata [4-5], method based on process algebra [6,7], method based on Petri net [8,9] and method based on LTS[10,11] according to the abstract behavior models.

The literature [4] describes of component behavior by finite-state automata, the paper puts forward adaptable method for incompatible components, and discusses about how to generate adaptor depending on the analysis of all of the execution trace of finite-state automata on the basis of interfaces mapping. The literature [5] also uses finite-state automata to model Web service behavior and generates adaptor model for duality interaction Web service according to interaction path. Because the action mapping relationship is not defined, the naming mismatch problem cannot be solved and the adaptor specification is generated with uncertainty. Method based on the finite state automata is relatively simple, but the limitation lies in its inadequate expression ability.

The literature [6,7] describe the component interface behavior protocol by π calculus, and then it gives out the method of generating adaptor. But the method merely works for solving duality interaction problems. Moreover, the complexities restrict its application in adaptation field.

The literature [8] puts forward modeling the semantic and behavior using Petri Net based on normalized structure of semantic service flow nets then converts Semantic Service
Flow Nets to $\pi$ calculus and provide a solution for generating behavior adaptor for Web services composition. The literatures [9] use colored Petri net to model Web service behavior described by WS-BPEL and construct the reachable graph to analyze adaptable thereafter generate Web service adaptor based on interface mapping rules. However, adaptor generated by this method does not have control logic nor to resolve adaptation under complicated scenarios. The method based on Petri net depends too much on state reachable graph in generating and verifying adaptor. When confronting with large-scaled adaptation problems, it will easily cause the state space explosion problem.

The literature [10] introduces using LTS to describe of components behavior and establish adaptation contract by means of analyzing the corresponding relation between synchronous messages. It solves two mismatch situations including naming mismatch and unexpected reception. The other mismatch situation of undefined expected receiving messages is still not being solved yet. As for the reorder mismatch they model components behavior using Petri Net. The deficiencies are: (1) it cannot be used to solve complicated adaptation problems; (2) the complexity of calculation is very high in the case that it requires multiple switches between LTS model and Petri Net model in confront of several mismatch situations. On the basis of [10] literature [11] simulates service protocol by STS(Symbolic Transition System). The deficiencies of it mainly lies in the fact that uses a centralized approach to check the whole system and irrelevant operation interwoven together thus increases the complexity of adaptation.

Compared with previous works the contribution of this paper mainly lies in: Firstly, Web service behavior is projected to different connectors thereby the behavior of Web service is orthogonalized and the overall system collaboration process is transformed into the P2P distributed collaboration sub processes, therefore, overall compatibility judgment and adaptation is converted into compatibility judgment and adaptation over different connectors, greatly reducing the computational complexity. Secondly, the adaptor protocol can be generated automatically if adaptable thus to ensure the correct interaction between two service projections over one connector with mismatch behavior. Finally, in dynamic network environment, if one web service is changed only local compatibility check and adaptation are needed, reducing the cost of system update and improving the flexibility of composition.

III. WEB SERVICES MODEL AND BEHAVIOR PROJECTION

A. Web services model

Definition 1(Web services model, WSM) Web services $WS$ can be defined as a quadruple $WSM=(M^S_{WS}, M^R_{WS}, L_{WS}, P_{WS})$, where:

$M^S_{WS}$ is the set of messages can be sent by service $WS$, corresponding to the operation invocations to others service components, namely $M^S_{WS} = \{m_1, m_2, \ldots, m_p\}$;

$M^R_{WS}$ is the set of messages can be received by service $WS$, corresponding to operations can be invoked by other service components, namely $M^R_{WS} = \{n_1, n_2, \ldots, n_q\}$;

$L_{WS}$ is the set of connector linked to service $WS$; it indicates the connection relationship of Web services in architecture, for every connector $l_i \in L_{WS}$, $l_i = \langle WS_i, InWS_i \rangle$, $WS_i$ indicates the services $WS_i$ connected to $WS$ through connector $l_i$ and their instances $InWS_i$;

$P_{WS}$ is the behavioral protocol of $WS$; expresses the sequential relations of operation invocations and is formally defined as a quintuple $P_{WS} = \{E_{WS}, S_{WS}, I_{WS}, F_{WS}, T_{WS}\}$, where:

- $E_{WS}$ is a set of events;
- $S_{WS}$ is a set of states;
- $I_{WS}$ is initial state, $I_{WS} \subseteq S_{WS}$;
- $F_{WS}$ is a set of terminal states, $F_{WS} \subseteq S_{WS}$;
- $T_{WS} \subseteq S_{WS} \times E_{WS} \times S_{WS}$ is a set of finite state transition in services interaction;

The events of Web services are divided into reception, emission and internal events and be expressed by $e$, $l$ and $r$ respectively; correspondingly, the reception event set, emission event set and initial event set are indicated by $E^r_{WS}$, $E^e_{WS}$, $\{r\}$, $E_{WS} = E^r_{WS} \cup E^e_{WS} \cup \{r\}$, where, $E^r_{WS}$ corresponds to operations, invoked by other services, defined in interface of $WS$, $E^e_{WS}$ corresponds to operations request to other services defined in $WS$ interface, $\{r\}$ corresponds to the function of independent evolution of Web service and $E^r_{WS} \cap E^e_{WS} \cap \{r\}$ are disjointed.

![Figure 1](image)

Figure 1. an e-business system based on the third payment platform

Fig. 1 shows the services and their connection relation of an e-business system based on the third payment platform, where service $SCustomer$ make the deal with service $SVendor$ through connector $l_1$, $SCustomer$ connects to the third payment platform $SPP$ through connector $l_2$ and $SPP$ connects to $SVendor$ through connector $l_3$. $SPP$ is in charge of collecting of trade charges and paying of the purchase. $SCustomer$ pays to $SPP$ after his or her order is confirmed and $SPP$ notice $SVendor$ to ship after receiving the payment. When receives the ordered goods $SCustomer$ notifies the $SPP$ to transfer money to $SVendor$ and end of the transaction.

Fig. 2 shows the different Web service behavior protocol expressed by LTS.

![Figure 2](image)

2(a) Behavior protocol of service $SCustomer$

2(b) Behavior protocol of service $SPP$
2(c) Behavior protocol of service SVendor

Figure 2. Behavior protocol of Web services

B. Web service behavior protocol projections

Web service behavior protocol LTS forms directed graph in which states are nodes and transitions are edges. In the application system composed of services interaction among multiple services can be viewed essentially as multiple interactions between two services over different connectors. Therefore, Multiple services are compatible if and only if any two interactive services are compatible, multiple services are adaptable when and only when any interaction between two services are adaptable. After defining the behavior of every service we can project the whole behavior of service to different connectors according to the connection relationship of system so as to make the behavior of service orthogonalized. On the basis of service projections the overall compatibility judgment and adaptation of the system can be converted into compatibility judgment and adaptation over different connectors thereby greatly reducing the computational complexity.

Definition 2(Web service behavioral projection, WSP) the actions set of service WS over connector $l_i$ is $E_i$, the behavior projection of service WS over connector $l_i$ still is a LTS, denoted by $P_{WS|l_i} = \{E_{WS|l_i}, S_{WS|l_i}, I_{WS|l_i}, T_{WS|l_i}\}$, where:

- $E_{WS|l_i} = \{s\} \cup E_i$;
- $T_{WS|l_i} = \{(u, e, u')\} \cup T_{WS|l_i} \cup \{(u, e, u')\} \in T_{WS|l_i}$.

Algorithm 1: WebServiceBehavioralProjection

Input: $P_{WS}$
Output: Web service projections over different connectors $P_{WS|l_i}$

1: For every connector $l_i$
2: Input $P_{WS}$
3: For every $t=(s, e, s') \in T_{WS}$
4: If $s \notin E_i$ Then
5: If $e=I_{WS}$ Then
6: {replace $s$ by $E_{WS}$ $T_{WS|l_i} = T_{WS}-t$;
7: If $s' \in F_{WS}$ Then
8: $F_{WS|l_i} = F_{WS} \cup s$;
9: else
10: $S_{WS|l_i} = S_{WS} \cup s'$;
11: End If
12: Else
13: $T_{WS|l_i} = T_{WS}-t$;
14: If $s' \in F_{WS}$ Then
15: Replace $s$ by $s$;
16: else
17: $S_{WS|l_i} = S_{WS} \cup s'$;
18: End If
19: End If //corresponding to sentence 5
20: End If //corresponding to sentence 4
21: End For //corresponding to sentence 3
22: For (every $t=(s, e, s') \in T_{WS|l_i} \wedge t=(s, e, s') \in T_{WS|l_i}$)
23: If $e=e'\tau$ Then
24: Replace $s$ by $s'$;
25: $T_{WS|l_i} = T_{WS|l_i}-t$;
26: $S_{WS|l_i} = S_{WS|l_i} \cup s'$;
27: End If
28: End For
29: Return $P_{WS|l_i} = \{E_{WS|l_i}, S_{WS|l_i}, I_{WS|l_i}, T_{WS|l_i}\}$;

30: End For

Firstly the algorithm inspect whether the action set $E_i$ of connector $l_i$, if not then remove the transition caused by this action from service LTS, In the removing process the initial state of service LTS be retained and if all transitions in one path be deleted then the terminal state of this path should be deleted too. Lastly, we can get service behavior projection over connector $l_i$ after merging continuous transitions caused by internal action $\tau$. The computational complexity of the algorithm is $O(|S|^n)$, where $S$ is the number of state in LTS and $n$ is number of connector connected to service WS. In the following of this paper compatibility judgment and adaptation are all studied between two service projections over one connector.

We can get service projections showed in Fig.3 according to the connection relation of Fig. 1 and algorithm 1.

3(a) Behavior projections of SCustomer over connector $l_1$ and $l_2$

3(b) Behavior projections of SPP over connector $l_1$ and $l_2$

3(c) Behavior projections of SVendor over connector $l_1$ and $l_2$

Figure 3 Behavior projections of Web services

For the e-business system based on the third payment platform, after getting the service projections we can check compatibility of component projections over different connectors according to algorithm of Literature [12]. In the connector $l_2$ service $SCustomer$ and $SPP$ can transit from their initial states to terminal states respectively under the effect of complementary events (,!Transfer(), ?Transfer()) and (,!AckDel(), ?AckDel()). It indicates that $SCustomer$ and $SPP$ are compatible over connector $l_2$. In a similar way $SVendor$ and $SPP$ are compatible over connector $l_1$. But in connector $l_1$ $SCustomer$ and $SVendor$ are incompatible because in their initial states two events $!User()$ and $?Query()$ are not complementary events. So adaptor in the connector $l_i$ must be developed to eliminate the behavior mismatch.

IV. ADAPTATION BASED ON SERVICE PROJECTION

A. Mismatch analysis

In the practice of composition, Web services are usually developed by different third vendors. Since the usage exception of the services is not exactly the same, the operation name, operation sequence and operation granularity in the services inevitable be defined differently.
The mismatch situations resulted by these differences are list below:

- Difference of operation name(1-1): the functions of operation are conformance but names are different;
- Superfluous message (1-0): sender send one message but no message corresponding to it in receiver;
- Missing message (0-1): receiver except to receive one message but no corresponding message in sender;
- One to multiple(1-n): one message in sender corresponds to multiple messages in receiver;
- Multiple to one(m-1): multiple messages in sender corresponding to one message in receiver;
- Disorder (n-n): sequence of sending messages is different from sequence of receiving messages.

As for the software system composed by web services, services cannot be assembled directly if mismatching. And meantime, for the system integrators, it is impossible to modify the codes of the service due to their black-box characters; the only way they can use is non-intrusive adaptation mechanism to eliminate the incompatibility among services. Therefore, introducing adaptors over different connectors to mediate the interaction between service projections with mismatch behavior and ensure the incompatible service projections interact correctly.

B. Adaptation contract

According to the aforementioned analysis, the adaptor is an independent coordinator, which is able to partly or wholly resolve the incompatibility problems among services. The functions of the adaptor include: (1) receive the message sent by sender and cache message if it cannot be forwarded immediately; (2) send the cached message to receiver; (3) filter the unanticipated reception messages; (4) merge several messages into one message if relation of messages is multiple to one or split one message to several messages if relation of messages is one to multiple.

In order to express the corresponding relation of synchronous messages between two service projections over one connector synchronous vector is defined in definition 3, each action in one synchronous vector is executed by one service and each vector represents the corresponding relationship between the messages that sent and received in services synchronous interaction.

Definition 3(synchronous vector) the synchronous vectors of two service projections, $P_{WS} I = \{ E_{WS I}, S_{WS I}, L_{WS}, F_{WS I}, T_{WS I} \}, i \in [1, 2]$, over one connector $I$ are $<WS I, e_1, WS I, e_2>$, where $WS I$ is service identifier, $e_1$, $e_2$ is action executed by service, $e_i \in (E \cup \{e\})$, if one service does not participate in a synchronization its action is expressed by $\epsilon$.

In an application system composed by Web services, if the required operation of one service is not provided by other system services then there exists function fault and we called the system no function fault as enclosed system. The unenclosed system itself is can’t run correctly and it is meaningless to discuss unenclosed system, therefore, this paper study the system adaptation under the premise of the enclosure system. There don’t exist synchronous vector such as $<e, \epsilon>$ in the enclosure system.

During the process of extracting synchronous vector, In order to avoid the same message appear repeat in synchronous vector set, if one message in sender corresponds to multiple messages in receiver only the synchronous relation of the sending message and the first receiving message is marked while in other vectors only receiving messages are denoted; if one message in receiver corresponds to multiple messages in sender then only the synchronous relation of the last sending message and the receiving message are marked and in other synchronous vectors only sending message are denoted.

For the e-business system based on the third payment platform the operations corresponding relations of service projections of $S_{Customer}$ and $S_{Vendor}$ over connector $I$ are:

The synchronous vectors of multiple operations corresponding to one operation are:

$V_{user} = \langle S_{Customer}!User\rangle$;
$V_{password} = \langle S_{Customer}!Password, S_{Vendor}!Login\rangle$

The synchronous vectors of one operation corresponding to one operation are:

$V_{query} = \langle S_{Customer}!Query, S_{Vendor}!Query\rangle$;
$V_{select} = \langle S_{Customer}!Select, S_{Vendor}!Select\rangle$;
$V_{delivery} = \langle S_{Customer}!Delivery, S_{Vendor}!Delivery\rangle$;
$V_{exit} = \langle S_{Customer}!Exit, S_{Vendor}!Exit\rangle$;
$V_{refuse} = \langle S_{Customer}!Refuse, S_{Vendor}!Refuse\rangle$

The synchronous vectors of one operation corresponding to multiple operations are:

$V_{order} = \langle S_{Customer}!Order, S_{Vendor}!Order\rangle$;
$V_{getAdd} = \langle S_{Vendor}!GetAdd\rangle$;
$V_{conf} = \langle S_{Customer}!Conf, S_{Vendor}!Conf\rangle$;
$V_{accept} = \langle S_{Customer}!Accept\rangle$

The synchronous vectors of one operation corresponding to none are:

$V_{input} = \langle S_{Vendor}!Input\rangle$;
$V_{obtain} = \langle S_{Customer}!Obtain\rangle$

Synchronous vector only express the corresponding relation of receive/send messages between service projections over one connector. In order to describe the message sequence relation between two projections during their interaction process, definition 4 still uses LTS to express the sequence of synchronous vectors.

Definition 4(Vector LTS) The LTS of synchronous vector set $V = (I_V, S_V, F_V, L_V)$, where labels are synchronous vector.

Service projection LTS shows the operation calling relation in one connector. Synchronous vector shows the corresponding relation between messages sent and received between two interaction service projections. Vector LTS indicates the sequence of synchronous vectors. As for the adaptation of interaction is concerned, three of the relations need to be met, namely adaptation contract.

Definition 5(Adaptor contract, AC) The adaptor contract of two service projections, $P_{WS} I = \{ E_{WS I}, S_{WS I}, L_{WS}, F_{WS I}, T_{WS I} \}, i \in [1, 2]$, over one connector $I$ is a tuple $(V, L)$, where $V$ is the set of synchronous vectors between two service projections, $L$ is vector LTS of $V$.

For the e-business system based on the third payment platform, after defining the synchronous vectors we design the adaptation contract shown in fig. 4.
C. Generation of adaptor protocol

From the point of view of adaptor, in any point of time, the sending and receiving relation must be one of the four situations listed below:

1. Two sides neither receiving nor sending message, indicates two sides in their terminal states.
2. One side receiving message and another side sending message can be subdivided into the following situations:
   - If the message sent by the sender is exactly what the receiver expects to receive, the adaptor will forward the expected message to the receiver directly. Both of two sides transit to their next states.
   - If the sending message isn’t the expected one by another side, the adaptor will receive and cache this message, the sender continues to transit, while the state of the receiver doesn’t update.
   - If the sending message corresponding to multiple messages of receiver and the message receiver desired to receive just one of them then adaptor saves the received message and sends the excepted message to the receiver. Both of sender and receiver transit to their next states. Else adaptor saves the received message and the sender transits to its next state but while the state of the receiver doesn’t update.
   - If there is no message in receiver corresponds to the message sending by sender then adaptor cache the message, but it will be filtered finally.

3. Receiving simultaneously, can be subdivided into the following situations:
   - Inspect whether the expected messages are all already cached in adaptor, if they are, get message from cache and send them, both sides continue to transit.
   - If the message just expected by one side is cached in adaptor, then this side transit further and the other side, which doesn’t have the matching message, doesn’t transit.
   - If messages desired by the two sides that are all not saved by adaptor then deadlock and both sides cannot transit further, adaptation fails.

4. Sending simultaneously: Adaptor caches the received messages and both transit further.

On the basis of forming adaptation contract, if adaptable, adaptor protocol can be generated according to the receiving/sending message relation of adaptor.

Definition 6 (Adaptor Protocol, AP) The adaptor protocol, $AP_l$, of two service projections over one connector $l$, $P_{WS_l} = \{E_{WS_l}, E_{WS_l}, E_{WS_l}, I_{WS_l}, T_{WS_l} \}, l \in \{1,2\}$, is a LTS $=\{E_1, I_1, I_2, I_3, I_4, T_1 \}$ where: $S_1 \subseteq S_{WS_l} \times \times S_{WS_l}$ is state set of adaptor, $E_4 \subseteq F_{WS_l} \downarrow l \cup F_{WS_2} \downarrow l$ is event set of adaptor, $I_4 = I_{WS_1} \times I_{WS_2}$ is the initial state of the adaptor, $F_4 \subseteq F_{WS_1} \downarrow l \times F_{WS_2} \downarrow l$ is set of terminal states, and $T_4 \subseteq S_1 \times A_1 \times S_1$ is the transition set of the adaptor.

According to above situations, algorithm can be designed and the adaptor protocol can be generated automatically. For express convenience, we define several symbols: $\text{em}(s) = \{m(s_1, l, s') \in T \land s \in S\}$ represents the set of messages that can be sent from state $s$, $\text{rec}(s) = \{(l \land s_2, s') \in T \land s \in S\}$ represents the set of messages which can receive from state $s$. $Q$ is the current set of messages saved in adaptor’s cache. $S_A$ is the state set of the adaptor and $T_A$ is the transition set of the adaptor. In initial state: $Q = \emptyset$, $S_A = \emptyset$, $T_A = \emptyset$.

Algorithm 3: Generate Adaptor Protocol

Input: $P_{WS_l} = \{E_{WS_l}, E_{WS_l}, E_{WS_l}, I_{WS_l}, T_{WS_l} \}, l \in \{1,2\}$ and adaptation contract $AC = (V, L)$.

Output: If adaptable then adaptable!(true) and output the adaptor protocol $AP_l = \{E_1, I_1, I_2, I_3, I_4, T_1\}$, else output adaptable!(false).

1. $\text{LTS}_{AC} = (V, L)$;
2. $S_A = S_{AC}$, $T_A = \emptyset$; $Q = \emptyset$; adaptable!(true);
3. For all $t = (v = (s_1, s_2), e_1 \in \text{rec}(s_1)) \in \text{LTS}_{AC}$ do
4. For $u = 1$ to 2
5. $v \neq u$
6. If $e_1 \in \text{em}(s)$ Then
7. If $(v = (s_1, s_2), e_1 \in \text{rec}(s_1))$ Then
8. $t = (s_1, e_1, s_2)$; $y = (s_1, e_1, s_2)$; $s = s$; $s' = s$; $s = s$; $s = s$;
9. $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
10. $S_1 = S_1 \cup s_1$; $s_1 = s$;
11. else if $(v = (s_1, s_2), e_1 \in \text{rec}(s_1))$ Then
12. If $(v = (s_1, s_2), e_1 \in \text{rec}(s_1))$ Then
13. $t = (s_1, e_1, s_2)$; $y = (s_1, e_1, s_2)$; $s = s$; $s' = s$; $s = s$;
14. $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
15. $S_1 = S_1 \cup s_1$; $s_1 = s$;
16. else
17. $t = (s_1, e_1, s_2)$; $s = s$; $s = s$; $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
18. $S_1 = S_1 \cup s_1$; $s_1 = s$;
19. end if //corresponding to sentence 12
20. end if //corresponding to sentence 11
21. end if //corresponding to sentence 7
22. else if $e_1 \in \text{em}(s)$ Then
23. $Q = Q \cup e_1$; $y = (s_1, e_1, s_2)$; $y = (s_1, e_1, s_2)$;
24. $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
25. $S_1 = S_1 \cup s_1$; $s_1 = s$;
26. end if //corresponding to sentence 22
27. end if //corresponding to sentence 6
28. if $e_1 \in \text{rec}(s_1)$ and $e_1 \in \text{rec}(s_1)$ Then
29. if $s \in Q$ and $s \in \emptyset$ Then
30. $t = (s_1, e_1, s_2)$; $y = (s_1, e_1, s_2)$;
31. $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
32. $S_1 = S_1 \cup s_1$; $s_1 = s$; $Q = Q \cup e_1$;
33. else if $e_1 \in \emptyset$ and $s \in \emptyset$ Then
34. $t = (s_1, e_1, s_2)$; $y = (s_1, e_1, s_2)$;
35. $Q = Q \cup e_1$; $S_1 = S_1 \cup s_1$;
36. $T_1 = T_1 \cup \{(s_1, e_1, s_2), (s_1, e_1, s_2)\}$;
Regardless of looping, Web service projections LTS and the adaptation contract form directed acyclic graphs, and algorithm use depth first traversals to inspect the transition in adaptation contract at the same time inspects transition in each Web service LTS. The algorithm complexity is $O(|S|^3)$, and the $S$ is the maximum state number of service projection LTSs and adaptation contract LTS and 3 is two of service projections plus one of adaptation contract.

To e-business system based on the third payment platform the adaptor protocol over connector $l_i$ generated automatically is showed in figure 5.

![Figure 5 adaptor protocol of SCustomer and SVendor over connector $l_i$.](image)

Definition 7 (correctness of adaptor protocol). Two component projections, $P_{WS}|\!\!l_1 = \{E_{WS}, S_{WS}, I_{WS}, F_{WS}, T_{WS}\}, \ j \in \{1, 2\}, \overline{\text{over connector } l_j}$, under the mediated of adaptor protocol API, every component projection can transit form its initial state to terminal state, namely no deadlock state in $P_{WS}|\!\!P_{API}|\!\!P_{WS}\ \forall l_i$.

To inspect the correctness of adaptor protocol, the synchronous product of $P_{WS}|\!\!l_1 = \{E_{WS}, S_{WS}, I_{WS}, F_{WS}, T_{WS}\}, \ j \in \{1, 2\}$ and LTS API=$\{E_A, S_A, I_A, F_A, T_A\}$ need to be calculated then judgments the compatibility by checking whether there is a deadlock state in it.

For e-business system based on the third payment platform we can use the generated adaptor to eliminate the mismatch behavior and ensure both of SCustomer and SVendor service projections transit from their initial states to terminal states.

V. CONCLUDING REMARKS

Usually Web services are provided by different software vendors and have the character of autonomy, although the function meets the system needs, behavior protocol incompatible lead to wrong actions or deadlock in composition is almost inevitable. Therefore, how to ensure the incompatible web service interaction correctly is important problem that must be solved in the service composition. This paper models web services behavior as LTS and projects service behavior to different connectors according to the connection relationship, thereafter, judges the compatibility over one connector by inspecting whether there is a deadlock in the synchronous product. When their incompatible, forming synchronous vector by analyzing their behavior relations and generate adaptation contract according to the sequence of synchronous vectors. In the following, with the guide of adaptation contract, combing the relationship of messages sending/receiving generates adaptor specification automatically so to ensure two service projections over one connector interact correctly. This paper only analyzes the control flow information, the control flow and data flow information will be considered unified in further research.

ACKNOWLEDGMENT

This work is supported by the project “research on component composition with self-adaptation mechanism in Internet environment” (No.61262012), funded by National Natural Science Foundation of China; project “Web service composition based on extended interface”(No.2008GZS0017), funded by science foundation of Jiangxi province, China, and Visiting scholar project funded by JXUFE.

REFERENCES


[12]Zhang C, Software Components Behavioral Adaptation Based on LTS[C]. The 7th International Conference on Computational Intelligence and Software Engineering (CiSE 2015),217-224.