Binary Activity Chain Modes for Compositional Web Service and Its Compliance Verification

Bo Chen  Chungui Li  Qixian Cai  
Department of Computer Engineering, Guangxi University of Technology, Liuzhou, 545006, China

Abstract—requirement-driven behavior verification for compositional Web service is one of hot research issues for Web computing. Modeling and analysis of the behavior requirements of compositional Web service plays an important role in behavioral verification. Traditional methods are expressing requirements as LTL like logic specifications which are based on activities or as MSC like graph forms which are based scenarios. In this paper, we propose the concept of behavior specification based on activity chain in which its atomic granularity is between activity and scenario. Four behavioral modes such as chain existence mode, chain absence mode, chain precondition mode and chain response mode are adopted to express usual requirement specifications. Encode them on Labeled Transition System LTS and then give them operation semantics. Check compositional Web services based on LTS corresponds with behavior modes or not. Give the sufficient, necessary condition and algorithm for checking.

Index Terms—Specification, Stateful Web Services, Composite Web Services, Model checking

I. INTRODUCTION

A fundamental goal of web services is to combine today’s simple Web services into more complex ones in order to achieve more sophisticated application purposes. The composite service adds two dimensions by comparison to the simple ones; they are stateful and they obey to an operational behavior. This raises many theoretical and practical issues which are part of ongoing research [1]. Recently, the works on web service verification are mainly focus on the following three issues 1) whether communication activities of compositional service accord with specification; their interactions are compatible? A given service can be replaced by another? [1], [2], [3], 2) whether the control and data flows of compositional service processes are correct? and whether the subservices comply with the constraint rules among them? [4], [5], [6], 3) whether a compositional service is compliant with specific requirements of the user (requirement-driven compliance verification)?

The ultimate motivation of web service composition is to offer satisfactory behavioral functions for users. It is important to study concise methods to express the requirements for behavior of compositional service and check whether the behavior is compliant with the requirement of user after checking that service process is correct and communication is available. Some works on this issue have been published. Pistore [12] expressed the goals and requirements of different roles in compositional service with formal Tropos language. Also the internal constraints and external dependencies to implant these goals and requirements have been formally presented. But the requirements are only for some component of specific roles, nor a whole behavior requirements of an user for a compositional service. Furthermore, the expressions of such requirements are in some LTL like forms not in concise manners. Rouached [13] expressed the time, casual and results of events occur in compositional web service with event calculus and attributes of behavior are expressed with first order logic. But a requirement expressed in such manner is in essence relation among some single events not relation among certain event sequences and its expression manner is more abstract. More existed works[7-13] generally expressed behavioral requirements in two methods such that one is in LTL,CTL like temporal logic specifications and another is in MSC, UML like graphic specification. The former’s basic element is activity. The latter’s basic element is scenario. The specifications describe some relation among activities or scenarios. In case of compositional web service, requirements for service behavior are often demands for composite behavior that are temporal relations between activity chains which belong to different subservice component. Since the direct object described by LTL like logic and MSC like graphic language is activity and scenario respectively, it is not suitable for LTL and MSC to direct describe such temporal relations based on activity chain which its granularity is between activity and scenario. It is nature to consider to transform the temporal relation based on activity chain to the temporal relation based on activity and then to express indirectly such behavioral requirement with LTL like logic. However, such a behavioral requirement contains two hierarchy temporal relations that first is the temporal relation between two activity chains and second is the chain order of activities in a specific activity chain. So obtained LTL formula after transformation will be very complicated. Not only is this formula difficult to read and understand, it is even more difficult to write correctly without some expertise in the idioms of specification language [15]. It is still a problem
to be explored how to suitably express the behavioral requirement based on activity chain in certain concise manner.

In current existed work on requirement-driven compliance verification, the principal approach is to translate a service behavior (BPEL process) into a mathematically well-founded model, considering only the semantic of elements that are relevant for the property to be verified. Then, model checking methods can be applied to the formal representation of the composite service behavior [1]. The behavior requirements to be checked are temporal relations based on activity or scenario. Fu [7] modeled the asynchronous communication of partners in compositional web service as a guarded mealy automaton. The global message chain observed by virtual observer is as the session model of compositional web service. In the case of finite input message queue, checked whether the session behaviors comply with the communication specification which is a LTL formula based on message. In his specification, the basic element is activity which sends or receives a message. Betin-Can [8] designed an interface protocol as communication contract expressed with guarded automata and checked whether a service complies with the contract by Java PathFinder. But the contract is only the specification of sub-service not that of global compositional service. Mongiello [9] abstracted a BPEL process as an execution chain and verified the chain against a specification expressed in CTL. The basic element of a specification was still activity. Foster [10] expressed behavior requirements with MSC and the obligation of service with fluentLTL and verified compliance of a compositional service in LTS. The basic element of a specification is scenario or activity with effect. Aalst [11] intercepted the flow of SOAP message of a compositional web service and translated them into Petri net as behavior model. The corresponding BPEL process was also translated into Petri net as a specification. The service implementation compliance verification was carried out in accordance with the specification which was a Petri net. Note that the basic elements of all specifications in above works are activity, scenario or even a state transition system. Their granularities are too small or too large to suitable to express such behavior requirements which granularity is middle level that is activity chain. How to express such behavior requirement in certain concise specification manner and how to verify that a compositional web service is compliant with such a specification or not? It is still an issue to be explored.

In this paper, we focus on the expression of binary behavior requirement based on activity chain and attempt to formally define such behavior specifications. Then we give the methods on checking the compliance of compositional web service against such specifications.

II. THE BASIC CONCEPT OF ACTIVITY CHAIN IN SERVICE AND EXAMPLE

A. The Definition of Activity Chain

Definition2.1. the activities of a compositional web service is inductively defined as below

\[ WS = \{ ws_1, ws_2, ..., ws_k, ws_\text{orch} \} \]

is a set of web service instance names, Where \( ws_\text{orch} \) is an orchestration engine.

\[ O_{ws} = \{ o_{ws} | o_{ws} = \begin{cases} \text{op} & [?m] \\ \text{op} & ![m] \end{cases} \} \]

is the set of operations in the port type of service ws, where op is a operation name, m is a message name, ![m] means receiving a message m, ![m] means sending a message m.

\[ A_{ws} = \{ a_{ws} | a_{ws} = \begin{cases} \text{receive}[o]_{ws} \text{orch} \text{ or } \text{reply}[o]_{ws} \text{orch} \end{cases} \} \]

is a set of basic activities of service ws. Activities are classified in two categories that one is receive a request for operation o of ws from service wsorch and another is answer to the request for operation o.

\[ A_{\text{orch}} = \{ a_{\text{orch}} | a_{\text{orch}} = \text{receive}[o]_{ws} \text{ or } \text{reply}[o]_{ws} \} \]

is the set of basic activities of service wsorch. invoke \( o_{ws} \) means wsorch invokes an operation ows of service ws.

\[ O = \bigcup \{ o_{ws} | ws \in WS \} \]

is the set of operations of compositional web service WS.

\[ Act = \bigcup \{ a_{ws} | ws \in A_{ws}, ws \in WS \} \]

is the set of activities of compositional web service WS.

In the definition above, every operation of a service must abide by the specification of WSDL. It belongs to one of four categories that is notification, solicit, request- response and solicit- response expressed in definition as \( o![m], o[?m], o[!m], o[?m] \) respectively. Since the so-called two-way operations of two latter categories are easily transformed into one-way operations of two former categories, we adopt only one-way operations such as \( o[!m] \) and \( o[?m] \) in this paper and let \( o \), a represents symbols of operation and activity respectively in case of unambiguity.

Definition2.2. an activity chain of a compositional web service with length n, \( \sigma = < a_1, ..., a_n > \), \( a_i \in Act, 1 \leq i \leq n \), is a tuple of finite activities occurred one after another in an execution of service.

Specially, if one infinite activity chain is composed of all activities occurred sequently in one execution of service, it is called a trace of service. Denote it \( \sigma \).

B. Binary Activity Chain Modes Definitions

Definition2.3. Let \( C = < a_1, ..., a_n > \) an activity chain of service WS with length n, \( \sigma = < \sigma_1, ..., \sigma_n > \), \( \sigma_i \in Act, 1 \leq i \), is a trace of WS. If there is a finite subchain of \( \sigma \) with length n, \( \sigma' = < \sigma_{i_1}, ..., \sigma_{i_n} > \), that \( \sigma_{i_j} = a_j, 1 \leq j \leq n \), then call C occurs in \( \sigma \).

Definition2.4. Let C an activity chain of service WS with length n and a is an activity of WS. If for any trace \( \sigma \) that C occurs in \( \sigma \), a must occur in \( \sigma \) precedence of C, then C and a satisfy chain precondition mode and is written a \( _{C_PR} A \).

Definition2.5. Let C an activity chain of service WS with length n and a is an activity of WS. If a occurs in \( \sigma \) trace, it will lead to C occur in \( \sigma \) after a occurs in \( \sigma \). Then C and a satisfy chain response mode and is written a \( _{C_RE} C \).
C. An Example

Flight and Hotel are two existing web services, which provide separately flight and hotel booking service for the client. Travel agency FH is the compositional service orchestrator, which provides integrated service for client. FH is responsible for invocation Flight operation and Hotel operation, and interact with client at the same time to provide integrity service process. Figure 2.1 describes an interactions scenario of compositional service. FH accepts the request from the client, and invokes Flight sub service and Hotel sub service concurrently, returns sorted flights list, lodging suggestions, then provides appointed flight and lodging plan according to client’s request, and waiting for client’s confirmation. At last, FH provides the client with services (the ticket and lodging order). We omit some communications between FH and Hotel for the sake of simplicity. Interactions of services are described by MSC like graph language. The difference here is that we use activity instead of message to label the transition of MSC. \( \rightarrow \) denotes the activity of service, the arrow denotes control flow direction when executing activities in interaction process.

When L executes an activity a, a, \((s, a, s')\) \(\rightarrow\), then it may become \(L'\), \(L'=\langle S, A, \rightarrow, s'\rangle\). Denote it \(L \xrightarrow{a} L\)', if \(s \xrightarrow{a} s'\), here, \(s \xrightarrow{a} s'\) is the same mean that of \((s, a, s')\) \(\rightarrow\).

**Definition 3.2.** Let LTS \(L_1, L_2\) are two LTS. The rules of parallel operation of two LTS are listed below

\[
L_1 || L_2 = \langle S, A, \rightarrow, s_0 \rangle
\]

\[
\rightarrow = \bigcup_{a \in A \cup \{\tau\}} \rightarrow_a
\]

\(s_0 = \langle s_{01}, s_{02} \rangle\). \(s_{01}\) is a LTS, \(s_{02}\) is another LTS. \(s_{01} \rightarrow_1 s_{11}, s_{12} \rightarrow_2 s_{21}, \ldots\) is an infinite or finite alternating chain of states and activity labels, where \(s_i \in S, i \geq 0, a_j \in A, j \geq 1, s_i \rightarrow a_j s_{ij}, i \geq 1, \rho\) is called an execution of L. \(\sigma = \langle a_1, a_2, \ldots\rangle\) is called the trace corresponding to \(\rho\).

**Definition 3.3.** Let \(L = \langle S, A, \rightarrow, s_0 \rangle, A = \alpha L\), is a LTS. \(\rho = s_0 a_1 s_{12} a_2 s_{22} \ldots\) is an infinite or finite alternating chain of states and activity labels, where \(s_i \in S, i \geq 0, a_j \in A, j \geq 1, s_i \rightarrow a_j s_{ij}, i \geq 1, \rho\) is called an execution of L. \(\sigma = \langle a_1, a_2, \ldots\rangle\) is called the trace corresponding to \(\rho\).

### TABLE 2.1. THE Binary Behavior Requirements Based on Activity Chains in Example

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Behavior mode</th>
<th>Activity chains</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1(3)</td>
<td>a-C-PR C1</td>
<td>a=receive[hotel_offers]C&lt;invoke[flights_filter]F&gt;</td>
</tr>
<tr>
<td>R1(4)</td>
<td>a-C-RE C1</td>
<td>a=invoke[nack]C&lt;onMessage[nack]C&gt;</td>
</tr>
</tbody>
</table>

R1.(1) The service must provide optional flights sorted by price after having received the client request.

R2.(3) Hotel candidate offers provision must be precondition of providing flights filter.

R2.(4) If client reply nack for offers, then service will not provide final service.

R1.1) is a liveness property that can be expressed as:

**Definition 3.1.** An LTS is a tuple \(L=(S,A,\rightarrow, s_0)\), where \(S\) is the set of finite states. \(A=\alpha L\subseteq Act\) is the set of finite activities. \(\rightarrow \subseteq S \times A \times S\) is a transition relation. \(A\tau= A\{\tau\}\), \(s\) is initial state and \(\tau\) is internal activity that is invisible to extern.
IV COMPLIANCE VERIFICATION WITH ACTIVITY CHAIN MODES

Compliance verification is to check whether every execution of specific compositional web service complies with the activity chain mode. In general, a specific LTS presents a ceaseless reactive system which its executable path is infinite. Thus if it goes into a terminate state, that is Post(s)≠Φ, s must be its deadlock state. However, a LTS represents a web service in this paper and its normal execution may be infinite or finite normal conclusion. If a state s is a terminate state, it may be possible a deadlock state or possible normal finite conclusion state. Compliance verification must go on in normal execution of web service. So it is necessary to exclude the deadlock situation from a web service first.

A. The Extension of LTS and Exclusion of Deadlock States

Assume the LTS L_W||L_2||...||L_k is a compositional web service is terminated in state s=(s_1,...,s_k), s_i, 1≤i≤k, is corresponding subservice state of the service, Post(s)=Φ.

Definition 4.1. If a state s is a terminate state of a compositional web service, then
(1) s is a normal conclusion state(also called accepting state) iff ∀i, 1≤i≤k. Post(s_i)=Φ and s_i is L_i normal conclusion state.
(2) s is a deadlock state iff ∃i∈1≤i≤k. Post(s_i)≠Φ.

To pick out deadlock state, we extend the LTS model L below.

1. If s_i is an accepting state of L_i, add a self-loop transition with activity label @L_i, called accepting activity label.
2. If LTS L=L_1||...||L_k and there are more than one subcomponents have accepting states, then regard their accepting activity labels as same, that is @L_i=α, i≠j.
3. The executions of accepting activities comply with parallel rule (3) in section 3.

When service L_W goes into accepting state s=(s_1,...,s_k), all of its subservices go into their accepting states and they will execute the same accepting activity labeled with @L_W. Thus a normal finite execution with accepting state will become infinite execution with postfix of @L_W s @L_W s...@L_W...,

Finally, check the compliance verification and then the characteristics about activity chain modes. Initially, check the compliance of web service for activity chain modes with reachable analysis of LTS.

Definition 4.2. Let L_M=<S_M,A_M,→M,S_M0> is a LTS of activity chain in figure 3.1. A_M=A_M1∪A_M2∪{[@L]} or A_M=A_M1∪A_M2, where A_M1= Act is normal activity labels, A_M2={a|a∈A_M1}, @L is an accepting activity label defined in 4.1. If a∈A_M, b∈Act, then b is called matching with a, iff a∈A_M1 ∧ b=a or a∈A_M2 ∧ b∈Act, denote it b=a. If b is an accepting activity label @L’, then b is called matching with a, iff a∈A_M1 or a∈A_M2, denote it b=a.

If b∈Act∪[@L] and ∃a∈A_M satisfy that b=a, then denote b∈A_M.

Definition 4.3. Let L_W=<S_W,A_W,→W,S_W0>,L_M=<S_M,A_M,→M,S_M0> is LTS of compositional web service and activity chain mode respectively. ω is an infinite or finite activity chain of L_W. The projection ↓M: A_W*→A_M* is defined inductively below.

(1) (.)↓M=().
(2) If \( a \in A_{\text{M}} \), then \((a) \rightarrow M(a) \sim (\sigma \downarrow M)\).

(3) If \( a \notin A_{\text{M}} \), then \((a) \rightarrow M(a) \sim (\sigma \downarrow M)\).

Where, \( \sim \) is a connector between two symbols. In fact, \( \downarrow M \) is a filter function for traces of service that only reserve the symbols in \( A_{\text{M}} \) and discard the symbols not in \( A_{\text{M}} \).

**Definition 4.4.** (compliance for activity chain mode) let \( L_W = < S_W, A_W, \rightarrow, s_{W0} > \) be a compositional web service. \( L_{\text{M}} \) is an activity chain mode defined in 2.2. \( L_W \) and \( L_{\text{M}} \) satisfy preconditions in section 4.1. If \( L_{\text{M}} \) and \( \sigma \downarrow M \) is a trace of \( L_{\text{M}} \) for any trace of \( L_W \), then call compositional web service \( L_W \) is compliant with activity chain mode \( L_{\text{M}} \) and denote it \( L_W = L_{\text{M}} \).

**Lemma 4.1.** Let \( \sigma \) be a trace of \( L_W = L_1||L_2||\ldots||L_k \), \( L_{\text{M}} = L_1 \), then \( \sigma_i \) is a trace of \( L_i \).

Proof: let \( \sigma_{\downarrow \tau} = a_1 \ldots a_k \), \( \sigma \downarrow M = a_1 \ldots a_k \). From the definition 4.3, \( a_1 \in A_1 \), \( k = 1 \) \ldots \( \ldots \) \( a_k \downarrow M \) is a activity chain of \( L_1 \), the activity labels \( a_1 \ldots a_k \) occurred before \( a_1 \) are not in \( A_1 \). So \( a_1 \) is first activity label in \( \sigma \) of \( L_1 \) that is also in \( A_1 \), \( a_1 \) is first activity label in \( L_1 \) from initial state related to the \( \sigma \). Similarly \( \sigma \) has no any other activity label in \( A_1 \) between \( a_i \) and \( a_{i-1} \). So \( \sigma_i \) is a \( L_i \) trace.

**Lemma 4.2.** Assume that \( L_{\text{C-PR}} \) is an activity chain precondition mode. If no considering the repetition of accepting activity label @\( L_{\text{C-PR}} \), the normal conclusion execution trace of \( L_{\text{C-PR}} \) will be the form \((!a_1)*a_1(!a_2)*a_2\ldots(!a_n)*a_n\). The proof of theorem is obvious. In figure 3.1 (3), for every normal conclusion execution of \( L_{\text{C-PR}} \), \( \rho \) must start from initial state and go into accepting state \( s_n \) and followed by infinite accepting activities @\( L_{\text{C-PR}} \). So the trace generated by \( \rho \) must be \((!a_1)*a_1(!a_2)*a_2\ldots(!a_n)*a_n \) @\( L_{\text{C-PR}} \)). The lemma holds without considering the infinite accepting activities postfix (@\( L_{\text{C-PR}} \)).

It is clear that every normal conclusion execution of \( L_{\text{C-PR}} \), \( \rho \), must contains all activities occurred in the order of that in \( C \) and activity \( a \). Activity \( a \) is the first activity apart from initial state and finally the execution must contain activities in \( C \). So the chain precondition mode \( L_{\text{C-PR}} \) holds. Conversely, if a behavior specification is expressed as a \( L_{\text{C-PR}} \), \( C \) and a must occur in any execution of compositional web service \( \rho \), then \( \rho \) must generate traces of the sub form \((!a_1)*a_1(!a_2)*a_2\ldots(!a_n)*a_n \). Since the parallel operation rule in definition 3.2, the LTS of the specification must also be such form. So \( L_{\text{C-PR}} \) reflects the exact operation semantic of activity chain precondition mode.

**Theorem 4.3.** Assume that \( L_W = < S_W, A_W, \rightarrow, s_W, s_W0 > \) is a compositional web service. \( C = \{a_1, a_2, \ldots, a_n\} \) is an activity chain. \( L_{\text{C-PR}} = < S_{\text{C-PR}}, A_{\text{C-PR}}, \rightarrow, s_{\text{C-PR}}, s_{\text{C-PR}0} \) is an activity chain precondition mode LTS. \( L = L_{\text{C-PR}}||L_{\text{C-RE}} = (S, A, \rightarrow, s_{\text{C-RE}}, s_{\text{C-RE}0}) \). Then compositional web service \( L_W \) is compliant with \( L_{\text{C-PR}} \) iff any ring in \( L \) is initial reachable and must contains the transition labeled with activity @\( L_{\text{C-PR}} \).

Proof: for sufficient case, let \( \rho \) is any execution of \( L \), \( \rho \) must be infinite execution because of preconditions. \( \rho \) must have a ring for only finite states. The transition labeled with the activity @\( L_{\text{C-PR}} \) must occur in \( \rho \) due to sufficient condition. Let \( \sigma \) is a trace corresponding to \( \rho \). So @\( L_{\text{C-PR}} \) must occur in \( \rho \). Let \( \sigma \downarrow \rightarrow \rho \) is a projection on @\( L_{\text{C-PR}} \), \( \sigma \downarrow \rightarrow \rho \) must be a trace of \( L_{\text{C-PR}} \) related to \( \rho \) for the lemma 4.1. Thus @\( L_{\text{C-PR}} \) must occur in \( \rho \). The accepting state \( s_n \) of \( L_{\text{C-PR}} \) must reach through the execution of \( \rho \), then conclude that @\( L_{\text{C-PR}} \) must be the form of \((!a_1)*a_1(!a_2)*a_2\ldots(!a_n)*a_n \) and \( \sigma \) must contain activity chain \( C \) and \( a \). Meanwhile a must occur before \( C \). Because \( \rho \) is any execution of \( L \). So a \( L_{\text{C-PR}} \) C holds.

For the necessary case, if there is a reachable ring that contains no @\( L_{\text{C-PR}} \) if \( (s_W, s_W) \) occur in this ring which \( s_W \) is accepting state of \( L_{\text{C-PR}} \). Then \( s_W \) has only one successor activity @\( L_{\text{C-PR}} \) and @\( L_{\text{C-PR}} \) must not occur for the assumption. So it is only the case that the direct successor activity of \( (s_W, s_W) \) will be \( a \) @\( A_{\text{C-PR}} \).

The compositional web service \( L_W \) has the infinite execution disjoint with @\( L_{\text{C-PR}} \) and \( s_W \) must not occur in this execution. It is a contradiction. So \( (s_W, s_W) \) does not occur in ring and the activity chain \( Q \) does not exist. □

An algorithm for checking the compliance of compositional web service can be obtained from the theorem 4.3 listed below.

**Check_C-PR_Chain(LTS Li, activity chain Q)**

(1) Constructing activity chain mode \( L_{\text{C-PR}} \) for \( Q \) according the rule 1 in figure 3.1.

(2) Extending \( L_{\text{C-PR}} \) and \( L \) according to the rules in section IV A.

(3) Let \( L = L_1||L_2||\ldots||L_n \) be LTS. DFS search throughout \( L \), pick out every ring in \( L \) if possible or if no any ring in \( L \) goes to (5).

(4) For every ring picked out, checking whether @\( L_{\text{C-PR}} \) is in the ring or not. If not, then \( L_W \) is incompatible with a \( L_{\text{C-PR}} \) and exit. If yes, goes to (3).

(5) If there is a ring in \( L \), then \( L_W \) is compliant with a \( L_{\text{C-PR}} \). C, otherwise, \( L_W \) is incompatible with a \( L_{\text{C-PR}} \).

**C. Compliance Verifications for Chain Response Mode**

Chain Response mode requires that when activity chain \( C \) occurs in certain trace of \( L_W \), \( \sigma \) must lead to activity \( a \) occur in the certain time of future. Their corresponding activity chain mode LTS may be extended by the rule in section IV A. Figure 3.1.4 \( L_{\text{C-RE}} \) is its corresponding LTS extended. \( L_{\text{C-RE}} \) are similar with chain precondition mode \( L_{\text{C-PR}} \) in structure. The compliance verifications is similar with the compliance verification for chain precondition mode.

**Theorem 4.4.** Assume that \( L_W = < S_W, A_W, \rightarrow, s_W, s_W0 > \) is a compositional web service. \( C = \{a_1, a_2, \ldots, a_n\} \) is an activity chain. \( L_{\text{C-RE}} = < S_{\text{C-RE}}, A_{\text{C-RE}}, \rightarrow, s_{\text{C-RE}}, s_{\text{C-RE}0} > \) is an LTS of activity chain precondition mode. \( L = L_{\text{C-RE}}\) is any LTS of activity chain precondition mode. \( L = L||L_{\text{C-RE}} = (S, A, \rightarrow, s, s_0) \). Then compositional web service \( L_W \) is compliant with \( L_{\text{C-RE}} \) C if any ring in \( L \) is initial reachable and must contains the transition labeled with activity @\( L_{\text{C-RE}} \).
V EXAMPLE ANALYSIS

In this section, we analysis the example presented in section II and check its compliance for Binary activity chain modes. In Figure 5.1, we give the LTS description of F_H and activity chain mode LTS. For the sake of simplicity, we omit other LTSs and its parallel of F and H. Because F and H only interact with F_H in the orchestration service and the client also interacts with F_H only. The activities in modes are activities of L F_H which are focused by Client. So we simply use L F_H instead of the parallel with L F and L H. It can be seen that there is a ring of L F-H||LC-PR that the activities labeled are not matching with the accepting activity @L=@L C- pr@= @L F H. In fact, the ring is (13, 2)e (15, 2) e*(13, 2). So the service is not compliant with activity chain precondition mode. Similarity, L F is not compliant with the activity chain mode a RE C.

VI CONCLUSION

In this paper, we propose the concept of behavior specification based on activity chain. Its basic element is activity chain and its granularity between activity and scenario. Referencing the idea of attribute patterns based on activity, we propose two binary behavior modes based on activity chain that are precondition mode, response mode. These modes can be used to describe behavior requirements for compositional web services. They are suitable for many practical cases. The scope in attribute pattern can not adopted in activity chain modes. The reason is that the scope can also be regarded as the result of “and” operation of multi modes, for example, the scope “before” can be seen as a precondition mode. Without the scope, activity chain modes can be more simple and concise. In order to verify the compliance for above modes, we extend LTS model to exclude deadlock state. Encode modes into LTS. Also by translating BPEL process into LTS, we give out the sufficient and necessary condition to check the compliance of compositional web service for modes. Finally, an example and its analysis are illustrated in

Figure 5.1. The LTSs of compositional web service F H and modes and paralle
paper. Future work may be consideration of optimization of verification.

REFERENCES


APPENDIX A: BPEL PROCESS OF COMPOSITIONAL WEB SERVICE F_H

xmlns:trv="http://f_htravel.com/bpel/travel/"
xmlns:fli="http://flighttravel.com/bpel/flight/"
<partnerlinks>
<partnerlink name="Travel" partnerLinkType="trv: TravelLT"
myRole="F_HService" partnerRole="TravelCustomer"/>
<partnerlink name="TravelFilter" partnerLinkType="trv:
TravelFilterLT" myRole="F_HFilterService"
partnerRole="TravelFilterCustomer"/>

Chungui Li, born on Aug. 1968, graduated from Beijing Institute of Technology for Ph.D. in artificial intelligence science in 2003. His main research interests include machine learning, intelligent systems, information retrieval and data mining.

He has been teaching and researching in Guangxi University of Technology, Liuzhou, China for many years. As an associate professor, he has published many papers in national academic journal and international conference.

Qixian Cai, born on Sept. 1948, as a graduate student, graduated from SouthEast University electronics in 1986, Nanjing, China. His major fields of study include temporal database technique, computer architecture, and computer based education. He was engaged in higher education work for 30 years . and once served as a vice-director of Computer engineering department, vice-director of Academic administration etc. Now, he is professor of Dept. of computer engineering of Guangxi University of Technology, in Liuzhou city of Guangxi province of China. He published 12 books and 3 softwares. Was a first author totaled to publish more than 50 papers in academic journals. The major publishing books and papers have: Computer Architecture, Beijing, China: Electronics Industrial Publisher; Computer Application in Education:Principle and Practice, GuiLin, China: GuangXi Normal University publisher; C Programming and Application, Beijing, China: Electronics Industrial Publisher; Research into C-Temporal Relation Data Model, Journal of Natural Science of Hunan Normal University, Changsha, China, 2004, vol.27, no.1, pp.18-22; Researching into the Method of Key-Time based on the C-TRDM and the Time Filtration Operator, Journal of natural science of Hunan Normal University, Changsha, China, Vol.28 No.3 Mar., 2004. pp13-17.

Bo Chen, born on Dec.1963, graduated from Chinese Science Institute for M.S. degree in the field of control theory in 1990 and now is a Ph.D. candidate of Tongji University in the field of software theory. His major study interests include trusted software and model checking. He has been for teaching and researching work for about twenty years in Guangxi University of Technology, Liuzhou, China. As an assistant professor, he has lectured many courses in computer science such as software engineering, database system, etc. He has published about twenty research papers in academic journal and international conference such as Computer Science, Geomatics and Information Science of Wuhan University, Chinese Journal of Computer, Niss 2008, APCIP 2009, etc.