ENVIRONMENT COMPATIBILITY OF SERVICES INTERACTION AND ITS REACHABLE ANALYSIS

B. Chen¹,²,*, X. Huang³

¹Service and Embedded Computing Key Lab of Ministry of Education, Tongji University, Shanghai, 201804, China
²College of Computer, Guangxi University of Science and Technology, Liuzhou, 545006, China
³Library of Guangxi University of Science and Technology, Liuzhou, 545006, China
* Corresponding author E-mail address: cb31@sina.com

The environment of services were not considered in most of analyses of interface interaction of composite services. In this paper, interface of interaction of services under specific environment are modelled as interface automata. The concept of environment compatibility of services is proposed. By giving the expression of criterion for compatibility checking and traversing the interaction model of composite services, reachable analysis for interaction environment compatibility checking can be carried. Some results of verifications are different from that of without environment and are accordance to actual situations.

Keywords: interface interaction, composite service, interface automaton, environment compatibility, and compatibility verification.

1. Introduction

A Service can combine with other atomic services or composite services through their interface interactions as a complex multifunctional software across organizations and platforms. However, combination mechanism also bring some risk and nondeterministic for composite services. Many scholars have modelled and analysed the behaviour of service interaction.

Bultan [1] regarded each peer participated in choreography as a finite automaton and studied the compatibility of interface interactions of two peers. Zhao [2] expressed the behaviour model of each participator in choreography with a special finite state machine named LOTOS. Cambronero [3] mapped the service interaction protocol WS-CDL to timed automata to express the behavioural of service. More scholars analysed interface interactions of composite web services with process algebra [4-7]. These studies are basically divided into two classes. One is the analysis of global interaction and answered whether one subservice in some interaction abided by the global agreement. Another were associated with local interactions and answered whether the interface interactions of two subservices is compatible?

This paper belongs to the latter. The difference is that the environment of interface interaction is introduced into the models and analysis. De Alfaro [8] discussed the interaction compatibility of two component under environment. His work focused on answering whether or not exists an environment, under which the interaction of two components is compatible. In this paper, we discuss the specific environment of composite web services and analyze their interaction compatibilities. It will been seen that if the interaction is compatible without considering the environment, the situation would be different.

2. Interface Automaton Model for Services and Interaction Environment

Interface automaton can be used to analyse the interface interaction of component system [8]. Emmi [9] noted that the system modelled by interface automaton has some advantage over the same system modelled by I/O automaton. Because some deadlock interaction in interface automaton system can be detected and that would not occur in I/O automaton system. We adopt interface automaton to model web service interaction for its characteristic is very similar to that of component system.

**Definition 2.1** an interface automaton is a hex-tuple, \( A=(S, I, Act^I, Act^O, Act^H, \delta) \), where

1. \( S \) is the set of finite states.
2. \( I \) is the set of initial states.
3. \( Act^I \) is the set of input activities
4. \( Act^O \) is the set of output activity symbols
5. \( Act^H \) is the set of noninteractive but visible activity symbols
Information and Computer Technologies

(6) $\delta \subseteq S \times (\text{Act} \cup \{\tau\}) \times S$ is a state transition relation, $\text{Act} = \text{Act}^0 \cup \text{Act}^1 \cup \text{Act}^H$

Specially, if $\text{Act}^I = \text{Act}^0 \cup \{\tau\}$, the corresponding interface automaton $A$ is called closed, otherwise, it is called open. If there exists a state transition, $(s, a, s') \in \delta$, then $a$ is called enabled in states. An interface automaton can be used to model interface interaction of web service.

**Definition 2.2** Let $A = (S_A, I_A, \text{Act}^0_A, \text{Act}^I_A, \text{Act}^H_A, \delta_A), B = (S_B, I_B, \text{Act}^0_B, \text{Act}^I_B, \text{Act}^H_B, \delta_B)$ are two interface automata. They are composable if they are unconstrained in signature, i.e. $\text{Act}^0_A \cap \text{Act}^I_B = \emptyset, \text{Act}^0_A \cap \text{Act}^I_B = \emptyset, \text{Act}^H_A \cap \text{Act}^H_B = \emptyset, \text{Act}^H_A \cap \text{Act}^H_B = \emptyset$.

**Definition 2.3** Let $A, B$ be two composable interface automata. Their composition is an interface automaton, $A \otimes B = (S, I, \text{Act}^0, \text{Act}^I, \text{Act}^H, \delta)$, where

1. $S = S_A \times S_B$, $I = (I_A, I_B)$.
2. $\text{Shared}(A, B) = \text{Act}^0_A \cap \text{Act}^0_B = (\text{Act}^0_A \cup \text{Act}^0_B \cup \text{Act}^I_A) \cap (\text{Act}^0_B \cup \text{Act}^0_A \cup \text{Act}^I_B) = \text{Act}^0_A \cap \text{Act}^0_B \cap \text{Act}^I_A \cap \text{Act}^I_B$
3. $\text{Act}^I = \text{Act}^I_A \cup \text{Act}^I_B \cup \text{Shared}(A, B)$.
4. $\text{Act}^0 = \text{Act}^0_A \cup \text{Act}^0_B \cup \text{Shared}(A, B)$.
5. $\text{Act}^H = \text{Act}^H_A \cup \text{Act}^H_B \cup \text{Shared}(A, B)$.
6. $\delta = \{(u, v, a, (u', v')) | u \in S_A \land u' \in S_A \land v \in S_B \land (u, a, u') \in \delta_A \land a \notin \text{Shared}(A, B)\} \cup \{(u, v), a, (u', v') | v \in S_B \land v' \in S_B \land (v, a, v') \in \delta_B \land a \notin \text{Shared}(A, B)\} \cup \{(u, v), a, (u', v') | u \in S_A \land u' \in S_A \land v \in S_B \land (u, a, u') \in \delta_A \land (v, a, v') \in \delta_B \land a \in \text{Shared}(A, B) \cup \{s\}\}$. Easily see that the composition above is associative and commutative.

**Definition 2.4** Let $A_E = (S_E, I_E, \text{Act}^0_E, \text{Act}^I_E, \text{Act}^H_E, \delta_E), A = (S_A, I_A, \text{Act}^0_A, \text{Act}^I_A, \text{Act}^H_A, \delta_A)$ are two interface automata. $A_E$ is called the environment of $A$ if $A_E$ satisfies that

1. $A_E$ and $A$ are composable in signature, i.e. $\text{Act}^I_E \cap \text{Act}^I_A = \text{Act}^I_E \cap \text{Act}^I_A = \text{Act}^I_A \cap \text{Act}^I_E = \emptyset$
2. $A_E$ is not empty (means that its initial set is not empty).
3. $A \otimes A_E$ is closed

The condition (3) means that interface automaton and its environment compose a closed system. Factually, BPEL4people standard supports role based people interaction. It is reasonable to consider the client activities as a subservice and the composite web service is closed.

3. The Example of the Interface of a Service

![Diagram](image1)

**Figure 1.** The interface automaton model of services in Online Book Service System

Online Book Service System: the system includes three services, Network Book Purchase, Service (NB for short), Book Provider Service (PV for short) and Online Bank (BA for short). For brevity, NB is considered as the combination of client and service interface and it represents client to interact with PV. Their interaction models can be described in Figure1.
4. The Compatibility of Interface Interaction

The compatibility definitions in existed works basically accorded with three situations in Bordeaux [10]. The compatibilities in this section are under specific environment and are distinguished with that without considering environment. The weak compatibility is just like the second definition in Bordeaux [10] under environment and the strong compatibility excludes one deadlock situation in the definition of Bordeaux [10].

**Definition 4.1** let A and B are two composable service interfaces. \((u,v)\) is a reachable state of composite interface \(A \circledast B\). If there exists an activity symbol \(a \in \text{shared}(A,B)\) such that \(a \in \text{Act}_u^A(u)\) (or \(a \in \text{Act}_u^B(v)\)) and \(a \notin \text{Act}_u^A(u)\) (or \(a \notin \text{Act}_u^B(v)\)), then \((u,v)\) is called an irresponsive state of composite interface \(A \circledast B\).

The set of all irresponsive states is denoted as \(\text{Irresponsive}(A,B)\). Easily proof the following formulae.

\[
(u, v) \in S_A \times S_B \mid \exists a \in \text{shared}(A,B) \begin{cases} \ a \in \text{Act}_u^A(u) \land a \notin \text{Act}_u^B(v) \\ \lor \ a \in \text{Act}_u^B(v) \land a \notin \text{Act}_u^A(u) \end{cases} 
\]

(1)

\[
\text{Irresponsive}(A \circledast B, C) = \text{Irresponsive}(A, B \circledast C)
\]

(2)

\[
(u, v, w) \in S_A \times S_B \times S_C \mid \exists a \in \text{Act}_u^B(v) \land a \notin \text{Act}_u^A(u) \lor a \in \text{Act}_u^A(u) \land a \notin \text{Act}_u^B(v)
\]

(3)

**Definition 4.2** A and B are two composable service interfaces. \((u,v)\) is a reachable state of composite interface \(A \circledast B\). If there exist two activity symbols \(a, b \in \text{shared}(A,B)\) and \(a \neq b\) such that \(a \in \text{Act}_u^A(u), b \in \text{Act}_u^B(v), a \notin \text{Act}_u^B(v), b \notin \text{Act}_u^A(u)\) satisfy that \(\forall c \in (\text{shared}(A,B) \setminus \{a, b\})\), A and B cannot execute through co-operating c, then called the state \((u,v)\) co-unrequested state of composite interface \(A \circledast B\).

\[
\text{Co-Unrequested}(A,B) = \begin{cases} (u, v) \in S_A \times S_B \mid \exists a, b \in \text{shared}(A,B) \land a \neq b \land a \in \text{Act}_u^A(u) \land a \notin \text{Act}_u^B(v) \\ \lor b \in \text{Act}_u^B(v) \land b \notin \text{Act}_u^A(u) \end{cases}
\]

(4)

The symbol \(c \in \text{Act}_u^A(u), \hat{c} \notin \text{Act}_v^A(v)\) means that if \(c\) is the activity symbol in state \(u\) of A, then \(\hat{c}\) is the corresponding reverse activity in state \(v\) of B (the reverse of an input activity is the corresponding output activity and vice versa.)
Information and Computer Technologies

Co-Unrequested(A ⊗ B, C) = Co-Unrequested(A, B ⊗ C) (5)
Co-Unrequested(A, B ⊗ C) =
(u, v, w) ∈ S_A × S_B × S_C |
\[
\begin{align*}
  a, b &\in shared(A, B ⊗ C) ∧ a \neq b, \\
  b &\in (Act_A^I(u) ∧ ¬Act_B^O(v) ∧ ¬Act_C^O(w)) \\
\end{align*}
\]
\[
\exists a, b \vee
\]
\[
\begin{align*}
  a, b &\in shared(B, C) ∧ a \neq b, \\
  b &\in (Act_B^I(v) ∧ ¬Act_C^O(w) ∧ ¬Act_A^O(u)) \\
\end{align*}
\]
\[
\forall c \in Act_A^I(u) \backslash \{a, b\} \land ¬Act_B^O(v) \land ¬Act_C^O(w)
\]

**Definition 4.3** let A_w = A_1 ⊗ A_1 ⊗ A_2, an interface of composite service, where A_1 and A_2 are two service interfaces. A_0 is an environment of them. If they satisfy following condition (1) and (2), then A_1 and A_2 is called environment weak compatible. Written as A ≈_w B. If they satisfy all condition, then A_1 and A_2 under environment A_0 is called environment strong compatible, written as A ≋_w B.

1. the interface of composite service, A_w, is closed, namely Act_A^I(u) = Act_B^O(v) = Act_C^O(w) = \varnothing.
2. there is no any reachable irresponsive state in composite interface interaction, namely, irresponsive(A_w) = \varnothing.
3. there is no any reachable co-unrequested state in composite service interface, i.e. Unrequested(A_w) = \varnothing.

\[\begin{array}{c}
\text{Figure 2. Service A and B with co-unrequested state and environment C}
\end{array}\]

Three interfaces are illustrated in Figure 2. The composite of service A and B is not closed. The interface of service C is their environment. Without considering environment, A and B are incompatible because there is a co-unrequested state. After considering the environment C, the co-unrequested state is unreachable and the interface interaction of A and B is compatible under environment C.

**Definition 4.4** let A_0 is the environment of service A. If Irresponsive(A_0, A) = \varnothing, then the interface interaction of A and its environment A_0 is called self environment weak compatible, written as A ≈_e A_0.

**Definition 4.5** let A_0 is the environment of service A. If Irresponsive(A_0, A) = \varnothing and Co-Unrequested(A_0, A) = \varnothing, then the interface interaction of A and its environment A_0 is called self environment strong compatible, written as A ≋_e A_0.

5. Compatible Checking for Interface Interaction

The compatible checking for interface interaction of services can be carried out by traversing the interface of composite service A_w and judging whether there is irresponsive state and co-unrequested state. The algorithm traverses very state of composite service interface automaton and judges whether there is irresponsive state and co-unrequested state according formulae (3) and (6). For space limited, only the procedure for judging if a state is or not an irresponsive state has been listed above, i.e. checking irresponsive(s). The sub procedure checking co-unrequested is similar. The number of the former sub-procedure loops is the total of activity labels in A_0 ⊗ (A_1 ⊗ A_2). The number of the latter sub-
Information and Computer Technologies

procedure loops depends its two loop variables. Suppose the N is activity labels in $A_i \otimes (A_1 \otimes A_2)$. The time complexity of two sub-procedures is $O(9N)$ and $O(4N^2)$ respectively. Where 9 and 4 is most number of judges in one loop of two sub-procedures respectively. ErrorFound procedure traverse $A_i \otimes (A_1 \otimes A_2)$ in depth-first fashion. The time complexity if $O(n^2)$, where n is the number of nodes in graph.

**Algorithm 5.1** (environment strong compatible checking):

IsCompatible_Envi

Input: $A_i = (S_i, I_i, Act^{I_i}_i, Act^{O_i}_i, \delta_i), i=1,2$ are two service interfaces, $A_e = (S_e, I_e, Act^{I_e}_e, Act^{O_e}_e, \delta_e)$ is the environment. $A = A_e \otimes (A_1 \otimes A_2)$ is the composition of service interface.

Output: "yes" if $A$ contains a irresponsive state or unrequested state, otherwise "no".

{  Set of states $R:=\emptyset$; // the set of visited states in first outer DFS.
  Stack of states $U:=\varepsilon$; //the stack for first outer DFS.
  Boolean error_found:=false; irresponsive:=false; unrequested:=false; State $I:=(I_e, I_1, I_2)$;
  State $s:=I$;
  errorfound(s);
  if error_found  return ("yes")
  else return("no");
}

Procedure Boolean Errorfound (state $s$)

{  push(s,U); R←\{s\};
  repeat {
    $s'\leftarrow$ top(U);
    if $Post(s') \notin R$ then {
      //the successor of $s'$ not been visited exists
      let $s:=$Post(s') \cup R;
      push(s',U); // push the unvisited successor of $s'$ and mark it reachable
      R←\{s\};
    }
    else{
      pop(U); // DFS finished for $s'$
      checking_irresponsive(s);
      if irresponsive==false checking_unrequested(s); error_found:=irresponsive\lor unrequested;
    }
  }until ((U ==\varepsilon) \lor error_found)
  return(error_found);
}

procedure Boolean Checking_irresponsive( state $s$)

Input: shared($A_i, A_1 \otimes A_2$); shared($A_1, A_2$); Act; state $s$.

Output: the irresponsive value of state $s$.

Step1: activity label $a$; boolean irresponsive:=false; Act←Act$_1 \cup Act_2 \cup Act_e$; //initialize variables.

Step 2: if (irresponsive==false \&\& Act\notin\emptyset) Let $a \in Act$;

if ($a \in shared(A_e, A_1 \otimes A_2)\ )$ {
  if ($a \in Act^{O_a}_E \otimes A_2 \otimes Act^{I_a}_E$) {
    if $a \in Act^{O_a}_1(v)\land Act^{I_a}_2(w)\land Act^{O_a}_A(u)$ irresponsive:=true;
    if $a \in Act^{O_a}_2(w)\land Act^{I_a}_1(v)\land Act^{O_a}_A(u)$ irresponsive:=true;
  }
  if ($a \in Act^{O_a}_1 \otimes A_2 \otimes Act^{I_a}_E$) {
    if $a \in Act^{O_a}_A(u)\land Act^{I_a}_1(v)\land Act^{O_a}_2(w)$ irresponsive:=true;
  }
}

if ($a \in shared(A_1, A_2)\land Act^{I_a}_E \otimes Act^{O_a}_A$) {
  if $a \in Act^{O_a}_A(u)\land Act^{I_a}_1(v)\land Act^{O_a}_2(w)$ irresponsive:=true;
  if $a \in Act^{O_a}_1(v)\land Act^{O_a}_2(w)\land Act^{I_a}_A(u)$ irresponsive:=true;
  if $a \in Act^{O_a}_A(w)\land Act^{I_a}_1(v)\land Act^{O_a}_2(u)$ irresponsive:=true;
}
Information and Computer Technologies

if (a:\text{shared}(A_1 \cdot Act) \land a:\text{shared}(Act_{AB})) \{
  if a:\text{Act}^A(v) \land a:\text{Act}^A(w) \land a:\text{Act}^B(u) \text{ irresponsive} → \text{true};

  if a:\text{Act}^A(v) \land a:\text{Act}^B(w) \land a:\text{Act}^B(v) \land a:\text{Act}^A(u) \text{ irresponsive} → \text{true};
\}

Step 3: if (irresponsive = \text{true} \land \text{Act} \neq \emptyset) \text{ Act} ← \text{Act} \setminus \{a\}; \text{ go to step 2};

Step 4: if (irresponsive = \text{false} \land \text{Act} = \emptyset) \text{ return} (irresponsive).

Without considering the composite environment, service NB and PV in Online Book Service System in section 3 are incompatible for the state 7 is a reachable irresponsive state. By introducing the environment BA, the reachable state 7 in their composite interface interaction model is limited as a unreachable state. Thus their interface interaction is environment strong compatible.

6. Conclusion

The interface automaton is applied to model composite web service interface interaction in this paper. By introducing the environment factor, the environment strong and weak compatibilities of service interaction under specific environment are proposed. We think this consideration is accord with factual service situation. After given the criterion of judge for irresponsible state and co-unrequested state, we can check the environment compatibility through reachable analysis. Future work will focus on global interaction analysis.

Acknowledgment

This research is supported by Open Project Program of Service and Embedded Computing Key Lab of Ministry of Education in Tongji University (No.2011-2), Research Program of Education Department of Guangxi (No. 201203YB124), The Program of Doctoral Fund of Guangxi University of Science and Technology (11Z05).

References


Received on the 10th of December 2013