Semantic Component Behavior Equivalence  
Analysis based on Higher-order Typed $\pi$ Calculus  
Qing Wu, Chunbo Zhao, and Weihua Hu  
College of Computer Science, Hangzhou Dianzi University  
Email: wuqing@hdu.edu.cn

Abstract—It is an important issue to analyze the behavioral equivalence of semantic components when studying the dynamic replacement and recombination of them for middleware adaptation. However, it is difficult to check the equivalence of behaviors rapidly and precisely. In order to improve the precision of judging and guarantee the normality and stability of system after replacing and recombining components, the paper mainly uses and extends the theories of equivalence analysis based on higher-order typed $\pi$ calculus. By analyzing the equivalence of semantic components, it can ensure the behavioral consistency of the new component and replaced component, also effectively guarantee the stability and normality of the whole middleware system’s dynamic adaptation.

Index Terms—semantic component, component behavior equivalence

I. INTRODUCTION

Nowadays, it attracts much attention on making research on component based software architecture, in which replacing and recombinant of component has become one of the most attractive topics. In order to ensure the stability and reliability of the whole system, we have to analyze the behavioral equivalence of new component and replaced component.

Component behavior equivalence requires not merely interface matching, but consistency of function behavior. Many researchers neglect the function behavior and this leads to the one-sidedness of research results. Some others have used interface automata to model component behavior flow [1], [2], those behaviors having same action sequence is considered as equal. However, when one flow has more than one action sequence, the respectively equivalence of each sequence doesn’t mean the equivalence of the whole flow, thus the accuracy is not high enough. Besides, process algebra bisimulation theory [2], [3] has also been applied to analyze the behavior equivalence. Nevertheless, no standard has been made for analyzing the component interactive behavior equivalence [4]–[6], meanwhile the one-sidedness and inaccuracy still exist.

Aiming at these problems, this paper extends the bisimulation theory of $\pi$ calculus, uses high-order typed $\pi$ calculus to formalized model and analyze equivalence. The following content of the paper is organized as below. Section 2 describes how to use high-order typed $\pi$ calculus to formalized express the semantic component behavior. In section 3, we propose some theories on behavioral equivalence and analyze it. Some related work is stated in section 4. Finally, the conclusion and further research work are put forward.

II. FORMALIZED MODEL OF SEMANTIC  
COMPONENT BEHAVIOR

In this section, we give an overview of higher-order typed $\pi$ calculus. And then, the semantic component dynamic behaviors modeling based on higher-order typed $\pi$ calculus is presented in detail.

A. Overview of higher-order typed $\pi$ calculus

The $\pi$ calculus is one of formal methods to model and reason about concurrency and mobility. It extends CCS [7] with the ability to create and remove communication links between processes. One extension of the $\pi$ calculus is the higher-order typed $\pi$ calculus by D.Sangiorgi [8], where the objects transmitted can also be processes. Process, name and abstraction are three parts of the higher-order typed $\pi$ calculus. Process is a working unit of current running entity, and uses name to define channels and objects transmitted on the channel. Each process interacts with other process via a shared channel. We use $P$, $Q$, $R$, ... to range over processes. Name is a reference of one object. We use a, b, X, Y, ... to range over value names and object(process) names. Abstraction is a non-concrete process with some parameters. Thus, the class of processes is given by the following grammar.

\[ P ::= 0 | \alpha \cdot (X).P | \alpha ! (Y).P | P + Q | P || Q | (\nu X)P | [X = Y]P | A(X) \]

(1) $0$ is blank process which can do nothing.  
(2) $\alpha \cdot (X).P$ is an input prefix process. It means one name $Z$ is received along the channel $\alpha$, and $X$ is a placeholder for the receive name. After this input, it will continue as process $P$ and $X$ will be replaced by the newly received name $Z$, which is described as $P[Z/X]$.  
(3) $\alpha ! (Y).P$ is an output prefix process. It means the name $Y$ is sent through the channel $\alpha$, and after that it will continue as process $P$.  
(4) $P + Q$ is a sum process, which represents a process that can either $P$ or $Q$.  
(5) $P || Q$ is a parallel composition process, which means the combined behavior of $P$ and $Q$ in parallel. $P$ and $Q$ can act independently, and may also communicate if one performs an output and the other an input along the shared channel.
is a restriction process, which behaves as P. Since the name is a local name in P and it can’t be used to communicate with other processes.

(7) \([X = Y]P\) is a match process. The process will behave as P if X and Y are the same name, otherwise it does nothing.

(8) \(A(K)\) is an abstraction with concrete parameters process. A is an abstraction, defined as \(\hat{x} A \hat{x}\), is a set of process formal parameters, and \(\hat{K}\) is a set of process actual parameters. So \(A(\hat{K}/\hat{x})\) is conducted.

Besides, process reduction and transition rules are defined in Pi-calculus with particular semantics. The reduction rules consist of R-COM, R-PAR, R-RES, and R-STRUCT.

B. Semantic component dynamic behaviors modeling

We have developed a semantic and adaptive middleware platform called ScudWare [9], which is based on semantic information and conformed to a lightweight CCM (CORBA component model) specification [10]. In ScudWare architecture, the semantic components are essential software entities. These components implement some application logic and can execute special functions when they are instantiated. The structure properties, function behaviors, and inner adaptive behaviors are three important parts of the semantic components. Specially, the component inner adaptive behaviors can change component resources consumption states to get satisfying execution effect required from other components in the ScudWare middleware system. The system components can provide runtime environments infrastructure such as context-aware information and adaptive behaviors managements for semantic components. As a result, the semantic and system components are executors of functional and non-functional behaviors of ScudWare middleware.

The semantic component behaviors have dynamic and concurrent characters. In addition, the component interacts with others through its service request ports and service supply ports, whose interaction is mainly embodied in messages transfer. Similarly, the processes of \(\pi\) calculus transfer messages with others through its channels. Thus we can map the component ports to the \(\pi\) calculus process channels. And the transceivers of messages by components correspond to the transceivers of messages by \(\pi\) calculus process.

According to the different transfer forms of message, the component atomic behaviors can be divided into three kinds. The first is send only(S). The second is receive only(R). And the third are send before receive(SR) and receive before send(RS).

The set of all communication ports is described as GT.

\(GT = \{gt_1, gt_2, \cdots, gt_n\}\)

The set of all input messages is defined as \(M_{in}\), including input names and processes.

\(i_s\) represents one of the messages. \(M_{in} = \{i_1, i_2, \cdots, i_s\}\).

The set of all output messages is defined as \(M_{out}\), including output names and processes. \(o_j\) represents one of the output messages. \(M_{out} = \{o_1, o_2, \cdots, o_j\}\).

\(gt_n(i_s)\) denotes receiving message \(i_s\) through port \(gt_n\), while \(gt_n(o_j)\) denotes sending message \(o_j\) through port \(gt_n\). The component atomic behaviors of S, R, SR and RS through port \(gt_m\) can be described as follows:

\(P_{send} = gt_n(i_j, i_{j+1}, \cdots, i_{k-1}, i_k).0\)

\(P_{receive} = gt_n(i_j, i_{j+1}, \cdots, i_{k-1}, i_k).\pi.0\)

\(P_{send, receive} = gt_n(o_j, o_{j+1}, \cdots, o_{k-1}, o_k).0\)

\(P_{receive, send} = gt_n(i_j, i_{j+1}, \cdots, i_k).\pi. gt_m(o_j, o_{j+1}, \cdots, o_k).0\)

For instance, a network-based search engine in ubiquitous computing environments have one component called Chinese search component P, which has three subcomponents those are TranslaotCE, EnglishSearchEngine and TranslatorEC. The input and output channels of P are named as \(p_{IN}\) and \(p_{OUT}\), while the channels of TranslatorCE, EnglishSearchEngine and EnglishSearchEngine and TranslatorEC are called \(p_{EC}, p_{SEC}\) and \(p_{CE}\). The Chinese search component dynamic behaviors are illustrated in figure 1.

If \(P_{CSI}\) represents the dynamic behaviors of P, then it can be described as follows:

\(P_{CSI} = P_{IN}(m, P)\)

\(P = ([f_{IN}(i_n).p_{EC}(\text{in}.p_{EC}(\text{out}.f_{OUT}(\text{out}.0))])f_{OUT}(\text{out}.0).f_{OUT}(\text{out}.0)\)

The component evolution is shown as below:

\(P = (v_{IN}(i_n).p_{SEC}(\text{in}.p_{SEC}(\text{out}.f_{OUT}(\text{out}.0))])f_{OUT}(\text{out}.0).f_{OUT}(\text{out}.0)\)

\(P_{CSI} = P_{IN}(m, P)\)

\(P = (v_{IN}(i_n).p_{SEC}(\text{in}.p_{SEC}(\text{out}.f_{OUT}(\text{out}.0))])f_{OUT}(\text{out}.0).f_{OUT}(\text{out}.0)\)

Figure 1. Behavior View of Chinese Search Component P.
According to the Chinese search component P in ubiquitous computing environments mentioned above, we can create some other Chinese search components, then analyze and verify the behavioral equivalence of them.

Firstly, a Chinese search component named Q is created, which can return Chinese search result according to the input search terms. After obtaining search words, Q will choose one of the following processing methods: a. use Chinese search engine directly; b. first convert Chinese search terms into English, then use English search engine, at last convert search result back to Chinese. The behavior view of component Q is concretely described in figure 2.

Figure 2. Behavior View of Chinese Search Component Q.

Assume that Q have input and output channels successively named as pt\textsubscript{Q\_IN} and pt\textsubscript{Q\_OUT}, while the channels of Chinese search engine, English search engine, Translator\textsubscript{EC} and Translator\textsubscript{CE} is successively described as pt\textsubscript{Q\_SCE}, pt\textsubscript{Q\_SEC}, pt\textsubscript{Q\_EC} and pt\textsubscript{Q\_CE}. If process Q\textsubscript{CSE} represents interactive behavior of Q, then it can be described as follows.

1. \(Q_{CSE} = pt_{QSEC}(\text{in})\cdot pt_{QEC}(\text{out})\).
2. \(Q_1 = (\text{random})\cdot pt_{PADOH}(\text{random})\).
3. \(Q_2 = pt_{QSEC}(\text{in})\cdot pt_{QEC}(\text{out})\).
4. \(Q_3 = (\text{select})\cdot pt_{QSEC}(\text{in})\cdot pt_{QEC}(\text{out})\).
5. \(Q_3 = (\text{select})\cdot pt_{QSEC}(\text{in})\cdot pt_{QEC}(\text{out})\).
6. \(Q_3 = (\text{select})\cdot pt_{QSEC}(\text{in})\cdot pt_{QEC}(\text{out})\).

The evolution of process Q3 is similar to the process P1 in P\textsubscript{CSE} and it’s no longer repeated here.

III. ANALYSIS OF SEMANTIC COMPONENT BEHAVIORS EQUIVALENCE

In terms of the semantic component model and its dynamic behavior formalization, we will propose the analysis and verification of semantic component behaviors equivalence.

A. Component Context (CC)

Here, we divide component context into external environment and internal environment. The former is reciprocal behaviors of inside subcomponent interactive behaviors, while the latter is reciprocal behaviors of interactions with other components. If a component P has a set of outer behaviors named as X, a set of inner behaviors named as Y, and a context CP, well then CP can be described as follows.

\[\begin{align*}
\sigma_P &= \alpha^+ + \chi P' \alpha^-
\end{align*}\]

Much attention should be paid to the bisimulation strength difference between HSB and HWB. As HSB’s over high bisimulation and HWB’s regardless of internal
actions, HWB is much more widely used than HSB. However, HSB is always needed in high stability and reliability system.

D. Context Compatibility Bisimulation (CCB)

CCB (Φ) is a relation between processes such that

\[ P \triangleleft_\Phi Q \text{ implies:} \]

1. \( C_P \) is a context of \( P \);
2. If \( P \mid C_P \rightarrow \cdot \cdot \cdot .0 \) then \( Q \mid C_P \rightarrow \cdot \cdot \cdot .0 \).

Aiming at the compatibility of new component with replaced component context, CCB always is a weaker bisimulation than HWB.

IV. RELATED WORK

It is an important issue to analyze the behavioral equivalence of components when studying the dynamic replacement and recombination of them for software adaptation. T. Basten [1], Zhang [2] has used interface automata to express component behavior, and it’s intuitive to describe behaviors in graphics mode, but the computational complexity will increase rapidly when interactive behaviors become involved. Xu [3] has put forward some bisimulation equivalence theories of higher-order π calculus, but no practical case based on these theories has been given. Gao [11] has used calculus to model behavior, but they haven’t verified the equivalence of different behaviors. In order to ensure the stability and high performance of the whole system after replacing or recombining components, we must analyze and ensure the equivalence of new component and replaced component. Focusing on this problem, this paper emphatically model and analyze the equivalence of behaviors among components.

V. CONCLUSION AND FUTURE WORK

This paper firstly describes how to use higher-order typed π calculus to formalized the model of component behavior, and then we put forward some theories for equivalence analysis. Through the precise analysis of equivalence of component behavior, we can ensure the behavioral consistency of new component and replaced component, also effectively guarantee the stability and normality of the whole middleware system’s dynamic adaptation. In the next work, the semantic component evolution behaviors should be considered in our methods.

REFERENCES