Web Service Composition Modeling Based on Pi Calculus

Yang Yan[1], LI Rong[1], WANG Sai[1]

1. Computer School, Central China Normal University, Wuhan, China

Email: ms_yangyan@163.com

Keywords: Pi Calculus; Web Service; Service Composition; Composition Validation

Abstract. Web service composition has been an important research topic in the field of service computing. Through the web service process modeling and the formal verification of the composition, we make sure the composite services meet the designer’s anticipated targets. Pi calculus is process algebra which is suitable to describe the dynamic Topological structure of concurrent systems. In this paper we introduce the basic grammar of Pi calculus, give a formal description based on Pi calculus for BPEL4WS concept mapping, provide a formal model to check and validate the composition modeling. And finally the validation process of the models is illustrated through examples.

Introduction

Web services composition is a temporarily assembled association of some autonomous service according to certain business logic. Generally speaking, individual atom services for the composition come from different organizations or institutions. In order to ensure this temporary service jointly work normally to reach the business objectives, the processes of the service composition and the business logic must be consistent, and each atom service must do the normal cooperation and interaction to each other. Thus the smooth use and execution of each atom service can turn the normal circulation of service data within the atom services into reality. Therefore, the validation before the combined process service being put out to run is quite vital.

Now a wide research of the validation for web service validation has been carried out at home and abroad. These studies are mostly based on formal method, such as the use of Petri net as [1], based on automata theory as [2] and based on process algebra as [3] . For service composition verification, the above three kinds of formal methods have basically the same ability in the expression for web services validation, but they have big difference in the use convenience degree and the computational complexity. Using Petri net or automata to describe the service composition will be more intuitive, but will cause state space explosion when the service process scale greatens, service quantity grows and service interaction become more complicated. Compared with them, the method based on process algebra by describing the process in text system, has stronger ability of expression and concise form. The main reason of using Pi calculus as the web service composition modeling language is that the Pi calculus can directly describe the dynamic Topology structure. On the other hand, Pi calculus can directly express the circulation which can express the channel as a value, thus makes the Pi calculus especially suitable for describing web services dynamic composition [4].

Higher-order Pi Calculus

With the development of communication system and network distributed computing, there is greater need to design software systems which have multiple concurrent activity components. However, traditional calculation models, such as Turing machines and lambda calculations, have deficiency in modeling of interactive systems. Their basic activity is to read the stored contents or function use, and they’re also unable to describe the interaction and concurrent execution software system. By the 1990s Milner has got the Pi calculus when broadening CCS. Pi calculus is the most important concurrent computation model in computer parallel field, higher-order polymorphism Pi calculus has got a development based on first-order Pi calculus by Sangiorgi in order to describe the changing concurrent systems of the structure and the behavior.

Pi calculus has three basic entities: process, name and abstraction. Process is a unit of concurrently run entity, and it gives a unifying definition of channel and object sent by channel under name. Each process has many contacting channels with other processes. It interacts with them by the shared channel. The difference between higher-order polymorphism Pi calculus and the first-order Pi calculus mainly lies in that the object's name itself can be the process, which makes the data of the interaction communication can be the process, and the process can be passed. Abstraction which is based on the process is a process with parameters. Concretizing the abstract parameters will get the general process. The parameters and the objects transferred in a channel have the types in higher-order polymorphism Pi calculus. We need to pay attention to the type compatible when concretizing the abstract parameters and the process interactions.

Definition 1. Put N says a set of name, name is represented by small letters, $\overline{N}\underline{\underline{\Delta}}(\overline{\alpha}\mid\alpha\in N)$, $\alpha\in N\cup\overline{N}$,P,Q means the process of high-order Pi calculus, then the high-order polymorphism Pi calculus can be defined as:

$$P::=0\mid\alpha x.P\mid\overline{\alpha}y.P\mid P+Q\mid P\mid Q\mid(\nu x)P\mid\left[x=y\right]P\mid\pi\text{ , }\pi::=\ x(y)\mid\overline{x}y\cdot$$

It can be explained as follows:

0 stands for non-active process, which cannot interact with any other processes.

αx.P:α refers to the importing port of one channel, the behavior for this process is to firstly pass the channel name α to the receiving object y, then to activate the process P[y/x],x is the local name,[y/x] stands for the operation of changing name of a.

$\overline{\alpha}$ y.P: $\overline{\alpha}$ refers to the exporting port of one channel, the behavior for this process is to export the object name y through the channel nameα,then to activate the process P,y is global name.

P+Q: This is an uncertain computing form of the process, this process will activate the process P or Q according to some certain incentives, its behavior is only the behavior for the process which is activated.

P|Q: This is a process of parallel operation form, in this process P and Q parallel exists, they can exchange the operations with other processes separately, they can also communicate with each other through shared channel.

(vx).P: In this process, x is the domain name of process P, it makes the process P unable to communicate with other process via channel x.

[x=y]P: This is a conditions process, if x=y, it will activate the process P, otherwise it will not take any operations and will be the non-active process 0;

In the polymorphism Pi calculus, the data passed by the channel can be a name vector, such as $x(y_1...y_n).p$ or $x(\overline{y}).p$, $\overline{x}y_1...y_n.p$ or $\overline{xy}.p$ etc. We mark $x(y_1...y_n).p$ as $x.(\lambda y_1...y_n)p$, and mark $\overline{x}y_1...y_n.p$ as $\overline{x}.[y_1...y_n]p$, and defines, process abstract: $k=(\lambda y_1...y_n)p$,process concretization: $C=[\lambda y_1...y_n]p$

The high-order polymorphism Pi calculus defines the evolution of the process through reduction rules and transition rules. The reduction rules and transition rules can be refereed to literature.

Formal description based on Pi calculus

BPEL cooperation partner relationship and basic composition

BPEL4WS provides an XML format language to describe the business process and business interaction agreement. The executable business process with the definition of BPEL will involve inter-departmental and cross enterprise service providers, they are called partners. Not only BPEL will involve messaging, partners also need to send messages to BPEL program. BPEL uses the statement of partner link types and partner link to express this cooperation relationship. In BPEL the elements used to define the partner releationship are partnerLinkType, partnerLink and partner. The relationship of the three is as what the picture shows, partnerLinkType examples partnerLink, and partner is the collection of partnerLink. After defining the partnerLinkType in BPEL program interface, we can define the concrete partnerLink in BPEL process. Multiple partnerLink can have one partnerLinkType. Partner is a group set of partnerLink. PartnerLink defines the business partner relationship from interaction, but partner is from cooperation partner to define some partner who must provide the service. Since one partnerLink only represents the relationship with one partner, it can only belong to one partner. In BPEL process, the partner elements are optional.

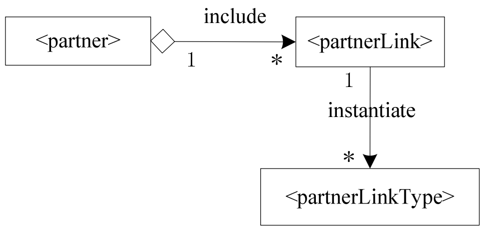

Figure 1. The relationship of partner. partnerLink and partnerLinkType

BPEL4WS defines all kinds of different atom activities and structural activities to describe a combined business process. Each atom activity of BPEL4WS（receive. reply. invoke. assign. wait. empty. throw. Compensate） has the following four elements: partnerLinks refers to the engine service and the link relationship with the external service, operation refers to the method of engine and the external service. Variable stores the parameters, one operation has import and export messages, so variable has inputVariable and outputVariable. The structural activity includes (flow. sequence. if. pick. while) to standardize the execution series of web service, which is the interaction with external environment, and the activities with the customer port and web service integration. The structured activity for BPEL4WS is similar to the control statements of the traditional programming language. The XML elements of the control statements can include other activities, as Figure 2 shows.

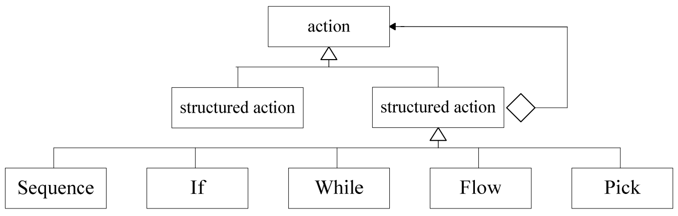

Figure 2. BPEL4WS structured activities

Pi calculus modeling on BPEL4WS

In order to facilitate the modeling for all kinds of behaviors and the handier of BPEL by using Pi calculus, we firstly create a mapping of BPEL4WS concept to Pi calculus:

Business process	Pi calculus process
All kinds of activities	Pi calculus process
Handier	Pi calculus process
Interaction relationship between the web service	Pi calculus process communications
Variables	Message got from Pi calculus process communications
partnerLink, portType and operation	Channels for Pi calculus process communications

In order to formally describe BPEL4WS easily, we first give below definitions:

Definition 2. For a four-unit group ChannelName=(PartnerLink, portType, operation, variableName) corresponds an exclusive channel name, we set all the attributes as a channel partnerLink\ PortType\operation\VariableName, take Variable as the composition of the channel, and the name passed in the parentheses is the Variable name.

BPEL4WS uses a web service, it ends after using complete, so the atom activity modeling is P=···.0, one activity ends after being used by engine, and it reaches an empty activity state.

In order to make the grammar rules agree with the following used modeling validation tool, we use $\grave{a}$<x> to express the channel export name x, use $\hat{v}$ to express a constraint name (internal name).

1. Through the channel order Customer sends the order request name orderreq to Travel, and the dynamic channel named ack is waiting for the result at the same time at channel ack. Customer model by Pi calculus is as follows:

$\mathrm{Customer}(\mathrm{order}, \mathrm{orderreq}, \mathrm{ack})$

$= \text{`}\mathrm{order} < \mathrm{ack} > . \text{`}\mathrm{order}<\mathrm{orderreq}>.$

$(\mathrm{ack}(\mathrm{resulta}).0|\mathrm{ack}(\mathrm{resulth}).0)$

2. Travel receives the channel name ack and the order request orderreq from order through channel air. It sends the order to Airline and Hotel and the internal channel airresp. hotelresp, waiting for the resulta and resultb from the internal channel airresp. hotelresp. Finally it gets back to Customer. Pi calculus process modeling for Travel is as follows:

$Travel(order, air, hotel, ack)$

$= (\text{`}airresp, hotelresp)order(ack).$

$order(orderreq).(\text{`}air < orderreq > .$

$\text{`}air < airresp > .airresp(resulta).$

$ack < resulta > .0 \,|\, 'hotel < orderreq > .$

$'hotel < hotelresp > .hotelresp(result).$

$'ack < result > .0)$

3. Airline gets the orderreq and the internal channel name airresp from channel air. According to the ticket situation it gets back to Travel through airresp. Pi calculus process modeling for Airline is as follows:

$Airline(air, resulta) =$

$('airresp)air(orderreq).air(airresp).$

$'airresp < resulta > .0$

Similarly, using a Pi calculus process to model Hotel is as follows:

$Hotel(hotel, result) =$

$('hotelresp)hotel(orderreq).$

$hotel(hotelresp).hotelresp < result > .0$

4. Travel Service composed by Travel, Airline and Hotel is as follows:

$TravelService(order, resulta, resulth) =$

$Travel \,|\, Airline \,|\, Hotel =$

$(\hat{a}ir, hotel, ariresp, hotelresp)(order(ack).$

$order(orderreq).('air < orderreq > .$

$'air < airresp > .airresp(resulta).'ack$

$< resulta > .0 \,|\, 'hotel < orderreq > .'hotel$

$< hotelresp > .hotelresp(result).'ack$

$< resulth > .0) \,|\, air(orderreq).air(airresp).$

$'airresp < resulta > .0 \,|\, hotel(orderreq).$

$hotel(hotelreq).hotel(hotelresp).$

$hotelresp < resulth > .0)$

On this basis, we use the MWB tool to check the grammar correctness, whether there's deadlock, and use the step demand to track the interaction of the service. When validating whether the composite service meets the customer's requirement, we use behavior inversion method, which is the exchange of import and export. If the exchange is week similar, then the composite service can meet the customer's requirement.

The other important content of the modeling check is to validate whether the designed composite service can meet the customer's requirement. For that, for example, we validate whether Travel Service and the Customer can interact correctly, which is the behavior matching nature validation of the composite service TravelService and Customers. Here we use a behavior inversion method, which is the import and export exchange, then get the mirror process RevCustomer of the Customer process. If RevCustomer process is weak similar as TravelService, then TravelService matches Customer.

Conclusion

One composite web service is made up of a group of sub services. These sub services concurrently interact with each other to meet the customer's requirements. And the interactions among them are completed by the communications and exchanging of messages. One of the key research problems for web service composition is to ensure the correctness of the composition. In this paper we provide a formal model to check and validate the composition modeling, and also give a formal description based on Pi calculus for the most important web service composition standardization BPEL4WS concept mapping. Finally we describe the model validation method with the case study. The future work includes: how to deal with the time and event in web service composition; how to use Pi calculus to create web service automatic composition model and the automatically exchange problem for the web service composition description language and the Pi calculus modeling.

References

[1] J. He, Y. Geng and K. Pahlavan, Modeling Indoor TOA Ranging Error for Body Mounted Sensors, 2012 IEEE 23nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), Sydney, Australia Sep. 2012 (page 682-686)

[2] Y. Geng, J. Chen, K. Pahlavan, Motion detection using RF signals for the first responder in emergency operations: A PHASER project[C], 2013 IEEE 24nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), London,Britain Sep. 2013

[3] S. Li, Y. Geng, J. He, K. Pahlavan,Analysis of Three-dimensional Maximum Likelihood Algorithm for Capsule Endoscopy Localization, 2012 5th International Conference on Biomedical Engineering and Informatics (BMEI), Chongqing, China Oct. 2012 (page 721-725)

[4] Y. Geng, J. He, H. Deng and K. Pahlavan, Modeling the Effect of Human Body on TOA Ranging for Indoor Human Tracking with Wrist Mounted Sensor, 16th International Symposium on Wireless Personal Multimedia Communications (WPMC), Atlantic City, NJ, Jun. 2013.