面向应用的 Web 服务组合设计和验证框架

辜希武,李瑞轩,卢正鼎
GU Xi-wu,LI Rui-xuan,LU Zheng-ding

华中科技大学 计算机学院 应用系,武汉 430074
College of Computer Science and Technology,Huazhong University of Science and Technology,Wuhan 430074,China
E-mail:guxw_wang@sina.com

GU Xi-wu,LI Rui-xuan,LU Zheng-ding.Application oriented framework for Web services composition design and verification.Computer Engineering and Applications,2008,44(34):132-136.

Abstract:To deal with the issue that Web services composition design specifications lack of formal semantics and formal verification methods,the paper outlines a top-down framework-iFrame4WS for Web services composition design and verification.In iFrame4WS,the process of Web services composition is divided into description level,abstract level and execution level.The formal model and verification approach are defined in abstract level so as to ensure the correctness of Web services composition.
Key words:Web services;Web services composition;formal model

摘 要:针对 Web 服务组合设计规范缺乏形式化的语义和验证方法的问题,提出了一个自顶向下的 Web 服务设计和验证的框架-iFrame4WS。在 iFrame4WS 中,将 Web 服务组合的设计方案划分为描述层、抽象层和执行层,并通过抽象层的形式化模型和形式化验证来检查 Web 服务组合的正确性。
关键词:Web 服务;Web 服务组合;形式化模型
DOI:10.3778/j.issn.1002-8331.2008.34.041 **文章编号**:1002-8331(2008)34-0132-05 **文献标识码**:A **中图分类号**:TP311

1 引言

随着 Web 服务技术的迅速发展,越来越多的的 Web 服务运行在 Internet 上,如何将已有的、运行在异构平台上的 Web 服务组合起来,以提供给用户更为强大和增值的功能,成为 Web 服务研究领域中的一个热点。

基于工作流的 Web 服务组合方法可以分为两类[1]:基于 Choreography 和基于 Orchestration。基于 Choreography 的组合方法从一个全局的角度定义每个参与者(子服务)之间的会话和协作;基于 Orchestration 的方法则是局部的,它从一个参与者的角度来定义该参与者如何与其它子服务交互。目前,基于 Orchestration 的 Web 服务组合描述规范主要是 Web 服务商业流程执行语言[2](Business Process Execution Language for Web Services,BPEL4WS),基于 Choreography 的 Web 服务组合描述规范主要是 Web 服务编排描述语言[3](Web Services Choreography Description Language,WS-CDI)。这两种方法不是孤立的,而是位于两个不同层次上而被结合起来用于实现 Web 服务的组合:首先利用 WS-CDL 规范从全局的角度描述参与组合的各个子服务(WS-CDL 称之为角色)之间的互相协作的过程,然后将全局的协作过程映射到每个角色上,从而得到每个角色的局部行为(用 BPEL4WS 描述)。

但是 BPEL4WS、WS-CDL 都是基于 XML 的描述性语言,缺乏一种形式化模型来表达语言的语义以及形式化验证方法来保证用这些规范所定义的 Web 服务组合的正确性。同时,由于参与 Web 服务组合的各个子服务都运行在分布式异构的环境下,因此想依靠组合的 Web 服务的实际运行来检测组合中的错误是代价昂贵的并且几乎是不可能的,所以对 Web 服务组合的形式化验证也是必须的而且是非常重要的。

本文提出了一个面向实际应用的 Web 服务组合的框架-iFrame4WS(Integration Framework for Web Services)。该框架集成了针对 BPEL4WS 和 WS-CDL 规范提出的形式化模型,以使得 Web 服务组合的开发者能够以一种自顶向下的方式设计、验证 Web 服务组合方案。

2 框架体系结构

图 1 给出了 iFrame4WS 的体系结构,框架分为 3 层:描述层、抽象层和执行层。其中各层的功能将在第 3~5 章详细介绍。

3 框架描述层

在 iFrame4WS 的描述层,已有 Web 服务的 WSDL 接口描述通过自定义的 Register API 注册到 UDDI 注册中心;Web 服

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60403027,No.60773191);国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z403)。
作者简介:辜希武,男,讲师,博士,主要研究领域为 Web 服务、语义 Web 和中间件;李瑞轩,男,副教授,主要研究领域为分布式计算、分布式系统安全;卢正鼎,男,教授,博士生导师,主要研究领域为分布式系统集成、Web 数据管理、信息安全、数据库系统。
收稿日期:2008-05-20 修回日期:2008-06-23

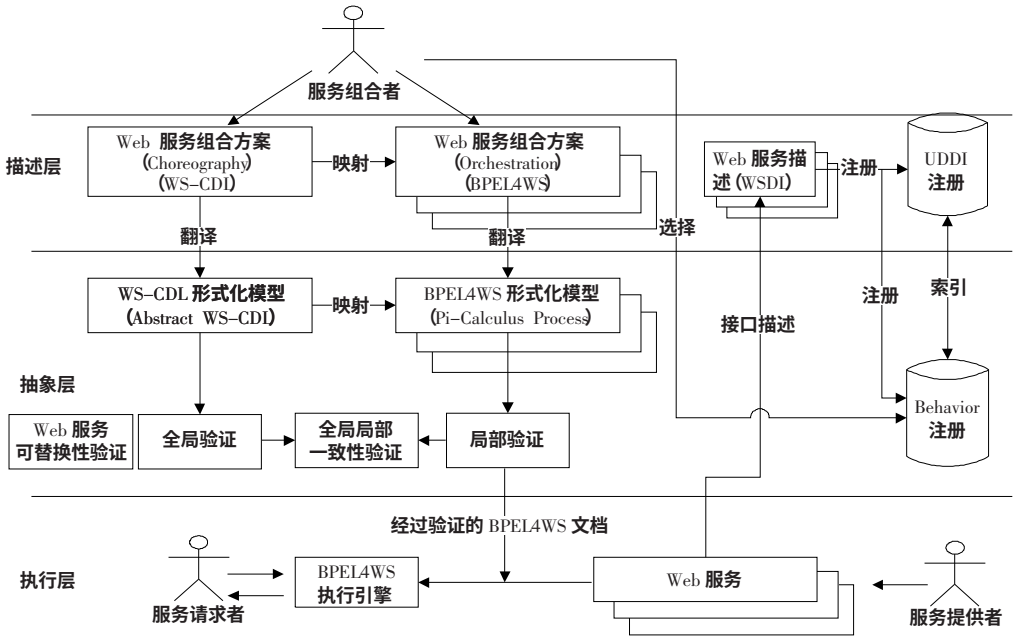

图 1 iFrame4WS 体系结构

务组合设计者用 WS-CDL 或 BPEL4WS 规范来定义 Web 服务组合方案，在制定方案时可以查询选择已有的 Web 服务参与组合(根据 Web 服务的 WSDL 接口描述和动态行为描述)。

3.1 Web 服务接口描述及注册

对于每个已有的 Web 服务，其接口由其 WSDL 描述文档中的 portType 元素来描述。例如，以下的 WSDL 文档描述了 Web 服务 WebShop 的接口：

```
<portType name="WebShop">
    <operation name="ReceiveOrder">
        <input message="Order">
    </operation>
    <operation name="Notification">
        <output message="OrderNotify">
    </operation>
</portType >
```

上面的 WSDL 文档描述了 Web 服务 WebShop 提供的 2 个外部可访问的接口：ReceiveOrder 和 Notification 以及这两个接口的输入/输出参数以及参数类型。

所有已有的 Web 服务接口描述通过自定义的 Register API，将接口作为 UDDI 的 tMode 元素被注册到一个本地的 UDDI 注册中心。一旦被注册,该 Web 服务接口就被赋予一个唯一的 UUID(称为 tModeKey)作为其索引,并且可以被任何已授权的用户检索和引用。

Web 服务组合的设计者在定义一个 Web 服务组合方案前,可以通过 UDDI 注册中心查询、选择已有的 Web 服务参与组合。通过 UDDI 中心，设计者可以得到已有 Web 服务的 WSDL 接口描述，通过 WSDL 接口描述和 Web 服务动态行为的一一对应关系(一一对应关系的建立在 4.4 节描述)，设计者可以得到已有 Web 服务的动态行为描述。

3.2 基于 Choreography 和 Orchestration 的 Web 服务组合描述

Web 服务组合的设计者首先以 WS-CDL 规范定义一个基于 Choreography 的 Web 服务组合方案。基于 Choreography 的 Web 服务组合方案定义了参与 Web 服务组合的各个角色(即各个子 Web 服务)以及角色间的交互行为。基于 Choreography 的 Web 服务组合方案由 WS-CDL 文档的根元素 pakcge 来描述,各个角色间的会话以 Choreography 元素来描述。

当一个基于 WS-CDL 的 Web 服务组合方案被定义好后，通过从 WS-CDL 到 BPEL4WS 的映射可以得到每个角色的基于 Orchestration 的局部行为描述。如果基于 WS-CDL 的全局组合方案里规定了 n 个参与角色，那 WS-CDL 文档被映射分解为 n 个 BPEL4WS 文档，这 n 个 BPEL4WS 文档描述了基于 Orchestration 的 Web 服务组合方案。

3.3 从 WS-CDL 到 BPEL4WS 的映射

从 WS-CDL 到 BPEL4WS 的映射指将定义基于 Choreography 的 Web 服务组合方案的 WS-CDL 文档映射为多个 BPEL4WS 文档，其中每个 BPEL4WS 文档描述了每个角色的局部行为。J.Mendling 和 M.Hafner[4]给出了从 WS-CDL 到 BPEL4WS 的映射规则并且实现了一个转换原型系统，但是这种转换是直接在 WS-CDL 和 BPEL4WS 之间进行,缺乏形式化的验证机制。表 1 为 WS-CDL 主要元素到 BPEL4WS 主要元素的映射关系。

表 1 WS-CDL 主要元素到 BPEL4WS 主要元素的映射

WS-CDL 元素	BPEL4WS 元素
relationshipType	partnerLinkType
roleType	portType
channelType	operation
variableDefinitions	variables
sequence	sequence
parallel	flow
choice	switch,pick
workUnit	switch+while
request Interaction	invoke (fromRole) receive (toRole)
respond Interaction	receive (fromRole) reply (toRole)
request-respond Interaction	invoke+receive (fromRole) receive+reply (toRole)

4 框架抽象层

iFrame4WS 的抽象层的主要功能有：将用户提出的基于 WS-CDL 的 Web 服务组合的方案转换成基于 Abstract WS-CDL 的形式化描述；将基于 BPEL4WS 的 Web 服务组合的方案转换成基于 Pi-演算的 BPEL4WS 形式化描述；Abstract WS-CDL 形式化描述到 BPEL4WS 形式化描述的映射；对 Abstract WS-CDL 形式化描述的验证、对 BPEL4WS 形式化描述的验证以及 Choreography 和 Orchestration 的一致性验证；Web 服务的可替换性验证；同时 iFrame4WS 的抽象层还能够将已有的 Web 服务的动态行为注册和保存在 Behavior 注册中心。

4.1 Web Services Choreography 的形式化描述和验证

在文献[5]中，针对 WS-CDL 提出了一个形式化模型 Abstract WS-CDL。利用 Abstract WS-CDL，Web 服务之间的全局交互行为可以用 *Session* S 来形式化描述，S 的定义如下：

$S::=S_{atom}|S_1+S_2|S_1||S_2|S_1.S_2|[P_g][P_{rep}]^*S$

$S_{atom}::=S_{req}|S_{resp}|S_{req-resp}|NULL$

$S_{req}::=req(r_1,r_2,ch@r_2.o,r_1[x]\rightarrow r_2[y])$

$S_{resp}::=resp(r_1,r_2,ch@r_1.o,r_1[x]\rightarrow r_2[y])$

$S_{req-resp}::=req\text{-}resp(r_1,r_2,ch@r_2.o,r_1[x]\rightarrow r_2[y],r_1[s]\rightarrow r_2[t])$

其中 S_1+S_2、$S_1||S_2$、$S_1.S_2$ 分别表示 *Session* S_1、S_2 的选择、并行和顺序执行。S_{req}、S_{resp} 和 $S_{req-resp}$ 描述了 Web 服务之间最基本的三类交互行为：S_{req} 指角色 r_1 通过通道 ch 向角色 r_2 发出一次请求，角色 r_1 上变量 x 的值 $r_1[x]$ 被发送到角色上 r_2 的变量 $r_2[y]$；S_{resp} 指角色 r_2 通过通道 ch 向角色 r_1 的一次应答；$S_{req-resp}$ 指角色 r_1 和角色 r_2 通过通道 ch 发生的一次请求-应答。$[P_g][P_{rep}]^*S$ 指会话 S 的迭代执行。*NULL* 代表不执行任何交互动作的会话。WS-CDL 规范中的主要交互行为和 Abstract WS-CDL 会话间的对应关系如表 2 所示，其中 $|A\rangle_{acdl}$ 表示一个由 WS-CDL 规范所定义的交互行为 A 所对应的 Abstract WS-CDL Session。

表 2 WS-CDL 主要交互行为的形式化描述

WS-CDL 交互行为	Abstract WS-CDL Session
Request-Respond	$req\text{-}resp(r_1,r_2,ch@r_2.o,r_1[x]\rightarrow r_2[y],r_1[s]\leftarrow r_2[t])$
Request	$req(r_1,r_2,ch@r_2.o,r_1[x]\rightarrow r_2[y])$
Respond	$resp(r_1,r_2,ch@r_1.o,r_1[x]\leftarrow r_2[y])$
Sequence $(A_1,\cdots,A_n)$	$\|A_1\rangle_{acdl}\cdot\cdots\cdot\|A_n\rangle_{acdl}$
Choice $(A_1,\cdots,A_n)$	$\|A_1\rangle_{acdl}+\cdots+\|A_n\rangle_{acdl}$
WorkUnit (g,r,A)	$[P_g][P_{rep}]^*\|A\rangle_{acdl}$
Parallel $(A_1,\cdots,A_n)$	$\|A_1\rangle_{acdl}\|\|\cdots\|\|A_n\rangle_{acdl}$

Abstract WS-CDL 的操作语义由形如 $S\xrightarrow{p,\alpha}S'$ 的符号变迁系统来描述，$S\xrightarrow{p,\alpha}S'$ 指当命题 p 为真同时会话 S 执行一个原子不可分的基本会话 α 后其行为表现为会话 S'（即会话 S 变迁为会话 S'），其中 $\alpha=S_{atom}$。例如，原子会话 S_{req} 执行以后变迁为会话 *NULL*，用操作语义可描述为：$S_{req}\xrightarrow{true,S_{req}}NULL$。Abstract WS-CDL 操作语义包含的主要推演规则有：

$$\frac{}{\alpha\xrightarrow{true,\alpha}NULL}\ (\text{ATOM})\qquad \frac{S\xrightarrow{p,\alpha}S'}{S.T\xrightarrow{true,\alpha}S'.T}\ (\text{SEQ})$$

$$\frac{S\xrightarrow{p,\alpha}S'}{S||T\xrightarrow{p,\alpha}S'||T}\ (\text{PARA})\qquad \frac{S_1\xrightarrow{p,\alpha}S_1'}{S_1+S_2\xrightarrow{p,\alpha}S_1'}\ (\text{CHOICE})$$

每条规则的横线上方为前提，下方为结论。利用 Abstract WS-CDL 定义的操作语义，可以对形式化描述的 Web Services Choreography 的执行过程进行形式化推演和验证。由于篇幅所限，这里不详细列出推演过程。

4.2 Web Services Orchestration 的形式化描述和验证

Web Services Orchestration 从一个 Web 服务组合的参与者（即某一个 Web 服务）的角度来看该 Web 服务是如何与其他参与组合的 Web 服务进行交互，即 Orchestration 描述了 Web 服务的局部交互行为。在这里采用进程代数 Pi-演算[6]来形式化描述一个 Web 服务的局部行为，一个 Pi 演算的进程 P 可以定义如下：

$P::=\mathbf{0}|\bar{a}x.P|a(x).P|\tau.P|P_1+P_2|[x=y]P|(\nu x)P|!P|P_1|P_2$

其中输出前缀 $\bar{a}(x)$ 表示名字 x 沿着通道 a 输出；输入前缀 $a(x)$ 表示从通道 a 接受一个名字；前缀 τ 表示一个进程外部不可见的内部动作；$\mathbf{0}$ 表示一个结束了的进程；P_1+P_2 表示进程的选择执行；$P_1|P_2$ 表示进程的并发执行；$[x=y]P$ 表示当名字 $x=y$ 时，进程 P 被执行；$!P$ 表示 P 被重复复制无穷多次；$(\nu x)P$ 表示名字 x 被局限在进程 P 内部。

利用 Pi-演算对 Web 服务的局部行为进行形式化描述时，Web 服务的局部行为被描述为 Pi-演算的进程；而 Web 服务之间的交互可以被抽象为彼此之间沿着某一通道发生的输入输出动作，通道则是 Web 服务 WSDL 接口中的 Operation 的抽象。Web Services Orchestration 的主要规范 BPEL4WS 中的行为的 Pi-演算描述如表 3 所示，其中 $|A\rangle_{PI}$ 表示一个 BPEL4WS 规范所定义的交互行为 A 所对应的 Pi-演算进程描述。

表 3 BPEL4WSL 主要交互行为的形式化描述

BPEL4WS 行为	对应的 Pi-演算进程
invoke	$\overline{cho}\ inputVariable.chi(outputVariable).\mathbf{0}$
receive	$chi(variable).\mathbf{0}$
reply	$\overline{cho}\ variable.\mathbf{0}$
Sequence $(A_1,\cdots,A_n)$	$\|Sequence\rangle_{PI}=\|A_1\rangle_{PI}\cdot\cdots\cdot\|A_n\rangle_{PI}$
Switch (S_N,S_A,D) $S_N=\{N_1,\cdots,N_n\}$ $S_A=\{A_1,\cdots,A_n\}$	$\|Switch\rangle_{PI}=\sum_{i=1}^{n}[x=N_i]\|A_i\rangle_{PI}+[x\neq N_1]\cdots[x\neq N_n]\|D\rangle_{PI}$
While (N,A)	$\|While\rangle_{PI}=[x=N]\|A\rangle_{PI}\cdot\|While\rangle_{PI}+[x\neq N].\mathbf{0}$
Pick (S_E,n) $S_E=\{(che_i,var_i,A)\ \|i=1,n\}$	$\|Pick\rangle_{PI}=\sum_{i=1}^{n}che_i(var_i)\cdot\|A_i\rangle_{PI}$
Flow $(A_1,\cdots,A_n)$	$\|Flow\rangle_{PI}=\|A_1\rangle_{PI}\|\cdots\|\|A_n\rangle_{PI}$

Pi-演算的操作语义由形如 $P\xrightarrow{\alpha}Q$ 的符号变迁系统来描述，$P\xrightarrow{\alpha}Q$ 表示表示进程 P 经过动作 α 变迁（演化）成为进程 Q，其中动作 α 可以为内部哑动作 τ、输入动作和输出动作。利用 Pi-演算的操作语义，可以对 Pi-演算进程的执行过程进行推演和验证。Pi-演算的验证还可采用基于 Pi-演算描述的并发系统的验证工具 Mobility Workbench[7]（MWB）来进行。利用该工具可以对一个基于 Pi-演算描述的 Web 服务组合进行验证以排查出可能存在的错误如死锁，可以跟踪服务的行为以及验证服务的等价性。

4.3 Choreography 到 Orchestration 的映射

正如在描述层可以将定义 Web 服务组合全局交互行为的 WS-CDL 文档映射为多个描述了每个角色的局部行为 BPEL4WS 文档一样，可以在抽象层将 Abstract WS-CDL 的会话 S 映射到参与该会话的每个角色上，得到描述每个角色局部行为的 Pi-演算进程。一个会话 S 对角色 r 的映射记为 $|S\rangle_r$，

即$|S\rangle_r$为角色 r 所对应的 Pi-演算进程。映射的主要规则为：

(1) $|S_{req}\rangle_r=\overline{ch}x$ (if $r=r_1$) $|S_{req}\rangle_r=ch(y)$ (if $r=r_2$) $|S_{req}\rangle_r=skip$ (otherwise)

(2) $|S_{resp}\rangle_r=ch(x)$ (if $r=r_1$) $|S_{resp}\rangle_r=\overline{ch}y$ (if $r=r_2$) $|S_{resp}\rangle_r=skip$ (otherwise)

(3) $|S_{req-resp}\rangle_r=\overline{ch}1x.ch2(y)$ (if $r=r_1$) $|S_{req-resp}\rangle_r=ch1(y).\overline{ch}2t$ (if $r=r_2$) $|S_{req-resp}\rangle_r=skip$ (otherwise)

(4) $|S_1||S_2\rangle_r=|S_1\rangle_r|S_2\rangle_r$

(5) $|S_1.S_2\rangle_r=|S_1\rangle_r.|S_2\rangle_r$

(6) $|S_1+S_2\rangle_r=|S_1\rangle_r+|S_2\rangle_r$

通过从 Choreography 到 Orchestration 映射,可以在 Choreography 和 Orchestration 二个不同的层次来对 Web 服务组合的设计方案进行形式化验证以排除设计过程的错误。除此之外,还必须对 Choreography 和 Orchestration 的一致性进行形式化验证,一致性的验证见本文 4.5 节。从 Choreography 到 Orchestration 映射的其他规则见文献[5]。

4.4 Web 服务动态行为的形式化描述和注册

WSDL 描述了一个 Web 服务的接口。如 3.1 节给出的 WSDL 文档就描述了 Web 服务 WebShop 的接口具有两个操作:*ReceiveOrder* 和 *Notification*。但是 WSDL 规范仅仅给出了这两个操作的输入/输出参数及类型,它无法描述 *WebShop* 的动态行为,即 *ReceiveOrder* 和 *Notification* 的调用顺序。如果一个客户首先调用操作 *Notification* 就可能会导致客户和 *WebShop* 之间的死锁。因此,为了保证 Web 服务组合的设计者能够选择动态行为相匹配的服务参与组合以及寻找一个动态行为相匹配的新的服务来替换已有的服务(见 4.6 节),必须提供一种机制来描述、注册保存、检索 Web 服务的动态行为。

Web 服务的动态行为用 Pi-演算来描述,如果 *WebShop* 的 *ReceiveOrder* 操作必须先于 *Notification* 被调用,则 *WebShop* 的动态行为可以被 Pi-演算进程描述为：

$WebShop\_ReceiveOrder(Order).\overline{WebShop\_Notification}Notify.\mathbf{0}$

其中通道名的命名规则为 *portTypeName_operationName*。

在给出了 Web 服务的动态行为描述后,可以将 Web 服务的动态行为描述保存在一个本地的数据库里(称为 Behavior 注册中心),但必须要解决如何将 Web 服务的动态行为和注册在 UDDI 注册中心里的 WSDL 接口描述一一对应的问题。解决办法是将 UDDI API 加以封装,形成一个自定义的 Register API。当将一个 Web 服务的 WSDL 描述作为一个 tMode 元素注册到 UDDI 注册中心时,可以得到唯一标识 tMode 元素的 tModeKey,接着将 tModeKey 作为 Web 服务的动态行为描述的主键将 Web 服务的动态行为描述保存在数据库中,这样通过 tModeKey 就建立了 Web 服务的动态行为和 WSDL 接口描述的一一对应关系,这个过程如图 2 所示。

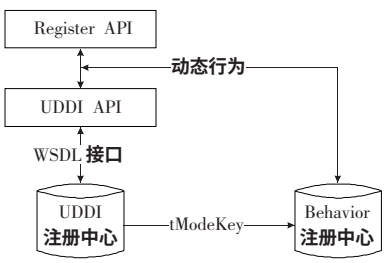

图 2 封装 UDDI API

4.5 Choreography 和 Orchestration 一致性的形式化描述和验证

Choreography 和 Orchestration 分别从全局和局部两个层次描述了 Web 服务的组合,当用户用 WS-CDL 和 BPEL4WS 规范分别定义了 Web 服务组合的全局方案和局部方案后,必须保证两个层次的组合方案是一致的。因此,给出如下相关定义和一致性的判定准则。

定义 1 对于一个由 Abstract WS-CDL 描述的基于 Choreography 的会话 S,其对应的 Orchestration 记为 $Orch_s$,且 $Orch_s=\prod P_r(r\in S[R])$。其中:$S[R]$为参与会话 S 的角色的集合,P_r 为参与会话 S 的角色 r 对应的 Pi 演算进程,$\prod$ 代表多个 Pi-演算进程的"|"运算。

定义 2 如果全局会话 S 和其对应的局部行为 $Orch_S$ 是一致的,则记为 $S\Leftrightarrow Orch_S$。

定义 3 对于一个由 Abstract WS-CDL 描述的一个全局会话 S,如果其对应的局部行为 $Orch_S=\prod|S\rangle_r(r\in S[R])$,则 $S\Leftrightarrow Orch_S$。其中,$|S\rangle_r$ 为参与会话 S 的角色 r 由全局会话 S 分解得到的局部行为的 Pi-演算进程描述。

在定义 3 的基础上,利用 Pi-演算理论中进程的动态行为的等价性质和互相似理论来分析如何判定 Web 服务组合的全局交互行为和局部行为的一致性。互相似和动态行为的等价是 Pi-演算为了证明不同并发系统的行为等价性而建立的一套概念和定理。互相似是指从系统外观察者的角度来看不同并发系统的行为的等价性。在 Pi-演算理论里,进程 P 和 Q 是互相似的蕴涵着当进程 $P\xrightarrow{\alpha}P'$,则必定存在进程 Q',使得 $Q\xrightarrow{\alpha}Q'$,且 P'和 Q'仍然是互相似的。如果 P 和 Q 是互相似的,则记为 $P\dot{\sim}Q$。

定义 4 全局会话 S 和其对应的局部行为 $Orch_S$ 是一致的,即 $S\Leftrightarrow Orch_S$,如果 $Orch_s=\prod P_r(r\in S[R])$,且 $\forall r\in S[R]$,有 $P_r\dot{\sim}|S\rangle_r$。其中,P_r 为描述参与会话 S 的角色 r 的局部行为的 Pi 演算进程,$|S\rangle_r$ 为全局会话 S 对角色 r 作映射而得到的 Pi-演算进程。

4.6 Web 服务可替换性的形式化验证

本节中的 Web 服务都由一个 Pi-演算进程来描述。Web 服务的可替换性是指:如果一个组合的 Web 服务 C 由 N 个并发的 Web 子服务$\{P_1,P_2,\cdots,P_n\}$组合而成,即 $C=\prod P_j(j=1-n)$。如果其中一个 Web 服务(假设为 P_1)由于某种原因不能正常工作,需要寻找另一个 Web 服务 P_1' 来替代 P_1 或者 Web 子服务 P_1 本身版本升级到 P_1',那么这时必须要保证组合的 Web 服务 C 仍然能正确地工作,并且对客户完全透明。

显然,如果组合的 Web 服务 C 能正确地工作,意味着 N 个并发的 Web 子服务$\{P_1,P_2,\cdots,P_n\}$是互相容的。因此,如果 Web 子服务 P_1 被替换为 P_1',则必须要保证$\{P_1',P_2,\cdots,P_n\}$仍然是互相容的。在文献[8]中,给出了 Pi-演算进程间的投影操作,并在此基础上给出了 Web 服务互相容的形式化定义。将 Web 服务 P 对 Q 的投影操作记为$|P\rangle_Q$,即$|P\rangle_Q$为 P 对 Q 进行投影操作后得到的 Pi-演算进程,同时 Web 服务 P 和 Q 互相容记为 $P\tilde{\nabla}\triangle$

Q。在文献[8]的工作基础上给出如下 Web 服务可替换性的判断准侧：

定义 5 如果组合的 Web 服务 C 由 n 个并发的 Web 子服务 $\{P_1,P_2,\cdots,P_n\}$ 组成，即 $C=\prod P_j\ (j=1-n)$，则 Web 子服务 P_i $(1\leqslant i\leqslant n)$ 可以被 Web 服务 Q 替换，当 Q 满足：对 $\forall P_j\in\{P_m|1\leqslant m\leqslant n,m\neq i\}$，有 $|Q\rangle_{P_j}\tilde{\nabla}_{\triangle}|P_j\rangle_Q$。

定义 6 如果组合的 Web 服务 C 由 n 个并发的 Web 子服务 $\{P_1,P_2,\cdots,P_n\}$ 组成，即 $C=\prod P_j\ (j=1-n)$，则 Web 子服务 P_i $(1\leqslant i\leqslant n)$ 可以被 Web 服务 Q 替换，当 Web 子服务 P_i $(1\leqslant i\leqslant n)$ 被 Web 服务 Q 替换而得到的 Web 服务集合 $S=\{P_m|1\leqslant m\leqslant n,m\neq i\}\cup\{Q\}$ 满足：

$Q\tilde{\nabla}_{\triangle}\prod P_j\ (j\neq i,P_j\in S)$ 且 $\forall P_m\in S-\{Q\}$ 有 $P_m\tilde{\nabla}_{\triangle}Q|\prod P_n\ (n\neq m,P_n\in S-\{Q\})$

定义 7 如果组合的 Web 服务 C 由 n 个并发的 Web 子服务 $\{P_1,P_2,\cdots,P_n\}$ 组成，即 $C=\prod P_j\ (j=1-n)$，则 Web 子服务 P_i $(1\leqslant i\leqslant n)$ 可以被 Web 服务 Q 替换，当 Web 子服务 P_i $(1\leqslant i\leqslant n)$ 被 Web 服务 Q 替换而得到的 Web 服务集合 $S=\{P_m|1\leqslant m\leqslant n,m\neq i\}\cup\{Q\}$ 满足：$Q|\prod P_j\ (P_j\in S-\{Q\})$ 最终会变迁为 0 且只能通过一系列内部动作变迁为 0。

5 框架描执行层

当抽象层所有的验证通过后，描述每个角色局部行为的 BPEL4WS 文档被一个 BPEL4WS 执行引擎加载并运行。BPEL4WS 执行引擎采用 BPWS4J－一个基于 Java 的执行 BPEL4WS 流程的引擎。对于每个 BPEL4WS 流程，BPWS4J 将自动加载描述该流程的 BPEL4WS 文档，同时，该流程在执行过程中需要调用的其他 Web 服务（该流程的 partner）的 WSDL 文档也会被动态加载以调用相应的服务。

6 相关工作

V.De Antonellis，M.Melchiori，B.Pernici，et al[9]提出了一个 Web 服务协作的虚拟环境，在该环境中，利用形式化的方法来描述 Web 服务的接口，用本体描述语言 DAML-S 来描述 Web 服务的动态行为，并在此基础上来验证 Web 服务组合以及 Web 服务的可替换性；W.L Yeung，J.Wang，W.Dong[10]利用进程代数 CSP 来形式化描述和验证 Web Services Choreography；M. Koshkina，F.van Breugel[11]提出了一个自定义的进程代数 BPE-Calculus，将 BPEL4WS 定义的商业流程翻译成 BPE-Calculus 进程，并利用扩展的 CCS 自动验证工具 CWB-NC 对 BPE-Calculus 进程进行验证；饶元、冯博琴、李尊朝[12]从软件体系结构的角度出发，对 Web 服务及组合给出了形式化的描述和分析，提出了一种基于体系结构生命周期的Web 服务合成框架。

7 结束语

提出了一个 Web 服务组合的设计和验证框架iFrame4WS。在该框架中，Web 服务组合设计者定义的 Web 服务组合方案在描述层以 WS-CDL 和 BPEL4WS 来定义；在抽象层则分别从 Chreography 和 Oechestration 两个不同的角度以 Abstract WS-CDL Session 和 Pi-演算进程来形式化描述和验证，并通过 Abstract WS-CDL Session 和 Pi-演算进程之间的映射规则和一致性的形式化定义来保证全局行为和局部行为的一致性；对已有 Web 服务的选择和替换则可以通过对 Web 服务动态行为的描述以及 Web 服务可替换性的判断准则来完成；通过验证的 Web 服务组合方案通过 iFrame4WS 的执行层的 BPEL4WS 执行引擎来执行。

参考文献：

[1] Bucchiarone A，Gnesi S.A survey on services composition languages and models [C]//Proceedings of the International Workshop on Web Services Modeling and Testing，2006：51-63.

[2] Andrews T，Cubera F，Dholakia H，et al.Business process execution language for Web services version1.1[EB/OL].[2005-05].http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[3] Kavantzas N，Burdett D，Rizinger G，et al.Web service choreography description language version 1.0[EB/OL].[2005-11].http://www.w3.org/TR/ws-cdl-10/.

[4] Mendling J，Hafner M.From WS-CDL choreography to BPEL process orchestration[EB/OL].[2005-11].http://wi.wu-wien.ac.at/home/mendling/publications/TR06-CDL.pdf.

[5] 辜希武，卢正鼎.Web 服务编排描述语言 WS-CDL 的形式化模型框架[J].计算机科学，2007，34(9)：5-11.

[6] Milner R，Parrow J，Walker D.A calculus of mobile process (Parts I and II) [J].Information and Computation，1992，100：1-77.

[7] Victor B，Moller F.The mobility workbench-a tool for the Pi-calculus[C]//LNCS 818：Proceedings of the 6th International Conference on Computer Aided Verification，California，USA，1994.London，UK：Springer-Verlag，1994：428-440.

[8] 辜希武，卢正鼎.Web 服务相容性的形式化描述和分析[J].计算机工程与应用，2007，43(27)：28-33.

[9] de Antonellis V，Melchiori M，Pernici B，et al.A methodology for e-service substitutability in a virtual district environment[C]//Eder J，Missikoff M.LNCS 2681：Proceedings of the 15th International Conference on Advanced Information Systems Engineering，Klagenfurt，Austria，2003.London，UK：Springer-Verlag，2003：552-567.

[10] Yeung W L，Wang J，Dong W.Verifying choreographic descriptions of Web services based on CSP[C]//Proceedings of the IEEE Services Computing Workshops (SCW'06).Washington D C，USA：IEEE Computer Society Press，2006：97-104.

[11] Koshkina M，van Breugel F.Modelling and verifying Web service orchestration by means of the concurrency workbench [J].ACM SIGSOFT Software Engineering Notes，2004，29(5)：1-10.

[12] 饶元，冯博琴，李尊朝.ALBC4WS：一种基于软件体系结构生命周期的动态服务合成框架[J].计算机研究与发展，2005，42(12)：2063-2069.