

# 基于 Pi-演算的 Web 服务组合的描述和验证

廖 军 谭 浩 刘锦德

(电子科技大学计算机科学与工程学院 成都 610054)

**摘 要** 形式化方法对于建模和验证软件系统是一种有效的方法,所以对 Web 服务的形式化描述和验证是一个重要的研究方向.对于 Web 服务及其组合来说,保证其组合正确性以实现其服务增值是十分必要的. Pi-演算是一种移动进程代数,可用于对并发和动态变化的系统进行建模. 该文基于 Pi-演算对 Web 服务及其组合进行形式化描述和建模. 文中说明了 Pi-演算与以前形式化方法的不同之处,分析了 Pi-演算应用于 Web 服务组合需要解决的问题. 讨论了 Pi-演算与 Web 服务协议栈的对应关系,说明了利用 Pi-演算建立 Web 服务组合模型的规则,指出了如何寻找代理和通道. 最后建立了一个实际的模型,并利用形式化工具对建立的组合模型是否正确以及是否满足需求进行了验证.

**关键词** Pi-演算;进程代数;Web 服务;服务组合;服务形式化

**中图法分类号** TP311

## Describing and Verifying Web Service Using Pi-Calculus

LIAO Jun TAN Hao LIU Jin-De

(College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

**Abstract** Using formal method is an effective methodology for modeling and verifying software system. Describing and verifying Web services by formal method is an important research. Guaranteeing the validity of Web services composition is necessary for enhancing the value of services. Pi-calculus is a kind of mobile process algebra which can be used to model concurrent and dynamic systems. Web services and their composition are described and modeled based on Pi-calculus in this paper. Some difference among the Pi-calculus and other formal methods is discussed. Rules about applying Pi-calculus to Web services are explained and the method of working out agents and channels is proposed. Relationship between Pi-calculus and Web services protocol stack is illustrated too. Finally, a demo is constructed and the validity of composition model is verified. Compared to process algebra method based on CCS, the Pi-Calculus can be used for describing and reasoning dynamic system and appropriate for modeling Web services composition.

**Keywords** Pi-calculus; process algebra; Web service; service composition; formal service

## 1 引 言

近年来 Web 服务的理论和技术取得了长足的

发展,其保证互操作性的协议栈下层在学术界和工业界已基本达成一致.作为新一代的开放系统技术,Web 服务与 CORBA、Java 等技术相比,在保证互操作上更为成功.从技术上说,Web 服务的价值在于

收稿日期:2004-12-06;修改稿收到日期:2005-01-31. 本课题得到“十五”国防预研项目基金(41315010103)资助.廖 军,男,1977 年生,博士研究生,主要研究方向为 Web 服务、开放系统和中间件技术. E-mail:liao@uestc.edu.cn.谭 浩,男,1970 年生,博士,副教授,主要研究方向为中间件和 Web 服务、连续流媒体.刘锦德,男,1930 年生,教授,博士生导师,主要研究领域为开放分布式系统、中间件和 Web 服务、普适计算以及信息安全.

服务重用,而重用的目的则是使服务增值.至于保证服务通信和重用的基础则是互操作协议栈的下层一致性.除了服务本身的重用外,组合不同的 Web 服务以产生不同于各个单一 Web 服务的功能,也是研究的重点之一. Web 服务组合是指:由各个小粒度的 Web 服务相互之间通信和协作来实现大粒度的服务功能;通过有效地联合各种不同功能的 Web 服务,服务开发者可以借此解决较为复杂的问题,实现增值功能.比如常见的旅行预定服务、订单服务等,都是组合已有的组织内和组织外的服务来实现单个服务所不能完成的复杂功能.

但是,如何使得 Web 服务真正进入实用的阶段,使得 Web 服务实现跨组织、跨管理域的系统集成和自动交互,还面临着诸多的问题;其中一些问题是在传统的中间件应用中已经解决了的,而另外一些则是新问题<sup>[1]</sup>,比如 Web 服务如何组合、能否自动组合以及组合的正确性验证,即是否与期望中的服务等价等问题.为了解决服务组合的这些问题,学术界和工业界提出了多种方法;总体来说,其思路可以分为两类:一类是用流程组合服务的思想,比如 BPEL4WS<sup>①</sup>,这类方法的建模和验证常使用 Petri 网和进程代数<sup>[2]</sup>等形式化方法;另一类是借助人工智能的思想,如语义 Web<sup>[3]</sup>,常基于时序逻辑等方法.由于 Web 服务是在整个 Internet 上进行交互和组合,其面对的组合对象和环境(如连接状况等)复杂多变,同时鉴于专门用于描述移动进程的 Pi-演算<sup>[4,5]</sup>在描述这类问题上具有一定的优势,本文将主要讨论前一类方法,并使用基于 Pi-演算的方法.

本文第 2 节讨论了 Pi-演算的相关工作;第 3 节用 Pi-演算对 Web 服务组合进行描述,讨论了如何描述服务的动态变化问题,并利用得到的抽象模型对组合的正确性进行了验证;第 4 节介绍了相关工作;最后进行全文总结并说明了下一步的工作.

## 2 Pi-演算

Pi-演算是 Robin Milner 提出的以进程间的移动通信为研究重点的并发理论,它是对 CCS (Calculus of Communication System) 的发展.其基本计算实体为名字和进程,进程之间的通信是通过传递名字来完成.由于 Pi-演算不但可以传递 CCS 中的变量和值等,还可以传递通道名,并且将这几种实体都统称为名字而不再作区分,这使得 Pi-演算具有了建立新通道的能力,因此 Pi-演算可以用来描述结构

不断变化的并发系统.

围绕 Pi-演算有很多研究工作已展开,大致可以分为两方面:(1)Pi-演算本身的理论研究,如 Pi-演算的互模拟等价性、类型系统以及 Pi-演算的有穷公理化问题等<sup>[4~6]</sup>;(2)Pi-演算作为建模工具在计算机软件系统各研究方向上的应用,如人工智能、分布式组件技术<sup>[7,8]</sup>等.

Pi-演算可由三方面定义,分别为语法定义、操作语义和结构等价规则.

**定义 1.** Pi-演算的语法定义.

设  $N$  为无限名字集,用  $u, v, w, x, y, z$  等小写字母表示名字集上的名字;进程标识符用  $A, B, C$  等大写字母表示;进程表达式用  $P, Q, R$  等大写字母表示,进程表达式为以下 7 种之一:

(1)和式.  $\sum_{i \in I} P_i$ , 其中索引集  $I$  为有限集,使用 0 来表达空进程,即不活动的进程.

如  $P_1 + P_2$  表示选择执行  $P_1$  或  $P_2$ ;

(2)前缀表达式.  $\bar{y}x.P, y(x).P, \tau.P$ .

其中  $\bar{y}x$  称为负前缀,  $\bar{y}x.P$  表示在端口  $y$  上输出名字  $x$ , 然后执行  $P$ ;

$y(x)$  称为正前缀,  $y(z/x).P$  表示在端口  $y$  上输入任意名字  $z$ , 然后执行  $P\{z/x\}$ , 其中  $z/x$  表示用  $z$  替换  $x$ , 名字  $x$  (类似于编程语言中的形式参数)限制在正前缀  $y(x)$  中;

将前缀表达式中连接互补端口  $y$  和  $\bar{y}$  的链接称为通道  $y$ .

$\tau$  称为哑前缀,  $\tau.P$  表示做哑动作  $\tau$  然后执行  $P$ , 一般来说,  $\tau$  用于表示进程外部不可见的内部动作;

(3)并行表达式.  $P_1 | P_2$  表示并行执行  $P_1$  和  $P_2$ ;

(4)限制表达式.  $(x)P$  表示  $P$  在新通道  $x$  上的外部动作被禁止,但  $P$  通过通道  $x$  的内部通信是允许的,它们都成为了  $\tau$  动作,即将  $x$  局限在  $P$  中,另外的表示法包括  $(\nu x)P$  或  $\text{new } c \text{ in } P$ , 即  $\text{let } c = \text{new channel}() \text{ in } P$ ;

(5)匹配表达式.  $[x = y]P$  表示当  $x$  和  $y$  是同一通道名时,其行为与  $P$  相同,否则为 0 进程;

(6)进程标识符.  $A(y_1, y_2, \dots, y_n)$  对每个进程标识符来说,必须有其定义  $A(x_1, x_2, \dots, x_n) ::= P$ , 其中  $x_1, x_2, \dots, x_n$  是  $P$  中的自由名;

(7)重复表达式.  $!P$  表示  $P$  的无穷个复制.

至此,我们将 Pi-演算定义如下:

① <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>

$P ::= 0 \mid \bar{y}x.P \mid y(x).P \mid P +$

$Q \mid P \mid Q(x)P \mid [x=y]P \mid !P \mid A(y_1, y_2, \dots, y_n).$

在此,用  $fn(P)$  表示进程  $P$  的自由名集,  $bn(P)$  表示进程  $P$  的受限名集,则有

$fn(\bar{y}x) = \{y, x\},$

$fn(y(x).P) = \{a\} \cup (fn(P) - \{x\}),$

$fn((x)P) = fn(P) - \{x\}.$

定义 2. Pi-演算的操作语义.

TAU-ACT:  $\frac{}{\tau.P \xrightarrow{\tau} P}$  表示任何情况下,  $\tau.P$

经过  $\tau$  动作成为  $P$ ;

OUTPUT-ACT:  $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$  表示任何情况

下,  $\bar{x}y.P$  经过  $\bar{x}y$  动作成为  $P$ ;

INPUT-ACT:  $\frac{}{x(z).P \xrightarrow{x(y)} P\{y/z\}}, y \notin fn((z)P)$

表示输入  $y$  替代形式参数  $z$ , 则  $P$  中也以  $y$  替换形参  $z$ , 只要  $y$  不在  $(z)P$  的自由名集中;

SUM:  $\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$ , 其中  $\alpha$  为动作, 表示  $x(y)$ ,

$\bar{x}(y)$  或者  $\tau$ , SUM 规则表示若  $P$  可以通过动作  $\alpha$  而成为  $P'$ , 则  $P + Q$  也可成为  $P'$ ;

除以上所列规则外, 还有以下语义规则:

MATCH:  $\frac{P \xrightarrow{\alpha} P'}{[x=x]P \xrightarrow{\alpha} P'}$ ;

PAR:  $\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$   $bn(\alpha) \cap fn(Q) = \emptyset$ ;

COM:  $\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{y/z\}}$ ;

CLOSE:  $\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} (w)P' \mid Q'}$ ;

RES:  $\frac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'}$ ,  $y \notin n(\alpha)$ ;

OPEN:  $\frac{P \xrightarrow{\bar{x}y} P'}{(y)P \xrightarrow{\bar{x}(z)} P'\{z/y\}}$ ,  $y \neq x, z \notin fn((y)P')$ ;

REP:  $\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$ .

定义 3. Pi-演算的结构等价规则.

$P \mid Q \equiv Q \mid P, (P \mid Q) \mid R \equiv P \mid (Q \mid R),$

$P + Q \equiv Q + P, (P + Q) + R \equiv P + (Q + R),$

$(x)0 \equiv 0, (x)(y)P \equiv (y)(x)P,$

$((x)P) \mid Q \equiv (x)(P \mid Q)$  if  $x \notin fn(Q),$

$x(y).P \equiv x(z).(\{z/y\}P)$  if  $z \notin fn(P),$

$(y)P \equiv (z)(\{z/y\}P)$  if  $z \notin fn(P).$

### 3 用 Pi-演算建模 Web 服务组合

#### 3.1 Pi-演算与 Web 服务元素的对应关系

用 Pi-演算来描述 Web 服务及其组合, 有三个问题必须解决: (1) Web 服务的元素如何与 Pi-演算的基本元素对应; (2) Web 服务组合如何表示; (3) 如何用 Pi-演算来形式化地描述 Web 服务组合. 其中第 1 个问题在微软的 BizTalk Server 使用的 XLANG 语言中有所涉及<sup>①</sup>, XLANG 是用于描述 Web 服务组合的一种语言, 它是基于 Pi-演算模型的. 在 XLANG 中, Pi-演算基本元素和 WSDL 元素的对应规则如表 1 所示.

表 1 XLANG 中 Web 服务与 Pi-演算元素对应关系

Web 服务	Pi-演算
Type	Sort
Message	Message
Operation	Action
PortType	Sort
Binding	Interaction
Port	Port
Service	Process

在我们的推演中, 参考表 1 中的对应关系, 并结合演算建模并发通信系统的常用方法, 用两种图来描述一个 Web 服务(及其组合): (1) 流图(flow graph); (2) 变迁图(transition graph). 其中流图描述服务(进程)之间的连接关系以及它们与外界的连接点, 而变迁图则表示服务状态如何进行变化.

第 2 个问题涉及 Web 服务的组合描述语言的问题, 正如引言中已提到的, 现有的 Web 服务组合描述语言主要有两类: (1) 人工智能的方法, 如语义 Web; (2) 基于流程的方法, 如 BPML, WSCI, BPEL4WS (源自 WSFL, WSCI 和 XLANG) 等. 虽然 W3C 推出了 Web 服务协作描述语言 WS-CDL<sup>②</sup> (Web Service Choreography Description Language), 但它主要是用于流程组合层之上的协作描述层的, 它和其它规范的关系如图 1 所示(其中右边虚线椭圆框中的部分为基于语义 Web 的方法).

① [http://www.gotdotnet.com/team/xml\\_wsspecs/xlang-c](http://www.gotdotnet.com/team/xml_wsspecs/xlang-c)

② <http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/>

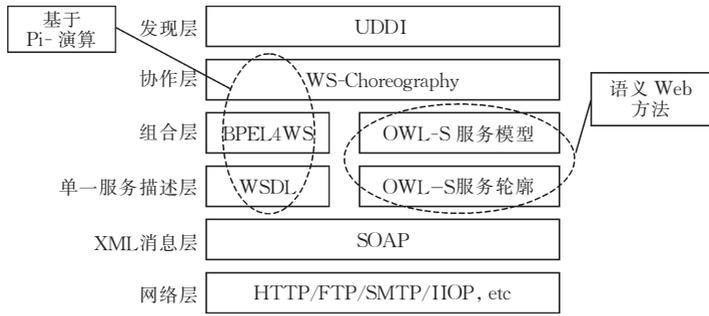


图 1 Web 服务组合与协作协议栈

实际上, Web 服务协作描述语言 WS-CDL 就是基于 Pi-演算的, 我们的目标是使得下层的服务组合和 WSDL 都能够与 Pi-演算结合起来, 这样从下至上就有了统一的形式化基础, 便于从上到下的正确性和一致性验证。

本文中, 主要是采用基于流程描述 Web 服务的方法, 讨论它如何与 Pi-演算对应和结合起来。至此, 重点集中到第 1 个和第 3 个问题上, 即将 Web 服务和 Web 服务的组合分别用 Pi-演算表示出来。

首先, 将 Web 服务表示为 Pi-演算的进程。在此, Pi-演算的进程并不对应于计算机程序中的进程, 而是泛指对应一个服务; 但是, 由于 Pi-演算把实体统称为名字, 所以其进程还可以对应成值。为了不引起混淆, 下文一般将进程认为是某个外部或内部服务。其基本对应关系如图 2 的流图所示。

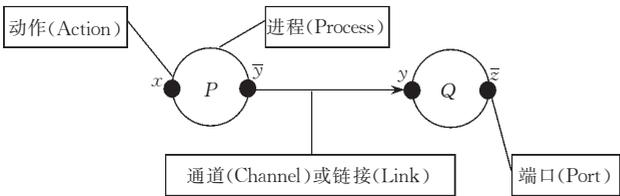


图 2 Pi-演算流图

这是一个最简单的交互动作,  $P$  进程沿着通道发出消息,  $Q$  进程从通道接收消息。通常我们把通道名和动作名作为同一含义使用, 如: 动作  $y$  发出消息, 即理解为  $P$  沿着通道  $y$  发出消息, 而  $Q$  在同一通道  $y$  上接收消息。

### 3.2 机票预定服务的描述

即使相对简单的服务也可能包含复杂的交互, 我们以一个机票预定服务为例具体说明这些问题。因为重点是讨论如何建模 Web 服务及其组合, 所以我们的机票预定服务适当作了一些简化, 省略了部分细节问题, 其逻辑结构如图 3 所示。

机票预定服务接受客户代理的请求, 根据机票数据, 确定是否能够满足客户的请求。当确认后, 再向银行发出请求, 若银行确认用户已付帐, 则预定服务发送确认信息给客户代理。

其中各消息表示如下(为了简化后面的表示, 在此给每个消息都定义一个缩写):

Request: 客户代理的订票请求 (Req);

AskInfo: 订票服务详细询问客户代理的需求 (Ask);

ProvInfo: 客户代理将客户的具体需求返回给订票服务, 如始发地、目的地、起飞时间等信息

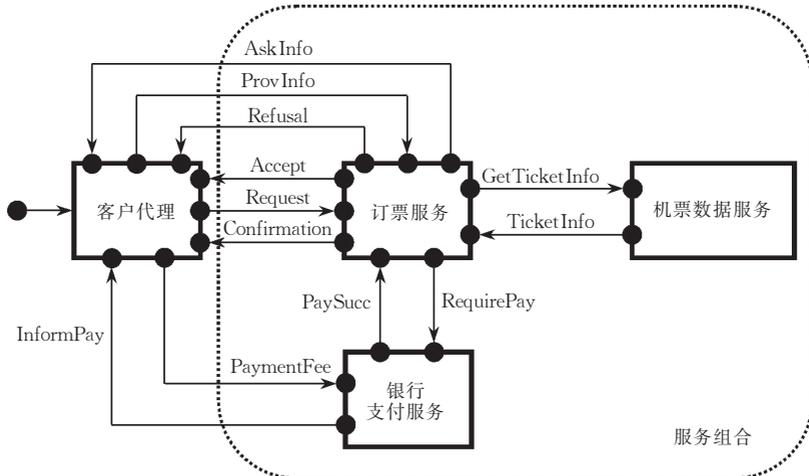


图 3 机票预定的 Web 服务

(Pro);

Refusal: 订票服务根据客户代理提供的信息和机票数据返回的信息,发现不能满足客户需求时,发出无法继续服务的拒绝消息(Ref);

Accept: 能够满足客户需求,订票服务接受了订票请求(Acc);

GetTicketInfo: 订票服务向机票数据服务请求当前可获得的航班数据(Get);

TicketInfo: 机票数据服务返回相关数据(Tic);

RequirePay: 订票服务向银行服务要求付款(假设先付款后送票,而且通过客户的银行帐号直接进行)(ReP);

InformPay: 银行支付服务向客户代理发出付款请求(Inf);

PaymentFee: 客户代理付款并确认(Pay);

PaySucc: 银行支付服务通知订票服务,客户已付款(PaS);

Confirmation: 订票服务向客户发出确认信息,订票成功(Con).

Pi-演算是表示这种复杂行为的有效方式,可以清楚地表示出系统行为.但是将 Web 服务组合的逻辑关系映射成 Pi-演算的表达式时,需要解决两个问题:(1)如何识别出系统中的参与进程(也叫做 Pi-演算的代理);(2)如何识别出代理使用的通道有哪些.在此给出寻找代理和通道的三条规则.

**规则 1.** 逻辑上单独的原子服务作为一个代理,也就是说 Web 服务组合中的每个子 Web 服务作为一个代理出现;当需要对某个复杂的单一服务进行内部逻辑验证和流程分析时,也可将其分割为几个相对独立的代理.

**规则 2.** 当消息在两个代理间以一定顺序出现时(非并行),它们将在某一通道上进行传递;

**规则 3.** 需要通信的两个代理间至少存在一条通道.

下面就按照以上 3 条规则进行转化.根据以上规则,将图 3 抽象为如图 4 所示的流图.

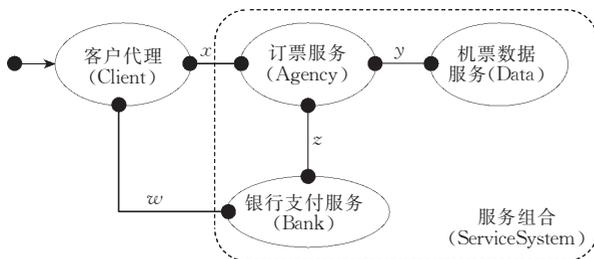


图 4 Web 服务组合抽象成 Pi-演算流图

其中  $x$  为客户代理 Client 与订票服务 Agency 之间的通道, $y$  为订票服务 Agency 与机票数据服务 Data 之间的通道, $z$  为银行支付服务 Bank 与订票服务 Agency 之间的通道, $w$  为客户代理 Client 与银行支付服务 Bank 之间的通道.

对系统进行建模如下(为了清楚地表示沿通道发出消息,发出的消息均增加了尖括号 $\langle \rangle$ ):

设

$$\begin{aligned} \bar{c} = \{x, w, Req, Ask, Pro, Ref, Acc, Inf, Pay, Con\}, \\ Client(\bar{c}) = \bar{x}\langle Req \rangle.x(msg).([\text{msg} = Ask] \\ (\bar{x}\langle Pro \rangle.x(msg1)).([\text{msg1} = Ref]Client(\bar{c}) + \\ [\text{msg1} = Acc]w(msg2).[\text{msg2} = \\ Inf]\bar{w}\langle Pay \rangle.x(msg3).[\text{msg3} = Con]Client(\bar{c})). \end{aligned}$$

类似地,订票服务、机票数据服务和银行支付服务分别建模如下:

订票服务.

设

$$\bar{a} = \{x, y, z, Req, Ask, Pro, Get, Tic, Ref, Acc, ReP, PaS, Con\},$$

$$Agency(\bar{a}) =$$

$$\begin{aligned} x(msg).[\text{msg} = Req]\bar{x}\langle Ask \rangle.x(msg1).[\text{msg1} = \\ Pro]\bar{y}\langle Get \rangle.y(msg2).[\text{msg2} = \\ Tic](\bar{x}\langle Ref \rangle.Agency(\bar{a}) + \\ \bar{x}\langle Acc \rangle.\bar{z}\langle ReP \rangle.z(msg3).[\text{msg3} = PaS]\bar{x}\langle Con \rangle. \\ Agency(\bar{a})). \end{aligned}$$

机票数据服务.

设

$$\bar{d} = \{y, Get, Tic\},$$

$$Data(\bar{d}) = y(msg)[\text{msg} = Get]\bar{y}\langle Tic \rangle.Data(\bar{d}).$$

银行支付服务.

设

$$\bar{b} = \{z, w, ReP, Inf, Pay, PaS\},$$

$$Bank(\bar{b}) =$$

$$\begin{aligned} z(msg).[\text{msg} = ReP]\bar{w}\langle Inf \rangle.w(msg1).[\text{msg1} = \\ Pay]\bar{z}\langle PaS \rangle.Bank(\bar{b}). \end{aligned}$$

整个服务系统 ServiceSystem 由订票服务、机票数据服务和银行支付服务共同组成.

对于 ServiceSystem 来说, $\{y, z\}$  是其内部私有通道,所以应该作为受限名出现.令

$\bar{s} = \{x, w, Req, Ask, Pro, Ref, Acc, Inf, Pay, Con\}$ , ServiceSystem 的定义如下:

$$\begin{aligned} ServiceSystem(\bar{s}) = (y, z)(Agency(\bar{a}) | Bank(\bar{b}) | \\ Data(\bar{d})). \end{aligned}$$

在寻找代理和通道的规则 1 中,提到当需要对某个复杂的单一服务进行分析和验证时,也可将其分割为几个相对独立的代理.以 Agency 为例,可以将其分割为如图 5 所示的三部分.

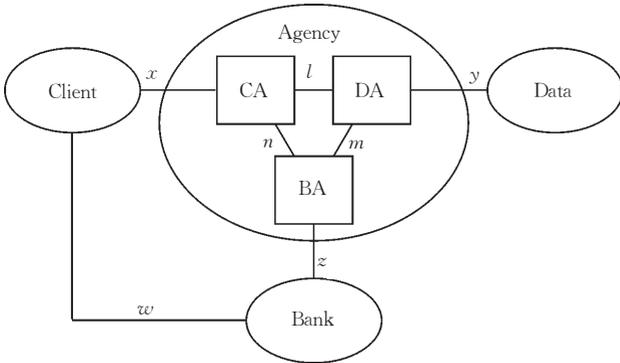


图 5 Agency 的内部服务分割图

图 5 中 CA 为专门负责与客户交互的服务,DA 是用于获取机票数据的服务,而 BA 负责与银行服务间的事务.三个服务间通过私有的  $l, m, n$  三条通道链接起来.如果将 Agency 表达出来,则为三个代理,分别负责 Agency 的三条向外通道上的通信.三个代理在受限通道名  $l, m, n$  上组合成为 Agency 服务.其描述思路与 ServiceSystem 的一致,通过这种描述可以清楚地看出:服务可以用类似的方式从小到大组合起来,通过将底层的原子(支撑)服务组合形成非常复杂的高层(应用)服务.

在这个实例中, Pi-演算有充分的能力来描述案例中的主要需求,并且保持一定的抽象程度,这使系统得以形式化和建模.

### 3.3 描述服务的动态变化

Pi-演算比起其前身 CCS 来说,支持传递通道名是其重要特征;这使得 Pi-演算能够描述结构动态变化的系统,如移动通信系统<sup>[9]</sup>等.同样,我们使用它来描述服务组合的动态变化.假设机票数据服务有一个备份服务器,当主服务器要临时关闭时,需要通知订票服务 Agency,让它将服务链接切换到备份服务器通道,如图 6 所示.

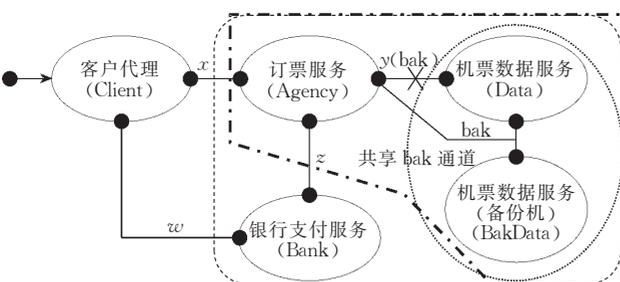


图 6 服务的动态切换

图 6 表示机票数据服务 Data 在停止服务前,通过  $y$  通道将原来属于自己和备份服务器之间的私有通道 bak 传递给了订票服务 Agency,相当于 Data 扩大了私有通道 bak 的使用范围(即由图 6 中虚线椭圆框的范围扩大到了点划线多边形框的范围). Agency 在获取到 bak 通道后,便可使用 bak 通道继续请求和接收数据服务.对系统来说,有如下转变发生:

$$(bak)(\bar{y}\langle bak \rangle, Data' | BakData) | y(z). Agency' \\ \xrightarrow{\tau} (bak)(Data' | BakData | Agency' \{bak/z\}).$$

系统的拓扑结构发生改变,使得 Agency 可以通过备份数据通道访问备份数据服务器.

### 3.4 利用 Pi-演算对系统进行推演和验证

使用形式化工具的目标是协助系统设计和验证.在设计阶段使用 Pi-演算有助于清楚地描述系统的交互行为;而在系统模型建立后,则可利用 Pi-演算来推演系统的行为,同时验证模型的正确性,如发现系统行为不完整、死锁、缺少同步等.每次修改模型时,都要进行这些检查,以保证系统是一直按照需求设计,并且在正确地实现.

Pi-演算与有限状态机、下推自动机和图灵机相比,其优势主要体现在两方面:一是对等价的概念作了强化,人们认为,自动机等价是指语言等价;但是 Robin Milner 发现语言等价有时候并不表示两个自动机是完全等价的,所以在 Pi-演算中引入了强模拟、弱模拟、强互模拟以及弱互模拟等概念,以区分语言等价所不能描述的自动机之间的区别.这同时也给了我们验证两个系统是否等价的有力方法:如果系统  $P$  强模拟系统  $Q$ ,则表示  $P$  至少能完成  $Q$  所能完成的所有任务和变迁;如果系统  $P$  和  $Q$  弱互模拟,则表示  $P$  和  $Q$  表现出来的外部行为是等价的.对于互模拟概念的详细讨论超出了本文的内容,参见文献[4,5].

作为一种强大和成熟的形式化方法, Pi-演算有支持其正确性验证和相关应用的工具,如 JACK 工具集、基于 Pi-演算的语言 PICT、可执行的 Pi-演算 EPI 以及传值进程代数工具 VPAM<sup>[10]</sup>等.本文采用基于 New Jersey SML 语言编译器<sup>[11]</sup>的 MWB (Mobility Workbench)工具<sup>[12]</sup>. MWB 工具是用于操作和分析移动并发系统的自动化 Pi-演算工具,它使用函数式编程语言 SML(即标准元语言)构建,具体使用的是朗讯贝尔实验室实现的 SML/NJ 110 版.验证时的软件实验环境为: Windows2000,

SML/NJ 110.0.3 版, MWB'99 版. 下面就在此实验环境中对组合的 Web 服务模型进行推演和验证.

使用 MWB 工具对代数表达式进行语法分析可以发现进程定义的一些基本错误:如类型错误、缺少同步、不完整的行为等,并可利用工具的推理功能排除一些最常见的错误. 使用 `deadlocks` 命令可以检查进程是否存在死锁(即进程终止)的情况,并且可以使用 `step` 命令查看进程行为,即进行系统行为的跟踪推演. 完成这些基本步骤后,再验证系统确实符合当初设计的需求,即设计出的系统与客户需求等价. 对于如何验证系统与需求一致,文献[13]提出了一种简单而有效的反转证明法,该方法是用 CCS 系统的观察等价的证明,因为观察等价实际上就是用户需求行为的对偶与服务系统所提供的行为一致,这就证明了系统满足了用户需求. 但是,对于 Pi-演算来说,因为通道的存在,这种方法不能直接在 MWB 工具中使用. 但是,我们仍然将客户代理的行为求反,采用行为跟踪和理论推演结合的方式来验证其功能等价性.

在前面的建模步骤中,我们将客户代理也建模为一个进程,但是该进程不存在于 Web 服务组合进程中. 这种方法的目的是将客户代理的进程作为系统设计的目标,即我们要实现的系统是满足客户代理需求的系统. 所以图 3 虚线框中的 Web 服务组合应该满足客户代理的需求. 用 Pi-演算来描述这个逻辑,即图 3 虚线框中的 Web 服务组合与客户代理应该是观察等价的,即客户代理的行为与服务系统动作互补. 如果系统满足客户需求,则客户代理的补集所形成的进程应该与服务系统的进程等价.

客户代理的对偶系统进程定义为

$ReverseClient(\bar{c}) =$

$$x(msg).[msg=Req]\bar{x}\langle Ask \rangle.x(msg1).[msg1=Pro](\bar{x}\langle Ref \rangle.ReverseClient(\bar{c}) + \bar{x}\langle Acc \rangle.\bar{w}\langle Inf \rangle.w(msg2)[msg2=Pay]\bar{x}\langle Con \rangle.ReverseClient(\bar{c})),$$

其中

$$\bar{c} = \{x, w, Req, Ask, Pro, Ref, Acc, Inf, Pay, Con\}.$$

根据 ServiceSystem 的定义,  $\{Get, Tic, ReP, PaS\}$  是该服务系统通过通道  $\{y, z\}$  的内部动作, 对外它们是完全不可见的, 相当于一个  $\tau$  动作. 在 ServiceSystem 中将  $\tau$  动作消去后的系统与原 ServiceSystem 是观察等价的. 可以看到, 消去  $\tau$  动作后 ServiceSystem 的可观察动作集与 ReverseClient 完全相同, 即它们使用相同的通道与外界通信.

使用 MWB 工具的 `step` 命令分别跟踪 ReverseClient 的行为和消去  $\tau$  动作后 ServiceSystem 的行为<sup>①</sup>, 可以发现, 虽然两者的内部结构和变迁不一样(主要区别在  $\tau$  动作上), 但它们表现出来的外部功能是等价的, 即二者是弱互模拟的. 因此, ServiceSystem 这个服务组合满足了系统设计需求, 能够提供客户代理所需要的服务, 同时也说明前面提出的建模方法和寻找代理的规则是正确的.

## 4 相关工作

在研究利用进程代数方法描述验证 Web 服务的过程中, 研究人员提出了基于各种演算的方法和模型, 也有基于其它形式化方法(如 Petri 网等)的研究在开展. 本节概述其中的一些重要工作.

Nakajima 描述了如何使用 SPIN 模型检查工具来验证 Web 服务组合. 其组合语言为 WSFL. 为了使用 SPIN 工具来验证, 首先要将 WSFL 描述的业务流程转换成 Promela 规范语言来描述, SPIN 提供模型检查功能, 但不提供等价性的检查.

Karamanolis, Giannakopoulou, Magee 和 Wheeler 将 Web 服务流程转换成 FSP 进程(Finite State Process)并使用 LTSA 工具(the Labeled Transition System Analyzer)进行模型检查<sup>[14]</sup>. LTSA 工具允许用户以确定性 FSP 进程来指定属性, 在文献[15]中, Foster, Magee 等人描述了 LTSA 工具的 BPEL4WS 插件. 他们把 BPEL4WS 程序转换成 FSP 进程并使用 LTSA 工具来验证 FSP 进程. LTSA 工具仅仅支持模型检查, 而且 FSP 进程所能表达的属性集小于  $\mu$ -演算所能支持的属性集.

Koshkina 和 van Breugel 引入基于 BPEL4WS 的 BPE 演算<sup>[16]</sup>. 使用 PAC 工具(Process Algebra Compiler)和 CWB-NC 工具(Concurrency Workbench of the New Century)来建模和验证 Web 服务协作. BPE 演算包含了 BPEL4WS 的控制流结构, 基于结构化操作语义, BPE 演算作为 PAC 的输入, 其输出作为 CWB 的输入, 从而验证服务组合.

此外, 基于 Petri 网的 Web 服务组合建模方法也有相关研究在开展, 文献[17]中提出的 WS-Net 是一种可执行的体系结构描述语言, 它支持着色 Petri 网的语义以及面向对象概念. 在 WS-Net 中,

<sup>①</sup> 在使用 MWB 工具验证时, 各进程式需按照 MWB 的符号规则进行表示, 如补号  $\bar{x}$  用 'x 表示等, 在此不再赘述.

每个 Web 服务组件在三个层次被描述:接口网声明组件提供给其它组件的服务;互连接网指定组件要完成其任务需要使用到的服务;互操作层描述组件的内部操作行为. WS-Net 实际是提出了一种体系结构模型,用于形式化体系结构的拓扑结构以及每个服务组件的行为和系统行为,但是 WS-Net 并没有提供将 WSDL 转换成 WS-Net 模型的自动转换工具.

## 5 结论和下一步的研究

Web 服务及其组合的形式化描述和验证,是 Web 服务中一个重要的研究方向. 本文从 Pi-演算与 Web 服务协议栈的关系入手,讨论了 Pi-演算在 Web 服务中可能的应用场合,并用 Pi-演算建模和验证了一个实际问题,给出了可以使用 Pi-演算建模和验证 Web 服务组合的结论. 与已有的其它相关研究工作比起来,本文主要是对如何将 Pi-演算用于描述和验证 Web 服务组合作了探索,讨论了 Pi-演算和 Web 服务元素的对应关系、建模规则以及重要的验证点和验证方法.

许多现有的 Web 服务及其组合描述语言都是半形式化的,容易出错且不易检测和验证,也没有相应形式化工具的支持,这使得 Web 服务组合的正确性难以保证,其流程也无法在较抽象的层次上跟踪. 在这种背景下,出现了许多形式化方法,如基于 Petri 网、进程代数的方法. 我们所讨论的基于 Pi-演算的方法则除了具有传统进程代数的优势外,还具有可用于建模动态变化的系统以及检测死锁、同步等的的能力,所以使用 Pi-演算来描述 Web 服务及其组合不但理论上完全可行,实际上也具有前人所用方法所不具有的优势. 此外, Pi-演算有 MWB 等模型工具的支持,这使得模型的测试和跟踪成为可能.

下一步的工作中,需要建立更加完善的模型工具,实现服务描述和代数描述之间的自动双向转换. 其中描述语言包括各个层次的语言,如 Web 服务描述语言(WSDL)、组合描述语言(BPEL4WS)和协作描述语言(WS-CDL)等. 研究它们和 Pi-演算以及其它常用进程代数方法之间的自动转换,是支持模型自动验证的基础. 此外,利用 Pi-演算研究 Web 服务的自动组合模型也是重要的课题.

## 参 考 文 献

- Koehler J., Srivastava B. Web service composition; Current solutions and open problems. In: Proceedings of the 13th International Conference on Automated Planning & Scheduling, Trento, Italy, 2003, 28~35
- Milner R. Communication and Concurrency. Englewood Cliffs; Prentice-Hall, 1989
- Fensel D. The semantic web and its languages. IEEE Intelligent Systems, 2000, 15(6): 67~73
- Milner R. Communicating and Mobile Systems; The Pi-Calculus. Cambridge; Cambridge University Press, 1999
- Milner R., Parrow J., Walker D. A calculus of mobile processes, part I/II. Journal of Information and Computation, 1992, 100(1): 1~77
- Lin Hui-Min. Complete proof systems for observation congruences in finite control pi-calculus. In: Proceedings of the 25th International Colloquium on Automata, Languages and Programming, Aalborg, Denmark, 1998, 443~454
- Jiao Wen-Pin, Zhou Ming-Hui, Wang Qian-Xiang. Formal framework for adaptive multi-agent Systems. In: Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology, Halifax, Canada, 2003, 442~445
- Markus Lumpe. A Pi-calculus based approach to software composition [Ph. D. dissertation]. Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland. 1999
- Orava F., Parrow J. An algebraic verification of a mobile network. Formal Aspect of Computing, 1992, 4(6): 497~543
- Lin Hui-Min. A verification tool for value-passing process algebras. IFIP Transactions C-16: Protocol Specification, Testing and Verification, North-Holland, 1993, 79~92
- Appel A. W., MacQueen D. B. Standard ML of new Jersey. In: Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming, New York, 1991, 1~13
- Victor B., Moller F. The mobility workbench; A tool for the pi-calculus. In: Proceedings of the 6th International Conference on Computer Aided Verification, California, USA, 1994, 428~440
- Salaün G., Bordeaux L., Schaerf M. Describing and reasoning on Web services using process algebra. In: Proceedings of the 2nd IEEE International Conference on Web Services, San Diego, California, USA, 2004, 43~50
- Karamanolis C., Giannakopoulou D., Magee J., Wheeler S. M. Model checking of workflow schemas. In: Proceedings of the 4th International Enterprise Distributed Object Computing Conference, Makuhari, Japan, 2000, 170~179
- Foster H., Uchitel S., Magee J., Kramer J. Model based verification of web service compositions. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering, Montreal, Canada, 2003, 152~161
- Koshkina M., van Breugel F. Modelling and verifying Web service orchestration by means of the concurrency workbench. ACM SIGSOFT SEN, 2004, 29(5): 1~10
- Zhang Jia, Chung Jen-Yao, Chang C. K., Kim S. WS-Net: A petri-net based specification model for web services. In: Proceedings of the 2nd IEEE International Conference on Web Services, San Diego, California, USA, 2004, 420~427



**LIAO Jun**, born in 1977, Ph. D. candidate. His research interests include Web services, open system and middleware.

**TAN Hao**, born in 1970, Ph. D., associate professor. His research interests include middleware and Web services, continuous multimedia.

**LIU Jin-De**, born in 1930, professor, Ph. D. supervisor. His research interests include open distributed systems, middleware and Web services, pervasive computing, and security.

## Background

This project is supported by the National Defense Pre-research Foundation of the “tenth five-year-plan” of China under grant No. 41315010103.

Open System and Middleware Laboratory of UESTC has been dedicated to the research of open distributed system. The research has been supported by National Defense Pre-research Foundation since “eighth five-year-plan”. The research team studied open system environment based on CORBA (including real-time CORBA-rtORB and embedded CORBA-emORB), Java (Jini) and a agent based computing basic architecture (ABCBA). Several important results were achieved: Open System Profile and Framework which was

awarded the third class prize of National Defense; Core Framework of Software Radio based on emORB which is being focused by National Navy who intends to use it; ABCBA on which a securities administration system was successfully developed and put into use.

Now Web services and pervasive computing middleware (pvcwCORBA) are used as middleware for integrating all of the former works. Pi-calculus is used as a tool of formal method to describing and verifying services and software component composition. In this paper, several rules of describing Web services and a method of reasoning Web services based on pi-calculus are presented.