ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.3001−3014
DOI: 10.1360/jos183001

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Web 服务行为兼容性的判定与计算*

邓水光, 李 莹, 吴 健+, 邝 砾, 吴朝晖

(浙江大学 计算机科学与技术学院,浙江 杭州 310027)

Determination and Computation of Behavioral Compatibility for Web Services

DENG Shui-Guang, LI Ying, WU Jian+, KUANG Li, WU Zhao-Hui

(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: Phn: +86-571-87951647, E-mail: dengsg@zju.edu.cn, http://middleware.zju.edu.cn/ ~shuiguang

Deng SG, Li Y, Wu J, Kuang L, Wu ZH. Determination and computation of behavioral compatibility for Web services. *Journal of Software*, 2007,18(12):3001−3014. http://www.jos.org.cn/1000-9825/18/3001.htm

Abstract: How to ensure services compatible at the behavioral level is an important issue for services integration and collaboration in a seamless way. Based on the proposed concept of service view, a formal definition of behavioral compatibility between services is proposed. Then, a π-calculus-based method is proposed to qualitatively determine and quantitatively compute behavioral compatibility. First, it transforms service behaviors and interactions between services into π-calculus processes using an algorithm automatically. Second, it determines qualitatively whether two services are behavioral compatible with the help of operational and transitional semantics and a formal deduction. After that it proposes an algorithm based on the Expansion Law to compute the compatibility degree between services quantitatively. The application of the method in the scenarios of composing and replacing services dynamically shows that it is very useful for correctly building and reliably executing service compositions.

Key words: Web service; service behavior; behavioral compatible; π calculus; process

摘 要: 确保 Web 服务行为兼容是实现 Web 服务无缝集成与协作的一个重要问题.在服务视图概念的基础上,给出了 Web 服务行为兼容性的相关定义.提出一种基于π演算的 Web 服务行为兼容性的定性判定与定量计算方法.该方法首先通过算法自动地将 Web 服务行为和 Web 服务间的交互行为表达成π演算进程,然后借助π演算的操作语义和形式化推演实现服务行为兼容性自动的定性判定;随后在π演算的进程变换理论的基础上提出算法实现服务兼容性自动的定量计算.该方法在服务动态组合与服务动态替换中的典型应用表明,该方法对于服务组合的正确建立和可靠执行具有重要作用.

关键词: Web 服务;服务行为;行为兼容; π演算;进程

中图法分类号: TP393 **文献标识码**: A

* Supported by the National Key Technology R&D Program of China under Grant No.2006BAH02A01 (国家科技支撑计划); the National Natural Science Foundation of China under Grant Nos.60603025, 60503018 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z171 (国家高技术研究发展计划(863)); the Natural Science Foundation of Zhejiang Province of China under Grant No.Y105463 (浙江省自然科学基金)

Received 2007-06-10; Accepted 2007-10-16

Web 服务作为一种新型的分布式构件模型已经在电子商务、企业应用集成等领域扮演着越来越重要的角色,特别是 Web 服务的组合技术,因其能实现服务的重用和增值而成为学术界和工业界共同关注的焦点,众多的服务组合方法和研究项目被提了出来[1].为实现服务组合中各成员服务之间的无缝集成与协作,如何保证各成员服务之间的兼容性至关重要.

近年来,随着 Web 服务技术的发展,Web 服务模型也从最初的黑盒式的接口模型,如 WSDL 服务模型,发展成具有多视图的服务模型,如 OWL-S(http://www.w3.org/Submission/OWL-S/)和 WSMO(http://www.w3.org/Submission/WSMO/)等.这些服务模型包含了用于描述 Web 服务各个侧面的多个视图,如 OWL-S,它既包括了用于注册和发现的服务概要信息 Service Profile,也包含了描述服务内部实现细节的流程模型 Service Model.而随着这些新型的 Web 服务模型的不断出现和持续发展,越来越多的研究人员指出,服务不仅只包含静态的语法、语义等信息,还应包含内部控制流、数据流、交互协议、状态变迁等动态行为属性[2].而这一观点伴随服务粒度的不断增大越发受到重视.由于目前网络上的服务大多是原子级的简单服务,服务内部的操作彼此孤立,均可单独调用.然而,随着服务组合方法的不断成熟和应用,网络上势必会出现大量大粒度的流程式组合服务,这些组合服务内部是暗含业务流程逻辑的,它所提供的操作在调用上存在一定的时序关系.因此,将这些服务进行再组合时,不仅需要保证各成员服务之间在语法和语义上的兼容性,更要保证它们在行为上的兼容性.因此,行为兼容性是服务组合中各成员服务之间兼容性的一个重要方面.目前,服务行为兼容性的判定引起了国内外学者的关注[3−8].但这些工作大多基于 Petri 网(如文献[3])、有限状态机(如文献[5])或者自动机理论(如文献[8])展开,一旦服务行为、服务间的交互过程变得错综复杂时,这些方法都将出现空间爆炸,从而导致计算和判定的复杂度增大.此外,这些方法均只实现兼容性的定性判定.为此,本文采用π演算实现 Web 服务行为及服务交互行为的建模,在此基础上,借助π演算的操作语义、自动推演能力、进程变换等基础理论和工具,实现 Web 服务行为兼容性的自动定性判定和自动定量计算.

文章第 1 节给出相关概念.第 2 节首先说明如何利用π演算刻画服务行为以及服务间的交互,给出算法自动得到服务的进程表达式;之后介绍利用π演算理论定性判定和定量计算服务间的兼容度.第 3 节说明服务行为兼容性的定量判定和定量计算在服务动态组合及服务动态替换中的应用.第 4 节是相关工作和比较.最后是本文的总结与展望.

1 基本定义

1.1 Web服务视图

图 1 显示的服务 Vendor,包含两个端口类型(PortType):*PT*1 和 *PT*2,其中,*PT*1 包含了 *Op*1,*Op*2 和 *Op*3 这 3 个操作(operation),*PT*2 包含了 *Op*4 和 *Op*5 这两个操作.该服务在接收到订单消息 *PO* 之后生成它的一个实例并开始执行,其内部流程逻辑描述为:服务在初始状态 *A* 获得用户的一个订单消息 *PO* 之后进入状态 *B*.检查库存后:1) 若库存无法满足用户订单需求(即 *NoStock*=Yes),则向用户发送请求被拒绝的消息 *REF* 进入结束状态 *D*,服务实例结束;2) 若库存满足用户订单需求(即 *NoStock*=NO),则向用户发送货物运输消息 *DEL* 进入状态 *C*,等待现金支付消息 *CP* 或者银行转帐消息 *BTP*,只要获得其中任意一个消息,服务就进入结束状态 *D*,从而完成了一次交易.

如图 1 所示的 Vendor 服务包含了外部接口视图和内部行为视图:从外界来看,服务提供了端口和操作供外界调用从而完成消息的接收和发送;而从内部来看,服务是根据一定的业务过程在消息收发动作的触发下,从起始状态经过状态的变迁最终到达结束状态.

定义(服务接口视图). 服务接口视图 *IV* 可形式化定义为一个五元组 $IV=(T,P,M,f_p,f_m)$,其中:

(1) *T* 为该服务的端口类型集合;

(2) *P* 为该服务包含的操作集合,对于 *P* 中一个元素 $p\in P$,引入标记 *type*(*p*)用于标识该操作所属的类型,$type(p)\in\{OneWay,Notification\}$;

(3) *M* 为该服务的消息集合;

(4) f_p:$T\rightarrow 2^P$ 为端口类型集合到操作集合的超集的映射,说明每一个端口类型所包含的操作;

(5) f_m:$P\rightarrow 2^M$ 为操作到消息集合的超集的映射,说明每一个操作接收或者发送的消息的集合.

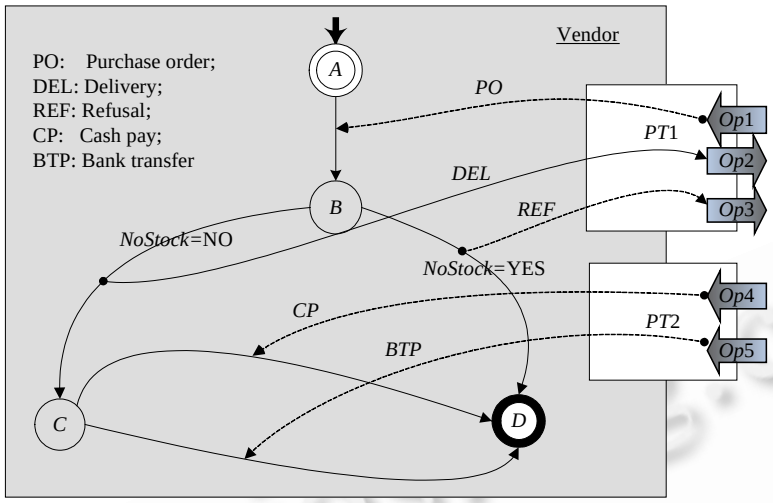


Fig.1 A Vendor service with behavior

图 1 一个具有行为的服务 Vendor

服务接口视图描述了服务外部端口类型、接口和消息.值得注意的是,在 WSDL 中定义了 4 种不同类型的操作,即 *OneWay*,*Request-Response*,*Solicit-Response* 和 *Notification*,但由于 *Request-Response* 类型和 *Solicit-Response* 类型的操作可以拆分成一个 *OneWay* 类型的操作和一个 *Notification* 类型的操作来表示,因此,在服务接口视图的定义中,*type*(*p*)的取值范围为 *type*(*p*)∈{*OneWay*,*Notification*}.根据服务接口视图的定义,如图 1 所示的 Vendor 服务的接口视图可定义为 $IV_{Vendor}=(T,P,M,f_p,f_m)$,其中:

(1) $T=\{PT1,PT2\}$;

(2) $P=\{Op1,Op2,Op3,Op4,Op5\}$;

(3) $M=\{PO,DEL,REF,CP,BTP\}$;

(4) $f_p(PT1)=\{Op1,Op2,Op3\}$;$f_p(PT2)=\{Op4,Op5\}$;

(5) $f_m(Op1)=\{PO\}$;$f_m(Op2)=\{DEL\}$;$f_m(Op3)=\{REF\}$;$f_m(Op4)=\{CP\}$;$f_m(Op5)=\{BTP\}$.

定义(服务行为视图). 服务行为视图 *BV* 可形式化定义为一个四元组 $BV=(S,s_o,s_f,R)$,其中:

(1) S 为状态集合;

(2) s_o 为起始状;

(3) s_f 为结束状态;

(4) R 为状态迁移集合,每一个状态迁移 r 可以形式化为一个五元组 $r=(s_b,s_e,C,a,M)$,其中:

a) s_b 为该迁移的起始状态;

b) s_e 为迁移的目标状态;

c) C 为迁移能够发生的条件集合;

d) a 为迁移的触发动作(消息接收或者发送动作),即 $a\in\{receive,send\}$;

e) M 为触发动作所关联的消息集合.

事实上,服务行为视图是通过有限状态机来刻画服务内部的逻辑流程,该流程在消息驱动下实现状态的转移.根据该定义,如图 1 所示的服务行为视图可以定义为 $BV_{Vendor}=(S,s_o,s_f,R)$,其中:

(1) $S=(A,B,C,D)$;

(2) $s_o=A$;

(3) $s_f=D$;

(4) $R=\{r_1,r_2,r_3,r_4,r_5\}$:$r_1=(A,B,\varnothing,receive,\{+PO\})$;$r_2=(B,C,\{NoStock=NO\},send,\{-DEL\})$;
$r_3=(B,D,\{NoStock=YES\},send,\{-REF\})$;$r_4=(C,D,\varnothing,receive,\{+CP\})$;$r_5=(C,D,\varnothing,receive,\{+BTP\})$.

定义(服务视图). 服务视图 SV 可形式化定义为一个三元组 $SV=(IV,BV,f_{IB})$,其中:

(1) IV 为服务接口视图;

(2) BV 为服务行为视图;

(3) f_{IB}:$BV.R\rightarrow IV.P$ 为行为视图与接口视图的关联映射,用于说明行为视图中迁移的触发动作与接口视图中操作的对应关系.

服务行为视图中的迁移都由触发动作发起,触发动作最终对应于接口视图中操作的调用,而迁移中触发动作所关联的消息则对应于操作调用中接收或发送的消息,如 *receive* 类型的触发动作源于对接口视图中 *OneWay* 类型操作的调用,而 *send* 类型的触发动作则源于对接口视图中 *Notification* 类型操作的调用.给定行为视图中的一个迁移 r,$f_{IB}(r)$为该迁移在接口视图中所对应的操作.

定义(操作序列/完全操作序列). 给定服务视图 $SV=(IV,BV,f_{IB})$,在其行为视图 BV 中,从起始状态 s_o 到 BV 中其他任意一个状态 s 的一条状态迁移路径 $st=\langle r_i,\ldots,r_j,\ldots,r_k\rangle$上的操作组成的有序队列 $PQ=\langle f_{IB}(r_i),\ldots,f_{IB}(r_j),\ldots,f_{IB}(r_k)\rangle$称为该服务视图的一条操作序列,其长度记为 $l_{PQ}=|PQ|$.若 s 为结束状态,则该操作序列为完全操作序列.

在如图 1 所示的服务视图中包含了 3 条完全操作序列,分别为 $PQ_{V1}=\langle Op1,Op2,Op4\rangle$,$PQ_{V2}=\langle Op1,Op2,Op5\rangle$和 $PQ_{V3}=\langle Op1,Op3\rangle$.

1.2 Web服务行为兼容性

Web 服务行为兼容性是针对 Web 服务的交互过程而言的.如图 2 所示的两个服务 Vendor(V)和 Customer(C)的交互场景(其中,Vendor 为图 1 所示的服务)存在如下 3 种可能的交互过程.

1) Customer 发送 *PO* 消息;Vendor 接收 *PO* 之后检查库存,因订单请求可满足,所以发送 *DEL* 消息;Customer 接到 *DEL* 消息之后,选择发送 *CP* 消息进入结束状态 *D*;Vendor 接收到 *CP* 消息之后也进入结束状态.至此,两个服务之间的一次正常交互结束;

2) 与 1)类似,但 Customer 在状态 *C* 选择发送 *BTP* 消息而非 *CP* 消息,进入结束状态 *D*;Vendor 接收到 *BTP* 消息之后也进入状态 *D*.至此,两个服务之间的一次交互结束;

3) Customer 向 Vendor 发送 *PO* 消息;Vendor 检查库存之后,因订单请求无法满足,所以向 Customer 发送 *REF* 消息并结束;而 Customer 因无法接收 *REF* 消息而停滞在状态 *C* 无法正常结束.

上述 3 个交互过程表明,对于两个服务之间的交互,从内部行为视图来看可以表现为两者内部状态之间的迁移,而从各自外部的接口视图来看,表现为两者操作之间的一系列调用.

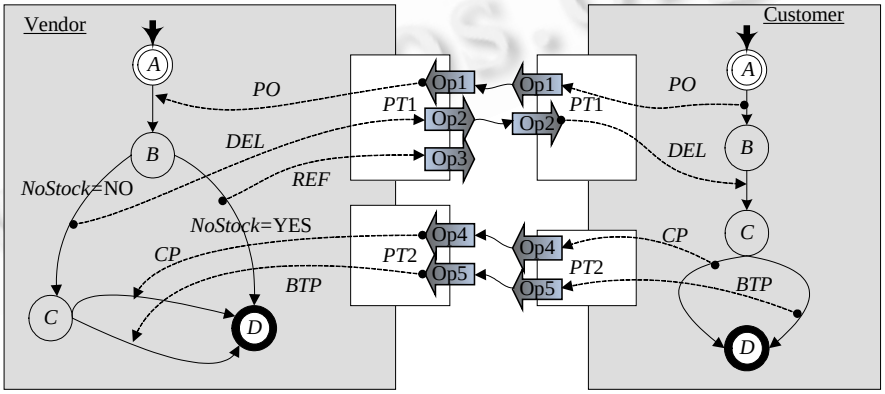


Fig.2 Interaction between Vendor and Customer services

图 2 服务 Vendor 与服务 Customer 之间的交互

定义(正常交互元). 两个服务视图 $SV_1=(IV_1,BV_1,f_{IB_1})$ 和 $SV_2=(IV_2,BV_2,f_{IB_2})$ 之间的一个交互元可以表示为一个三元组$\infty=(p_-,p_+,M)$,其中:

(1) $p_-\in\{p|p\in IV_1.P\cup IV_2.P,type(p)=Notification\}$;

(2) $p_+\in\{p|p\in IV_1.P\cup IV_2.P,type(p)=OneWay\}$;

(3) p_-和 p_+满足关系:$p_-\in IV_1.P\wedge p_+\in IV_2.P$ 或 $p_-\in IV_2.P\wedge p_+\in IV_1.P$;

(4) $M=f_m(p_-)$且 $M\subseteq f_m(p_+)$.

交互元为两个服务在交互过程中的一个步骤,它要求交互的两个操作类型互为对偶,且类型为 *OneWay* 的操作能接收类型为 *Notification* 的操作所发出的所有消息.

定义(异常交互元). 给定两个服务视图 $SV_1=(IV_1,BV_1,f_{IB_1})$ 和 $SV_2=(IV_2,BV_2,f_{IB_2})$,以及一个类型为 *Notification* 的操作 $p_-\in IV_1.P$,若任意一个类型为 *OneWay* 的操作 $p_+\in IV_2.P$,关系 $f_m(p_-)\subseteq f_m(p_+)$均不成立,则$\infty=(p_-,null,M)$为异常交互元,其中,$M=f_m(p_-)$.

该定义表明,异常交互异元表示两个服务视图之间一次不成功的交互步骤,即交互一方通过操作发出消息 *M*,而另一方没有相应操作接收该消息.我们将正常交互元与异常交互元统称为交互元.如图 2 所示中第 3 个交互过程的第 3 个交互元就是一个异常交互元,即$\infty=(V.Op3,null,\{REF\})$,表示 Vendor 通过 *Op*3 发送消息 *REF*,而 Customer 无法接收,因此出现异常.给定一个交互元∞,引入函数$\vartheta(\infty,s_1)$得到服务 s_1 参与该交互元的操作,如图 2 所示中第 1 个交互过程中的第 1 个交互元$\infty=(C.Op1,V.Op1,\{PO\})$,$\vartheta(\infty,V)=Op1$.

定义(交互路径). 两个服务视图 $SV_1=(IV_1,BV_1,f_{IB_1})$ 和 $SV_2=(IV_2,BV_2,f_{IB_2})$ 间的一条交互路径为满足如下条件的一个交互元有序队列 $IP=\langle\infty_1,\infty_2,\ldots,\infty_n\rangle$.

(1) $P_{s_1}=\langle\vartheta(\infty_1,s_1),\vartheta(\infty_2,s_1),...,\vartheta(\infty_n,s_1)\rangle$ 为 SV_1 的操作序列;

(2) $P_{s_2}=\langle\vartheta(\infty_1,s_2),\vartheta(\infty_2,s_2),...,\vartheta(\infty_n,s_2)\rangle$ 为 SV_2 的操作序列;

(3) $\infty_1,\infty_2,\ldots,\infty_{n-1}$ 均为正常交互元,即$\infty_i.P_+\neq null(1\leq i\leq n-1)$;

(4) ∞_n 为异常交互元或按照 IP 交互之后 SV_1 和 SV_2 均到达结束状态.

定义表明,交互路径是两个服务均从起始状态开始交互直至交互结束(即两个服务均到达结束状态)或直至出现异常交互元而产生的交互元序列.如图 2 所示的 3 个交互过程分别对应如下 3 条交互路径:

$IP_1=\langle\infty_1=(C.Op1,V.Op1,\{PO\}),\infty_2=(V.Op2,C.Op2,\{DEL\}),\infty_3=(C.Op4,V.Op4,\{CP\})\rangle$

$IP_2=\langle\infty_1=(C.Op1,V.Op1,\{PO\}),\infty_2=(V.Op2,C.Op2,\{DEL\}),\infty_3=(C.Op5,V.Op5,\{BTP\})\rangle$

$IP_3=\langle\infty_1=(C.Op1,V.Op1,\{PO\}),\infty_2=(V.Op2,null,\{REF\})\rangle$

定义(有效交互路径). 服务视图 SV_1 和 SV_2 的一条交互路径 $IP=\langle\infty_1,\infty_2,\ldots,\infty_n\rangle$是有效交互路径,当且仅当:

(1) $P_{s_1}=\langle\vartheta(\infty_1,s_1),\vartheta(\infty_2,s_1),...,\vartheta(\infty_n,s_1)\rangle$ 为 SV_1 的完全操作序列;

(2) $P_{s_2}=\langle\vartheta(\infty_1,s_2),\vartheta(\infty_2,s_2),...,\vartheta(\infty_n,s_2)\rangle$ 为 SV_2 的完全操作序列.

定义表明,有效交互路径全部由正常交互元组成,根据该交互路径进行交互之后,两个服务视图均能到达结束状态.由此可知,上述 3 条交互路径中,IP_1 和 IP_2 为有效交互路径.

定义(完全兼容/部分兼容/可兼容/不兼容). 若两个服务视图之间的所有交互路径均为有效交互路径,则这两个服务视图是完全兼容的;若至少存在 1 条有效交互路径,则这两个服务视图是局部兼容的;完全兼容与部分兼容统称为可兼容;若不存在有效交互路径,则两个服务视图是不兼容的.

定义(兼容度). 若两个服务视图之间存在 *n* 条交互路径,而有效交互路径的数目为 *m*,则它们之间的兼容度为 $\varpi(s_1,s_2)=m/n$.

由此可知,如图 2 所示的两个服务视图是部分兼容,且兼容度为 2/3.

2 Web 服务行为兼容性的自动判定与计算

2.1 π演算基本语法

在π演算中,进程代表并发运行实体的单位,进程和名字是π演算中的两个基本元素.一个进程 *t* 可以被定义

成如下形式中的一种:

$$t ::= o \mid \overline{a}\langle x\rangle.P \mid a\langle x\rangle.P \mid \tau.P \mid P+Q \mid P \mid Q \mid (va)P \mid \text{if } x=y \text{ then } P$$

(1) 0:空进程,表示不做任何操作的进程,也可由 NIL 表示;

(2) $\overline{a}\langle x\rangle.P$:输出前缀表达式,表示进程从通道 a 发出 x 之后执行进程 P;

(3) $a\langle x\rangle.P$:输入前缀表达式,表示进程从通道 a 接收 x 之后执行进程 P;

(4) $\tau.P$:哑前缀表达式,表示进程无须执行任何动作就可以执行进程 P;

(5) $P+Q$:和表达式,表示在进程 P 和 Q 之间选择执行一个进程;

(6) $P|Q$:并行表达式,表示 P 和 Q 可以并发地独立执行.此外,若 P 和 Q 中有一个在某个通道上有发出动作,而另一个在同一个通道上有接收动作,则 P 和 Q 之间还可以发生内部同步操作;

(7) $(va)P$:限制表达式,表示进程 P 对外通道 a 被限制,即 P 无法通过通道 a 与外部进程进行通信,但仍可以在 P 的内部进行通信;

(8) if $x=y$ then P:匹配表达式,表示如果名字 x 与 y 相同,则执行进程 P;否则,该进程为空进程,不执行任何动作;匹配表达式还可以表示成$[x=y]P$.

有关π演算的操作语义、行为理论等,详见文献[10].

2.2 Web服务行为及其交互的π演算表达

对于服务视图中的接口视图,我们将操作表达成进程的通道,而在该操作上接收或者发送的消息视为该通道上收发的消息.表 1 为 WSDL 中的 4 类操作及其对应的基本进程表达式.

Table 1 Process expressions for Web service operations

表 1 Web 服务操作的进程表达

Operation type	Example	Process expression
OneWay	⟨*operation name*="*a*"⟩ ⟨*input message*="*m*"/⟩ ⟨/*operation*⟩	$a\langle m\rangle$
Request-Response	⟨*operation name*="*a*"⟩ ⟨*input message*="*m*"/⟩ ⟨*output message*="*n*"/⟩ ⟨/*operation*⟩	$a\langle m\rangle.\overline{a}\langle n\rangle$
Solicit-Response	⟨*operation name*="*a*"⟩ ⟨*output message*="*m*"/⟩ ⟨*input message*="*n*"/⟩ ⟨/*operation*⟩	$\overline{a}\langle m\rangle.a\langle n\rangle$
Notification	⟨*operation name*="*a*"⟩ ⟨*output message*="*m*"/⟩ ⟨/*operation*⟩	$\overline{a}\langle m\rangle$

对于 Web 服务的行为视图,其内部由状态迁移构成的流程结构可由不同的结构化进程表达式来表达.如图 1 所示的服务 Vendor 可表达成如式(1)所示的进程 P_V.

$$P_V = Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \tag{1}$$

由于 Web 服务本身可视为一个自治系统,因此,Web 服务之间的交互可被视为一个并发系统,可通过进程之间的并发组合来表示.在如图 2 所示的服务交互中,服务 Customer 可以被表达成如式(2)所示的进程 P_C.该进程通过通道 $Op1$,$Op2$,$Op4$ 和 $Op5$ 与外部环境通信,其中,$Op1$,$Op4$ 和 $Op5$ 为消息发送通道,$Op2$ 为消息接收通道.

$$P_C = \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \tag{2}$$

当服务 Vendor 与 Customer 都被表达成相应的进程之后,两者之间的交互可以表达成式(3)所示的组合进程.

$$P_{(V,C)} = P_V \mid P_C = Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \tag{3}$$

算法 ServiceView2Process 以一个服务视图作为输入,输出该服务视图对应的π演算进程表达式.该算法通过递归调用方法 $Exp(r)$生成进程表达式.若服务行为视图的变迁数目为 n,完全操作序列的最大长度为 m,则该算法

的复杂度为 $O(n\times m)$.

算法. ServiceView2Process.

INPUT: $SV=(IV,BV,f_{IB})$ where $IV=(T,P,M,f_p,f_m)$, $BV=(S,s_o,s_f,R)$, $f_{IB}:BV.R\rightarrow IV.P$;

OUTPUT: The corresponding π-calculus process P_{sv} for SV.

1. **FOR** each r in $BV.R$
2. **IF** ($r.s_b==s_o$){
3. **SET** $r_0=r$;
4. **Break**;
5. }
6. $P_{sv}=Exp(r_0)$;

METHOD: ***Exp*(*r*)**

1. **SET** $S=NULL$;
2. **FOR** each r' in $BV.R$
3. **IF** ($r'.s_b==r.s_e$)
4. $S.add(r')$;
5. **SET** $n=sizeof(S)$;
6. **IF** ($n==0$)
7. **RETURN** $Trans(r)$;
8. **IF** ($n==1$)
9. **RETURN** $Trans(r).Exp(S[0])$;
10. **ELSE**
11. **RETURN** $Trans(r).(Exp(S[0]+\ldots+Exp(S[n-1])))$;

METHOD: ***Trans*(*r*)**

1. **SET** $Op=SV.f_{IB}(r)$;
2. **SET** $type=type(Op)$;
3. **SET** $message=SV.IV.f_m(Op)$;
4. **IF** ($type==Notification$)
5. **RETURN** $\overline{Op}\langle message\rangle$;
6. **IF** ($type==OneWay$)
7. **RETURN** $Op\langle message\rangle$.

2.3 Web服务行为兼容性的自动定性判定

将 Web 服务视图及其交互表达成进程表达式之后,可以通过π演算的操作语义以及自动推演实现服务兼容性自动的定性判定.以如图 2 所示的交互为例,对其并发组合进程表达式(3)进行推演,得到式(4)~式(6)这 3 个推演过程.

$$\begin{aligned}P_{(V,C)} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle+Op5\langle BTP\rangle)+\overline{Op3}\langle REF\rangle)\mid\\ &\quad \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle)\\ &\xrightarrow{PO}(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle+Op5\langle BTP\rangle)+\overline{Op3}\langle REF\rangle)\mid(Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle)\\ &\xrightarrow{\text{DEL}}Op4\langle CP\rangle+Op5\langle BTP\rangle\mid\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle\xrightarrow{CP}0\mid 0=0\end{aligned}\tag{4}$$

$$\begin{aligned}P_{(V,C)} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\ &\quad \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{PO} (\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid (Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{DEL} Op4\langle CP\rangle + Op5\langle BTP\rangle \mid \overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle \xrightarrow{BTP} 0 \mid 0 = 0\end{aligned} \tag{5}$$

$$\begin{aligned}P_{(V,C)} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\ &\quad \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{PO} (\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid (Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{REF}\end{aligned} \tag{6}$$

以上 3 个推演过程也分别对应了服务 Vendor 与服务 Customer 之间的 3 条交互路径.其中,推演过程(4)和推演过程(5)最终都变成 0 进程,这说明两个服务按照这两种方式进行交互最终都能正常结束,因此,这两条交互路径为有效交互路径;而推演过程(6)在当 Vendor 进程通过通道 *Op*3 向 Customer 进程发出消息 *REF* 之后,由于 Customer 进程没有对应的通道接收该消息,致使组合进程表达式无法再继续往下进行迁移,这说明两个服务在这种情况下无法完成正常的交互,即这个推演过程所代表的交互路径为非有效交互路径,这也说明了服务 Vendor 和 Customer 是局部兼容.再考察图 3 所示的交互过程,其中,服务 NewCustomer(*NC*)对应的进程表达式如式(7)所示:

$$P_{NC} = \overline{Op1}\langle PO\rangle.(Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle) \tag{7}$$

服务 Vendor 与服务 NewCustomer 之间的交互对应的并发进程表达式如式(8)所示.

$$\begin{aligned}P_{(V,NC)} = P_V \mid P_{NC} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\ &\quad \overline{Op1}\langle PO\rangle.(Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle)\end{aligned} \tag{8}$$

对如式(8)所示的并发进程推演得到如下两个推演过程:

$$\begin{aligned}P_{(V,NC)} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\ &\quad \overline{Op1}\langle PO\rangle.(Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle) \\ &\xrightarrow{PO} (\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid (Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle) \\ &\xrightarrow{REF} 0 \mid 0 = 0\end{aligned} \tag{9}$$

$$\begin{aligned}P_{(V,NC)} &= Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\ &\quad \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{PO} (\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid (Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle) \\ &\xrightarrow{DEL} Op4\langle CP\rangle + Op5\langle BTP\rangle \mid \overline{Op4}\langle CP\rangle + \overline{Op5}\langle BTP\rangle \xrightarrow{CP} 0 \mid 0 = 0\end{aligned} \tag{10}$$

以上两个推演过程也分别对应了服务 Vendor 与服务 NewCustomer 之间的两种不同的交互过程,即交互路径.由于式(9)和式(10)最后均演变成空进程,说明这两条交互路径均为有效交互路径,因此,两个服务是完全兼容的.尽管 NewCustomer 服务与图 2 所示的 Customer 服务相比在状态 *C* 只能发送消息 *CP* 而并不能发送 *BTP* 消息,但这并不影响与 Vendor 服务的正常交互,它与 Vendor 服务之间只存在上述两种可能的交互方式.

定理. 给定两个服务对应的进程表达式 *P* 和 *Q*,若这两个服务是可兼容的,则这两个服务进程必定满足条件 $P|Q \Rightarrow 0 (\Rightarrow \overset{def}{=} \xrightarrow{\tau}{}^{*}$,即⇒为迁移符号→的传递自反闭包).

证明:(1) 根据服务可兼容的定义可知,若两个服务是可兼容的,则两个服务在交互的过程中至少存在 1 条有效交互路径,使得两个服务均能从起始状态演变成结束状态.此时,进程 *P* 和 *Q* 可按照该有效交互路径产生的消息进行内部通信,从而使得 $P|Q$ 经过若干内部同步操作演变成空进程,即条件 $P|Q \Rightarrow 0$ 满足;(2) 若条件 $P|Q \Rightarrow 0$ 满足,则说明进程表达式 $P|Q$ 可通过内部同步操作(即进程 *P* 和 *Q* 之间进行同步通信)演变成空进程,而在演变过程中,*P* 和 *Q* 之间形成了一条消息序列.此时,这两个服务根据该消息序列也能使得两个服务均从起始状态到达结束状态,因此,这两个服务是可兼容的. □

定理表明,可以将服务之间的兼容性判定转化成对服务的并发组合进程的推演,并从推演的结果可以判定两个服务是否可兼容.而这一判定过程可以借用π演算的辅助工具,如 MWB(mobility workbench)[11]等自动完成.

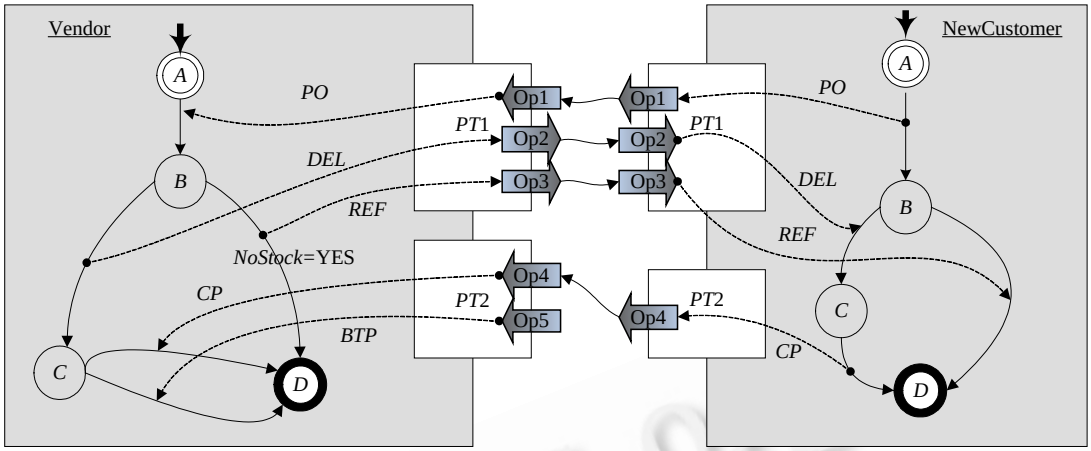


Fig.3 Interaction between Vendor and NewCustomer services

图 3 服务 Vendor 与服务 NewCustomer 之间的交互

2.4 Web服务行为兼容性的自动定量计算

为实现 Web 服务兼容性的自动定量计算,需要自动求得两个服务之间所有的交互路径,并检验每一条交互路径是否为有效交互路径.为此,可采用π演算的扩展规则对两个服务进程的并发组合进程进行变换.

扩展规则(expansion law). 设进程 $P=(P_1[f_1]\ldots P_n[f_n])\backslash L, n\geq 1$($\backslash L$ 表示集合 L 中的名字全部为受限名;f_i 为换名函数),则 P 可扩展成如下等价形式:

$$P=\sum\{f_i(\alpha).(P_1[f_1]|\ldots|P_i[f_i]|\ldots|P_n[f_n])\backslash L : P_i \xrightarrow{\alpha} P_i',\ f_i(\alpha)\notin L\cup\overline{L}\}+$$
$$\sum\{\tau.(P_1[f_1]|\ldots|P_i[f_i]|\ldots|P_j'[f_j]\ldots|P_n[f_n])\backslash L : P_i \xrightarrow{l_1} P_i', P_j \xrightarrow{l_2} P_j',\ f_i(l_1)=\overline{f_j(l_2)}, i<j\}.$$

扩展规则的目的是将若干进程并发组合而成的进程转变成若干个进程的和表达式.和表达式中任意一个子进程均表示了该并发进程的一个潜在的可能执行方式.将图 2 对应的如表达式(3)所示的并发进程作如下扩展,其中,$L=\{Op1,Op2,Op3,Op4\}$.值得注意的是,在使用扩展规则进行扩展时,我们将进程间内部同步操作形成的正常交互元记录在哑动作下标处,以此来表明交互路径的形成过程.

$$\begin{aligned}
P_{(V,C)} &= (Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle+Op5\langle BTP\rangle)+\overline{Op3}\langle REF\rangle)\,|\\
&\quad \overline{Op1}\langle PO\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))\backslash L\\
&= \tau_{\langle(CC.Op1,V.OP1,\{PO\})\rangle}.((\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle+Op5\langle BTP\rangle)+\overline{Op3}\langle REF\rangle)\,|\\
&\quad Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))\backslash L\\
&= \tau_{\langle(CC.Op1,V.OP1,\{PO\})\rangle}.(\tau_{\langle(V.Op2,CC.OP2,\{DEL\})\rangle}.((Op4\langle CP\rangle+Op5\langle BTP\rangle)\,|\\
&\quad (\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))+\overline{Op3}\langle REF\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))\backslash L\\
&= (\tau_{\langle(CC.Op1,V.Op1,\{PO\}),(V.Op2,CC.Op2,\{DEL\}),(CC.OP4,V.Op4,\{CP\})\rangle}+\tau_{\langle(CC.Op1,V.Op1,\{PO\}),(V.Op2,CC.Op2,\{DEL\}),(CC.Op5,V.Op5,\{BTP\})\rangle}+\\
&\quad \tau_{\langle(CC.Op1,V.OP1,\{PO\})\rangle}.\overline{Op3}\langle REF\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))\backslash L
\end{aligned}\tag{11}$$

扩展过程(11)的最后结果中共有 3 个子进程,其中有两个是空进程,它们对应了两条有效交互路径;此外,还包含了一个非空进程 $\tau_{\langle(CC.Op1,V.OP1,\{PO\})\rangle}.\overline{Op3}\langle REF\rangle.Op2\langle DEL\rangle.(\overline{Op4}\langle CP\rangle+\overline{Op5}\langle BTP\rangle))$,该进程表明,两个进程交互到 $\overline{Op3}\langle REF\rangle$ 时无法再彼此发生同步,即进程 P_V 选择通过通道 $Op3$ 发送 REF 消息时,进程 P_C 没有相应的通道可以接收,因此无法进一步实现内部同步通信.由此可知,该非空进程对应了一条非有效交互路径,这也验证了之前的结果,即这两个服务是局部兼容,且兼容度为 2/3.

考察图 3 的交互过程,同样利用扩展规则对式(8)的并发进程进行扩展,得到如扩展过程(12)所示的结果.

$$\begin{aligned}
P_{(V,NC)} &= (Op1\langle PO\rangle.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle) \mid \\
&\quad \overline{Op1}\langle PO\rangle.(Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle)) \setminus L \\
&= \tau_{\langle (C.Op1,V.OP1,\{PO\})\rangle}.(\overline{Op2}\langle DEL\rangle.(Op4\langle CP\rangle + Op5\langle BTP\rangle) + \overline{Op3}\langle REF\rangle \mid \\
&\quad Op2\langle DEL\rangle.\overline{Op4}\langle CP\rangle + Op3\langle REF\rangle) \setminus L \\
&= (\tau_{\langle (C.Op1,V.OP1,\{PO\}),(V.OP2,C.Op2,\{DEL\})\rangle}.((Op4\langle CP\rangle + Op5\langle BTP\rangle) \mid \overline{Op4}\langle CP\rangle) + \tau_{PO}.\tau_{REF}) \setminus L \\
&= \tau_{\langle (C.Op1,V.Op1,\{PO\}),(V.Op2,C.Op2,\{DEL\}),(C.OP4,V.Op4,\{CP\})\rangle} + \tau_{\langle (C.Op1,V.Op1,\{PO\}),(V.Op2,C.Op2,\{DEL\})\rangle}.Op5\langle BTP\rangle.\overline{Op4}\langle CP\rangle + \\
&\quad \tau_{\langle (V.Op1,C.Op1,\{PO\}),(V.Op3,C.Op3,\{REF\})\rangle}
\end{aligned} \tag{12}$$

扩展过程(12)的最后结果中包含两个空进程,分别对应一条有效交互路径;此外,还包含了非空进程:

$$\tau_{\langle (CB.Op1,V.Op1,\{PO\}),(V.Op2,CB.Op2,\{DEL\})\rangle}.Op5\langle BTP\rangle.\overline{Op4}\langle CP\rangle\ .$$

值得注意的是,该子进程并不表示服务 Vendor 与 NewCustomer 之间的一条交互路径,这是因为该子进程的第 1 个未执行的动作为 $Op5\langle BTP\rangle$,该动作是一个被动的消息接收动作,该子进程在两个服务的交互过程中是不会发生的,因此,该子进程并不对应一条交互路径.所以,服务 Vendor 与 NewCustomer 的交互路径数和有效几乎路径数均为 2,从而也证实了这两个服务是完全兼容的.算法 CDC 实现了两个服务之间兼容度的定量计算.

算法. CDC (compatibility-degree-calculator).

Input: Two Services A and B and their corresponding π-calculus processes P_A and P_B;

Output: The compatibility degree between A and B, i.e., $\varpi(A,B)$.

1. **SET** n as the number of interaction path between A and B;
2. **SET** m as the number of valid interaction path between s_1 and s_2;
3. **SET** $L=\varnothing$;
4. **SET** CN_1, CN_2 as the sets of channel names in P_A and P_B, respectively;
5. **FOR** each element $cn\in CN_1$
6. **IF** ($cn\in CN_2$)
7. put cn into L;
8. **SET** $P=(Ps_1 \mid Ps_2)\setminus L$;
9. Transform P into sum-process according to Expansion Law, thus get $P=\sum_i P_i$;
10. **FOR** each sub-process $P_i\in P$ **{**
11. **IF** ($P_i\equiv 0$) **{**
12. n++;
13. m++;
14. **} ELSE IF** p_i starts with an output prefix **{**
15. $n=n+1$;
16. **}**
17. **}**
18. **RETURN** $\varpi(A,B)=(n>0)?m/n:0$;

需要注意的是,对于以输入前缀为起始动作的子进程,并不对应两个服务间的一条交互路径,因此,在算法 CDC 的 14 行中仅计算以输出前缀为起始动作的子进程.若交互的两个进程的交互路径数目分别为 m 和 n,则算法 CDC 在第 9 行(即通过扩展规则进行扩展)的时间复杂度为 $O(n\times m)$,因此,算法 CDC 的时间复杂度为 $O(2\times m\times n)$.值得强调的是,算法 CDC 不仅能够定量计算两个服务之间的兼容度,而且还能得出两个服务之间的所有有效交互路径.

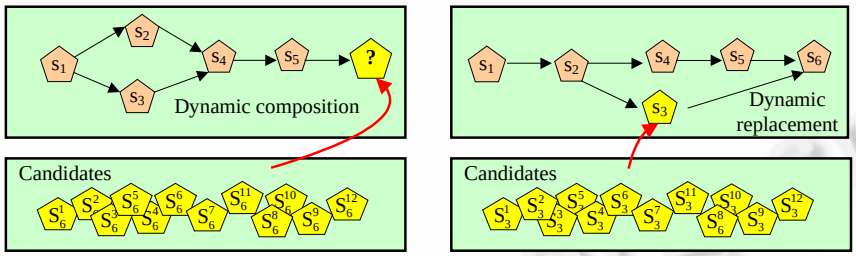
3 应用分析

Web 服务行为兼容性自动地定性判定和定量计算,对于实施服务的动态组合和服务的动态替换具有重要的指导作用.

对于服务的动态组合而言,如图 4(a)所示,为了从候选服务集合中选择一个最为合适的服务动态接入到组合服务中以达到服务增值的目的,可采用本文方法按照如下步骤选择最为合适的服务进行动态组合:1) 采用服务行为兼容的定性判定方法,逐一对候选服务与组合服务进行行为兼容性的定性判定;2) 若所有候选服务与组合服务之间都是完全不兼容的,则说明该组合服务在当前服务集合中无法再继续组装,动态组合过程随即结束;否则,将可兼容的候选服务加入服务集合 S;3) 采用服务行为兼容的定性计算方法,依次计算集合 S 中的每一个服务与组合服务之间的兼容度,将兼容度按降序排列;4) 若存在兼容度为 1 的服务,则选择该服务动态接入到组合服务中,从而进入下一步动态接入过程;若不存在兼容度为 1 的服务,则选择兼容度最高的服务动态接入到组合服务中,同时对该服务与组合服务之间的非有效交互路径进行异常处理,或者对此路径上的服务交互过程进行适配,从而最大限度地减少组合服务在执行过程中出现错误或者异常的可能.

而在服务组合的执行过程中,一旦发现某个服务变得不可用,则需要找到一个可以取而代之的服务进行动态替换,从而使得组合服务的执行得以继续下去.如图 4(b)所示,当组合服务的执行进入到服务 S_3 这一步时,发现 S_3 不可用,则此时需要从候选服务集中选择一个最佳服务进行动态替换.除了保证候选服务能够提供 S_3 拟被调功能之外,还要求保证候选服务与组合服务的交互是行为兼容的.此时,可采用服务动态组合中服务选取的类似过程选择最佳候选服务.

以上过程的分析和实例论证均说明了本文所提出的服务行为兼容性的定性判定与定量计算方法能够辅助用户正确且自动地完成服务动态接入与服务替换,一方面保证了服务组合建立结果的正确性与可用性,而另一方面能在动态开放、复杂难控的网络环境下确保服务组合正常、可靠地执行,完成业务逻辑,达到既定的业务目标.



(a) Dynamic service composition (b) Dynamic service replacement

(a) 服务的动态组合 (b) 服务的动态替换

Fig.4 Dynamic service composition and replacement

图 4 动态服务组合与替换

4 相关工作

构件行为是基于构件的软件系统设计和开发中需要考虑的一个重要方面[12,13],特别是构件行为兼容性的判定一直都是国内外学者研究的一个热点问题.目前,在这一问题上出现了一系列研究成果,如文献[14].但由于服务模型和语言与传统构件模型和语言具有较大的差异性,因此,这些成果均无法直接应用于服务行为兼容性的判定,这也引发了国内外学者对服务行为兼容性分析、验证和判定的研究.一般而言,服务行为兼容性的研究均基于某种形式化方法进行,纵观目前国内外相关的研究文献,Petri 网、自动机理论和进程代数是使用最多的 3 种形式化方法,本节将对这些代表性研究成果进行简单介绍.

文献[3]采用工作流 Petri 网对 Web 服务流程进行建模,将 Web 服务组合转化成工作流网的组合,从而将服

务组合中服务行为的兼容性验证和判定问题转变成工作流 Petri 网的活性(live)、有界性(bound)和死锁/活锁检验的问题,并通过 Petri 网已有工具完成验证和判定的过程.文献[4]则以服务描述语言 DAML-S 为基础,提出将 DAML-S 中的服务流程模型(原子流程和复合流程)通过工具自动转变成 Petri 网,进而利用 Petri 网对服务组合流程进行自动的定量分析、验证和仿真,从而完成流程中服务行为的兼容性验证和判定.

文献[5]将服务组合转化成有限状态机,之后,采用符号转移系统分析工具(labeled transition system analyzer, 简称 LTSA)对 FSP 进行编译,通过检查其活性、状态可达性和死锁情况来分析组合服务中各成员服务之间的行为兼容性.与此类似,文献[6,7]也都采用有限状态机分别对会话类 Web 服务和流程式服务进行建模,并以此来保证服务组合过程和服务发现过程中的服务兼容性.

文献[8]采用 Guarded 自动机模型对 BPEL 所描述的服务组合进行建模,并使用时序逻辑来描述交互过程需满足的目标属性(如要求某条交互路径可终结、某状态可达等),然后将该模型转换到 Promela 语言,并以此作为模型检验工具 SPIN 的输入,来验证模型是否满足目标属性,因此,该方法可以判定 BPEL 中各成员服务在某一交互路径上是否行为兼容.但在采用该方法时,需要事先采用时序逻辑描述出服务间的交互路径,因此无法实现服务行为兼容性完全的自动判定.文献[9]也采用了自动机来刻画两个服务之间的交互过程,进而判断两个服务之间的行为兼容性.

以上基于 Petri 网、有限状态机和自动机的方法在服务行为的描述上采用图形方式较为直观,但当服务行为和服务间的交互过程变得较为复杂时,这种图形刻画的方式容易出现状态空间爆炸,其计算、验证和判定的复杂度也随即增加.因此,有研究人员提出采用进程代数的方法(如通信系统演算 CCS、通信顺序进程 CSP 和π演算)来刻画服务行为和服务之间的交互[15,16].文献[17]采用 CCS 对 Web 服务编排协议 WSCI 进行形式化建模,并对 WSCI 中服务行为兼容性进行描述性说明,但缺乏进一步严格判定的方法.文献[18]从特征层、会话层和协议层 3 个层次描述了服务接口,其中,协议层采用符号迁移系统 LTS 进行刻画,并依据进程代数的互模拟理论给出服务协议相容的判定条件.

除了上述采用成熟的形式化方法和工具进行服务兼容性研究之外,也有研究人员通过自定义的形式化语言或者表达方式进行服务行为的刻画,在此基础上提出算法进行行为兼容性判定.文献[19]从服务接口的语法信息、条件约束信息和协议信息 3 个层面进行刻画,其中,协议信息即服务行为.作者提出了协议规格语言描述协议过程和协议包含的属性,之后提出协议检验算法,通过一系列协议验证规则完成协议正确性的验证,从而实现服务协议的兼容与可替换的判定.

上述相关工作都为本文进一步展开服务行为兼容的判定与计算提供了较好的研究基础.而与上述工作相比,本文的特色之处主要体现在两个方面:1) 采用了π演算刻画服务行为和服务交互行为,其表达能力与其他形式化方法相当,但因其类似数学表达式的文本刻画方式,避免了状态空间爆炸的缺陷,判定的复杂度低,并且采用较为成熟的π演算验证工具实现服务行为兼容性的自动判定;2) 方法不仅支持服务行为兼容性的定性判定,还能实现定量的计算,而且在定量计算过程中能够明确服务之间的有效交互路径和非有效交互路径,从而指明服务交互过程中何种情况下无法正常完成交互、何种情况下将导致交互异常的出现,这对于服务的动态组合与动态替换具有重要的参考意义.

5 结束语

本文围绕 Web 服务组合中各成员服务间的行为兼容性展开研究,提出了一种基于π演算的 Web 服务行为兼容性的判定与计算方法.未来的研究工作主要包含两个方面:1) 进一步研究一个服务与多个服务的兼容性判定和计算.考察一个服务 s_0 与多个服务组成的集合 $S=\{s_1,s_2,\ldots,s_n\}$的交互,我们将集合 S 中的每一个服务对应的服务视图通过算法 ServiceView2Process 分别得到其对应的π演算进程,然后将这些进程通过π演算的并行操作组合成一个进程表达式 p,从而将一个服务与多个服务之间的交互过程变成一个进程与多个进程组合而成的复合进程之间的推演过程,进而可以利用本文方法实现一个服务与多个服务之间的兼容性判定与计算;2) 对于不兼容的或者局部兼容的服务间交互,如何基于π演算理论实现服务的自动适配提高服务间交互的可兼容性,文献

[20]为本文展开这一方面的工作提供了较好的参考.我们将在此基础上,在两个局部兼容或者完全不兼容的服务之间增加一个适配进程,以协调两个服务进程之间的交互,从而减少服务交互异常的出现.

References:

[1] Dustdar S, Schreiner W, Schreiner W. A survey on Web services composition. Int'l Journal of Web and Grid Services, 2005,1(1): 1−30.

[2] Bultan T, Fu X, Hull R, Su J. Conversation specification: A new approach to design and analysis of E-service composition. In: Hencsey G, White B, eds. Proc. of the 12th Int'l World Wide Web Conf. (WWW). Budapest: ACM, 2003. 403−410.

[3] Martens A. On compatibility of Web services. Petri Net Newsletter, 2003,65:12−20.

[4] Narayanan S, McIlraith S. Analysis and simulation of Web services. Computer Networks, 2003,42(5):675−693.

[5] Foster H, Uchitel S, Magee J, Kramer J. Compatibility verification for Web service choreography. In: Zhang LJ, ed. Proc. of the Int'l Conf. on Web Service (ICWS). San Diego: IEEE Computer Society, 2004. 738−741.

[6] Zhang WT, Peng Y, Chen JL. Interface compatibility and composition of session-oriented E-service. Chinese Journal of Computers, 2006,29(7):1047−1056 (in Chinese with English abstract).

[7] Wombacher A, Fankhauser P, Mahleko B, Neuhold E. Matchmaking for business processes based on choreographies. Int'l Journal of Web Services Research, 2004,1(4):14−32.

[8] Fu X, Bultan T, Su JW. Analysis of interacting BPEL Web services. In: Hencsey G, White B, eds. Proc. of the Int'l World Wide Web Conf. (WWW). Budapest: ACM, 2004. 621−630.

[9] Shi YL, Zhang L, Liu FF, Lin LL, Shi BL. Compatibility analysis of Web services. In: Skowron A, ed. Proc. of the IEEE/WIC/ACM Int'l Conf. on Web Intelligence. Compiegne: IEEE Computer Society, 2005. 483−486.

[10] Milner R, Milner-Gulland R. Communicating and Mobile Systems: The π-Calculus. Cambridge University Press, 1999.

[11] Victor B, Moller F. The mobility workbench: A tool for the π-calculus. In: Dill DL, eds. Proc. of the Int'l Conf. on Computer Aided Verification. Springer-Verlag, 1994. 428−40.

[12] Mei H, Shen JR. Progress of research on software architecture. Journal of Software, 2006,17(6):1257−1275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1257.htm

[13] Hu J, Zhang Y, Yu XF, Zhang T, Li XD, Zheng GL. Scenario-Driven component behavior derivation. Journal of Software, 2007, 18(1):50−61 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/50.htm

[14] Hu HY, Lü J, Ma XX, Tao XP. Study on behavioral compatibility of components in software architecture using object-oriented paradigm. Journal of Software, 2006,17(6):1276−1286 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1276.htm

[15] Bordeaux L, Salaün G. Using process algebra for Web services: Early results and perspectives. In: Nascimento MA, eds. Proc. of the 5th VLDB Workshop on Technologies for E-Services (VLDB-TES). Toronto: Springer-Verlag, 2004.

[16] Yang H, Zhao X, Qiu Z, Pu G. A formal model for Web service choreography description language (WS-CDL). In: Feig E, ed. Proc. of the IEEE Int'l Conf. on Web Services. Chicago: IEEE Computer Society, 2006.

[17] Brogi A, Canal C, Pimentel E, Vallecillo A. Formalizing Web service choreographies. In: Proc. of the 1st Int'l Workshop on Web Services and Formal Methods. Pisa: Elsevier, 2004.

[18] Chen Z, Wang J, Dong W, Qi Z. An interface model for service-oriented software architecture. Journal of Software, 2006,17(6): 1459−1469 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1459.htm

[19] Beyer D, Chakrabarti A, Hengzinger T. Web service interfaces. In: Carr L, eds. Proc. of the Int'l World Wide Web Conf. (WWW). Chiba: ACM, 2005. 148−159.

[20] Nezhad H, Benatallah B, Martens A, Curbera F, Casati F. Semi-Automated adaptation of service interactions. In: Williamson CL, eds. Proc. of the Int'l World Wide Web Conf. (WWW). Banff: ACM, 2007. 993−1002.

附中文参考文献:

[6] 张文涛,彭泳,陈俊亮.会话类 E-Service 的接口兼容和服务组合分析.计算机学报,2006,29(7):1047−1056.

[12] 梅宏,申峻嵘.软件体系结构研究进展.软件学报,2006,17(6):1257−1275. http://www.jos.org.cn/1000-9825/17/1257.htm

[13] 胡军,张岩,于笑丰,张天,李宣东,郑国梁.场景驱动的构件行为抽取.软件学报,2007,18(1):50−61. http://www.jos.org.cn/1000-9825/18/50.htm

[14] 胡海洋,吕建,马晓星,陶先平.面向对象范型体系结构中构件行为相容性研究.软件学报,2006,17(6):1276−1286. http://www.jos.org.cn/1000-9825/17/1276.htm

[18] 陈振邦,王戟,董威,齐治昌.面向服务软件体系结构的接口模型.软件学报,2006,17(6):1459−1469. http://www.jos.org.cn/1000-9825/17/1459.htm

邓水光(1979－),男,浙江杭州人,博士,CCF 学生会员,主要研究领域为流程管理,面向服务的计算.

李莹(1973－),男,博士,副教授,主要研究领域为中间件技术,软件体系架构,编译技术.

吴健(1976－),男,副教授,CCF 会员,主要研究领域为 Web 服务,网格计算,数据挖掘.

邝砾(1982－),女,博士生,CCF 学生会员,主要研究领域为面向服务的计算.

吴朝晖(1966－),男,教授,博士生导师,CCF 高级会员,主要研究为网格计算,嵌入式计算,普适计算.