

基于进程代数的 Web 服务合成的替换分析

刘方方 史玉良 张 亮 施伯乐

(复旦大学计算机与信息技术系 上海 200433)

摘 要 Web 服务合成是 Web 服务研究领域的热点问题, Web 服务的替换与 Web 服务合成密切相关且对于 Web 服务合成系统的健壮性有重要作用. 使用进程代数作为形式化的工具描述 Web 服务, 对合成中的行为特征进行推导, 分析由于异步交互对 Web 服务合成设计与实现产生的影响. 在此之上, 基于服务合成正确性的定义, 提出一种一致性关系, 若新服务与将要被替换的参与合成的服务之间存在这种关系, 那么替换后的服务合成仍然是正确的, 并且替换是上下文无关的.

关键词 Web 服务; 合成; 替换; 进程代数

中图法分类号 TP311

Substitution Analysis of Web Service Composition via Process Algebra

LIU Fang-Fang SHI Yu-Liang ZHANG Liang SHI Bo-Le

(Department of Computing and Information Technology, Fudan University, Shanghai 200433)

Abstract Web services composition is a key issue in Web service research area. Substitution of service is closely related with composition and important to robustness of service composition. In this paper, we use process algebra as formalism foundation modeling and specifying Web services and reasoning on behavioral features of Web services composition. We analyze some cases that have effects on design and implementation of composition. Upon that, and based on definition of composition, we study substitution. As to the problem of how to substitute a component Web service, we present a relation. Any new selected Web services can substitute old component service independent of context and take part in composition successfully in the case that they satisfy criteria of this relation.

Keywords Web service; composition; substitution; process algebra

1 引 言

Web 服务的出现为网络分布应用程序开发提供了一种非常有用的方法. Web 服务架构也带来许多值得研究的问题, 热点之一是 Web 服务合成. 当用户需求无法由单一的 Web 服务实现时, 需要不同的 Web 服务一起工作以满足用户需求, 这就是

Web 服务合成所阐述的问题. Web 服务替换是与 Web 服务合成密切相关的一个问题, “是一个硬币的两面^[1]”. 一个由 Web 服务合成的系统, 当其中的关键服务遇到来自网络的威胁和本身的软硬件故障, 无法继续提供服务时, 需要选取新的服务来替换, 而替换的成功与否取决于替换之后的合成是否仍是正确的.

Web 服务标准, 如 W3C 的接口描述语言 WSDL,

用于消息传递的 SOAP 和 OASIS 的服务注册机制 UDDI,建立了 Web 服务架构的基础.在这样的架构中,Web 服务是自治、异步和松耦合的,Web 服务之间的通信又是异步的,这些特征使得 Web 服务合成变得复杂.Web 服务合成的研究,已有了一些标准,如 BPEL4WS^[2],WSC1^[3],它们主要用来描述 Web 服务合成的实现流程.形式化方法被用来分析 Web 服务合成的行为特性.文献[4]提出自上而下的方法分析异步交互下的 Web 服务合成.文献[5]用 pi 算子解决 Web 服务的自动发现、合成与验证. Petri 网在文献[6]中用于为 Web 服务的合成进行建模与推导.文献[7]提出 Colombo 框架,以建立一种完整、正确且可终止的 Web 服务合成算法.文献[8]对众多使用进程代数的研究方法予以总结.上述研究仅针对 Web 服务合成未考虑替换问题.

Web 服务的替换是基于合成的.虽然 WSDL 层面的服务替换只检查 Web 服务的接口是否包含将要被替换的 Web 服务的操作,但在 Web 服务行为特性层面情况却是不一样的^[1].替换后的 Web 服务合成中消息的执行顺序是否仍可以得到保证,是否所有的消息都被处理了等等问题都需要解决.仅从 WSDL 层面不能保证 Web 服务合成在替换之后仍是正确的,需要考虑服务的行为特性.

已有研究表明,Web 服务合成验证需花费的代价随参与合成的服务数量增加快速增长^[9].若替换频繁发生,不断重复的合成正确性验证必然会影响系统运行效率.为避免上述情况,服务替换需是上下文无关的,即不需合成验证的情况下就可判断选取的服务是否能够正确进行替换.此处上下文指合成系统中与要被替换的服务交互的所有部件服务.

本文中,基于文献[1,10-11]的研究工作,针对 Web 服务的自治异构和服务之间消息异步交互特性,并受到文献[12]中要替换的服务需满足已有服务的描述和基于消息传递的系统中程序设计与实现关系^[13-14]的启发,我们提供一种基于进程代数的 Web 服务合成的替换分析方法.替换是与合成分不开的,替换的正确性由合成正确性来判定.因此首先,为使用文献[1]中 Web 服务合成的正确性描述,我们分析消息异步传递对合成设计与实现的影响,补充文献[1]中 Web 服务的合成模型.在有了 Web 服务合成正确性的定义之后,Web 服务合成的替换分析也有了依据,上下文无关替换取决于已有的服务与要对其进行替换的服务之间的关系,对此,我们提出一种一致性关系,并证明如果已参与合成的

Web 服务与将要对其进行替换的 Web 服务之间存在这种一致性关系,服务替换即是上下文无关的.

本文第 2 节为相关工作介绍;第 3 节,根据文献[1]中的研究工作,分析 Web 服务异步交互对合成的影响,定义服务合成;给出可以保证上下文无关替换的一致性关系,并给出判断算法;第 4 节为结论与未来的工作.

2 相关工作

就我们所知,将合成与替换一起进行的研究有代表性的是文献[1,10,12].文献[12]中给出了 Web 服务相容性(compatibility)的一种定义,提供了相容性验证算法.作者用形式化方法描述 Web 服务,以“契约(contract)”形式给出目标服务,讨论参与合成的服务为实现目标服务的契约所要满足条件,即新服务需要与已有服务的契约保持一致方能替换已有服务.文中 Web 服务之间的通信是同步的.文献[10]的主要研究内容是两个 Web 服务之间的相容性和可替换性(substitutability).作者用进程代数 LTS 描述 Web 服务,服务之间相互操作的行为特性按 LTS 的推导规则进行.文中给出两个 Web 服务之间三种不同层次的相容性定义,这三层定义条件限制逐步放宽,以适应 Web 服务特性决定的 Web 服务的设计和实现的多样性.服务可替换性基于服务相容性.对应相容性,作者给出可替换性概念,并说明可替换性分为上下文无关和上下文相关.在条件限定严格的相容性定义下,服务替换可以是上下文无关的.文献[10]中的分析主要针对同步情况,未考虑异步环境带来的一些问题,也没有详细说明上下文无关的可替换性需满足的条件.文献[1]中,作者使用进程代数 CCS(Communication and Concurrency System)为 Web 服务的合成描述语言 WSC1 提供形式化描述,Web 服务的行为特性按 CCS 中的规则推导.文献[1]中作者不再区分服务间不同层面的消息传递,只关注所有参与合成的服务的消息是否以正确的顺序交互且每个服务均达到终止状态.作者给出了服务正确合成的要求,称可正确合成的服务为相容的.在相容性基础上,讨论可替换性.认为满足一种 subtype 关系的情况下,服务替换可以上下文无关进行.但文献[1]中虽然描述 Web 服务时区分了全局/局部选择分支,作者没有明确考虑这是异步环境对服务合成的影响所决定的,且按文献[1]中的模型与服务相容性的要求进行的服务合成,在异步

环境中不一定正确. 而对于可替换性, 作者虽列出 subtype 关系的 3 个条件, 但未证明此关系是上下文无关替换的充分条件.

3 Web 服务合成的替换

3.1 CCS 基础知识

文献[1]中, 使用 CCS 的过程和 CCS 的推导规则描述 Web 服务, 推理 Web 服务合成的行为特性. 因此, 我们首先介绍 CCS^[15]的基础知识.

一个 CCS 过程(由 P, Q, R, M 等表示)在 CCS 中定义如下:

$$P ::= 0 \mid \alpha.Q \mid P+Q \mid P \parallel Q \mid P \setminus sm,$$

$$\alpha ::= a?(x) \mid a!(x) \mid \tau.$$

α 表示通过一个信道或发送消息 $a!(x)$, 或接收消息 $a?(x)$, 或是过程内部的隐藏操作 τ . 一个终止的过程记为 0. 通常, 一个 CCS 过程由一系列的形如 $\alpha.P$ 的过程组成. 一个过程可能执行选择操作 $P+Q$; 或者是由一些子过程做并行合成: $P \parallel Q$. 一个过程可能会有一些限制, 用 $P \setminus sm$ 表示, sm 为信道名称集合, 意思是过程 P 的一个子过程想通过属于 sm 的一个信道 m 发送消息, 只有当存在另一个 P 的子过程可以通过信道 m 接收消息时才能成功地执行. 这样由同一个过程的两个子过程之间发送和接收消息的动作称为隐藏动作, 用 τ 表示.

因为消息传递过程中的参数及其数据类型不对 Web 服务的行为特性分析设计造成影响, 因而为模型分析之便, 下文中省去消息信道中的参数.

CCS 的一些推导规则如下:

$$\text{规则 1. } \frac{}{a.P \xrightarrow{a} P};$$

$$\text{规则 2. } \frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'};$$

$$\text{规则 3. } \frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q};$$

$$\text{规则 4. } \frac{Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P \parallel Q'};$$

$$\text{规则 5. } \frac{P \xrightarrow{a?(y)} P', Q \xrightarrow{a!(x)} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q' \{x/y\}}.$$

规则 1 表明一个过程 $a.P$ 可以转化为过程 P ; 规则 2 表示含有选择结构的过程沿着某一支进行转化; 规则 3、规则 4 和规则 5 表示合成的过程可以通过活动 α (接收或发送消息)或者内部隐藏动作 τ

进行转化. 规则 3 和规则 4 是对称的.

用 WSCI 描述的 Web 服务可以转变成为一个 CCS 过程. 如文献[1]中的 *BookingTrip* 服务: 服务提供者收到订票请求后发送确认, 如果服务可用, 订票请求被接收, 否则, 订票请求被拒绝.

BookingTrip =

bookingRequest?.bookingAck!.(τ .bookingConfirm!.0 + τ .bookingRefusal!.0).

合成就是几个 Web 服务的交互过程. 如旅行者 *Traveler* 的操作流程描述如下:

Travller =

bookingRequest!.bookingAck?.(bookingConfirm?.0 + bookingRefusal?.0).

它们的合成 *BookingTrip* \parallel *Travler* 经过隐藏操作之后, *BookingTrip* 和 *Traveler* 都到达终止状态.

3.2 Web 服务的合成与替换分析

有了上述准备之后, 文献[1]中, 作者给出了服务合成正确性的描述, 并在此基础上分析替换. 由于某些情况下, 合成过程可能不会终止而是不断地重复运行下去, 这种情况在客户/服务架构中最为常见, 为此, 作者从否定的描述——失败(failure)开始来探讨 Web 服务合成的正确性. 如果服务的合成在经过有限次的隐藏操作之后, 未到达终止状态 0, 或是某个重复出现的状态, 而又无法与合成系统内部的服务继续交互执行下去, 那么合成就是失败的. 因此, 合成的正确性描述为: 如果合成在经过有限次的隐藏操作之后不会处于失败状态, 那么合成就是成功的.

同时, 文献[1]中区分了一个 Web 服务中的全局/本地(global/local)选择. 本地选择分支是服务中的条件分支结构(WSCI 中 switch 结构), 每个分支都带有隐藏操作 τ 表示选择的条件, 如上节中的 *BookingTrip* 服务的选择分支. 在分析 Web 服务合成时, 每一个本地选择分支都要考虑, 因为系统无法预见在执行的时候究竟哪一个分支被选择, 为避免合成系统运行出现错误, 必须考虑每个本地选择. 而全局选择分支, 指的是一个 Web 服务等待几个输入消息中的一个到达(WSCI 中 choice 结构), 如上节中 *Traveler* 的操作流程, 只有当合成系统中存在有部件服务提供发送消息到此消息信道的操作, Web 服务才会选择这个全局分支进行下去, 也就是说只有在合成系统中还有其它的服务中有向此选择分支发送消息的操作时, 才需考虑这个分支.

但是,在 Web 服务实际环境下,消息是异步传递的,这使得在合成设计时,按照合成正确性描述判断是正确的合成,在实际执行的过程中,却会出现不正确的结果.下面看一个例子,这个例子表明,即使按照文献[1]中的 Web 服务合成的正确性描述,参与合成的 Web 服务交互之后不会处于失败状态,但如果 Web 服务的模型中没有对异步交互的消息进行缓存,具体的执行过程中,Web 服务可能无法正确合成.图 1 中,3 个要合成的 Web 服务 W_{s_1} , W_{s_2} , W_{s_3} 由 3 个过程 P , Q 和 R 表示.可以看到,在设计时, $P \parallel Q \parallel R = a?.b? \parallel a! \parallel b!$, 经过状态转换后可以到达终止状态 0. 但是在实际执行中,如果 Web 服务 W_{s_3} 通过信道发送的消息 b 早于服务 W_{s_2} 通过信道发送的消息 a 到达 W_{s_1} , W_{s_1} 不缓存消息 b , 而是丢弃,则合成失败.

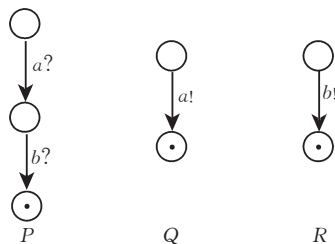


图 1 3 个 Web 服务合成

未考虑异步交互的影响会造成实际中合成结果与设计不符.下节中我们修正文献[1]中 Web 服务模型的不足.此外,全局/本地选择的区分实质上也是由于消息异步传递引起的,关于这点也将在下节中解释.

关于服务替换,文献[1]的作者基于服务合成正确性描述,提出了一种 subtype 关系分析替换.服务 $W_{s'_1}$ 要替换 W_{s_1} , 如果它们之间满足 3 个条件: (1) 服务 $W_{s'_1}$ 的任意全局选择分支, $W_{s'_1}$ 都能提供; (2) $W_{s'_1}$ 不会扩展 W_{s_1} 的操作,即 $W_{s'_1}$ 的发送或接收消息操作都会出现在 W_{s_1} 中; (3) W_{s_1} 终止, $W_{s'_1}$ 也终止.作者根据已有的工作基础,认为满足以上 3 个条件, $W_{s'_1}$ 可以替换 W_{s_1} .

分析上述 3 个条件,条件(1)要求 W_{s_1} 的全局选择分支, $W_{s'_1}$ 都要提供,这是因为全局选择分支在等待 W_{s_1} 以外的合成系统的服务发送消息,如果有某个等待接收消息的分支 $W_{s'_1}$ 不提供,就无法处理发送来的消息,合成失败.条件(2)是限定 $W_{s'_1}$ 中不要出现新的消息操作,使得合成系统内的其它服务无法处理,造成合成失败.条件(3)则是保证 $W_{s'_1}$ 不仅可以完成一次会话,且当 W_{s_1} 终止的时候,也不会重

复或继续运行.

然而,上述条件中没有提到, $W_{s'_1}$ 虽然不能扩展 W_{s_1} 的操作,可任何出现在 W_{s_1} 中的关键操作, $W_{s'_1}$ 也必须具备.这些关键操作包括全局选择的全部分支,这一点条件(1)可以保证,而内部选择同样需要考虑.如果合成系统中有服务等待 W_{s_1} 的内部选择至少发送某种类型的消息方可继续执行,而 $W_{s'_1}$ 没有对应的发送消息行为,那么替换后的合成无法继续进行.因此内部选择是必须进行的操作,不能够忽略.此外,其它类型的关键操作也要得到保证.为此,我们在下节中提出一种一致性关系,并证明它可以保证上下文无关的替换是正确的.

3.3 异步合成的上下文无关替换

为了使用文献[1]中合成正确性的描述,Web 服务需要为接收到的消息设置缓冲区.但是,即使在有缓冲区的情况下,缓冲不区分消息类型,即每个服务只有一个消息队列,图 1 中的情况仍无法处理.为此,根据文献[14]中异步消息传递系统的做法,我们有如图 2 中所示的 Web 服务交互模型,每一个 Web 服务为每种接收到的消息类型都设置缓存区,这样图 1 中的消息 b 可以被缓存,待消息 a 到达并被处理之后再取出消息 b 取出进行处理,合成可以成功地进行.

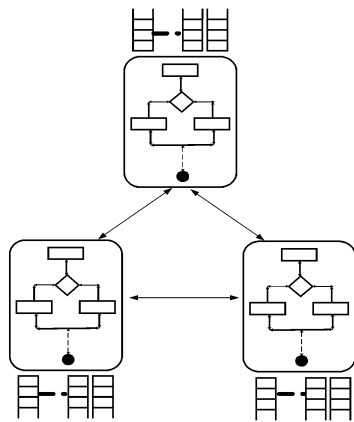


图 2 Web 服务合成模型

区分全局/本地分支选择也是异步交互所必需的.本文中,我们将其称为外部/内部(external/internal)选择.图 3 中两个 Web 服务 W_{s_1} , W_{s_2} 由过程 P 与 Q 表示,由于异步交互,它们的合成过程 $P \parallel Q = (a?.b! + \tau.b!) \parallel (b? + \tau.a!.Q)$ 在两种分支路径下都可以转换到终止状态.一种是 W_{s_1} 通过内部决定发送消息 b 给 W_{s_2} , W_{s_2} 接收消息,两者都到达终止状态.另一种情况是 W_{s_2} 通过内部决定发送消息 a 给 W_{s_1} , W_{s_1} 接收消息后,再发送消息 b 给 W_{s_2} , 此时已

经完成了一次任务执行的 W_{s_2} , 在接收到新的消息 b 后, 再次启动服务处理消息 b , 并且达到终止状态. 我们可以看到合成设计时的期望是所有参与合成的服务在一次会话过程中完成消息的处理, 虽然在后一种情况中, 合成最终终止, 但是合成的结果, 由于服务之间的异步通信, 却并不符合设计要求. 为了避免这种情况引起的混乱, 需要区分 Web 服务中的外部选择 (external choice) 和内部选择 (internal choice). 外部选择接收到的消息由外部环境决定; 内部选择发送的消息由 Web 服务本身决定. 限定 Web 服务中两种选择不能同时存在.

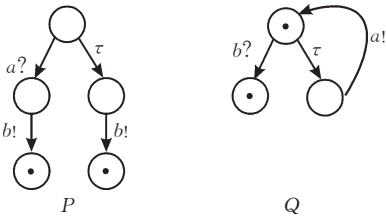


图 3 外部选择与内部选择对 Web 服务的影响

有了上述限定, 对应于文献[1]中 Web 服务正确合成的描述, 我们给出形式化的定义.

定义 1. Web 服务 $W_{s_1}, W_{s_2}, \dots, W_{s_l}$, 由 CCS 过程 M_1, M_2, \dots, M_l 表示, $W_{s_1}, W_{s_2}, \dots, W_{s_l}$ 的合成 $P = M_1 \parallel M_2 \parallel \dots \parallel M_l$ 是失败的, 如果 P 经过有限次的隐藏操作后到达失败状态, 即 P 经过有限次的隐藏操作后不处于终止状态 0, 而又无法再继续执行下去.

定义 2. Web 服务 $W_{s_1}, W_{s_2}, \dots, W_{s_l}$, 由 CCS 过程 M_1, M_2, \dots, M_l 表示, $W_{s_1}, W_{s_2}, \dots, W_{s_l}$ 的合成 $P = M_1 \parallel M_2 \parallel \dots \parallel M_l$ 是正确的, 如果 P 经过有限次的隐藏操作后不会到达失败状态.

对于 Web 服务替换, 为了做到上下文无关, 我们定义一种一致性关系, 它有 4 个条件, 包括了文献[1]中定义的 subtype 关系中的 (1), (3), 并保证 W_{s_1} 中的关键操作 W_{s_1}' 都具备, 同时对于内部选择, 可以放松限制, 只需执行至少一个内部选择分支即可, 且可以证明要替换的服务与已有的服务之间存在一致性关系时, 替换上下文无关.

Web 服务 $W_{s_1}, W_{s_2}, \dots, W_{s_l}$, 用 CCS 过程 M_1, M_2, \dots, M_l 表示, 它们的合成记为 $P = M_1 \parallel M_2 \parallel \dots \parallel M_l$, 用 R 表示服务 W_{s_1} 的合成上下文 M_2, M_3, \dots, M_l , 合成可以记为 $P = M_1 \parallel R$.

定义 3. Web 服务 W_{s_1}, W_{s_1}' 用过程 P 和 Q 表示, Q 与 P 是一致的, 记为 $Q \triangleright P$, 如果下列条件满足:

C1. 如果 $Q \xrightarrow{\tau^* a} Q'$, 那么存在 P' , 使得 $P \xrightarrow{\tau^* a} P'$, 且 $Q' \triangleright P'$.

C2. 如果 P 有外部选择 $a_1?.P_1 + a_2?.P_2 + \dots + a_n?.P_n$, 那么对每一个 $1 \leq i \leq n$, 存在 Q_i , 使得 $Q \rightarrow Q'$ 且 $Q' \xrightarrow{\tau^* a_i ?} Q_i$.

C3. 如果 P 有内部选择 $\tau.b_1!.P_1 + \tau.b_2!.P_2 + \dots + \tau.b_n!.P_n$, 那么存在一个 $1 \leq i \leq n$, 使得 $Q \rightarrow Q'$ 且 $Q' \xrightarrow{\tau^* b_i !} Q_i$.

C4. 如果 P 终止, 那么 Q 也终止.

定理 1. Web 服务 W_{s_1} 与 W_{s_1}' , 由 CCS 过程 M_1 与 M_{sub_1} 表示, R 代表服务 W_{s_1} 的合成环境, 如果 $M_{sub_1} \triangleright M_1$, 且 $M_1 \parallel R$ 是正确的, 那么 $M_{sub_1} \parallel R$ 也是正确的.

证明. 用反证法. 假设合成 $M_{sub_1} \parallel R$ 不正确, 根据定义 1, 存在 M_{sub_1}' 和 R' , $M_{sub_1} \parallel R \xrightarrow{\tau^*} M_{sub_1}' \parallel R'$, 而 $M_{sub_1}' \parallel R'$ 处于失败状态, 即没有到达终止状态或是重复出现的状态, 又无法继续运行下去.

记 $\alpha = a_1 a_2 \dots a_k$, 其中 $a_i (1 \leq i \leq k)$ 表示 M_{sub_1} 中的发送或接收消息操作, $\bar{a} = \bar{a}_1 \bar{a}_2 \dots \bar{a}_k$, $\bar{a}_i (1 \leq i \leq k)$ 是 $a_i (1 \leq i \leq k)$ 的逆操作, 即 a_i 表示发送消息, \bar{a}_i 表示接收消息, 反之亦然, 使 $M_{sub_1} \xrightarrow{\alpha} M_{sub_1}'$, $R \xrightarrow{\bar{\alpha}} R'$.

由条件 $M_{sub_1} \triangleright M_1$, 根据定义 3 中的 C1, 若 $M_{sub_1} \xrightarrow{\tau^* a_1} M_{sub_1}''$, 我们可得到 $M_1 \xrightarrow{\tau^* a_1} M_1''$, 且 $M_{sub_1}'' \triangleright M_1''$. 因此, 对于 $a_i (1 \leq i \leq k)$, 有 $M_1 \xrightarrow{\tau^* a_1 \tau^* a_2 \dots \tau^* a_k} M_1'$ 且 $M_{sub_1}' \triangleright M_1'$, 进而, $M_1 \parallel R \xrightarrow{\tau^*} M_1' \parallel R'$. 根据已知条件, $M_1' \parallel R'$ 不会处于失败状态.

现在, 我们有 $M_{sub_1}' \parallel R'$ 是失败的, $M_{sub_1}' \triangleright M_1'$.

如果 M_{sub_1}' 处于终止状态而 R' 没有, 据 $M_{sub_1}' \triangleright M_1'$ 和定义 3 中的 C4, M_1' 也处于终止状态. 而由 $M_1 \parallel R$ 是正确的, 可知 R' 也是终止的, 与假设矛盾, 所以不会出现 M_{sub_1}' 处于终止状态而 R' 不处于终止状态的情况. 同样, 也不会有 R' 处于终止状态而 M_{sub_1}' 不处于终止状态的情况.

如果 M_{sub_1}' 等待外界发送消息 b_1 , 而 R' 不能发送消息 b_1 . 据 $M_{sub_1}' \triangleright M_1'$, M_1' 也存在等待接收消息 b_1 的操作. 因为 $M_1' \parallel R'$ 是正确的, 如果 b_1 是 M_1' 唯一可接收的消息, 则 R' 一定可以发送消息 b_1 , 与假设矛盾. 如果 b_1 不是 M_1' 唯一可接收的消息, 则必

定存在其它可以由 M'_1 接收的、 R' 发送的消息 b_2 ，而根据定义 3 中的 C2，得： $M_{sub'_1}$ 必定可以接收 R' 发送的消息 b_2 ，即 $M_{sub'_1} \parallel R'$ 可以继续执行，不处于失败状态，与假设矛盾。

同样，若 $M_{sub'_1}$ 发送消息 c_1 ，而 R' 无法接收 c_1 。据 $M_{sub'_1} \triangleright M'_1$ ， M'_1 也存在发送消息 c_1 的操作，因为 $M'_1 \parallel R'$ 是正确的，只要是 M'_1 发送的消息，如定义 3 中的 C3，则 R' 一定可以接收，所以 R' 可以接收 c_1 ，与前提矛盾。

由上可知，在任何情况下假设都是不成立的。所以如果 $M_{sub_1} \triangleright M_1$ ，且 $M_1 \parallel R$ 是正确的，那么 $M_{sub_1} \parallel R$ 也是正确的。证毕。

利用上述定理，在进行 Web 服务替换时，不用考虑服务的合成上下文，且不用在服务执行的过程中去验证服务的正确性。只需找到和已有服务存在一致性关系的服务，就可以判定替换后的合成正确与否。仍然以订票系统为例，如果订票系统定义为 $BookingTrip' =$

$bookingRequest?.bookingAck!.bookingConfirm!.0$ ，根据定义 3， $BookingTrip' \triangleright BookingTrip$ ，那么 $BookingTrip'$ 可以替换 $BookingTrip$ ，而 $BookingTrip \parallel Traveler$ 是正确的。

图 4 给出了判断两个由 CCS 过程描述的 Web 服务是否存在一致性关系的算法。

```

Set U; //for states already satisfied cond. C1 of coformance
relation
Set C; //for all channels appearing in composite processes
Set T and J;
Conformance (Process P, Q)
1. if P is structure congruence with 0, i. e.  $P \equiv 0$ 
   and Q only has derivation  $Q \xrightarrow{\tau^*} Q'$  and  $Q' \equiv 0$ 
   then return P and Q have conformance relation
   else return P and Q don't have conformance relation
2. for  $\forall a \in C$ 
   if  $Q \xrightarrow{a} Q'$  and  $P \xrightarrow{\tau^* a} P'$ , add  $\langle P, Q \rangle$  to U
   else return P and Q don't have conformance relation
if P has external choice
  for  $\forall b \in C$ , if  $P \xrightarrow{b} P_i$ ,  $Q \xrightarrow{\tau^* b} Q_i$ ,
    and if it doesn't exist that  $P_i \equiv P_j$ ,  $Q_i \equiv Q_j$ 
    and  $\langle P_i, Q_i \rangle \in U$  add  $\langle P_i, Q_i \rangle$  to T;
if T is not NULL
  for each item  $\langle P_i, Q_i \rangle \in T$ 
    do Conformance  $(P_i, Q_i)$ ;
if P has internal choice
  if  $\exists b_1, b_2, \dots, b_k \in C$ ,  $P \xrightarrow{b_j} P_j$ ,  $Q \xrightarrow{\tau^* b_j} Q_j$ ,
    and if it doesn't exist that  $P_j \equiv P_m$ , and  $Q_j \equiv Q_m$ 
    add  $\langle P_j, Q_j \rangle$  to J;
if J is not NULL
  for each item  $\langle P_j, Q_j \rangle \in J$ 
    do Conformance  $(P_j, Q_j)$ ;

```

图 4 一致性关系判断算法

4 结束语

Web 服务合成与替换问题密不可分。我们在文中使用 CCS 的方法描述并推导了 Web 服务合成，分析了异步交互对合成的影响，并给出了一致性关系来限制将要替换已参与合成服务的新服务。有了这样的限制，替换可以上下文无关而又保证替换后合成的正确性。未来还有一系列的工作需要进行研究。当 Web 服务之间不相容时，如果不挑选新的服务，如何调节现有的服务使它们可以正确的交互。文献[1]中对此问题进行了一些研究，并提出了自动产生调节器的方法恢复 Web 服务之间的相容性。此问题还可以进一步探讨。CCS 描述 Web 服务一直未设计合成之上协同工作的层面，也就未包括事务特性、补偿特性及其它 Web 服务协同工作中非常重要的性质。这方面的问题值得深入研究。

参 考 文 献

- [1] Brogi A et al. Formalizing Web services choreographies. *Electronic Notes in Computer Science*, 2004, 105: 73-94
- [2] IBM. Business Process Execution Language for Web Services (BPEL4WS). <http://www.ibm.com/developworkers/library/ws-bpel>, 2002
- [3] W3C. Web Service Choreography Interface (WSCl), World W3C, 2002. <http://www.w3.org/TR/wsc1>
- [4] Bultan T, Fu X, Hull R, Su J. Conversation specification: A new approach to design and analysis of E-service composition//*Proceedings of the WWW2003*. Budapest, Hungary, 2003: 403-410
- [5] Meredith G, Bjorg S. Contracts and types. *Communications of the ACM*, 2003, 46(10): 41-47
- [6] Hamadi R, Benatallah B. A Petri Net-based model for Web service composition//*Proceedings of the 14th Australasian Database Conference on Database Technologies 2003*. Australasian, 2003: 191-200
- [7] Berardi D, Calvanese D, Giacomo G D, Hull R, Mecella M. Automatic composition of transition-based semantic Web services with messageing//*Proceedings of the 31st VLDB Conference*. Trondheim, Norway, 2005: 613-624
- [8] Bordeaux L, Slaun G. Using process algebra for Web services: Early results and perspectives//*Proceedings of the TES 2004*. Toronto, Canada, 2005: 54-68
- [9] Milanovic N, Malek M. Current solutions for Web service composition. *IEEE Internet Computing*, 2004: 51-59
- [10] Bordeaux L, Salun G, Berardi D, Mecella M. When are two Web services compatible//*Proceedings of the TES 2004*. Toronto, Canada, 2005: 15-28

[11] Liu F F, Shi Y S, Zhang L, Shi B S. Analysis of Web services composition and substitution via CCS//Proceedings of the DEECS'06. San Francisco, CA, USA, 2006: 236-245

[12] Mecella M, Pernici B, Craca P. Compatibility of E-services in a cooperative multi-platform environment//Proceedings of the TES 2001. Rome Italy, 2001: 44-57

[13] Fournet C, Hoare T, Rajamani S K, Rehof J. Stuck-free conformance theory for CCS. Microsoft Technical Report: MSR-TR-2004-69, 2004

[14] Rajamani S K, Rehof J. Conformance checking for models of asynchronous message passing software//Proceedings of the CAV2002. Copenhagen, Denmark, 2002: 166-179

[15] Milner R. Communication and Concurrency. Englewood Cliffs: Prentice Hall, 1989



LIU Fang-Fang, born in 1979, Ph.D.. Her research interests include Web service composition and active database.

SHI Yu-Liang, born in 1978, Ph.D.. His research interests include Web service and Web service composition.

ZHANG Liang, born in 1963, professor, Ph.D. supervisor. His research interests include service computing, information integrating and biology information.

SHI Bo-Le, born in 1935, professor, Ph.D. supervisor. His research interests include database and knowledge base.

Background

This paper is supported by the National Basic Research Program(973 Program) of China under grant No. 2005CB321905. This project aims at the designing a model to improve robustness of the system and developing some algorithms that are based on the model. And it is mainly focuses on this field: Survivability of service composition and recovery of business

process which is composed by different component services. This paper focuses on the substitution of service in the service composition. It gives a relation to guarantee the correctness of the composition after the substitution. And the algorithm is also provided.