ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.2980−2990
DOI: 10.1360/jos182980

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

一种基于扩展有限自动机验证组合 Web 服务的方法*

雷丽晖+, 段振华

(西安电子科技大学 计算理论与技术研究所,陕西 西安 710071)

An Extended Deterministic Finite Automata Based Method for the Verification of Composite Web Services

LEI Li-Hui+, DUAN Zhen-Hua

(Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

+ Corresponding author: Phn: +86-29-88223960, E-mail: leilihui@gmail.com

Lei LH, Duan ZH. An extended deterministic finite automata based method for the verification of composite Web services. *Journal of Software*, 2007,18(12):2980−2990. http://www.jos.org.cn/1000-9825/18/2980.htm

Abstract: To simplify and automate the verification of composite Web services, a method based on extended deterministic finite automata (EDFA) is presented. EDFA can describe Web services in an accurate way: the nodes represent states maintained by a service during the interactions between the service and its clients; the state transitions represent message exchanges between the service and its clients. Therefore, the automaton depicts the temporal sequences of messages, i.e. the behavior of the service. With the EDFA-based method for the verification of composite Web services, whether the capabilities of a service meet system requirements and whether there exist logic errors in the interactions between a service and its clients can be verified. Compared with other methods, this method is more suitable for the verification of composite Web services in an open environment.

Key words: composite Web services; deterministic finite automata; formal verification

摘 要: 为简化并自动化组合 Web 服务验证,提出一种基于扩展有限自动机(extended deterministic finite automata,简称 EDFA)验证组合 Web 服务的方法.使用 EDFA 可以准确地描述 Web 服务:EDFA 的状态表达 Web 服务在与用户交互的过程中维护的状态;EDFA 的状态转移及其标注描述 Web 服务与用户间的消息交换.EDFA 给出 Web 服务交互过程的所有消息交换序列,刻画出 Web 服务的动态行为.使用基于 EDFA 的组合 Web 服务验证方法不但可以验证组合 Web 服务是否满足系统需求,还可以验证组合 Web 服务运行过程是否有逻辑错误.与其他方法相比,该方法更适于验证开放式环境下的组合 Web 服务.

关键词: 组合 Web 服务;确定有限自动机;形式化验证

中图法分类号: TP393 **文献标识码**: A

Web 服务适于构建跨平台的松耦合分布式应用,是动态电子商务实现方案的关键技术.Web 服务组合是整合 Internet 上现有多种异构 Web 服务(称为基本 Web 服务)的过程,产生具有新功能的 Web 服务(称为组合 Web

* Supported by the National Natural Science Foundation of China under Grant Nos.60373103, 60433010 (国家自然科学基金); the Defence Pre-Research Project under Grant No.51315050105 (装备预先研究项目)

Received 2007-06-10; Accepted 2007-10-26

服务).目前,越来越多的企业正在尝试用 Web 服务组合实现企业应用集成,即用单个 Web 服务封装简单、独立的业务流程,通过 Web 服务组合实现复杂的新业务流程.使用 Web 服务组合实现企业应用集成必须考虑两个重要问题:第一,如何判定组合 Web 服务可实现企业要求的业务功能;第二,如何判定组合 Web 服务运行过程不会出现企业不期望的情况.解决上述问题的较好方法是对组合 Web 服务进行形式化验证,可分为两类:一是验证组合 Web 服务的功能;二是验证组合 Web 服务的运行过程.现有方法[1,2]将这两类验证分开处理,使得组合 Web 服务的验证更加复杂.本文提出一种基于扩展确定有限自动机(extended deterministic finite automata,简称 EDFA)的方法来处理组合 Web 服务的两类验证,有利于组合 Web 服务修正之后验证过程的重用及验证过程的自动化.

1 相关工作

现有 Web 服务组合方式可以分为两种,即编制(orchestration)和编排(choreography)[3,4].两者的目标都是以一种面向流程的方式把多个 Web 服务组织起来,完成一个复杂的新业务流程,实现单一基本 Web 服务无法实现的功能.编制需要重用多个 Web 服务的内部流程,以形成一个新业务流程,并由一个工作流引擎完成该业务流程的执行.编排是指不同 Web 服务协作完成一个新业务流程,即该业务流程的执行依赖于多个 Web 服务协作完成,而不是由单一工作流引擎来完成.目前,许多组合 Web 服务生成工具,如 JBoss jBPM,Oracle BPEL Process Manager 以及 WebLogic Integration BPM 等等,都采用了编制方式生成组合 Web 服务,所以本文研究如何描述和验证使用编制方式生成的组合 Web 服务.

使用编制方式生成的组合 Web 服务由一个工作流引擎来执行.当用户与组合 Web 服务交互时,工作流引擎根据组合 Web 服务的定义调度各个基本 Web 服务按照指定顺序和条件执行,并在它们之间路由数据.为了验证组合 Web 服务的运行过程不会出现用户不期望的情况(如死锁),需要对用户、工作流引擎及基本 Web 服务的动态行为进行形式化描述.为了验证组合 Web 服务可以实现用户要求的业务功能,还需要对用户提出的系统需求(如商品交易过程中,用户付款后一定有订单产生,且订单一定不会在用户付款前产生)进行形式化描述.

组合 Web 服务可以使用不同的形式化方法描述和验证.目前常见的组合 Web 服务建模及其验证技术可以分为 3 类:状态转移模型(state transition model)及其验证技术,使用 Petri 网或者有限自动机;进程代数模型(process algebra model)及其验证技术,使用π演算或者 BPE 演算[5];时序逻辑模型(temporal logic model)及其验证技术,使用线性时序逻辑或者分支时序逻辑.这 3 类验证方法的特点各不相同.Petri 网能够自然地描述并发、同步、资源竞争等系统特性,易于描述 Web 服务的动态行为,故基于 Petri 网的组合 Web 服务模型适于验证组合 Web 服务的运行过程.文献[6]根据 Web 服务中消息和行为的关系,提出面向消息的基于行为的 Petri 网模型 Moap.该模型包括消息域和服务过程,前者是服务协同以及服务和用户通信的通道,后者是基于 Petri 网的 Web 服务行为过程描述.Moap 能够很好地描述 Web 服务的并行调用,可用于验证组合 Web 服务的运行过程.进程代数可用来对并发和动态变化的系统进行建模,是目前描述 Web 服务的数学理论之一.利用进程的迁移可以刻画 Web 服务的动态行为,利用进程的互模拟性可以验证 Web 服务的行为能力是否等价.文献[7]提出用π演算的进程表达式表示 Web 服务的行为语义,并在此基础上给出 Web 服务组合的可行性验证算法以及需求可满足性的验证方法.但是,进程代数比较抽象,难以被广泛推广.与前两者相比,时序逻辑能够自然地描述事件的时序关系,可以方便地验证系统需求和性质.基于时序逻辑的模型检测(model checking)技术已应用于软件系统和硬件系统的验证.经过近年来的不断发展,时序逻辑也可以用来描述具有典型并发特征的工作流系统[2],因此也可以描述和验证组合 Web 服务的运行过程.此外,时序逻辑可利用偏序归约(partial-order reduction)、组合验证(compositional verification)等较为成熟的方法,有效地减少验证组合 Web 服务所需的状态空间,提高验证计算的效率.然而,与进程代数相似,时序逻辑也比较抽象,难以得到广泛推广.

基于有限自动机的 Web 服务模型与验证也有不少研究成果.Fu[8−10]等人给出一套技术和工具(guarded automata、Promela 语言、SPIN 模型检测器)以及一个完整的系统分析和验证组合 Web 服务的方法,可方便地使用工具分析组合 Web 服务的正确性.然而,该方法目前只能处理同步消息交换,因为在处理异步消息交换时,Guarded Automata 的状态集合是无限的.Wombacher[11]等人提出的 aDFA 模型使用 aDFA 的状态转移描述

Web 服务与用户的消息交换(一条消息对应一个状态转移,用消息名称标注状态转移),并且将自动机的状态与逻辑公式相关联,以说明在 Web 服务运行的某个状态下,服务在收到同一个消息后可能返回的不同消息.将有限自动机与 Petri 网进行比较:两者在交运算下都是封闭的;虽然 Petri 网的表达能力比有限自动机的表达能力更强,但 Petri 网的计算要比有限自动机的计算具有更高的空间复杂度.如果限制 Petri 网有界,则可以降低 Petri 网的计算空间复杂度.但对有界 Petri 网而言,其可达图可以用复杂的有限自动机来表示,所以有界 Petri 网并没有超过有限自动机的表达能力[11].

本文提出了扩展确定有限自动机 EDFA,使用 EDFA 描述 Web 服务,并在此基础上对组合 Web 服务进行两类形式化验证,即不但证明组合 Web 服务可实现用户要求的业务功能,而且证明组合 Web 服务的运行过程不会出现用户不期望的情况.与上述两个基于有限自动机的 Web 服务模型相比,EDFA 的优点在于考虑了 Web 服务的语义信息.在开放式网络环境下,依赖 Web 服务的语义信息可以实现 Web 服务的自动组合.EDFA 能够处理 Web 服务中的语义信息,这一特点使其更适合于验证开放式环境下的组合 Web 服务.

2 扩展确定有限自动机 EDFA

以下组合 Web 服务实例是讨论描述和验证组合 Web 服务的基础.设组合 Web 服务 BuyGoods 由 3 个基本 Web 服务 BookSeller,CDSeller 和 Bank 使用编制方式组合而成,其结构如图 1 所示.Agent 是执行组合服务 BuyGoods 的工作流引擎,BuyGoods 提供订购图书和 CD 的服务,业务流程如图 2 所示.查询图书和查询 CD 这两个过程互不相关,并发执行.BuyGoods 根据 Buyer 的输入查询图书和 CD,如果没有查到所需图书和 CD,那么,BuyGoods 的业务流程终止;否则,根据 Buyer 输入的付款信息完成货物付款并将订单返回给 Buyer.BuyGoods 业务流程包含的几个过程说明如下,斜体字是交易过程中的输入和输出参数.

- 查询图书(SearchBook).根据 Buyer 发送的书名 *BookName*,向 BookSeller 查询要订购的书:如果有库存,则向 Buyer 返回书的价格 *BookPrice* 和书的编号 *BookID*;否则,返回书无库存通知 *BOutStock*.
- 查询 CD(SearchCD).根据 Buyer 发送的 CD 名称 *CDTitle*,向 CDSeller 查询要订购的 CD:如果有库存,则向 Buyer 返回 CD 的价格 *CDPrice* 和 CD 编号 *CDID*;否则,返回 CD 无库存通知 *COutStock*.
- 货物付款(Payment).接收 Buyer 发送的付款信息 *PayInfo*,依据订购货物的价格 *Price*(*BookPrice* 与 *CDPrice* 之和)与 Bank 交互完成付款,返回 Buyer 付款确认信息 *PayConfirm*.
- 返回订单(Ordering).接收 Buyer 发送的邮寄信息 *Delivery*,根据订购货物的编号 *ID*(包括 *BookID* 和 *CDID*),返回给 Buyer 所购商品的订单 *Order*.

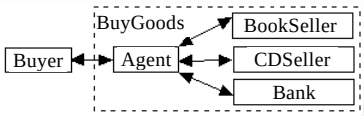

Fig.1 Service BuyGoods composed by orchestration

图 1 采用编制方式生成的服务 BuyGoods

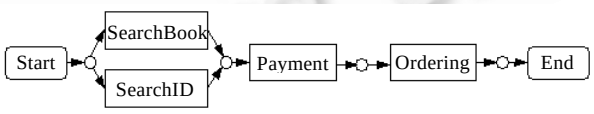

Fig.2 The business process of service BuyGoods

图 2 服务 BuyGoods 的业务流程

EDFA 是确定有限自动机(deterministic finite automata,简称 DFA)的扩展.本文将 EDFA 作为描述 Web 服务的形式化模型.EDFA 的状态表达 Web 服务在与用户交互的过程中维护的状态;EDFA 的一个状态转移及其标注描述 Web 服务与用户的一次交互,即 Web 服务与用户进行一次不可分割的消息交换.状态转移的标注是三元组,第 1 项表示消息的输入,第 2 项表示消息的输出,第 3 项表示消息的交换方式:*p* 表示 Web 服务接收调用请求并且返回调用结果(可以为空),*r* 表示 Web 服务发送调用请求并且接收调用结果(可以为空)*.状态转移表示交互过程完成之后,Web 服务从一个状态转移到另一个状态.显然,EDFA 可以给出 Web 服务交互过程的所有消息交换

* 作为服务请求者,如果消息交换为 one-way 模式,则 Web 服务发送调用请求,接收的调用结果值为空;如果消息交换为 notification 模式,则 Web 服务接收调用请求,返回的调用结果值为空.

序列,刻画了 Web 服务的动态行为.

定义 1(扩展确定有限自动机). 一个扩展确定有限自动机 $M=(D,I,O,A,Q,q_0,F,\delta,\gamma,\lambda)$,这里,

- D 是一个有穷字母表,
- I 和 O 是 D 的幂集的子集,即 $I\subseteq 2^D$ 且 $O\subseteq 2^D$,
- 标识符号集合 $A=\{p,r\}$,
- Q 是一个有限状态集合,
- q_0 是初始状态,F 是终止状态集合,且 $q_0\in Q,F\subseteq Q$,
- 状态转移函数 $\delta:Q\times(I\times O\times A)\to Q$,
- 输入函数 $\gamma:Q\to 2^I$,
- 输出函数 $\lambda:Q\times I\to 2^O$.

注意,对于一个交互过程,在接受同一输入后可能产生多个不同的输出(如 BookSeller 在接收 *BookName* 后可能产生 *BookPrice* 和 *BookID*,或产生 *BOutStock*),使组合 Web 服务进入不同的状态;然而,交互过程每执行一次只能产生其中一个输出(不能既产生 *BookPrice* 和 *BookID*,又产生 *BOutStock*).前一种情况使用输出函数 λ 描述,后一种情况使用状态转移函数 δ 描述,而输入函数 γ 描述在 Web 服务交互过程的某个状态所有可能接收的输入(如 BuyGoods 在初始状态可接收 *BookName* 或 *CDTitle*).服务 BuyGoods 的 EDFA 表示如图 3 所示.

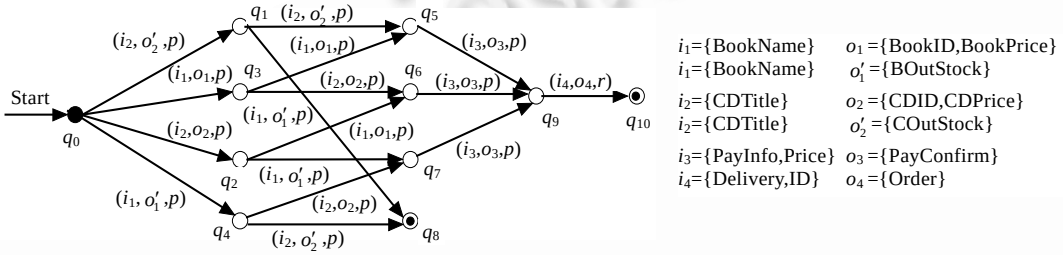

Fig.3 An EDFA representing service BuyGoods

图 3 表示服务 BuyGoods 的 EDFA

DFA 是一个行为模型,其刻画行为的 3 类基本元素是:有限数量的状态、状态之间的转移及每个状态转移的条件[12].DFA 的每个状态转移标注的是一个字母,而 EDFA 的每个状态转移标注的是一个三元组,但两者都是用来描述状态转移的条件.EDFA 的输入函数和输出函数不会改变模型中状态的数量,也不会改变状态转移及状态转移的条件,两者只是描述 Web 服务在交互过程的某个状态上所有可能接收的输入以及对于每个输入可能的输出.所以,EDFA 仍是一类确定有限自动机,所有确定有限自动机的相关理论同样适用于 EDFA.

3 基于 Web 服务行为的语义匹配

Web 服务行为通常使用消息序列来表示,所以现有的比较两个 Web 服务行为的方法[11,13]都是基于消息名匹配(即字符串匹配)来实现的.但是,这些方法都有使用前提,即两个 Web 服务必须有相同的调用接口描述(如 WSDL 文档),以保证名称相同的消息具有相同的语义.然而事实上,不同的 Web 服务可以使用不同的词汇表来描述自己的调用接口,所以,使用消息名匹配方法比较两个 Web 服务行为实现起来比较困难.

值得注意的是,一次不可分割的消息交换完成 Web 服务与用户的一次不可分割的信息交互,消息的输入参数和输出参数的语义可用来准确地表达该交互过程的语义.通过比较消息输入参数和输出参数的语义,可以确定两个交互过程的语义是否一致,这样的一个过程被称为消息语义匹配.消息语义匹配是确定两个 Web 服务共有行为的基础.虽然语义匹配比字符串匹配复杂,但在开放式环境中,语义匹配比字符串匹配更合理、可行,有利于开放式环境下的 Web 服务验证.下面说明基于 Web 服务行为的语义匹配,为组合 Web 服务形式化验证提供实现基础.

基于 Web 服务行为的语义匹配的基本思想是:首先用 EDFA 描述两个 Web 服务,然后,基于消息语义匹配求

两个 EDFA 的交,所得结果描述了两个 Web 服务的共有行为.由上述分析及正则语言的封闭性可知,两个 EDFA 的交(intersection)仍是一个 EDFA.

定义 2(两个 Web 服务的共有行为). 两个扩展确定有限自动机 $M_1=(D_1,I_1,O_1,A_1,Q_1,q_{10},F_1,\delta_1,\gamma_1,\lambda_1)$和 $M_2=(D_2,I_2,O_2,A_2,Q_2,q_{20},F_2,\delta_2,\gamma_2,\lambda_2)$分别表示 Web 服务 W_1 和 W_2,那么,W_1 和 W_2 的共有行为可用一个扩展确定有限自动机 M 来表示且 $M=M_1\cap_{bsm} M_2=(D,I,O,A,Q,q_0,F,\delta,\gamma,\lambda)$,这里,

- 有穷字母表 $D=D_1\cup D_2$,
- I 和 O 是 D 的幂集的子集,即 $I\subseteq 2^D$ 且 $O\subseteq 2^D$,
- 有限状态集合 $Q=Q_1\times Q_2$,
- 初始状态 $q_0=(q_{10},q_{20})$,终止状态集合 $F=F_1\times F_2$,且$(q_{10},q_{20})\in Q,F\subseteq Q$,
- 标识符号集合 $A=A_1=A_2=\{p,r\}$,
- 如果$\delta_1(q_{1k},(i_{1k},o_{1k},t_{1k}))=q_{1(k+1)},\delta_2(q_{2h},(i_{2h},o_{2h},t_{2h}))=q_{2(h+1)}$,并且有 $Match((i_{1k},o_{1k}),(i_{2h},o_{2h}))\geq threshhold,t_{1k}=t_{2h}$,那么,$\delta((q_{1k},q_{2h}),(i_{1k},o_{1k},t_{1k}))=(\delta_1(q_{1k},(i_{1k},o_{1k},t_{1k})),\delta_2(q_{2h},(i_{2h},o_{2h},t_{2h})))$,
- 输入函数γ:$Q\to 2^I$,
- 输出函数λ:$Q\times I\to 2^O$.

消息语义匹配与文献[14]提出的语义匹配方法相似,但是并不相同.消息语义匹配使用 4 个值 *exact*,*plugin*,*subsume* 和 *fail*(*exact*>*plugin*>*subsume*>*fail*)描述两个交互过程的语义匹配程度.令 $threshold\in\{exact,plugin,subsume,fail\}$,则 $Match((i_k,o_k),(i_h,o_h))\geq threshold$ 当且仅当 $match(i_k,i_h)\geq threshold$ 且 $match(o_k,o_h)\geq threshold,i_k,i_h\in I$ 且 $o_k,o_h\in O$.本文中约定 $match(\varnothing,\varnothing)=exact$.

设 a 和 b 是两个集合,a 的元素个数记为$|a|$,b 的元素个数记为$|b|$,$match(a,b)\geq threshold$ 当且仅当$|a|=|b|$,且对于 a 的任意一个元素 a_k,在 b 中总能找到一个元素 b_h,两者的语义匹配程度大于或等于系统允许使用的语义匹配程度值 *threshold*.虽然语义匹配只能给出两个概念的相似程度,但是只要给定了系统允许使用的语义匹配程度值,Web 服务就可据此判定下一步该怎么做.

4 组合 Web 服务验证

组合 Web 服务验证的基本思想是:首先,使用 EDFA 表示用户(可抽象为一个 Web 服务)、工作流引擎(也可抽象为一个 Web 服务)以及基本 Web 服务;然后,使用基于 Web 服务行为的语义匹配,求两个交互者的共有消息交换序列;最后,根据匹配结果判定组合 Web 服务运行过程是否出现用户不期望的情况以及组合 Web 服务是否满足用户提出的系统需求.

4.1 验证组合Web服务的运行过程

使用编制方式生成的组合 Web 服务由一个工作流引擎执行.该工作流引擎不但要与用户交互,而且还要以不同的用户角色与不同的基本 Web 服务交互.验证组合 Web 服务的运行过程就是判定上述这些交互过程是否有逻辑错误.以下用简单实例说明交互过程中可能出现的逻辑错误.如图 4 所示,服务 *A*,*B*,*C* 和 *D* 用 EDFA 表示,有 $o_{a1}=i_{b1},o_{b1}=i_{a1},o_{a2}=i_{b2},o_{b2}=i_{a2},o_{c1}=i_{d1},o_{d1}=i_{c1},o_{d2}=i_{c2},o_{c2}=i_{d2},o_{b2}\neq o'_{b2},i_{d2}\neq i'_{d2},o_{d2}\neq i_{c3}$.

- 未定义的接收:这是指 Web 服务在某个状态下收到一个消息,但却没有定义应该发生的状态转移,因此该 Web 服务的后续交互过程变得不可预知.如图 4 所示,假设 *A* 和 *B* 进行交互且到达各自的 q_1 状态.从图中可以直观地看到,下一步由 *A* 调用 *B*.*B* 在 q_1 状态接收 i_{b2} 后可能返回 o'_{b2},并且从状态 q_1 转移到状态 q_0.但是 *A* 没有定义接收 o'_{b2} 后应该转移到哪个状态,所以 *A* 与 *B* 的后续交互过程变得不可预知.

- 不可执行的交互:这是指 Web 服务定义的某个状态转移在交互过程中不可能被执行.如图 4 所示,设 *C* 和 *D* 进行交互且到达各自的 q_1 状态.从图中可以直观地看到,下一步若由 *D* 调用 *C*,*D* 先发送 o_{d2},*C* 接收 i_{c2} 后返回 o_{c2},并从状态 q_1 转移到状态 q_2,所以 *D* 只能接收 i_{d2},并从状态 q_1 转移到状态 q_2.显然,*D* 定义的从状态 q_1 到状态 q_3 的转移不可能执行,因为 $o_{c2}=i_{d2}\neq i'_{d2}$.此外,*C* 定义的从状态 q_1 到状态 q_0 的转移也不可能执行,因为 *D* 在 q_1 只能发送 o_{d2} 而 $o_{d2}\neq i_{c3}$.

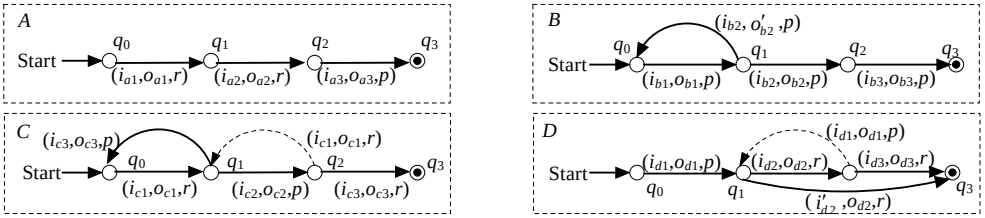

Fig.4 Clients and Web services represented by EDFAs

图 4 EDFA 表示的用户和 Web 服务

• 死锁:这是指 Web 服务与用户的交互过程进入某个状态(由 Web 服务运行状态和用户运行状态组成),既无法满足 Web 服务继续执行的条件,也无法满足用户继续执行的条件,使得 Web 服务与用户的交互过程无法继续进行.如图 4 所示,设 A 和 B 进行交互且到达各自的 q_2 状态.A 和 B 都无法得到想要的消息,停留在各自的 q_2 状态无法转移出去,交互过程无法继续.设 C 和 D 到达各自的 q_2 状态(为说明死锁,这里忽略虚线表示的状态转移).虽然 C 和 D 可各自发出调用请求,但两者都不能收到调用返回,所以停留在各自的 q_2 状态无法转移出去.

• 活锁:这是特殊的死锁.Web 服务与用户在各自的某些状态之间来回转移,不做有效的工作,这些状态不是 Web 服务与用户交互过程定义的终止状态,所以,Web 服务与用户的交互过程无法正常结束.如图 4 所示,设 C 和 D 到达各自的 q_2 状态,虚线表示的状态转移说明 C 可以在其状态 q_2 和 q_1 之间来回转移,D 可以在其状态 q_2 和 q_1 之间来回转移,但两者交互过程无法正常结束.

以下使用形式化方法判定上述逻辑错误.首先构建扩展确定有限自动机 M_1 和 M_2,分别描述 Web 服务及其用户;然后对 M_1 和 M_2 中所有状态转移标注(i,o,r)进行置换,即将 i 和 o 互换,并将消息交换方式标识 r 改为 p,得到 M'_1 和 M'_2;最后求出 $M=M'_1\cap_{bsm}M'_2$.显然,M 的一个状态是由 Web 服务的一个状态和用户的一个状态共同组成.一次不可分割的交互过程使 Web 服务和用户各自从当前状态转移到下一个状态.如果用户和 Web 服务都进入终止状态,两者的交互过程正常结束.

状态转移标注置换之后的 4 种可能匹配情况用图 5 来说明.为了使说明更加清晰,图中省略了状态标识,并用箭头表示置换过程.从图 5 可以直观地看出:A 出现不可执行的交互(b',a,r),B 出现未定义的接收(a,b',p),C 出现不可执行的交互(c,d,p)和(f,e,r),而 D 出现了死锁.

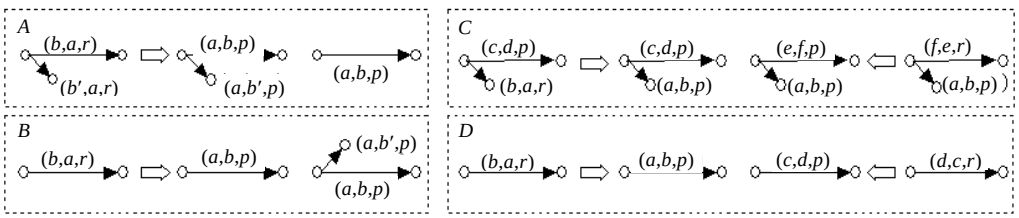

Fig.5 Four cases after state labels being changed

图 5 状态转移标注置换后的 4 种可能匹配情况

这里需要对状态转移标注的置换进行说明.对于服务请求者与服务提供者的每个交互过程,从服务提供者输出的数据就是服务请求者接收的数据,反之一样.为了简化消息语义匹配算法,我们将两者的状态转移标注进行置换.值得注意的是,我们只对状态转移标注(i,o,r)进行置换,即将 i 和 o 互换,并且将消息的交换方式标识 r 改为 p.这样做有一个重要的原因,从图 5 直接可以看出:如果我们不统一状态转移标注的第 3 项(即 p 或 r 都可以,所以可被忽略不计),那么,置换后 A 和 B 中待比较的交互过程是一样的;而事实上,A 和 B 中出现了两种不同的逻辑错误.

定义 3(未定义的接收和不可执行的交互). 两个扩展确定有限自动机 $M_1=(D_1,I_1,O_1,A_1,Q_1,q_{10},F_1,\delta_1,\gamma_1,\lambda_1)$和 $M_2=(D_2,I_2,O_2,A_2,Q_2,q_{20},F_2,\delta_2,\gamma_2,\lambda_2)$分别描述用户和 Web 服务.置换 M_1 和 M_2 中所有状态转移标注(i,o,r),即将 i

和 o 互换,并将消息交换方式标识 r 改为 p,得到 M_1' 和 M_2',求出 $M=M_1'\cap_{bsm}M_2'$.

1) M 的任意一条从初始状态到终止状态的路径,描述了 Web 服务与用户的一个交互过程;
2) 对 M 的一个从初始状态可达的状态(q_{1i},q_{2j}),如果$|\gamma((q_{1i},q_{2j}))|\neq 0$ 且 $\gamma_1'(q_{1i})\neq\gamma_2'(q_{2j})$,那么,包含该状态的交互过程存在不可执行的交互;
3) 对 M 的一个从初始状态可达的状态(q_{1i},q_{2j}),如果$|\gamma((q_{1i},q_{2j}))|\neq 0$ 且 $\gamma_1'(q_{1i})=\gamma_2'(q_{2j})$,
   - 当$\gamma_1(q_{1i})=\gamma_1'(q_{1i})$且$\gamma_2(q_{2j})\neq\gamma_2'(q_{2j})$时,若存在 $i_{1i}\in\gamma((q_{1i},q_{2j})),i_{2j}\in\gamma_2'(q_{2j})$且 $match(i_{1i},i_{2j})\geq threshold$
     当$|\lambda_2'(q_{2j},i_{2j})|<|\lambda_1(q_{1i},i_{1i})|$时,包含状态$(q_{1i},q_{2j})$的交互过程存在未定义的接收;
     当$|\lambda_2'(q_{2j},i_{2j})|>|\lambda_1(q_{1i},i_{1i})|$时,包含状态$(q_{1i},q_{2j})$的交互过程存在不可执行的交互;
   - 当$\gamma_1(q_{1i})\neq\gamma_1'(q_{1i})$且$\gamma_2(q_{2j})=\gamma_2'(q_{2j})$时,若存在 $i_{1i}\in\gamma((q_{1i},q_{2j})),i_{2j}\in\gamma_2'(q_{2j})$且 $match(i_{1i},i_{2j})\geq threshold$
     当$|\lambda_1'(q_{1i},i_{1i})|<|\lambda_2(q_{2j},i_{2j})|$时,包含状态$(q_{1i},q_{2j})$的交互过程存在未定义的接收;
     当$|\lambda_1'(q_{1i},i_{1i})|>|\lambda_2(q_{2j},i_{2j})|$时,包含状态$(q_{1i},q_{2j})$的交互过程存在不可执行的交互.

定义 4(死锁和活锁). 将定义 3 中的 M 视为一个有向图$(V,\{E\})$,图的顶点表示 M 的状态,图的有向弧表示 M 的状态转移,v_0 表示 M 的初始状态,$F\subseteq V$ 表示 M 的终止状态集合.

1) 如果 M 中存在一条路径$\langle v_0,v_1,\ldots,v_k\rangle$,但不存在 $v_j\in V(j\neq k)$使得$\langle v_k,v_j\rangle\in\{E\}$,且 v_k 表示的状态 $q_k\notin F$,那么,包含状态 q_k 的路径所描述的 Web 服务与用户的交互过程出现死锁.
2) 如果 M 中存在环$\langle v_i,v_j,\ldots,v_i\rangle$,而且以下两个条件都可以满足,那么包含$\langle v_i,v_j,\ldots,v_i\rangle$的路径所描述的 Web 服务与用户的交互过程出现活锁.
   - 由环$\langle v_i,v_j,\ldots,v_i\rangle$得到的状态集合$\{q_i,q_j,\ldots,q_i\}\cap F=\varnothing$;
   - 不存在这样一条边$\langle v_k,v_h\rangle\in\{E\},v_k\in\{v_i,v_j,\ldots,v_i\}$而 $v_h\notin\{v_i,v_j,\ldots,v_i\}$.

定义 5. Web 服务与用户的交互过程是定义良好的(well-formed),当且仅当两者的交互过程不存在未定义的接收和不可执行的交互.Web 服务与用户的交互过程是活的(live),当且仅当两者的交互过程没有死锁和活锁,并且不包含未定义的接收.交互过程是活的只能保证交互过程能够正常执行.Web 服务与用户的交互过程是安全的(safe)当且仅当两者的交互过程是活的且总能正常结束.

定义 6. 组合 Web 服务是定义良好的,当且仅当其包含的所有交互过程(包括与用户的交互过程和与基本 Web 服务的交互过程)都是定义良好的.组合 Web 服务是活的,当且仅当其包含的所有交互过程都是活的.组合 Web 服务是安全的,当且仅当其包含的所有交互过程都是安全的.

以下用伪代码描述的算法,可用来判定用户与 Web 服务的交互过程是否为定义良好的、活的和安全的.令 M_1 和 M_2 分别表示用户和 Web 服务,$M=M_1'\cap_{bsm}M_2'$,将 M 当作有向图处理,使用拓扑排序判定 M 中是否存在环.求出可达状态集合(*reachable_states*)与已访问状态集合(*visted_states*)的差集,该集合中的状态如果都在环上且都不是终止状态,交互过程会出现活锁.

```
boolean isLivelock(M,reachable_states,F) {
    InitStack(S); Livelock=true; visted_states.inital();
    for each state s_i∈reachable_states {
    if (|inDegree[s_i]|=0) then Push(S,s_i)};
    while (not isStackEmpty(S)) {p=Pop(S); visted_states.add(p);
        for each s_i∈p.getAdjcentStates {
            inDegree[s_i]=inDegree[s_i]−1;
            if (inDegree[s_i]=0) then Push(S,s_i);}}
    if visted_states.equals(reachable_states) then Livelock=false
    else for each state s_i∈reachable_states.subtract(visted_states) {\\ obtain the states that are not visited
                if (not existPath(s_i,s_i)) or (s_i∈F) then Livelock=false;}
                \\ existPath is used to determine whether there exists a path from s_i to s_i
```

retrun *Livelock*}

CheckEDFA(M,(q_{10},q_{20}),F,M_1, M_1' ,M_2, M_2') {

reachable_states=*FindReachableStates*(M,(q_{10},q_{20}));

paths=*FindAllPaths*(M,(q_{10},q_{20}),F); \\ a set of paths from the initial state to one of the final states

for each *state*(q_{1i},q_{2j})∈*reachable_states* {

if $|\gamma(q_{1i},q_{2j})|\neq 0$ and $\gamma_1'(q_{1i})\neq\gamma_2'(q_{2j})$ then *unexecutable.add* ((q_{1i},q_{2i}));

if $\gamma_1(q_{1i})=\gamma_1'(q_{1i})$ and $\gamma_2(q_{2j})\neq\gamma_2'(q_{2j})$ and $|\gamma(q_{1i},q_{2j})|\neq 0$ and $\gamma_1'(q_{1i})=\gamma_2'(q_{2j})$ then {

for each $i_{1k}\in\gamma(q_{1i},q_{2j})$ {i_{2h}=*Matched*(M_2' ,q_{2j},i_{1k});

if $|\lambda_2'(q_{2j},i_{2h})|<|\lambda_1(q_{1i},i_{1k})|$ then *unspecified.add*((q_{1i},q_{2j}))

else if $|\lambda_2'(q_{2j},i_{2h})|>|\lambda_1(q_{1i},i_{1k})|$ then *unexecutable.add*((q_{1i},q_{2j}));}

if $\gamma_1(q_{1i})\neq\gamma_1'(q_{1i})$ and $\gamma_2(q_{2j})=\gamma_2'(q_{2j})$ and $|\gamma(q_{1i},q_{2j})|\neq 0$ and $\gamma_1'(q_{1i})=\gamma_2'(q_{2j})$ then {

for each $i_{1k}\in\gamma(q_{1i},q_{2j})$ {i_{2h}=*Matched Matched*(M_2' ,q_{2j},i_{1k});

if $|\lambda_1'(q_{1i},i_{1k})|<|\lambda_2(q_{2j},i_{2h})|$ then *unspecified.add*((q_{1i},q_{2j}))

else if $|\lambda_1'(q_{1i},i_{1k})|>|\lambda_2(q_{2j},i_{2h})|$ then *unexecutable.add*((q_{1i},q_{2j}));}}

if *isEmpty*(*unspecified*) and *isEmpty*(*unexecutable*) then *Well-formed*=*true* else *Well-formed*=false;

if *isEmpty*(*unspecified*) and *isEmpty*(*deadlock*) and not *isLivelock*(*M*,*reachable_states*,*F*)

then *Live*=true else *Live*=false;

if *Live* and not *isEmpty*(*paths*) then *Safe*=true else *Safe*=false;}

图 6 和图 7 给出用户 Buyer、工作流引擎 Agent 和基本 Web 服务 BookSeller,CDSeller,Bank 的 EDFA 表示(图 6 的 Agent 与图 3 一致,故省略状态转移标注说明).工作流引擎与 3 个基本 Web 服务交互时有 3 个不同的用户角色:SearchBook,SearchCD 和 Payment,用图 7 中的 EDFA 表示.组合 Web 服务 BuyGoods 的运行过程包含 4 个用户与 Web 服务的交互过程:Buyer 与 Agent,SearchBook 与 BookSeller,SearchCD 与 CDSeller,Payment 与 Bank.验证 BuyGoods 运行过程需要判定上述 4 个交互过程是否有逻辑错误.假设所有消息语义匹配的结果都能满足系统要求,即匹配结果大于或等于系统允许使用的语义匹配程度值.使用上述方法可以判定 Buyer 和 Agent 的交互过程出现了未定义的接收,其他交互过程都是安全的,所以组合 Web 服务 BuyGoods 的运行过程不是安全的.

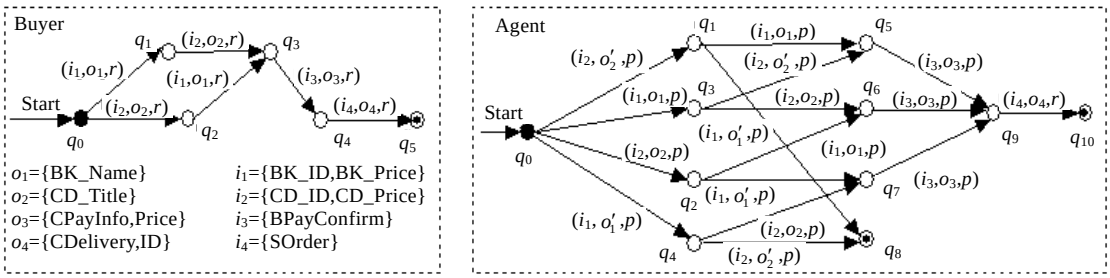

Fig.6 Two EDFAs representing client Buyer and service Agent

图 6 表示用户 Buyer 和服务 Agent 的两个 EDFA

需要注意的是,虽然每一个基本 Web 服务都与工作流引擎交互,但他们并不是组合 Web 服务的用户(如 BookSeller,CDSeller 和 Bank 都不是 BuyGoods 的用户).相反,工作流引擎是以不同的用户角色与基本 Web 服务进行交互,这是在 Web 服务组合阶段就决定了的.一般情况下,当使用 Web 服务组合实现复杂的业务过程时,需要选择合适的基本 Web 服务,并用各种组合方法确定这些基本 Web 服务的执行顺序和执行条件,以形成一个具有新功能的 Web 服务,即组合 Web 服务.在选择基本 Web 服务时,组合 Web 服务以什么用户角色与基本 Web 服务进行交互就已经确定了.

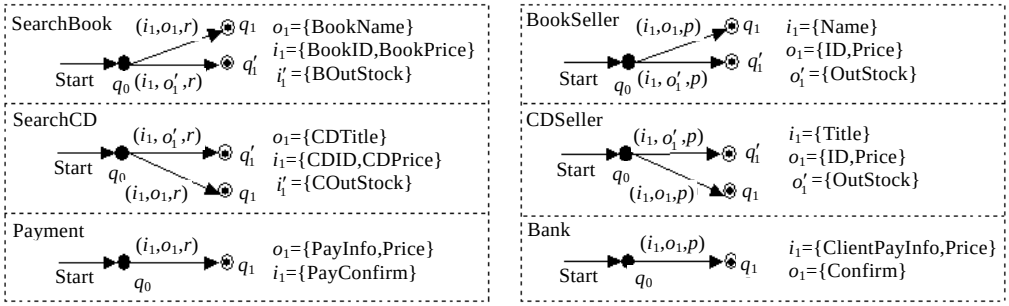

Fig.7 EDFAs represeing clients SearchBook, SearchCD, Payment and services BookSeller, CDSeller and bank

图 7 EDFA 表示用户 SearchBook,SearchCD,Payment 和 Web 服务 BookSeller,CDSeller,Bank

4.2 验证组合Web服务的功能

验证组合 Web 服务实现用户要求的业务功能,首先要确定用户提出的系统需求,然后证明组合 Web 服务可满足这些系统需求.不同的系统有不同的需求,然而,所有的系统需求都可用两种形式来描述:一种说明“坏事情永远都不会发生”(安全性);另一种说明“好事情最终会发生”(活性).验证的基本思想是:首先将用户提出的系统需求用 EDFA 来表示;然后将执行组合 Web 服务的工作流引擎用 EDFA 来表示;最后使用基于 Web 服务行为的语义匹配来判定组合 Web 服务是否满足用户提出的系统需求.

这里的难点是将用户提出的系统需求用 EDFA 来表示,我们可使用以下方法来实现.首先用时序逻辑公式表示系统需求,然后用已有的模型检测相关技术[16],将逻辑公式转换为确定有限自动机(状态迁移描述消息交换事件,自动机描述出所有可能的消息交换事件的时序关系),最后更改状态转移标注,将所得的确定有限自动机转换为 EDFA.

定义 7. 扩展确定有限自动机 $M=(D,I,O,A,Q,q_0,F,\delta,\gamma,\lambda)$和 $M_p=(D_p,I_p,O_p,A_p,Q_p,q_{p0},F_p,\delta_p,\gamma_p,\lambda_p)$分别用来描述执行组合 Web 服务的工作流引擎和用户提出的系统需求,置换 M 所有的状态转移**,即 i 和 o 互换,并将消息交换方式标识为 r 的改为 p, p 的改为 r,得到 M'.那么,组合 Web 服务满足这些系统需求,当且仅当 $M_p=M_p\cap_{bsm}M'$.

以 BuyGoods 为例,对一次有效的订购过程,如果需要付款,那么完成付款之后肯定返回订单.显然,用户期望“完成付款并返回订单”最终会发生(活性),而且“返回订单在完成付款之前”永远不会发生(安全性).如果不需要付款,则完成付款和返回订单都不发生.完成付款必须在所有查询结束后才能执行,否则返回订单可能包含未付款的书或 CD.上述需求用图 8 中 Req 的 EDFA 描述.假设验证过程中所有消息语义匹配的结果都能满足系统要求,使用上述方法可以判定,组合 Web 服务 BuyGoods 可满足图 8 中 Req 的 EDFA 描述的系统需求.判定算法可以用现有的有向图理论来解决,由于篇幅有限,这里不再讨论.

综上所述,验证组合 Web 服务的运行过程,对不同的组合 Web 服务,验证内容都是一样的,都是证明组合 Web 服务包含的交互过程不会出现逻辑错误;验证组合 Web 服务的功能,对不同的组合 Web 服务,验证内容可能不一样,因为对于不同的系统,用户提出的系统需求各不相同,所以,当验证组合 Web 服务的功能时,要首先确定用户提出的系统需求.

我们前期的工作已经实现了 Web 服务描述到 EDFA 的自动转换,以及基于 Web 服务行为的语义匹配[15].使用 EDFA 和基于 Web 服务行为的语义匹配来完成两类组合 Web 服务的验证,具有如下优点:

1) 在同一个形式化模型上验证组合 Web 服务的功能和组合 Web 服务的运行过程;
2) 用基于 Web 服务行为的语义匹配,更适于开放式环境中的组合 Web 服务验证;
3) 具有合理的计算时间和空间,验证计算的时间复杂度和空间复杂度都是多项式的;

** 验证 Web 服务功能时的状态转移标注置换与验证 Web 服务运行过程时的状态转移标注置换有所不同.因为,在验证 Web 服务功能时,只需得到 Web 服务与用户之间所有可能的消息交换事件的时序关系,无须考虑消息交换是由谁发起的.

4) 组合 Web 服务修改后,验证方法可以重用,提高了验证效率.

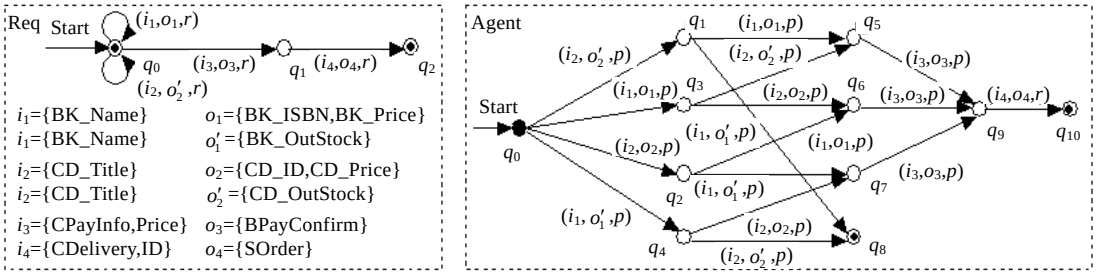

Fig.8 Two EDFAs representing system requirements and service Agent

图 8 表示用户需求和服务 Agent 的两个 EDFA

5 总 结

本文首先比较了两种组合 Web 服务构建方式,然后阐述用 EDFA 为 Web 服务建模的方法,最后基于 Web 服务行为的语义匹配提出判定方法,说明了在同一个形式化模型上,如何验证组合 Web 服务的功能和组合 Web 服务的运行过程.这种验证方法使得组合 Web 服务的验证过程直观、容易自动化且可以重用.由于验证方法使用 EDFA 作为 Web 服务的形式化模型,降低了验证计算的复杂度,提高了验证效率.此外,验证方法采用了基于 Web 服务行为的语义匹配,所以适于验证开放式环境中的组合 Web 服务.未来的工作需要设计和实现一个验证工具以检验这种组合 Web 服务验证方法.

References:

[1] Narayanan S, McIlraith SA. Simulation, verification and automated composition of Web services. In: Proc. of the 11th Int'l Conf. on World Wide Web. New York: ACM Press, 2002. 77−88.

[2] Solanki M, Cau A, Zedan H. Augmenting semantic Web service description with compositional specification. In: Proc. of the 13th Int'l Conf. on World Wide Web. New York: ACM Press, 2004. 544−552.

[3] Peltz C. Web services orchestration and choreography. IEEE Computer, 2003,36(10):46−52.

[4] Lei LH, Duan ZH. Automating Web service composition for collaborative business processes. In: Shen WM, ed. Proc. of the 12th Int'l Conf. on CSCWD. Melbourne: IEEE Press, 2007. 108−116.

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

[6] Qian ZZ, Lu SL, Xie L. Automatic composition of Petri net based Web services. Chinese Journal of Computers, 2006,29(7): 1057−1066 (in Chinese with English abstract).

[7] Hou LS, Jin Z, Wu BD. Modeling and verifying Web services driven by requirements: An ontology based approach. Science in China (Series E), 2006,36(10): 1189−1219 (in Chinese with English abstract).

[8] Fu X, Bultan T, Su JW. Analysis of interacting BPEL Web services. In: Proc. of the13th Int'l Conf. on World Wide Web. New York: ACM Press, 2004. 621−630.

[9] Fu X, Bultan T, Su JW. WSAT: A tool for formal analysis of Web services. In: Alur R, Peled D, eds. Proc. of the 16th Int'l Conf. on Computer Aided Verification. 2004. 510−514.

[10] Fu X, Bultan T, Su JW. Synchronizability of conversations among Web services. IEEE Trans. on Software Engineering, 2005, 31(12):1042−1055.

[11] 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.

[12] Hopcroft JE, Motwani R, Ullman JD. Introduction to Automata Theory, Languages, and Computation. 2nd ed., New York: Addison Wesley, 2001.

[13] Papazoglou M, Georgakopoulos D. Service-Oriented computing: Introduction. Communications of the ACM, 2003:25−28.

[14] Paolucci M, Kawamura T, Payne TR, Sycara K. Semantic matching of Web services capabilities. In: Proc. of the ISWC 2002. LNCS 2342, 2002. 333−347.

[15] Lei LH, Duan ZH. Transforming OWL-S process model into EDFA for service discovery. In: Proc. of the ICWS 2006. Chicago: IEEE Press, 2006. 108−116.

[16] Tian C，Duan ZH. Model checking propositional projection temporal logic based on SPIN. In：Proc. of the 9th international Annual Conf. on Formal Enginnering Method. LNCS 4789, 2007. 246−265.

附中文参考文献:

[6] 钱柱中,陆桑璐,谢立.基于 Petri 网的 Web 服务自动组合研究.计算机学报,2006,29(7):1057−1066.

[7] 侯丽珊,金芝,吴步丹.需求驱动的 Web 服务建模及其验证:一个基于本体的方法.中国科学 E 辑,2006,36(10):1189−1219.

雷丽晖(1976－),女,陕西西安人,博士,主要研究领域为网络计算,语义 Web.

段振华(1948－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,可信软件理论和技术.