Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2011-CJ-003

场景驱动的服务行为调控

柳溪,杨璐,潘敏学,王林章

Postprint Version. Originally Published in: Journal of Software,2011,22(6):1185−1198
[doi: 10.3724/SP.J.1001.2011.04019]

软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(6):1185−1198 [doi: 10.3724/SP.J.1001.2011.04019]

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

场景驱动的服务行为调控*

柳 溪[1,2+], 杨 璐[3,1], 潘敏学[1,2], 王林章[1,2+]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

[3](苏州大学 计算机科学与技术学院,江苏 苏州 215006)

Scenario-Driven Service Behavior Manipulation

LIU Xi[1,2+], YANG Lu[3,1], PAN Min-Xue[1,2], WANG Lin-Zhang[1,2+]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[3](School of Computer Science and Technology, Soochow University, Suzhou 215006, China)

+ Corresponding author: E-mail: liux@seg.nju.edu.cn, lzwang@nju.edu.cn

Liu X, Yang L, Pan MX, Wang LZ. Scenario-Driven service behavior manipulation. *Journal of Software*, 2011,22(6):1185−1198. http://www.jos.org.cn/1000-9825/4019.htm

Abstract: This paper proposes an approach for scenario-driven Web services behavior manipulation. First, the study uses UML sequence diagrams as the scenario-based specification to describe user's requirement on the behavior of the service and construct BPEL-Petri nets model (BPN model for short) to represent the service behavior based on its BPEL specification. Second, the service behavior is analyzed based on paths of the BPN model by utilizing the notion of concurrent transitions. The set of behavior with occurrence of the scenario depicted by the UML Sequence Diagram is obtained by traversing the BPN model. Finally, by using the result of behavior analysis, the study constructs the manipulator services to extract or filter out the behavior at run-time by listening to, checking, and filtering the messages exchanged between the user and the target service. In addition, the study has developed a prototype tool called BASIS to facilitate the behavior manipulation and conduct a case study to illustrate the feasibility of this approach.

Key words: Web service; scenario-based specification; behavior manipulation; BPEL; Petri net; automaton

摘 要: 提出了一个场景驱动的服务行为调控途径.首先,用 UML 顺序图模型作为场景规约以描述用户对服务行为的需求,并且基于目标服务的 BPEL 行为规约,构造表示服务行为的 BPEL-Petri 网模型(简称 BPN 模型).其次,基于并发变迁分析 BPN 模型上表示服务行为的路径,并通过遍历 BPN 模型获取包含 UML 顺序图描绘场景的服务行为集合;最后,根据行为分析的结果构建了调控服务,通过在运行时监听、检查并过滤用户与目标服务的消息交互,从目标服务中抽取或过滤顺序图描绘的场景.在此基础上,开发了原型工具 BASIS,以支撑场景驱动的服务行为调控途径,并通过实例研究展示了该方法的可行性.

* 基金项目: 国家自然科学基金(90818022, 91018006, 61021062); 国家重点基础研究发展计划(973)(2009CB320702); 核高基项目(2009z01036-001-001-3)

收稿时间: 2010-07-10; 定稿时间: 2011-03-29

关键词: Web 服务;场景规约;行为调控;BPEL;Petri 网;自动机
中图法分类号: TP311 文献标识码: A

1 引 文

面向服务的架构(SOA)在现代软件工业中发挥着异构系统之间的桥梁作用.在 SOA 中,业务功能常常需要通过第三方 Web 服务来实现.Web 服务的行为表现为服务与服务或服务与使用者之间的消息交互.对于某个第三方服务,用户:1) 可能需要更多更复杂的行为;2) 也可能仅需要该服务提供的一部分行为.对于第 1 种情况,可以利用服务组合的方式构造满足用户需求的服务,在这个领域已经有了很多的研究[2,6,7];而对于第 2 种情况,我们就需要调控目标服务的行为:保证用户期望发生的场景在目标服务的每次执行中出现——称为从目标服务中抽取该场景;或是使得用户不期望发生的场景在目标服务的执行中不出现——称为从目标服务中过滤该场景.同时,由于服务的使用者通常没有修改第三方服务的代码或部署平台的权限,所以只能利用服务消息驱动的特性来调控第三方服务的行为.

本文提出了场景驱动的 Web 服务行为调控途径,包括为输入的目标服务建立模型、分析目标服务的行为、构造行为约束自动机和调控服务.调控服务通过监听、检查和过滤用户与目标服务之间的消息,保证目标服务的恰当消息交互,实现场景驱动的服务行为抽取或过滤,完成对目标服务的行为调控.

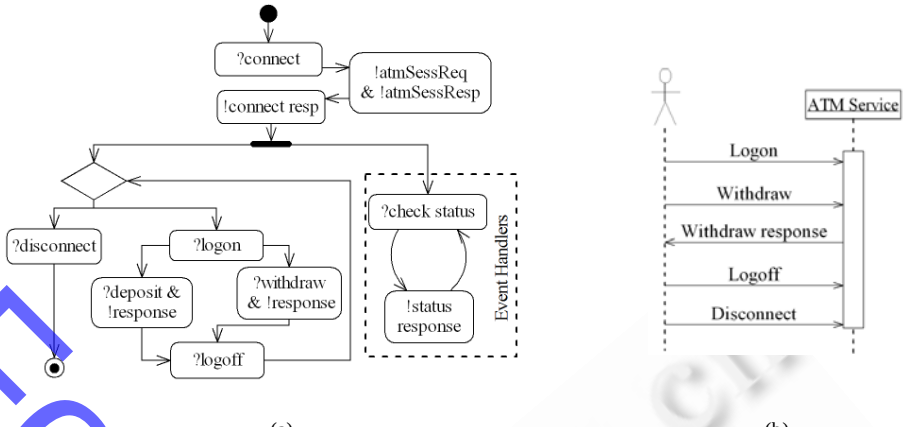
1.1 研究动机示例

我们首先介绍一个示例:ATM 服务(详细信息及完整的实例研究参见 http://seg.nju.edu.cn/BASIS010/atm_full.htm).该服务的 BPEL 代码来自于 IBM BPWS4J 项目(参见 http://www.alphaworks.ibm.com/tech/bpws4j)的示例.其行为如图 1(a)所示,其中,?*m* 表示接收消息 *m*,而!*m* 表示发送消息 *m*.ATM 服务接受 connect(连接)消息后启动,然后该服务发送会话请求,并从回复中获取当前会话信息,该会话信息作为 connect 的确认回复给客户;连接之后,ATM 可循环进行以下操作直到接受到 disconnect 消息后流程结束.ATM 可接受客户的 logon(登录系统)请求,然后接受 withdraw(取款)、deposit(存款)或 logoff(登出系统)请求并作相应处理和响应.在发送 connect 确认之后,ATM 服务还可以并行地在事件处理流程(event handlers,图 1(a)虚线框中活动)中响应并应答客户查询服务状态的请求.

假设对 ATM 服务用户给出的“取款”场景如图 1(b)所示,即 ATM 首先接收用户 logon,紧接着完成一次取款操作(withdraw 请求和响应消息对),然后用户从 ATM 服务中 logoff,最后 disconnect 并结束操作.取款场景在 ATM 服务中可以出现,却不能保证发生.例如,下面的消息交互序列 ms_1 和 ms_2 都可被 ATM 服务接受,而 ms_1 中包含了取款场景,ms_2 则不包含:

- ms_1=?connect→!atmSessRep→?atmSessResp→!connect resp→?logon→?withdraw→
  !withdraw response→?logoff→?disconnect;
- ms_2=?connect→!atmSessRep→?atmSessResp→!connect resp→?logon→?deposit→
  !deposit response→?logoff→?disconnect.

因此,我们需要对用户与目标服务之间的消息交互进行监听、检查和过滤.如果当前消息转发给 ATM 服务后,ATM 的行为能够满足用户对场景行为抽取或过滤的需求,就将该消息转发;否则,就立刻告知用户应输入的消息.例如,如用户需要抽取取款场景,对于 ms_2,由于转发 disconnect 消息后取款场景必不会发生了,所以不转发该消息,而告知用户应发送的消息(如 logon);若用户需要过滤取款场景,对于 ms_1,如果转发 disconnect 消息,取款场景就会完整地发生了,因此就不转发该消息,而告知用户应发送的消息(如 logon).此后,继续监听、检查和过滤用户与 ATM 之间的消息交互,以根据用户需求,实现对取款场景行为抽取或过滤.



(a) (b)

Fig.1 ATM service and the withdrawal scenario

图 1 ATM 服务及取款场景

1.2 研究问题:服务行为调控

为满足用户在使用第三方服务时,从目标服务中抽取或过滤部分行为的需求,我们研究了场景驱动的服务行为调控途径.本文中,第三方目标服务通过 BPEL[15]描述,而用户的行为需求中期望抽取或过滤的场景则用 UML 顺序图[8]模型描绘.服务行为调控的目的是,在不改变目标服务的前提下,利用服务的消息驱动的特性,监听、检查并过滤用户与目标服务之间的消息交互,使得目标服务在运行时能够收到恰当的消息,从而表现出满足用户需求的行为.如果用户期望抽取顺序图描绘的场景,则保证目标服务在执行时该场景的出现;而如果用户期望过滤顺序图描绘的场景,则避免目标服务在该场景执行时的出现.

本文提出的场景驱动服务行为调控途径的工作流程如图 2 所示,整个流程分为 3 个部分:行为建模、行为分析以及调控服务的构造.首先,在行为建模中为输入的目标服务建立 BPN 模型.然后,在行为分析中根据用户需求中期望过滤或抽取的 UML 顺序图模型描绘的场景,遍历转换得到的 BPN 模型,以获取 BPN 模型中所有包含顺序图描绘场景出现的路径.行为分析记录的路径中包含顺序图描绘的场景和必要的循环结构,并避免了与场景无关的并发分支的线性化.在构造调控服务前,判断分析的结果是否满足调控的需求,即行为分析找到了包含顺序图描绘场景出现的行为,并且当用户需求是行为过滤时,BPN 模型中包含顺序图描绘场景不出现的行为.如果分析结果满足调控需求,则构造调控服务.调控服务包括一个消息过滤包装服务、一个消息交互检查服务以及一个行为约束自动机.行为约束自动机是根据行为分析记录的路径构造的;而调控服务则在运行时监听消息交互,并根据行为约束自动机,检查是否应过滤当前消息.

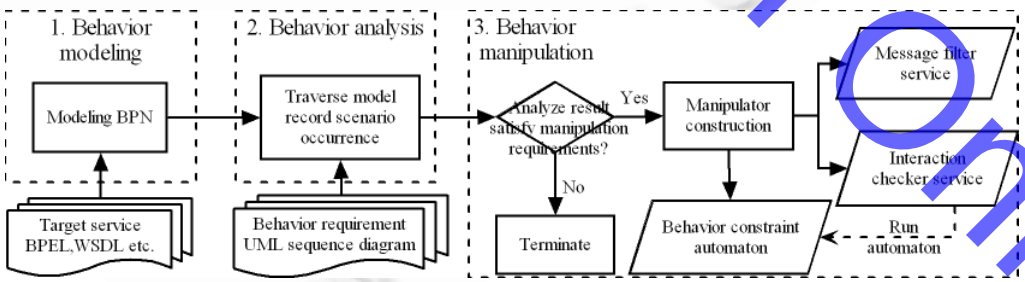


Fig.2 Process of Web service behavior manipulation approach

图 2 Web 服务行为调控途径的流程

1.3 相关工作

为了自动化地分析 BPEL 服务,首先需要为服务的 BPEL 代码构造形式化模型[4].基于 Petri 网的模型是最常用的建模工具之一.相关的工作中,比较有代表性的包括 Lohnmann[10]和 Ouyang 等人[13]使用的基于经典 Petri 网的工作流模型以及 Yi 等人[19]使用的高阶 Petri 网等.我们的前期工作[18]使用了 BPEL-Petri 网模型,并验证了转换得到的模型和 UML 顺序图的存在一致性和强制一致性[9].本文 BPEL 的建模以前期的工作为基础,并进行了扩展.与一致性验证相似,行为分析中我们也需要遍历模型的状态空间.针对行为调控的需求,我们还对前期工作中的遍历方法[9,18]作了优化,降低了遍历的时间和空间开销.

相比起服务行为的建模和一致性验证,Web 服务的行为调控是较新的研究课题,相关的研究还比较少.Bertoli 和 Marconi 等人[3,12]研究了如何利用人工智能规划的方法自动组合多个服务,以满足用户给出的组合服务的行为需求.与此工作相比,我们的工作关注于单个服务的行为抽取和过滤,通过对其消息交互的调控,使得目标服务的行为表现满足用户的需求.Lohnmann 等人[11]通过在服务模型上应用行为约束,产生操作指导(operating guidelines).用户的行为约束或者是定义在服务工作流模型的变迁上,或者是在服务工作流模型转换得到的服务自动机的状态和变迁上.这样,用户就需要了解服务的具体模型和内部状态,而行为约束也需要描述从服务初始状态到终止状态的完整行为.同时,约束模型必须非常谨慎地构造,考虑目标服务模型的状态和变迁.而我们的工作中,用户的需求描述为顺序图场景,其中只包含服务中用户需要抽取或过滤的行为片段;其次,我们构造调控服务,利用服务消息驱动的特性,能够在运行时保证目标服务满足恰当的消息交互.本文的算法直接根据满足行为需求的 BPN 模型的路径构造行为约束自动机和调控服务,而不是如 Lohnmann 等人[11]从所有可能的消息交互中去除不满足条件的分支,因此效率更高.此外,我们还提供了工具自动产生调控服务,以部署在相应的服务器上实现运行时行为调控.

本文的主要贡献在于:

- 定义了 BPEL 服务行为形式模型 BPEL-Petri 网(BPN 模型),以及从 BPEL 到 BPN 模型的转换;
- 分别为针对 UML 顺序图模型描绘的场景遍历 BPN 模型以获取路径和根据记录的路径构造调控服务的问题设计了算法,其中,遍历算法还基于并发变迁的概念,避免将不必要的并发线性化,从而缩减了行为分析的状态空间;
- 提出了完整的场景驱动服务行为调控途径,并以此为基础开发了原型工具 BASIS,以支撑服务行为调控的完整流程.

本文第 2 节介绍服务需求和服务行为建模.在第 3 节介绍按照顺序图场景规约对目标服务进行行为分析的方法.第 4 节介绍调控服务的构造,包括行为约束自动机的构造以及服务的运行时调控,原型工具 BASIS 也在这一部分介绍.第 5 节是本文的总结和进一步的工作.

2 服务建模

为了对目标服务行为进行调控,用户首先需要了解服务提供的行为,并描述用户自己需求的行为.服务行为调控中,由用户给出其行为需求模型,通过 UML 顺序图模型描绘,我们用 BPEL 作为目标服务的描述语言.为了分析目标服务的行为,我们构造了服务的 BPEL 行为规约到对应形式模型——BPN 模型的转换.

2.1 服务需求模型:UML顺序图

场景可通过操作的序列描述系统的行为片段.作为最为常见的场景规约,UML 顺序图模型(以下简称顺序图)[8]提供了一种直观、可视的方式来定义消息交互的时序关系,在工业界和学术界广泛应用.而 Web 服务是消息驱动的,所以其行为正是通过其消息交互表现的.本文中,用户对服务需求行为片段通过顺序图作为场景规约描绘,由用户给出并说明该顺序图描绘的场景是用户期望的还是不期望的行为,即需要从目标服务中抽取还是过滤该场景.图 1(b)中的取款场景就是一个顺序图.这里给出了顺序图的形式定义.

定义 1(顺序图). 一个顺序图是元组 $D=(O,E,M,G,V)$:O 是服务的有穷集;E 是消息收发事件的有穷集;M 是

消息的有穷集,对于每个消息 $m\in M$ 都存在 $e_s,e_r\in E$ 分别是的消息 m 的发送事件和接收事件;$G:E\to O$ 是消息收发事件到服务的映射,表示服务 $G(e)$发送(或接受)事件 e 所对应的消息,且$\forall e\in E,\exists o\in O,G(e)=o$;$V$ 是事件有序对的有穷集,对于任何$(e_1,e_2)\in V$ 均有 $e_1,e_2\in E$,且按照图上的顺序,e_1 应在 e_2 之前.

参照文献[17],我们说顺序图 D 的一个消息踪迹是一个消息收发事件的序列,这些事件的顺序满足 D 中定义的消息收发事件的顺序.设 *trail* 是 D 的一个消息踪迹,对于任何服务 $o\in O$,将 *trail* 中所有不属于服务 o 上的事件移除(即移除满足 $G(e)\neq o$ 的事件 e),我们称得到的 $trail_o$ 是顺序图 D 在服务 o 上的消息踪迹.

2.2 服务行为规约:Web服务业务流程执行语言(WS-BPEL)

Web 服务描述语言(WSDL)[16]描述了 Web 服务的最基本信息——服务的消息接口,但其本身不含服务的行为信息.为使用户能够增强对服务行为的了解,Web 服务还需要发布其行为规约,以作为用户选择服务的重要依据.Web 服务业务流程执行语言(WS-BPEL,简称 BPEL)[15]是 OASIS 标准,在工业界有着广泛的支持.其结构化和工作流特性已使其成为了 Web 服务标准族中最重要的标准之一,也是描述 Web 服务的行为规约的最佳选择.

BPEL 的操作使用基本活动来描述:receive(接收服务外部消息)、reply(发送请求的响应)、invoke(发送消息或调用其他服务)、wait(将当前流程暂时挂起)、assign(赋值)、empty(空活动占位符)、exit(结束流程)、throw(抛出异常)、rethrow(将已经捕获的异常再向上层抛出).BPEL 中通过以下 8 种结构化活动以构造更为复杂的行为:sequence 中的各个活动按照定义的顺序,序列化执行;if 中的各个活动当指定条件满足的时候执行;循环通过 while,repeatUntil 以及 forEach(parallel="no"时)表示;flow 中的各个活动并发地执行,而 forEach 中(parallel="yes"时)的 scope 实例也是并发的;pick 活动可以根据最先收到的消息或者设定的时间信号,选择一个分支运行;scope 则刻画了流程中的事务的概念,其中,错误、补偿和终止处理(FCT-Handler)以及事件处理(eventHandlers)在 scope 中定义.除了这些活动外,process 是 BPEL 的根活动,其可定义的属性和 scope 基本相同;控制链 links 则被用来描述并发活动之间的控制依赖关系.

2.3 服务行为建模:BPEL-Petri网模型

为了对 BPEL 服务的行为进行分析,以前期工作[18]为基础,我们对单托肯(single-token)的传统的 Petri 网加以扩展,定义了 BPEL-Petri 网模型.同时,我们还构造了 BPEL 服务行为规约到 BPEL-Petri 网模型的转换,以从 BPEL 服务代码中抽取出服务的形式化行为模型.

定义 2(Petri 网[14]). Petri 网是一个元组 $PN=(P,T,F,\mu_0)$,其中:P 是库所的有穷集;T 是变迁的有穷集,且 $P\cap T=\varnothing$;$F\subset(P\times T)\cup(T\times P)$是流关系;$\mu_0\subset P$ 是初始标识,其中,标识 μ 是 P 的子集.

对于库所 $p\in P$,我们定义 ${}^\bullet p=\{t\in T|(t,p)\in F\}$和 $p^\bullet=\{t\in T|(p,t)\in F\}$分别表示 p 的前驱和后继变迁.类似的,对于变迁 $t\in T$,${}^\bullet t=\{p\in P|(p,t)\in F\}$和 $t^\bullet=\{p\in P|(t,p)\in F\}$分别表示 t 的前驱和后继库所.如果 ${}^\bullet t\subseteq\mu$,则在标识 μ 下变迁 t 是使能的;否则,变迁 t 是非使能的.我们用 $enabled(\mu)$表示在在标识 μ 下所有使能的变迁.

Petri 网模型能够自然地描述工作流中并发、选择、循环等结构,适合于作为描述服务工作流的形式模型的基础[10,13].而为了更准确描述 BPEL 服务的行为规约,我们扩展了传统 Petri 网的概念,提出了 BPEL-Petri 网模型.

定义 3(BPEL-Petri 网模型). BPEL-Petri 网模型(简称为 BPN 模型)是一个元组 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$,其中,$PN=(P,T,F,\mu_0)$是一个 Petri 网;$\Lambda$是消息收发事件的有穷集,$\lambda:T\to\Lambda$是变迁上的事件关联函数,$\mu_F\subset P$ 是终止标识.在 BPN 模型中,μ_0 和 μ_F 都必须是单元素集.

对于 Λ 中的事件,我们用 $?m$ 表示接收消息 m 的事件,而用 $!m$ 表示发送消息 m 的事件.对于 N 中那些不对应消息收发事件的变迁 t,我们说这些变迁关联了空事件,用 $\lambda(t)=\varepsilon$来表示.

通过应用前期工作中 BPEL 流程建模的方法[18],可以得到给定服务的 BPN 模型.ATM 服务的 BPN 模型如图 3 所示,其中所有的库所和变迁都被标注了数字以互相区分.变迁关联的事件标注在变迁的旁边.从图中可以看出,该服务网的初始标识是 0 号库所构成的标识,而终止标识是 7 号的库所构成的标识.

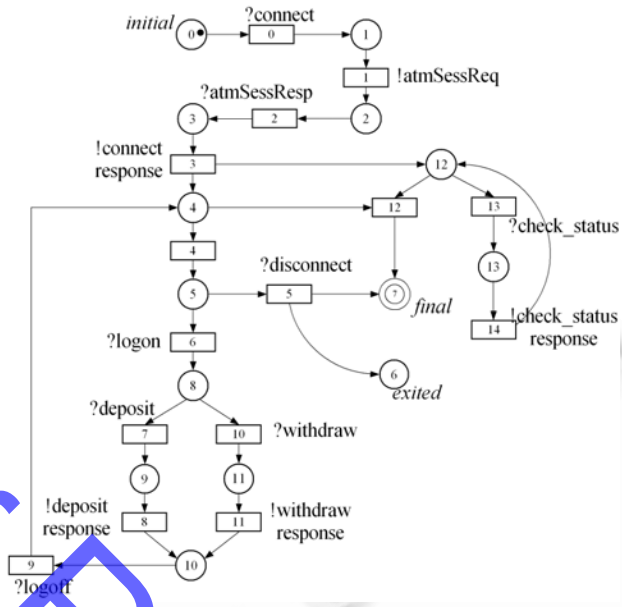


Fig.3 BPN model of the ATM service

图 3 ATM 服务的 BPN 模型

将 BPEL 服务转换为 BPN 模型之后,BPN 模型的执行语义通过运行来刻画.

定义 4(运行). BPN 模型 N 的运行是一个由标识和变迁组成的有穷或无穷的序列:

$$r=\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} \mu_n \xrightarrow{t_n} \ldots$$

其中,$n\geqslant 0$,且对于任何 $i(i\geqslant 0)$有 $t_i\in enabled(\mu_i)$,并且对于任何 $i(i\geqslant 1)$有 $\mu_i=(\mu_{i-1}-{}^{\bullet}t_{i-1})\cup t_{i-1}{}^{\bullet}$.

从运行 $r=\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} \mu_n \xrightarrow{t_n} \ldots$,可以得到变迁序列 $t_0\wedge t_1\wedge\ldots\wedge t_{n-1}\wedge t_n\wedge\ldots$,从这个变迁序列中移除关联空事件的变迁,可以得到 $t_i\wedge t_j\wedge\ldots\wedge t_k\wedge\ldots$,我们说 r 的事件序列是

$$trace(r)=\lambda(t_i)\wedge\lambda(t_j)\wedge\ldots\wedge\lambda(t_k)\wedge\ldots$$

3 服务行为分析

为了对目标服务的行为进行调控,即抽取或过滤指定行为场景,就需检查服务 BPN 模型中是否有包含顺序图描绘场景的行为,并记录下所有这样的行为.服务行为分析根据目标服务 BPN 行为模型和顺序图需求模型,遍历 BPN 模型的状态空间,获取其中包含顺序图描绘场景的行为.我们提出了一种基于并发变迁遍历 BPN 模型获取路径的算法,避免不必要的线性化,提高遍历的效率.在获取路径时,我们也记录了循环结构,并避免了循环带来的无穷行为.服务行为分析的结果是一组路径的集合,这些路径表示了目标服务中包含顺序图描绘场景的所有行为,并将作为构建行为调控服务的依据.

3.1 并发变迁和路径

基于 Petri 网的模型可以自然地表现并发.运行中,不同并发分支的变迁需在运行中交错触发,称为并发的线性化.在行为分析中,与顺序图描绘场景无关的变迁可能位于不同的并发分支中,而我们并不关心这样的变迁交错触发的顺序.同时,线性化的并发行为是各个并发分支的笛卡尔积.当并发分支的长度较长或数量较多时,BPN 模型的状态空间就会随着并发行为的线性化而迅速增长,导致状态空间爆炸.为此,针对行为分析的需求,我们借鉴并适当扩展了文献[5]中提出的并发变迁和路径的概念.

直觉上,并发变迁将所有本应在一个标识下分别触发的变迁同时触发.而由于行为分析的目的是找到 BPN 模型中包含顺序图描绘场景的行为,顺序图中事件对应的变迁仍分别触发,置于不同的并发变迁中.

定义 5(并发变迁). 设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$是 BPN 模型,$D=(O,E,M,G,V)$是顺序图,且 $m>0$.标识μ上的并发变迁τ是一个变迁的集合$\{t_1,t_2,\ldots,t_m\}$,其中:

1) 对于任何 $i(1\leqslant i\leqslant m),t_i\in enabled(\mu)$;

2) 对于任何 $i,j(1\leqslant i<j\leqslant m),{}^\bullet t_i\cap {}^\bullet t_j=\varnothing$;

3) 对于任何 $i(1\leqslant i\leqslant m)$,若$\lambda(t_i)\in E$,则$\tau$是单元素集,即 $m=1$.

并发变迁τ的触发即触发其中所有变迁 $t_1,t_2,\ldots,t_m$,新的标识 $\mu'=\bigcup_{1\leqslant i\leqslant m}(\mu-{}^\bullet t_i)\cup t_i^\bullet$,并用$\mu'=fire(\mu,\tau)$表示.

因此,从以上定义可以看出,标识μ下的并发变迁是一组变迁的集合,它们都在μ下使能.如果变迁关联的事件在 $D=(O,E,M,G,V)$的事件集 E 中,则该变迁应置于单元素的并发变迁中;否则,在该标识下使能的变迁可置于同一个并发变迁中.并且,每个并发变迁中不含冲突的变迁.所谓变迁 t_1 和 t_2 冲突,当且仅当它们不相同且有共同的前驱库所,即 $t_1\neq t_2\wedge {}^\bullet t_1\cap {}^\bullet t_2\neq\varnothing$.设 $m>0$,对于并发变迁$\tau=\{t_1,t_2,\ldots,t_m\}$,我们说其对应的并发事件是$\tau$中所有变迁关联事件组成的集合,用$\lambda(\tau)=\{\lambda(t_1),\lambda(t_2),\ldots,\lambda(t_m)\}$表示.

定义 6(路径、片段). 设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$是 BPN 模型.$N$ 的路径σ是一个由标识和并发变迁组成的有穷或无穷序列:$\sigma=\mu_0\xrightarrow{\tau_0}\mu_1\xrightarrow{\tau_1}\ldots$,且对于任何 $i(i>0),\mu_i=fire(\mu_{i-1},\tau_{i-1})$.称$\sigma$的任何一个子序列为$\sigma$的一个片段.

设ρ是 BPN 上的片段,我们用 $P(\rho),T(\rho)$分别表示该片段上所有的库所和变迁的集合.

此外,我们定义$\lambda(\rho)=\{\lambda(t)|t\in T(\rho)\}$.从片段的定义可见,路径可以看作是片段的特殊情况:当片段从 BPN 的初始标识开始时,片段即是路径.

运行可看作是路径的线性化执行,即路径中并发的变迁交错触发.因各并发分支中的变迁交错触发而形成的不同的运行,可用一条路径来表示.因此,我们可以通过路径更为简洁地刻画 BPN 模型的行为.

由于 BPN 模型循环中的行为往往不能丢弃,但循环却可能使路径的总数和长度变为无穷.为了将问题域收缩到有限范围内,同时能够记录下 BPN 模型中必要的循环结构,我们定义了简单和单次重复片段/路径.

定义 7(简单、单次重复片段/路径). 设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$为 BPN 模型,$D=(O,E,M,G,V)$是顺序图,而$\rho=\mu_m\xrightarrow{\tau_m}\mu_{m+1}\xrightarrow{\tau_{m+1}}\ldots(m\geqslant 0)$是 N 上某个路径的片段.我们说ρ是简单的,如果ρ中没有重复执行,即对于任何 i 和 $j,m\leqslant i<j,t_i\in\tau_i$ 且 $t_j\in\tau_j$,有${}^\bullet t_i\cap {}^\bullet t_j=\varnothing$.片段$\rho$是单次重复片段,如果对于任何 $i\geqslant m,\rho$中都最多只有一次 t_i 的重复执行,即最多一个 $\tau_j(j>i)$使得$\exists t_i\in\tau_l,t_j\in\tau_j,{}^\bullet t_i\cap {}^\bullet t_j\neq\varnothing$.若令 $m=0$,上述的简单片段和单次重复片段即相对应地为简单路径和单次重复路径.

可见,由于 BPN 模型中库所和变迁都是有穷的,所以简单和单次重复路径的数量和长度也必然是有穷的.

3.2 针对场景的行为分析

本节介绍针对场景行为分析的相关概念和算法.我们给出顺序图 D 描绘的场景在路径σ中出现的定义,并设计了遍历 BPN 模型 N 的状态空间的算法,以获取 D 所描绘场景出现的路径.

设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$是 BPN 模型,$D=(O,E,M,G,V)$是顺序图.对于 N 上任何路径σ的片段 $\rho=\mu_i\xrightarrow{\tau_i}\mu_{i+1}\xrightarrow{\tau_{i+1}}\ldots\xrightarrow{\tau_{j-1}}\mu_j$,我们可以得到并发事件的序列$\lambda(\tau_i)\wedge\lambda(\tau_{i+1})\wedge\ldots\wedge\lambda(\tau_j)$,从中移除所有空事件$\varepsilon$和空集,即可得到非空的事件集合的序列$\eta_0\wedge\eta_1\wedge\ldots\wedge\eta_m(m<j-i)$.如果对于任意 $k(0\leqslant k\leqslant m),\eta_k$ 都是单元素集,我们即可得到事件的序列$\xi=e_0\wedge e_1\wedge\ldots\wedge e_m$,其中,$e_k=\{\eta_k\}$.如果$\xi$是 D 所描绘场景在 N 对应服务上的消息踪迹,我们说顺序图 D 描绘的场景在ρ或σ中出现,并说ρ或σ是 D 的像.并且,如果$\eta_0=\lambda(\tau_l)$且$\eta_m=\lambda(\tau_j)$,我们说ρ是 D 的真像.

基于前面给出的定义,我们称 D 在 N 中出现的路径为Δ-路径.Δ-路径中包含 D 所描绘场景出现的真像片段,而同时将 N 中所有必要的循环结构记录下来.

定义 8(Δ-路径). 设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$为 BPN 模型,$D=(O,E,M,G,V)$是顺序图.我们称有穷路径:

$$\sigma=\mu_0\xrightarrow{\tau_0}\mu_1\xrightarrow{\tau_1}\ldots\xrightarrow{\tau_{l-1}}\mu_l\xrightarrow{\tau_l}\ldots\xrightarrow{\tau_{m-1}}\mu_m\xrightarrow{\tau_m}\ldots\xrightarrow{\tau_{n-1}}\mu_n\ (0\leqslant l<m<n),$$

是Δ-路径当且仅当下面的条件都满足:

1) $\mu_F\subseteq\mu_n$;

2) 片段 $\rho_{pre}=\mu_0\xrightarrow{\tau_0}\mu_1\xrightarrow{\tau_1}\ldots\xrightarrow{\tau_{l-1}}\mu_l$ 和 $\rho_{post}=\mu_m\xrightarrow{\tau_m}\mu_{m+1}\xrightarrow{\tau_{m+1}}\ldots\xrightarrow{\tau_{n-1}}\mu_n$ 都是单次重复片

段,且不是 D 的像;

3) 对于任意变迁 $t\in\tau_i$,其中,$i\in[0,l)\cup[m,n)$且$\lambda(t)\notin E$,都不存在这样的$j<i\wedge j\in[0,l]\cup[m,n)$满足 $t\in enabled(\mu_j)$,除非$\exists t'\in\tau_j, {}^\bullet t'\cap {}^\bullet t\neq\varnothing$;

4) $\rho_{im}=\mu_l\xrightarrow{\tau_l}\mu_{l+1}\xrightarrow{\tau_{l+1}}\ldots\xrightarrow{\tau_{m-1}}\mu_m$ 是 D 的真像,并且对于任何 $i,j\in[l,m)$,若不存在 $k\in[i,j]$使得τ_k是单元素集并且$\lambda(\tau_k)\subseteq E$,则片段 $\mu_i\xrightarrow{\tau_i}\ldots\xrightarrow{\tau_j}\mu_{j+1}$ 是简单片段.

其中,我们称ρ_{pre},ρ_{im}和ρ_{post}为σ的前像片段、真像片段和后像片段.

从定义 8 的条件 1)可见,每条Δ-路径都是有穷路径,且 BPN 模型 N 的终止标识是Δ-路径最后标识的子集.每条Δ-路径都仅存在一个真像片段对应顺序图 D 描绘的场景,而前像、后像片段分别是在真像片段之前和之后遍历的片段,因此,定义 8 中的条件 2)要求前像和后像片段都是单次重复片段,且只有真像片段对应 D 所描绘场景的出现.条件 3)是为了减少记录的刻画同样行为的路径数量,若变迁 t 在前像和后像片段的某个标识下使能,则在前像或后像片段中,要么 t 在其前的所有标识下都不使能,要么其前使能 t 的标识已经触发了 t 或与 t 冲突的变迁;条件 4)要求真像片段的任何子片段如果不是简单片段,那么其中一定要有对应 E 中事件的变迁构成的并发变迁.

设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$是 BPN 模型,$D=(O,E,M,G,V)$是顺序图.我们将算法 1 计算的 N 中的Δ-路径记录在集合$\Delta(N,D)$中.该算法通过触发并发变迁,从初始标识μ_0 以深度优先的顺序遍历 N,并利用简单和单次重复片段的概念处理循环.已遍历的路径片段记录在变量 *curpath* 中.对于通过触发并发变迁发现的新标识(算法中用变量 *node* 表示),我们将并发变迁和新的标识附加到 *curpath* 后,开始下一次迭代.当没有新的标识可以加入到 *curpath* 时则回溯,即从 *curpath* 中删去最后的节点和其前最后一个并发变迁.而在回溯之前,若 *curpath* 为Δ-路径,则将 *curpath* 加入到$\Delta(N,D)$;否则,若 *node* 包含终止标识μ_F,则将 *failcnt* 增加 1.这样,算法在遍历记录Δ-路径的同时,还将遍历过程中发现的包含以μ_F结束且不满足Δ-路径条件的路径数量统计在失败路径计数 *failcnt* 中.算法 1 的复杂度是与 N 中Δ-路径的前缀个数以及Δ-路径的最长前缀的长度成比例关系的.这里,前缀是指一个片段,该片段能够被进一步扩展为一条Δ-路径.

算法 1. 行为分析算法.

输入:服务 BPN 模型 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$、顺序图 $D=(O,E,M,G,V)$;

输出:$\Delta(N,D)$,*failcnt*.

令 $curpath=\mu_0$,$\Delta(N,D)=\varnothing$,*failcnt*=0;

repeat {

  令 *node* 为 *curpath* 的最后一个节点;

  if *node* 通过触发某个并发变迁τ可到达的新的后继节点 **then** {

    令 *node* 为通过触发该并发变迁τ得到的新后继节点;

    将 $\xrightarrow{\tau}$*node* 附加到 *curpath* 后}

  else {

    if *curpath* 对应的路径为Δ-路径 **then** {将 *curpath* 对应的路径加入$\Delta(N,D)$}

    elseif $\mu_F\in$*node* **then** {*failcnt*++}

    删除 *curpath* 的最后的节点及其前的最后一个并发变迁}}

until *curpath* 为空;

return $\Delta(N,D)$和 *failcnt*

定理 1. 令 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$为 BPN 模型,$D=(O,E,M,G,V)$是顺序图,则$\Delta(N,D)$是包含 N 中所有Δ-路径的最小集合.**

ATM 服务的 BPN 模型 N 如图 3 所示,用户的取款场景顺序图 D 如图 1(b)所示,则 ATM 服务在取款场景下

** 受篇幅所限,定理 1~定理 3 的证明参见我们的技术报告:http://seg.nju.edu.cn/BASIS010/mani_repo.pdf

的行为分析结果$\Delta(N,D)$中的 3 条Δ-路径如下所示:

$$\sigma_1=\{p_0\}\xrightarrow{\{t_0\}}\{p_1\}\xrightarrow{\{t_1\}}\{p_2\}\xrightarrow{\{t_2\}}\{p_3\}\xrightarrow{\{t_3\}}\{p_4,p_{12}\}\xrightarrow{\{t_4,t_{13}\}}\{p_5,p_{13}\}\xrightarrow{\{t_{14}\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_6\}}$$
$$\{\boldsymbol{p_8},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{10}\}}\{\boldsymbol{p_{11}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{11}\}}\{\boldsymbol{p_{10}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_9\}}\{\boldsymbol{p_4},\boldsymbol{p_{12}}\}\xrightarrow{\{t_4\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_5\}}\{\boldsymbol{p_6},\boldsymbol{p_7},\boldsymbol{p_{12}}\}$$

$$\sigma_2=\{p_0\}\xrightarrow{\{t_0\}}\{p_1\}\xrightarrow{\{t_1\}}\{p_2\}\xrightarrow{\{t_2\}}\{p_3\}\xrightarrow{\{t_3\}}\{p_4,p_{12}\}\xrightarrow{\{t_4,t_{13}\}}\{p_5,p_{13}\}\xrightarrow{\{t_{14}\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_6\}}$$
$$\{\boldsymbol{p_8},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{10}\}}\{\boldsymbol{p_{11}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{11}\}}\{\boldsymbol{p_{10}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_9\}}\{\boldsymbol{p_4},\boldsymbol{p_{12}}\}\xrightarrow{\{t_4\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_6\}}\{\boldsymbol{p_8},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{10}\}}$$
$$\{\boldsymbol{p_{11}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{11}\}}\{\boldsymbol{p_{10}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_9\}}\{\boldsymbol{p_4},\boldsymbol{p_{12}}\}\xrightarrow{\{t_4\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_5\}}\{\boldsymbol{p_6},\boldsymbol{p_7},\boldsymbol{p_{12}}\}$$

$$\sigma_3=\{p_0\}\xrightarrow{\{t_0\}}\{p_1\}\xrightarrow{\{t_1\}}\{p_2\}\xrightarrow{\{t_2\}}\{p_3\}\xrightarrow{\{t_3\}}\{p_4,p_{12}\}\xrightarrow{\{t_4,t_{13}\}}\{p_5,p_{13}\}\xrightarrow{\{t_{14}\}}\{p_5,p_{12}\}\xrightarrow{\{t_6\}}$$
$$\{p_8,p_{12}\}\xrightarrow{\{t_7\}}\{p_9,p_{12}\}\xrightarrow{\{t_8\}}\{p_{10},p_{12}\}\xrightarrow{\{t_9\}}\{p_4,p_{12}\}\xrightarrow{\{t_4\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_6\}}\{\boldsymbol{p_8},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{10}\}}$$
$$\{\boldsymbol{p_{11}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_{11}\}}\{\boldsymbol{p_{10}},\boldsymbol{p_{12}}\}\xrightarrow{\{t_9\}}\{\boldsymbol{p_4},\boldsymbol{p_{12}}\}\xrightarrow{\{t_4\}}\{\boldsymbol{p_5},\boldsymbol{p_{12}}\}\xrightarrow{\{t_5\}}\{\boldsymbol{p_6},\boldsymbol{p_7},\boldsymbol{p_{12}}\}$$

其中:库所用 p 加上图 3 中的数字标注表示;类似地,变迁用 t 加上图 3 中的数字标示表示;真像中的库所和变迁用加粗字体表示.因为取款场景在 ATM 服务中需要用循环来表现,所以本例中,Δ-路径的真像片段都不是简单片段.

通过行为分析,目标服务 BPN 模型 N 中顺序图 D 描绘场景出现的所有行为通过路径集合$\Delta(N,D)$记录下来.路径集合$\Delta(N,D)$将作为下面构造调控服务的基础.

4 构造调控服务

通过行为分析中对目标服务 BPN 模型的遍历,我们得到了一组Δ-路径以表示包含顺序图描绘场景的行为.如果行为分析的结果给出的$\Delta(N,D)$为空,说明 D 所描绘的场景在 N 中不出现:对于行为抽取,则无法使得该场景出现;对于行为过滤,则不需调控已可避免该场景的发生;但同时,如果算法 1 记录的 *failcnt* 为 0,则说明遍历的所有路径都是 D 场景的像,因而,若用户需要过滤该行为时,这个需求不能得到满足.因此,如果$\Delta(N,D)$不为空,且当用户需求是过滤 D 所描绘的场景时,*failcnt* 不为 0,我们就可以利用$\Delta(N,D)$中的路径为目标服务构造满足用户需求的行为约束自动机以及调控服务.

服务行为调控,即根据行为分析得到的Δ-路径,监听、检查和过滤用户与目标服务之间的消息,保证目标服务运行时有恰当的消息交互,从而表现出满足用户需求的行为.虽然Δ-路径能够刻画目标服务中包含顺序图描绘场景的行为,但是却不易直接用于调控目标服务的消息交互.因此,我们需要根据路径集合$\Delta(N,D)$构造能够“运行”的行为约束自动机和调控服务.首先,我们以事件自动机作为行为约束自动机的形式基础,并设计了算法为$\Delta(N,D)$中的每条路径分别构造对应的事件自动机,以刻画该Δ-路径所表示的包含 D 所描绘场景出现的服务行为;其次,将这些事件自动机整合并化简,得到最终完整的行为约束自动机;最后,我们介绍了调控服务如何在在运行时利用行为约束自动机,实现对目标服务的行为调控,以及原型工具 BASIS.

4.1 事件自动机

作为行为约束自动机的形式基础,我们首先定义事件自动机的概念,然后设计算法,为每条Δ-路径构造刻画其描述的行为的事件自动机.

定义 9(事件自动机). 设 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$为 BPN 模型,$D=(O,E,M,G,V)$是顺序图.事件自动机是一个元组 $U=(\alpha,S,\delta,s_0,S_F,S_{pre},S_{im},S_{post})$,其中:$\alpha\subseteq\Lambda$是事件集合;$S$ 是状态的有穷集;$\delta:S\times\alpha\to 2^S$是带事件的边函数;$s_0\in S$和$S_F\subseteq S$分别是初始状态和终止状态集;$S_{pre},S_{im},S_{post}\subseteq S$ 分别表示 U 中 D 所描绘场景出现之前、之中和之后的状态.

可以看出,事件自动机 U 能够接受的语言是消息收发事件组成的序列.由于希望 U 所接受的事件序列中都包含了 D 所描绘场景的发生,我们用 S 的子集 S_{pre},S_{im},S_{post} 分别表示 D 所描绘的场景还未发生时应在的状态、部分出现的状态以及已经完整出现后的状态.给定 $s_1,s_2\in S,e\in\alpha$,我们也用 $s_1\xrightarrow{e}s_2$ 来表示 $s_2\in\delta(s_1,e)$.

令路径$\sigma\in\Delta(N,D)$.我们为路径σ构造相应的事件自动机 $U^\sigma=(\alpha^\sigma,S^\sigma,\delta^\sigma s_0^\sigma,S_F^\sigma,S_{pre}^\sigma,S_{im}^\sigma,S_{post}^\sigma)$ 接受σ刻画的 D 所描绘场景出现的 N 的行为.此外,我们定义标签函数 $L^\sigma:S\to 2^{P(\sigma)}$,其中,$L^\sigma(s_0^\sigma)=\mu_0$ 且 $\forall s_F^\sigma\in S_F^\sigma,\mu_F\subseteq L^\sigma(s_F^\sigma)$.事

件自动机 U^σ的构造以及 L^σ的记录由算法 2 描述.其中,给定路径$\sigma\in\Delta(N,D)$的片段ρ、标识$\mu\subseteq P(\rho)$,路径σ上的事件$e\in\lambda(\rho)$(e 可为ε),我们用 $mov(\rho,\mu,e)$表示在片段ρ中,标识μ通过触发关联事件 e 的某个变迁而可达的标识集合.也就是说,$mov(\rho,\mu,e)=\{\mu'|\exists t_e\in T_e,\mu'=(\mu-{}^\bullet t_e)\cup t_e^\bullet\}$,其中,$T_e=\{t\in T(\rho)|\lambda(t)=e\wedge {}^\bullet t\subseteq\mu\}$.注意,算法 2 中的函数 *AddStatesNonIm* 和 *AddStatesIm* 均带有副作用,可以直接修改传入的 U^σ和 L^σ.

算法 2. 构造Δ-路径σ对应的事件自动机.

输入:$\sigma\in\Delta(N,D)$;

输出:U^σ和 L^σ.

将新状态 s_0^σ 加入 S^σ并设置 $L^\sigma(s_0^\sigma)=\mu_0$,$\alpha^\sigma=\lambda(\sigma)$;

AddStatesNonIm($\sigma,U^\sigma,L^\sigma,S_{pre}^\sigma$,false);

foreach $s\in S^\sigma$满足 $t_{im0}\in enabled(L^\sigma(s))$,其中,$t_{im0}$ 是ρ_{im} 的第 1 个并发变迁中唯一的变迁 **do** {

标记 s 为未标记};

AddStatesIm(σ,U^σ,L^σ);

将所有这样的状态 s:$s\in S_{post}^\sigma\wedge\forall e\in\lambda(\rho_{im}),\delta^\sigma(s,e)=\varnothing$设为未标记并加入 S_{post}^σ;

AddStatesNonIm($\sigma,U^\sigma,L^\sigma,S_{post}^\sigma$,true);

return U^σ和 L^σ

函数 *AddStatesIm*(σ,U^σ,L^σ):

foreach S^σ中未标记的状态 s **do** {

标记 s;

for τ_{im} 依次指代ρ_{im} 中从第 1 个到最后一个的并发变迁 **do** {

对于每个 $t\in\tau_{im}\cap enabled(L^\sigma(s))$,

在 $S^\sigma,\delta^\sigma(s,e)$和 S_{post}^σ 中加入新的未标记的新状态 s_{new},并令 $L^\sigma(s_{new})=(L^\sigma(s)-{}^\bullet t)\cup t^\bullet$}}

函数 *AddStatesNonIm*($\sigma,U^\sigma,L^\sigma,S_{cur},post$):

foreach S^σ中未标记的状态 s **do** {

标记 s;

foreach $P_e\in mov(\rho_{pre},s,e)\cup mov(\rho_{post},s,e)$,其中,$e\in\lambda(\rho_{pre})\cup\lambda(\rho_{post})$ **do** {

令 $S'=\{s'|(post\rightarrow s'\in S_{post}^\sigma)\wedge(\neg post\rightarrow s'\in S_{pre}^\sigma)\wedge P_e=L^\sigma(s')\}$;

if $S'=\varnothing$ **then** {

在 $S^\sigma,\delta^\sigma(s,e)$和 S_{cur} 中加入新的未标记状态 s_{new},并令 $L^\sigma(s_{new})=P_e$;

if *post* **then** {令$\delta^\sigma(s_{new},\varepsilon)=\{s_{pp}\in S_{pre}|L^\sigma(s_{pp})=L^\sigma(s_{new})\}$}}

else {令$\delta^\sigma(s,e)=\delta^\sigma(s,e)\cup S'$}}}

算法 2 从初始标识(同时也是σ的第 1 个标识)开始,依次根据σ的前像片段、真像片段和后像片段将状态和边加入 U^σ.新加入的状态都是未标记的,而新的状态和边通过在未标记状态上计算后继时加入.算法流程分别考虑σ的前像、真像和后像片段,是因为加入新状态和边的策略有所不同.如果当前处理片段是真像片段(即函数 *AddStatesIm*),只要按照真相片段中并发变迁的顺序,当前状态可通过触发该并发变迁中的某个变迁到达另一个状态,就加入一个新状态;如果当前片段不是真像片段时(即函数 *AddStatesNonIm*),新状态加入前需要检查是否存在处理当前片段时已加入的状态 s',使得 $L^\sigma(s')$等于从当前状态 s 通过触发关联事件 e 的变迁可达的标识.如果存在这样的状态,表示存在一个循环,则加入一条从当前状态到 s'标记事件为 e 的边;否则,加入新状态并设置 $L^\sigma(s_{new})$.进而,如果当前处理片段是后像片段(函数 *AddStatesNonIm* 中 *post*=true),我们加入从 s_{new} 发出的空边指向 S_{pre} 中的状态 s_{pp},其中,$L^\sigma(s_{pp})=L^\sigma(s_{new})$.事件自动机 U^σ的终止状态都是处理后像片段时加入的状态 S_{post}^σ,这样,在自动机接受消息交互事件序列、执行到终止状态时,顺序图描绘场景一定已经发生.由于前像、后像和真像片段的长度均是有穷的,因此,可见算法 2 必然能够终止.而算法 2 构造的路径σ的事件自动机将作为下节中构造

行为约束自动机的准备,故其正确性通过行为约束自动机的正确性保证(定理 2).

4.2 构造行为约束自动机

$\Delta(N,D)$中,路径σ的事件自动机 U^σ接受σ表示的包含顺序图 D 描绘场景的 BPN 模型 N 的事件序列.而通过将$\Delta(N,D)$中所有路径的事件自动机整合起来,我们就能构造出可用于整个服务的行为调控的事件自动机.对这样构造的事件自动机,在不影响顺序图描绘场景出现的前提下并进行化简之后,就得到了最终的服务行为约束自动机,其接受的事件序列是 BPN 模型 N 中所有包含顺序图 D 的事件序列,并仅接受这样的事件序列.

我们按照下面的步骤将$\Delta(N,D)$中每条路径σ的事件自动机 $U^\sigma=(\alpha^\sigma,S^\sigma,\delta^\sigma s_0^\sigma,S_F^\sigma,S_{pre}^\sigma,S_{im}^\sigma,S_{post}^\sigma)$ 整合起来,构造接受整个 BPN 模型中包含顺序图 D 出现的所有行为的事件自动机 $U(N,D)=(\alpha,S,\delta,s_0,S_F,S_{pre},S_{im},S_{post})$,并用 $L:S\to 2^P$ 作为 $U(N,D)$上的状态标签函数:

1. 将$\sigma\in\Delta(N,D)$对应自动机 U^σ中各元素依次加入 $U(N,D)$中,并向α中添加空事件,S 中添加新的初始状态 s_0,同时将该状态加入 S_{pre},并为该状态到 U^σ的各个初始状态 s_0^σ 加入空边,即

$$\alpha=\bigcup_{\sigma\in\Delta(N,D)}\alpha^\sigma\cup\{\varepsilon\},$$
$$S=\bigcup_{\sigma\in\Delta(N,D)}S^\sigma\cup\{s_0\},$$
$$\delta=\bigcup_{\sigma\in\Delta(N,D)}S^\sigma\cup\{(s_0,\varepsilon,s_0^\sigma)\mid\sigma\in\Delta(N,D)\},$$
$$S_F=\bigcup_{\sigma\in\Delta(N,D)}S_F^\sigma,$$
$$S_{pre}=\bigcup_{\sigma\in\Delta(N,D)}S_{pre}^\sigma\cup\{s_0\},$$
$$S_{im}=\bigcup_{\sigma\in\Delta(N,D)}S_{im}^\sigma,$$
$$S_{post}=\bigcup_{\sigma\in\Delta(N,D)}S_{post}^\sigma.$$

2. 令 s_0 上的标签为$\varnothing$,S 中其他状态的标签和 L^σ一样,即 $L=\{s_0\mapsto\varnothing\}\cup\bigcup_{\sigma\in\Delta(N,D)}L^\sigma$;
3. 根据 L 将 S_{pre} 划分为若干带有相同标签的最大子集(即划分得到的子集中的任意 s_1,s_2 都满足 $L(s_1)=L(s_2)$,且不存在该子集外的 s_3 而 $L(s_3)=L(s_1)$),合并同一子集中的状态,并更新和这些状态关联的边到合并后的状态;同时,相应更新标签函数 L;为 S_{post} 中的状态做同样的操作;
4. 若 $\exists s_0'\in S,\delta(s_0,\varepsilon)=\{s_0'\}$,则合并 s_0 和 s_0' 为新的初始状态,并更新和这些状态关联的边到合并后的状态;同时,相应地更新标签函数 L.

通过上面的步骤,我们将算法 2 构造的各个Δ-路径对应的事件自动机整合起来,得到的事件自动机记为 $U(N,D)$.而若事件自动机 U 接受 N 中所有包含 D 描绘的场景出现的事件序列,并且仅接受这样的事件序列,我们说 U 是 BPN 模型 N 在顺序图 D 下的行为约束自动机.

定理 2. 事件自动机 $U(N,D)=(\alpha,S,\delta,s_0,S_F,S_{pre},S_{im},S_{post})$是 BPN 模型 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$在顺序图 $D=(O,E,M,G,V)$下的行为约束自动机,即 $U(N,D)$接受 N 中所有包含 D 描绘的场景出现的事件序列,并且仅接受这样的事件序列.

尽管自动机 $U(N,D)$已可以直接用于服务行为调控,但是其中可能有空边(即带有空事件ε的边),且有可合并的状态.令 $imconnect\subseteq\delta$是 S_{im} 中状态之间事件非空的边,即 $imconnect=\{(s,e,s')\in\delta|s,s'\in S_{im}\wedge e\neq\varepsilon\}$.我们按照如下方法对自动机 $U(N,D)$化简,即消除空边并在不影响顺序图描绘场景出现的前提下对该自动机进行确定化:1) 为 $imconnect$ 中的事件构造改名函数 RN,使得改名前后的事件之间存在 1-1 映射的关系;2) 按照文献[1]中非确定有穷状态自动机确定化的方法,将按照 RN 改名之后的事件自动机确定化;3) 将确定化后事件自动机中的按照 RN 改名的事件改回原名,得到的自动机用 $M(N,D)$表示.事件自动机 $M(N,D)$即为最终的行为约束自动机.

定理 3. 由非确定行为约束自动机 $U(N,D)$化简得到的事件自动机 $M(N,D)$是 BPN 模型 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$在顺序图 $D=(O,E,M,G,V)$下的行为约束自动机.

ATM 服务在取款场景下的行为约束自动机 $M(N,D)$如图 4 所示.事件在自动机的边上标记,取款场景中的行

为用斜体表示.

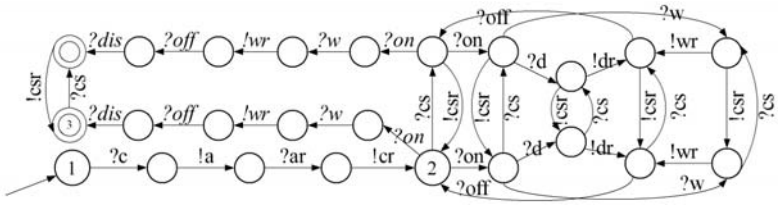


Fig.4 Behavior constraint automaton of ATM service under withdrawal scenario

图 4 ATM 服务在取款场景下的行为约束自动机

4.3 运行时的行为调控

令 N 是 BPN 模型,D 是顺序图.D 所描绘的场景驱动的行为调控需要在运行时完成.令 $M(N,D)=(\alpha,S,\delta,s_0,S_F,S_{pre},S_{im},S_{post})$是 N 在顺序图 D 下的行为约束自动机.由于 $M(N,D)$接受的 N 的事件序列都包含顺序图 D 描绘的场景的出现,而凡是行为约束自动机不能接受的事件序列必不含顺序图 D 描绘的场景的出现,调控服务可以根据行为约束自动机 $M(N,D)$对目标服务的输入消息进行过滤,使得用户能够与目标服务正确交互,保证目标服务的行为满足用户需求.

运行时的行为调控根据用户的调控需求,分别针对抽取或过滤顺序图描绘场景的两种情况进行.

当用户需求是抽取顺序图描绘的场景时,如果当前消息可能使得该场景发生,则应向目标服务转发,否则,过滤该消息并告知用户应当发送的消息,以保证该场景在目标服务的此次执行中出现.令 $Current\subseteq S$ 为当前使能状态集合,并在调控开始时初始化为$\{s_0\}$.调控服务监听用户与目标服务之间的消息交互.令函数 $Next$: $2^S\times\alpha\to 2^S$ 表示从 $Current$ 中所有状态通过事件 e 可达的状态集,即 $Next(Current,e)=\{s'\in S|\exists s\in Current,s'\in\delta(s,e)\}$,并且当前调控服务监听到的消息 m 在目标服务 BPN 模型中相应的事件是 e_m.若 $Next(Current,e_m)\neq\varnothing$,则将当前消息 m 转发,并更新 $Current$ 为 $Next(Current,e_m)$.否则,告知用户为满足行为抽取的需求应发送的消息,即从 $Current$ 的所有状态发出的边上的各个事件对应的消息 $Expect=\{m_x|\exists s\in Current,\delta(s,e_x)\neq\varnothing\}$,其中,$e_x$ 是消息 m_x 在目标服务 BPN 模型中相应的事件.

当用户需求是过滤顺序图描绘的场景时,如果当前消息不能避免指定场景的出现,则过滤该消息并告知用户应当发送的消息.否则,应向目标服务转发当前消息.设当前状态集合 $Current\subseteq S$ 和可达状态函数 $Next$:$2^S\times\alpha\to 2^S$ 与行为抽取情况下定义相同,且当前消息 m 在目标服务 BPN 模型中相应的事件是 e_m.如果 $Current$ 中存在某状态通过当前消息收发事件可达的状态在 S_{im} 中,即 $Next(Current,e_m)\cap S_{im}\neq\varnothing$,则告知用户为满足行为过滤的需求,应发送的消息 $Expect=\{m_x|Next(Current,e_x)\cap S_{im}=\varnothing\}$,其中,$e_x$ 是消息 m_x 在目标服务 BPN 模型中相应的事件;否则,将消息转发,并更新 $Current$ 为 $Next(Current,e_m)$.

对于第 1.1 节中的两个消息交互序列 ms_1 和 ms_2,如需抽取取款场景,调控服务从图 4 中标号为 1 的初始状态开始运行行为约束自动机.消息交互序列 ms_1 中的消息都应依次转发给 ATM 服务.流程终止于标号为 3 的终止状态上,取款场景发生.而对于 ms_2,则在收到?disconnect 消息时,$Current$ 中为图 4 中标号为 2 的状态,此时应回复用户为使取款场景发生,应发送消息为?logon 或?check_status.

4.4 原型工具

基于上述研究,我们用 Java 语言实现了原型工具 BASIS(behavior analyzer for interactive services)以支撑场景驱动的 Web 服务的行为调控途径(BASIS 详情和下载见:http://seg.nju.edu.cn/BASIS010/).该工具按照图 2 中的行为调控流程进行.BASIS 的输入是目标服务的 BPEL 行为规约和 WSDL 接口以及描绘了用户需求中场景的 UML 顺序图模型.工具的输出是转化得到的 BPN 模型、Δ-路径集合以及调控服务——包括行为约束自动机、一个 BPEL 消息过滤服务和一个 Java 消息交互检查服务.其中,BPEL 服务“包装”了目标服务,监听用户与目标服务的消息交互;收到消息之后,调用 Java 的消息交互检查服务,检查当前消息交互是否恰当:如果是,则 BPEL

消息过滤服务将当前消息转发,否则,告知用户为实现行为抽取或过滤的目标应发送的消息.用户应将调控服务部署在相应的服务引擎上,通过调用消息过滤服务,在运行时实现对目标服务的行为调控.我们使用 BASIS 为 ATM 服务抽取或过滤取款场景分别生成了调控服务,并进行了部署和测试,验证了本文提出的服务行为调控的可行性.

5 结论和进一步工作

Web 服务在异构和分布式系统的交互中发挥着重要作用.对用户来说,服务的行为调控是保证第三方服务满足其需要的重要手段.本文提出的场景驱动服务行为调控途径,通过在运行时监听、检查和转发用户和服务之间的消息交互,使得目标服务的执行满足用户抽取或过滤顺序图描绘的场景的需求.输入的服务 BPEL 行为规约首先被转为 BPN 模型,用户需求中期望抽取或过滤的行为通过顺序图场景来描绘.行为分析中,我们提出了算法遍历 BPN 模型,获取目标服务 BPN 模型中所有包含顺序图描绘场景出现的路径的集合,同时缩减了遍历 BPN 模型时所需的状态空间.行为分析的结果被用来构造行为约束自动机,接受 BPN 模型中顺序图描绘的场景出现的所有事件序列,并且仅接受这样的事件序列.我们还构造了调控服务,以在运行时利用行为约束自动机,检查并过滤输入消息,完成对目标服务的行为调控.同时,我们设计的原型工具 BASIS 对服务行为调控的整个过程提供了支撑.通过应用本文提出的服务行为调控途径,服务的使用者即可利用调控后的第三方服务构造满足其需求的业务流程.

在将来的工作中,我们会考虑 BPEL 规约中的错误、补偿和终止处理流程相关的行为,还将把数据流结合到行为调控途径中来,以增强行为调控的能力,使我们对服务行为的调控工作更为深入全面,并不断推进工业界的实际应用.

References:

[1] Aho AV, Sethi R, Ullman JD. Compilers: Principles, Techniques, and Tools. 2nd ed., Boston: Addison Wesley, 2006.

[2] Beek MH, Bucchiarone A, Gnesi S. Formal methods for service composition. Annals of Mathematics, Computing & Teleinformatics, 2007,1(5):1−14.

[3] Bertoli P, Pistore M, Traverso P. Automated composition of Web services via planning in asynchronous domains. Artificial Intelligence, 2010,174(3-4):316−361. [doi: 10.1016/j.artint.2009.12.002]

[4] Breugel F, Koshkina M. Models and verification of BPEL. Technical Report, Report No. M3J1P3, Toronto: York University, 2006.

[5] Chen MS, Qiu XK, Xu W, Wang LZ, Zhao JH, Li XD. UML activity diagram-based automatic test case generation for Java programs. The Computer Journal, 2009,52(5):545−556. [doi: 10.1093/comjnl/bxm057]

[6] Dustdar S, Schreiner W. A survey on Web services composition. Int'l Journal of Web and Grid Services, 2005,1(1):1−30. [doi: 10.1504/IJWGS.2005.007545]

[7] Fu X, Bultan T, Su JW. Analysis of interacting BPEL Web services. In: Proc. of the 13th Int'l Conf. on World Wide Web. 2004. 621−630. [doi: 10.1145/988672.988756]

[8] Object Management Group. Unified Modeling Language. version 1.4.2, ISO/IEC 19501, 2004.

[9] Li XD, Hu J, Bu L, Zhao JH, Zheng GL. Consistency checking of concurrent models for scenario-based specifications. In: Proc. of the 12th Int'l System Design Languages Forum. 2005. 298−312. [doi: 10.1007/11506843_21]

[10] Lohmann N. A feature-complete Petri net semantics for WS-BPEL 2.0. In: Proc. of the 4th Int'l Conf. on Web Services and Formal Methods. 2008. 77−91. [doi: 10.1007/978-3-540-79230-7_6]

[11] Lohnmann N, Massuthe P, Wolf K. Behavioral constraints for services. In: Proc. of the 5th Int'l Conf. on Business Process Management. 2007. 271−287. [doi: 10.1007/978-3-540-75183-0_20]

[12] Marconi A, Pistore M, Traverso P. Specifying data-flow requirements for the automated composition of Web services. In: Proc. of the 4th IEEE Int'l Conf. on Software Engineering and Formal Methods. 2006. 147−156. [doi: 10.1109/SEFM.2006.34]

[13] Ouyang C, Verbeek E, van der Aalst WMP, Breutel S, Dumas M, ter Hofstede AHM. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007,67(2-3):162−198. [doi: 10.1016/j.scico.2007.03.002]

[14] Peterson JL. Petri Net Theory and the Modeling of Systems. Prentice Hall, PTR, 1981.

[15] OASIS Web Services Business Process Execution Language (WSBPEL) TC. Web Services Business Process Execution Language, version 2.0, OASIS, 2007.

[16] World Wide Web Consortium (W3C). Web Services Description Language (WSDL) 1.1. W3C Note, 2001.

[17] Li XD, Wang LZ, Qiu XK, Lei B, Yuan JS, Zhao JH, Zheng GL. Runtime verification of Java programs for scenario-based specifications. In: Proc. of the Ada-Europe. 2006. 94−105. [doi: 10.1007/11767077_8]

[18] Yang L, Liu X, Wang LZ, Chen X, Li XD. Scenario-Based analysis and verification for Web services message flows. Chinese Journal of Computers, 2009,32(9):1759−1772 (in Chinese with English abstract).

[19] Yi XC, Kochut KJ. A CP-nets-based design and verification framework for Web services composition. In: Proc. of the IEEE Int'l Conf. on Web Services. 2004. 756−760. [doi: 10.1109/ICWS.2004.1314810]

附中文参考文献:

[18] 杨璐,柳溪,王林章,陈鑫,李宣东.面向基于场景规约的 Web 服务消息流分析与验证.计算机学报,2009,32(9):1759−1772.

柳溪(1984－),男,陕西西安人,博士生,主要研究领域为软件工程,软件验证,服务计算.

潘敏学(1983－),男,博士生,主要研究领域为模型验证,实时和并发系统的设计与分析.

杨璐(1982－),女,博士,讲师,CCF 会员,主要研究领域为软件工程,软件验证,软件监控.

王林章(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为模型驱动的软件测试与验证,软件测试自动化.