基于场景规约的服务行为调控*

柳溪[1,2+]，杨璐[3,1]，潘敏学[1,2]，王林章[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,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)

Service Behavior Manipulation according to Scenario-based Specifications

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

[1](State Key Laboratory of 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: Phn +86-13914798167, Fax +86-25-83594683, E-mail: liux@seg.nju.edu.cn, http://seg.nju.edu.cn/

Abstract: Web services provide one of the typical implementations for Software Oriented Architecture. Behavior manipulation is important to the users of third-party services when only part of the service behavior is of concern. It is a worth-researching issue to study how to allow the service users to extract desired behavior or filter out undesired behavior from the target service. This paper proposes an approach for web services behavior manipulation according to scenario-based specifications. Firstly, we use UML Sequence Diagram to describe user's requirement as scenario-based specifications, and provide modeling of BPEL-based behavior specification as BPEL-Petri nets model (BPN model for short) to represent the service behavior. Secondly, the service behavior is analyzed based on paths of the BPN model utilizing the notion of concurrent transitions, and the set of behavior according to scenario-based specifications is obtained by traversing the BPN model. Finally, by using the result of behavior analysis, we construct the manipulator services which listen, check and filter the message exchange between the user and the target service at run-time to extract or filter out the scenario specified by Sequence Diagram. Based on the research, we developed a prototype tool called BASIS to realize the behavior manipulation, and a case study is given to illustrate the feasibility of our approach.

Key words: web services; scenario-based specifications; behavior manipulation

摘 要: Web 服务是面向服务架构(SOA)的典型实现之一。当用户只关心第三方服务的部分行为时，行为调控对于用户来说非常重要，如何使得用户可以从目标服务中抽取期望出现的行为或过滤不期望出现的行为，成为一个值得研究的课题。本文提出了一个基于场景规约的服务行为调控途径。首先，我们用顺序图对用户指定的服务行为需求进行描绘以形成场景规约，并且基于服务的 BPEL 行为规约，构造表示服务行为的 BPEL-Petri 网的模型（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

* **作者简介**: **柳溪**(1984－),男,陕西西安人,博士研究生,主要研究领域为软件工程,软件验证和服务计算; **杨璐**(1982－),女,博士,讲师,主要研究领域为软件工程,软件验证和软件监控; **潘敏学**(1983－),男,博士研究生,主要研究领域为模型验证, 实时和并发系统的设计和分析; **王林章**(1973－),男,博士,副教授,主要研究领域为模型驱动的软件测试与验证和软件测试自动化.

遍历 BPN 模型获取符合场景规约的服务行为集合。最后，根据行为分析的结果，我们构建了行为调控服务，以在运行时监听、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。基于上述研究，我们开发了原型工具 BASIS，支持对服务行为的调控，并通过实例研究展示了本文方法的可行性。

关键词: web 服务; 场景规约; 行为调控

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。尽管 SOA 不一定是基于 web 的，但是 web 服务的标准化使其成为了 SOA 最佳实现之一。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的消息交互。对于某个第三方服务，用户 1)可能需要比该服务所提供的更多更复杂的行为，2)也可能仅需要该服务提供的一部分行为。对于第一种情况，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]。而对于第二种情况，我们就需要“调控”目标服务的行为，也就是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，由于用户通常没有修改第三方服务的代码或部署的平台权限，只能利用服务消息驱动的特性，通过调控与第三方目标服务的消息交互来调控其行为。

本文中，我们提出了基于场景规约的 web 服务行为调控的框架，为输入的目标服务建立模型，分析目标服务的行为，并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的消息，保证目标服务的恰当消息交互，实现针对 UML 顺序图所描绘场景的行为抽取或过滤，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[1]。该服务行为如图 1(A)所示：ATM 服务接收 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；接下来，ATM 接受客户的 logon（登录系统）请求，之后接受 withdraw（取款）、deposit（存款）或 logoff（登出系统）消息并作相应处理和必要响应；在收到 disconnect（断开连接）消息之前，ATM 还可继续接收 logon 请求让用户重新登录并再次进行上述操作，直到收到 disconnect 消息，流程结束。在发送 connect 确认之后，ATM 服务还可以并行的在事件处理流程（Event Handlers）中监听客户查询服务状态的请求，ATM 服务响应该请求，并以系统当前状态回复客户。

假设用户给出的 ATM 服务的取款场景如图 1(B)所示：ATM 首先接收 logon 消息，紧接着完成一次取款操作，该操作通过 withdraw 请求和响应消息对表现，然后用户从 ATM 服务中登出（发送 logoff 请求），最后断开连接(发送 disconnect 消息)并结束操作。取款场景在 ATM 服务中可以出现，却不是一定会发生，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。

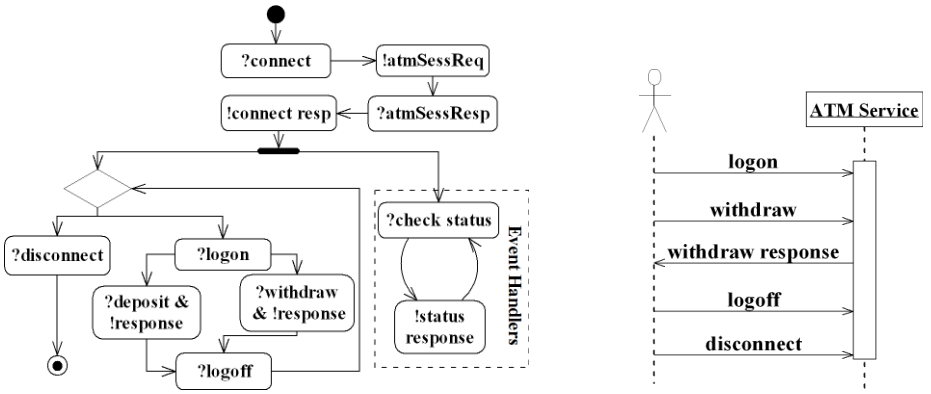
*ms*1=?connect→!atmSessRep→?atmSessResp→!connect resp→?logon→?withdraw→!withdraw response→?logoff→?disconnect;

*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，如需抽取取款场景，对于 *ms*2，由于转发 disconnect 消息后取款场景必不会出现了，所以不应转发该消息，而告知用户应发送的消息（如 logon）；如需过滤取款场景，对于 *ms*1，如果转发 disconnect 消息，取款场景就会完整的发生了，因此就不应转发该消息，而告知用户应发送的消息（如 logon）。此后，

1. 详见 http://seg.nju.edu.cn/BASIS010/atm_full.htm。

应继续监听、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* scenaior

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，本文研究基于场景规约的服务行为调控方法。第三方目标服务的行为规约通过 BPEL[15]描述，而用户的行为需求则用 UML 顺序图[8]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，则保证目标服务的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，则避免目标服务的执行时该场景的出现。

本文提出的基于场景规约服务行为调控框架的工作流程如图 2所示，整个流程分为三个部分：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了 BPEL-Petri 网模型（简称 BPN 模型）。接下来，在行为分析中遍历转换得到的 BPN 模型，以获取 BPN 模型中所有符合顺序图场景规约的行为。这些行为通过路径的概念描述，这些路径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，应对了基于线性化运行的遍历中状态空间爆炸的问题。接下来判断分析的结果是否满足调控的需求：行为分析找到了符合顺序图的行为，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。

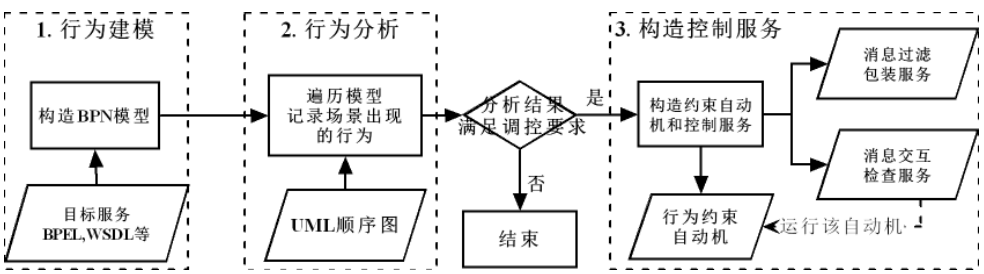


Fig.2 Process of web service behavior manipulation framework

图2 Web 服务行为调控框架流程

1.3 相关工作

BPEL 服务的行为分析吸引了很多的研究。其中为了自动化的分析 BPEL 服务，首先需要对 BPEL 建模[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[17]使用的高阶 Petri 网等。我们先前的工作[16]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 的建模以我们先前的工作为基础，并进行了扩展。和一致性验证相似，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，我们还对原有遍历方法[9,16]做了特别的优化，并应对状态空间爆炸的问题。

相比起服务行为的建模和一致性验证，web 服务的行为调控的是较新的研究课题，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。和这个工作比较起来，我们的工作关注于单个服务的行为的抽取和过滤，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。这样就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中止状态的完整行为。同时，约束模型必须非常谨慎的构造，考虑目标服务模型的状态和变迁。和该工作相比，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。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在运行时实现行为调控。

本文的主要贡献在于：

- 提出了一个基于场景规约的服务行为调控框架，包括对目标服务的建模、基于场景规约的行为分析以及调控服务的构造；
- 提出了基于并发变迁遍历 BPN 模型获取路径的算法，避免将不必要的并发线性化，从而减小了行为分析的状态空间；
- 开发了原型工具 BASIS 实现了服务行为调控的自动化。

本文的其余部分组织如下，第 2 部分介绍了服务需求和服务行为建模；在第 3 部分介绍了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 部分介绍调控服务的构造与实现，包括行为约束自动机的构造以及服务的运行时调控，我们的原型工具 BASIS 也在这一部分介绍；第 5 部分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的功能，并描述该用户需要的行为。本文选用 UML 顺序图作为用户需求模型，BPEL 作为服务行为的描述语言，并构造了服务的 BPEL 行为规约到 BPN 模型的转换。

2.1 服务需求建模：UML顺序图

*场景*可通过操作的序列描述系统的行为片段。作为最为常见的基于场景的规约，UML 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，用户需求行为片段通过 UML 顺序图描绘，图 1(B)中的取款场景就是一个 UML 顺序图。这里，我们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；$G:E\rightarrow O$ 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。

文献[8]中针对消息序列图（MSC）定义了消息踪迹的概念。由于顺序图与 MSC 非常接近，类似的我们可以定义顺序图 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 $trail_o$ 是称为顺序图 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 服务消息交互的接口通过 WSDL 描述。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[15]（WS-BPEL，简称 BPEL）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，这些活动包括：`receive`（接收服务外部消息）、`reply`（发送请求的响应）、`invoke`（发送消息或调用其它服务）、`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`（当其 `parallel` 属性为"no"时）表示；`flow` 中的各个活动并发的执行，而如果 `forEach` 的 `parallel` 属性为"yes"时，其所有子 `scope` 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、补偿和中止处理（FCT-Handler）以及事件处理（`eventHandlers`）在 `scope` 中定义。除了这些活动外，流程 `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。

描述 ATM 服务 BPEL 行为规约的参见：http://seg.nju.edu.cn/BASIS010/atm_full.htm。

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，我们构造了服务 BPEL 行为规约到 BPEL-Petri 网模型的转换。该模型与建模方法以我们先前工作[16]为基础，并进行了扩展。

BPEL-Petri 网模型的基础是 Petri 网。

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 $P \cap T = \varnothing$；$F \subset (P \times T) \cap (T \times P)$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 ${}^\bullet p=\{t \in T \mid (t,p) \in F\}$ 和 $p^\bullet=\{t \in T \mid (p,t) \in F\}$分别表示 p 的*前驱*和*后继*变迁。类似的，对于变迁 $t \in T$，${}^\bullet t=\{p \in P \mid (p,t) \in F\}$和 $t^\bullet=\{p \in P \mid (t,p) \in F\}$分别表示 t 的*前驱*和*后继*库所。如果${}^\bullet t \subseteq \mu$，则在标识$\mu$下变迁 t 是*使能*的；否者，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。

为了描述服务的 BPEL 行为规约，我们提出了 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*；$\mu_F \subset P$ 是服务网的*终止标识*。在 BPN 模型中，μ_0, μ_F 都是单元素集。

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。

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

在此基础上，一个有穷的运行是 N 上的一个*完整运行*当且仅当 N 的终止标识 μ_F 是该运行最后一个标识的子集；否则，该有穷运行是 N 上的一个*部分运行*。对于任何以下形式的运行 $r=\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} \mu_n \xrightarrow{t_n} \ldots$，其中 $n\geq 0$，如果不存在这样的标识 μ_i 和 μ_j $(0\leq i<j)$，使得 $\mu_i=\mu_j$，我们说 r 是一个*简单运行*；否则，称子序列 $\mu_i \xrightarrow{t_i} \mu_{i+1} \xrightarrow{t_{i+1}} \ldots \xrightarrow{t_{j-1}} \mu_j$ 是一个*循环*。

我们说变迁 t (或标识 μ 中库所)在 N 中是可达的，当且仅当存在运行 $r=\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots$ 使得对于 i $(i\geq 0)$，$t_i=t$（相应的，$\mu_i=\mu$）。

从 N 中的任何运行 $r=\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} \mu_n \xrightarrow{t_n} \ldots$，我们可以得到变迁序列 $t_0{}^\wedge t_1{}^\wedge \ldots {}^\wedge t_{n-1}{}^\wedge t_n{}^\wedge \ldots$。从这个变迁序列中移除关联空事件的变迁，我们可以得到新的变迁序列 $t_i{}^\wedge t_j{}^\wedge \ldots {}^\wedge t_k{}^\wedge \ldots$。我们说 r 的*事件序列*是 $trace(r)=\lambda(t_i)^\wedge\lambda(t_j)^\wedge\ldots^\wedge\lambda(t_k)^\wedge\ldots$。

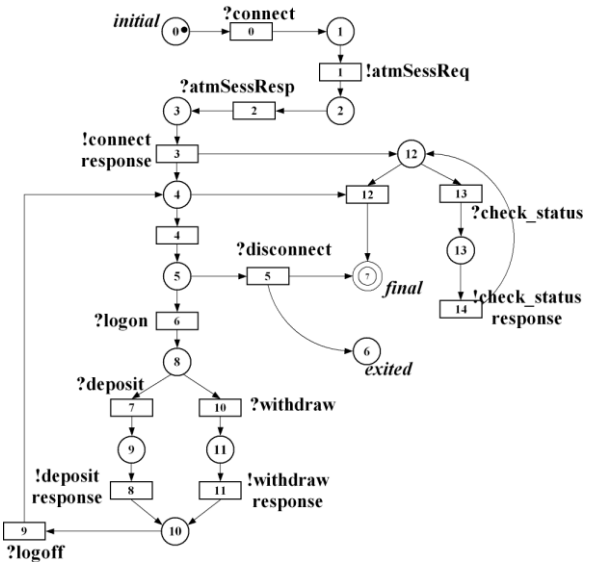


Fig. 3 Service net of the ATM Service

图3 ATM 服务的服务网模型

通过应用我们先前工作中 BPEL 流程建模的方法[16]，我们可以得到服务相应的 BPN 模型。

ATM 服务的 BPN 模型如图 3所示。图中，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。为了节省空间，我们用来标注这些事件的标签并没有包含完整消息收发信息（详细信息参见网页 http://seg.nju.edu.cn/BASIS010/atm_full.htm）。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。

3 服务行为分析

为了对目标服务的行为进行调控，抽取或过滤指定行为场景，就需检查服务 BPN 模型中是否有行为符合顺序图场景规约；如果可能则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，获取其中符合顺序图场景规约的行为。这里，我们提出了一个基于并发变迁遍历 BPN 模型获取路径的算法，以避免基于运行的遍历方法[9,16]中状态空间爆炸问题。在获取路径时，我们也记录了循环结构，同时避免循环带来无穷的行为。服务行为分析的结果，是一组路径的集合，以表示目标服务中符合顺序图场景规约的所有行为。这些路径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 网的模型可以自然的表现并发，运行的概念也可以方便的用来描述模型的行为。运行是由标识和变迁组成的序列，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，与用户需求场景事件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。而同时，线性化的并发行为是各个并发分支的笛卡尔积。这样，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，也就是状态空间爆炸。为此，针对行为分析的需求，我们借鉴了文献[5]提出的并发变迁和路径的概念，并做了适当扩展以适用于服务行为调控的问题。

路径是标识和并发变迁组成的序列。直觉上讲，所有本应在一个标识下分别触发的变迁在路径中由并发变迁一同触发。而由于行为分析的目的是找到 BPN 模型中，符合顺序图描绘的场景的行为，所以关联顺序图中事件的变迁仍应分别触发，即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 $m\geq 0$。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$i\,(1\le i\le m)$，$t_i \in enabled(\mu)$；
2) 对于任何$i, j\,(1\le i<j\le m)$，$\bullet t_i \cap \bullet t_j = \varnothing$；
3) 对于任何$i\,(1\le i\le m)$，若$\lambda(t_i)\in E$，则τ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$\mu'=\bigcup_{1\le i\le m}(\mu-\bullet t_i)\cup t_i\bullet$，并用$\mu'=fire(\mu,\tau)$表示。

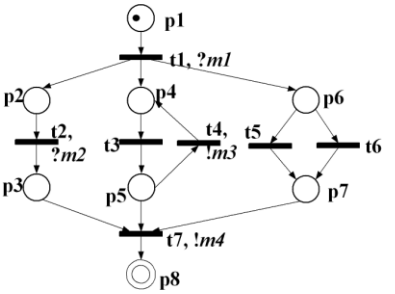


Fig. 4 Illustration of paths

图4 路径概念的简单示例

从上面的定义可以看出，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$m\ge 0$。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$\tau$中所有变迁关联的事件组成的集合，用$\lambda(\tau)=\{\lambda(t_1), \lambda(t_2), \ldots, \lambda(t_m)\}$表示。

基于并发变迁的概念，BPN 模型的行为也可以通过路径刻画。

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*$\sigma$是一个由标识和并发变迁组成的有穷或无穷的序列：$\sigma=\mu_0\xrightarrow{\tau_0}\mu_1\xrightarrow{\tau_1}\ldots$，对于任何$i\,(i>0)$，$\mu_i = fire(\mu_{i-1},\tau_{i-1})$。我们称$\sigma$的任何一个子序列为$\sigma$的一个*片段*。

运行可以看做是该路径的线性化执行，即路径中并发的变迁交错触发。例如，在图 4中的 BPN 中，如果$D=(O,E,M,G,V)$只包含事件 ?m1和 ?m2，那么如下的σ_1

$$\sigma_1=\{p1\}\xrightarrow{\{t1\}}\{p2,p4,p6\}\xrightarrow{\{t3,t5\}}\{p2,p5,p7\}\xrightarrow{\{t2\}}\{p3,p5,p7\}\xrightarrow{\{t7\}}\{p8\}$$

是一个该 BPN 的一个路径，而如下的r_1,r_2:

$$r_1=\{p1\}\xrightarrow{t1}\{p2,p4,p6\}\xrightarrow{t2}\{p3,p4,p6\}\xrightarrow{t3}\{p3,p5,p6\}\xrightarrow{t5}\{p3,p5,p7\}\xrightarrow{t7}\{p8\}$$

$$r_2=\{p1\}\xrightarrow{t1}\{p2,p4,p6\}\xrightarrow{t3}\{p3,p5,p6\}\xrightarrow{t2}\{p3,p5,p6\}\xrightarrow{t5}\{p3,p5,p7\}\xrightarrow{t7}\{p8\}$$

都是σ_1的线性执行，也是该 BPN 的运行。从这里可见，通过并发变迁和路径的概念，我们将并发中交错触发的变迁组织在一个并发变迁中同时触发，从而可以用较少的路径刻画 BPN 模型中的线性化运行。

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，路径可以看做是片段的特殊情况——当片段从 BPN 的初始标识开始时片段即是路径。

由于 BPN 模型中的循环（如图 3和图 4），循环中的行为往往不能丢弃，但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穷范围内，同时能够记录下 BPN 模型中必要的的循环结构，我们给出了简单和单次重复片段/路径的概念。

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$\rho=\mu_m\xrightarrow{\tau_m}\mu_{m+1}\xrightarrow{\tau_{m+1}}\ldots\xrightarrow{\tau_{n-1}}\mu_n\xrightarrow{\tau_n}\ldots$（0≤$m$≤n）是$N$上某个路径的片段。我们说$\rho$是*简单*的，如果$\rho$中没有重复执行，即对于任何$i$和$j$，$0\le i<j$，$t_i\in\tau_i$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$i\ge 0$，$\rho$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$\exists t_i\in\tau_i, t_j\in\tau_j.\bullet t_i\cap\bullet t_j\neq\varnothing$。若令$m=0$，上述的简单片段和单次重复片段即相对应的为*简单路径*和*单次重复路径*。

例如，前面给出的路径σ_1是简单路径（因此也是单次重复路径），而如下的

$$\sigma_2=\{p1\}\xrightarrow{\{t1\}}\{p2,p4,p6\}\xrightarrow{\{t3,t5\}}\{p2,p5,p7\}\xrightarrow{\{t4\}}\{p2,p4,p7\}\xrightarrow{\{t3\}}\{p2,p5,p7\}\xrightarrow{\{t2\}}\{p3,p5,p7\}\xrightarrow{\{t7\}}\{p8\}$$

则是单次重复路径而不是简单路径。图 4中由 $p4$, $t3$, $p5$ 和 $t4$ 构成的循环结构在σ_2 中通过片段 $\{p2, p4, p6\}^{\{t3,t5\}}\rightarrow \{p2, p5, p7\}^{\{t4\}}\rightarrow \{p2, p4, p7\}^{\{t3\}}\rightarrow\{p2, p5, p7\}$ 记录。

由于 BPN 中的库所和变迁的集合都是有穷的，所以单次重复和简单路径的数量也是有穷的。此外，通过单次重复片段还可以使我们记录下循环结构，而这在简单路径中是无法做到的。

3.2 针对场景的行为分析

本节介绍了针对场景的行为分析的具体方法。我们给出顺序图 D 描绘的场景在路径σ中出现的定义，以及一个算法，遍历 BPN 模型 N 的状态空间，获取符合顺序图 D 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 上任何某路径 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 ε 和空集，即可得到非空的事件集合的序列 $\zeta=\eta_0\wedge\eta_1\wedge\ldots\wedge\eta_m$ $(m<j-i)$。如果对于任意 k $(0\le k\le m)$，ζ 中 η_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 ξ 是顺序图 D 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*。而且，如果 $\eta_0=\lambda(\tau_i)$ 且 $\eta_m=\lambda(\tau_j)$，我们说 ρ 是 D 的*真像*。

基于前面给出的定义，我们称顺序图 D 在 N 中出现的路径为Δ-路径。Δ-路径需能够包含带有顺序图 D 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

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

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*。

从上面可见，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（定义 8 的条件 1)）。每条Δ-路径都仅存在一个真像片段对应顺序图 D 描绘的行为，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顺序图的出现。从直觉上说，定义 8 的条件 3) 是为了减少重复记录刻画同样行为的路径，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。我们用Δ(N,D)表示 N 中所有的Δ-路径。Δ-路径的集合Δ(N,D)通过算法 1 计算。

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：Δ(N,D), *failcnt*

令 $curpath=\mu_0$, Δ$(N,D)=\varnothing$, *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可到达的新的后继节点 **then {**
令 *node* 为通过触发某个并发变迁 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Δ(*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 Δ(*N*,*D*)和 *failcnt*

行为分析算法如算法 1 所示。算法 1 在计算Δ(*N*,*D*)的同时还将遍历发现的以包含μ_F结束的路径数统计在失败路径计数 *failcnt* 中。算法以触发并发变迁的方法从初始标识 μ_0 以深度优先的顺序遍历 *N* 的状态空间。已遍历的状态空间中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：从 *curpath* 中删去最后的节点和前面的并发变迁。而在回溯之前，如果 *curpath*为Δ-路径时，则将 *curpath* 加入到Δ(*N*,*D*)；而如果 *node* 包含终止标识μ_F则将 *failcnt* 增加 1。上面的算法中，对 BPN 模型的遍历是通过触发并发变迁进行的，这样就减小了并发带来的复杂性；且利用了简单和单次重复片段的概念，将循环带来的无穷行为的归结为有穷路径集合。这个算法的复杂度是和Δ(*N*,*D*)中Δ-路径的前缀个数以及Δ-路径的最长的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。算法的正确性通过定理 1 说明。

定理 1(行为分析算法正确性). 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。算法 1 返回的Δ(*N*,*D*)是包含 *N* 中所有Δ-路径的最小集合。

证明: 证明定理 1 即证明以下三点:(1)算法 1 能够结束;(2) 所有Δ-路径均在算法 1 结束后返回的 $\Delta(N,D)$ 中；(3) 算法 1 结束后返回的 $\Delta(N,D)$ 中的所有路径均是Δ-路径。

(1) 首先，由定义 8 可知Δ-路径的长度有穷。因为 BPN 中库所和变迁的集合是有穷集合，从给定标识开始的单次重复片段集合是有穷集，故而前像片段的集合有穷。而由于 *D* 中事件集合有穷，所以 *D* 的真像是有穷集。因此，后像片段集合的数量有穷。所以路径集合 $\Delta(N,D)$ 必是有穷集合且 $\Delta(N,D)$ 中路径长度有穷。

而同时，存在某次迭代，使得 *curpath* $=\langle\mu_0\rangle$ 且从 μ_0 没有通过触发某并发变迁能到达的新后继节点，则 μ_0 将从 *curpath*中删除，从而使得 *curpath* $=\langle\rangle$，迭代结束。因此，算法必定可以结束。

(2) 令算法 1 返回的集合为 Δ_1。假设存在一条Δ-路径 σ，但是 σ 不在集合 Δ_1 中。因为所有的路径都从 μ_0 开始且算法 1 从 μ_0 开始遍历，所以存在 σ 的某个前缀ρ_1，使得 ρ_1是 Δ_1 中某个路径的前缀。又因为 $\sigma \notin \Delta_1$，则存在 σ 的前缀 $\rho_2 = \rho_1 \xrightarrow{\tau} \mu$，使得 ρ_2 不是 Δ_1 中任意路径的前缀片段。因为 μ 是通过触发并发变迁 τ 所到达的新的后继结点，那么ρ_2 必不是任意Δ-路径的前缀片段（否则 $\xrightarrow{\tau}\mu$ 应当加入 *curpath* 进而加入 Δ_1 中）。然而ρ_2是Δ-路径 σ 的前缀片段，矛盾。故对于任何Δ-路径 σ， σ 在算法 1 返回的集合 Δ_1 中。

(3) 由于仅当 *node* 满足 *curpath*对应的路径是Δ-路径，才将 *curpath* 对应的路径加入 $\Delta(N,D)$，因此算法 1 结束后返回的 $\Delta(N,D)$ 中的所有路径均是Δ-路径。

因此，定理 1 得证。

□

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 服务在取款场景下的行为分析结果Δ(*N*,*D*)的其中 3 条Δ-路径如下所示（完整列表参见 http://seg.nju.edu.cn/BASIS010/atm_full.htm）:

$$\sigma 1=\{p0\}\xrightarrow{\{t0\}}\{p1\}\xrightarrow{\{t1\}}\{p2\}\xrightarrow{\{t2\}}\{p3\}\xrightarrow{\{t3\}}\{p4,p12\}\xrightarrow{\{t4,t13\}}\{p5,p13\}\xrightarrow{\{t14\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t6}\}}\{\mathbf{p8},\mathbf{p12}\}\xrightarrow{\{\mathbf{t10}\}}\{\mathbf{p11},\mathbf{p12}\}\xrightarrow{\{\mathbf{t11}\}}\{\mathbf{p10},\mathbf{p12}\}$$
$$\xrightarrow{\{\mathbf{t9}\}}\{\mathbf{p4},\mathbf{p12}\}\xrightarrow{\{\mathbf{t4}\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t5}\}}\{\mathbf{p6},\mathbf{p7},\mathbf{p12}\}$$

$$\sigma 2=\{p0\}\xrightarrow{\{t0\}}\{p1\}\xrightarrow{\{t1\}}\{p2\}\xrightarrow{\{t2\}}\{p3\}\xrightarrow{\{t3\}}\{p4,p12\}\xrightarrow{\{t4,t13\}}\{p5,p13\}\xrightarrow{\{t14\}}\{p5,p12\}\xrightarrow{\{t6\}}\{p8,p12\}\xrightarrow{\{t10\}}\{p11,p12\}\xrightarrow{\{t11\}}\{p10,p12\}$$
$$\xrightarrow{\{t9\}}\{p4,p12\}\xrightarrow{\{t4\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t6}\}}\{\mathbf{p8},\mathbf{p12}\}\xrightarrow{\{\mathbf{t10}\}}\{\mathbf{p11},\mathbf{p12}\}\xrightarrow{\{\mathbf{t11}\}}\{\mathbf{p10},\mathbf{p12}\}\xrightarrow{\{\mathbf{t9}\}}\{\mathbf{p4},\mathbf{p12}\}\xrightarrow{\{\mathbf{t4}\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t5}\}}\{\mathbf{p6},\mathbf{p7},\mathbf{p12}\}$$

$$\sigma 3=\{p0\}\xrightarrow{\{t0\}}\{p1\}\xrightarrow{\{t1\}}\{p2\}\xrightarrow{\{t2\}}\{p3\}\xrightarrow{\{t3\}}\{p4,p12\}\xrightarrow{\{t4,t13\}}\{p5,p13\}\xrightarrow{\{t14\}}\{p5,p12\}\xrightarrow{\{t6\}}\{p8,p12\}\xrightarrow{\{t7\}}\{p9,p12\}\xrightarrow{\{t8\}}\{p10,p12\}$$
$$\xrightarrow{\{t9\}}\{p4,p12\}\xrightarrow{\{t4\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t6}\}}\{\mathbf{p8},\mathbf{p12}\}\xrightarrow{\{\mathbf{t10}\}}\{\mathbf{p11},\mathbf{p12}\}\xrightarrow{\{\mathbf{t11}\}}\{\mathbf{p10},\mathbf{p12}\}\xrightarrow{\{\mathbf{t9}\}}\{\mathbf{p4},\mathbf{p12}\}\xrightarrow{\{\mathbf{t4}\}}\{\mathbf{p5},\mathbf{p12}\}\xrightarrow{\{\mathbf{t5}\}}\{\mathbf{p6},\mathbf{p7},\mathbf{p12}\}$$

… …

这些路径中的库所用 *p* 加上图 3中的数字标注表示，类似的变迁用 *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。这个例子中，Δ(*N*,*D*)中的路径都没有后像片段。因为取款场景在 ATM 服务中需要用循环来表现，所以路径的真像片段都不是简单片段。这些路径最后一个标识都包含了 *N* 的终止标识中的库所 *p*7。

通过行为分析，目标服务中带有顺序图出现的所有行为通过路径集合Δ(*N*,*D*)记录下来。Δ(*N*,*D*)中的各Δ-路径都存在一个片段对应的事件序列是 *D* 的消息踪迹。BPN 模型中的循环结构在Δ-路径中通过单次重复的片段记录；而由于Δ-路径是标识和并发变迁组成的序列，通过并发变迁将可能并发的变迁组织在一起，避免了对所有并发行为的线性化（除非需求的场景行为表现为并发行为的线性化），从而应对了并发带来的状态空间爆炸以及循环导致的无穷行为的问题。路径集合Δ(*N*,*D*)将作为下面构造调控服务的基础。

4 构造调控服务

服务的行为分析通过遍历目标服务的 BPN 模型，获取了一组Δ-路径以表示顺序图描绘场景出现的行为，其中行为中的循环通过单次重复片段等描述。而如果行为分析的结果给出的Δ(*N*,*D*)为空，说明用户指定顺序图场景在 *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用户给定顺序图场景的像，如果用户需要过滤该行为，则不能满足。因此，如果Δ(*N*,*D*)不为空，且当用户需求是行为过滤式 *failcnt* 不为 0，我们就可以利用Δ(*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，即根据顺序图描绘场景出现的所有行为，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顺序图描绘的场景出现的行为，但是却不易直接用于调控目标服务的消息交互。因此，本章中，我们需要根据路径集合Δ(*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Δ(*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-路径所表示的带有顺序图描绘场景出现的服务行为。接下来，将这些事件自动机整合并化简，得到的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。

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。接下来为每条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 中顺序图 D 描绘场景出现之前、之中和之后的状态。

设 $s_1,s_2\in S$ 且 $e\in\alpha$。如果 $s_2\in\delta(s_1,e)$，我们也用 $s_1\xrightarrow{e}s_2$ 来表示。可以看出，事件自动机 U 能接受的语言是

消息收发事件组成的序列。由于我们希望 U 所接受的事件序列中都包含了顺序图 D 描绘场景的发生，我们用 S 的子集 S_{pre}, S_{im}, S_{post} 分别表示顺序图 D 描绘的场景还未发生时应在的状态、D 描绘的场景部分出现的状态以及 D 描绘的场景已经出现后的状态。

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 刻画的包含顺序图 D 描绘的场景出现的 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 $\forall\, s_F{}^\sigma \in S_F{}^\sigma . \mu_F \subseteq L^\sigma(s_F{}^\sigma)$。$U^\sigma$ 的构造以及 L^σ 的记录由算法 2 描述。其中，令 ρ 是路径 $\sigma\in\Delta(N,D)$ 的片段，$\mu\subseteq P(\rho)$ 是一个标识，$e\in\lambda(\rho)$ 是该路径上的某个事件（e 可为 ε）。我们用 $mov(\rho,\mu,e)$ 表示在片段 ρ 中，标识 μ 通过触发某个关联事件 e 的变迁而可达的新的标识，这些标识中的库所是 μ 使能的变迁的后继库所，且这些变迁关联事件 e；也就是说 $mov(\rho,\mu,e) = \{\mu' \mid \exists\, t_e \in T_e . \mu' = (\mu - {}^\bullet t_e) \cup t_e{}^\bullet\}$，其中 $T_e = \{t \in T(\rho) \mid \lambda(t) = e \wedge {}^\bullet t \subseteq \mu\}$。

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(σ, U^σ, L^σ, false);

foreach $s \in S^\sigma$ 满足 $t_{im0} \in enabled(L^\sigma(s))$，其中 t_{im0} 是 ρ_{im} 的第一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);

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

addStatesNonIm(σ, U^σ, L^σ, 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} 中从第一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^σ, $\delta^\sigma(s,e)$ 和 S_{im}^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*(σ, U^σ, L^σ, *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 $S' = \{s' \in S^\sigma \mid P_e = L^\sigma(s')\}$;

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

向 S^σ、$\delta^\sigma(s,e)$ 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 $\delta^\sigma(s_{new}, \varepsilon) = \{s_{pp} \in S_{pre} \mid L^\sigma(s_{pp}) = L^\sigma(s_{new})\}$ **}}**

else { 令 $\delta^\sigma(s,e) = \delta^\sigma(s,e) \cup S'$，并将 S' 中所有状态置为未标记 **}}}**

算法 2 从初始标识（同时也是 σ 的第一个标识）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，而新的状态的加入都从未标记的状态开始，在从某个未标记状态计算后继时标记该状态。算法流程分别考虑 σ 的三个片段是因为加入新状态和边的策略有所区别。直觉上说，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 $L^{\sigma}(s_{pp}) = L^{\sigma}(s_{new})$，构成循环。$U^{\sigma}$的终止状态都是处理后像片段时加入的状态 $S_{post}{}^{\sigma}$，这样在自动机接受消息交互事件序列，执行到终止状态时，用户需求的顺序图场景一定已经发生。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。

ATM 服务在取款场景下$\Delta(N,D)$中的$\sigma 1$ 的事件自动机 $U^{\sigma 1}$ 如图 5所示。事件自动机的状态用椭圆表示，状态之间的边用箭头表示；标签标记在状态内部，事件则标记在边上。为了节省空间，事件上的消息采用了简短的标记（详见附录 2）。真像对应的状态标签和事件在图中用加粗的斜体字表示。

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表示的顺序图 *D* 描绘的场景出现的 *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场景的出现的前提下化简之后，就得到了最终的服务行为约束自动机，其接受的语言是 BPN 模型中所有包含顺序图出现的事件序列，并仅接受这样的事件序列。

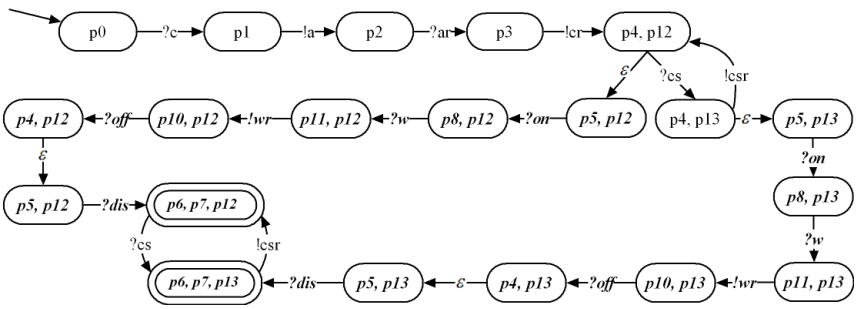


Fig. 5 The corresponding event automaton of Δ-path $\sigma 1$ of ATM Service under withdrawal behavior
图5 ATM 服务取款场景下Δ-路径$\sigma 1$ 对应的事件自动机

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 模型中带有顺序图 *D* 出现的行为的事件自动机 $U = (\alpha, S, \delta, s_0, S_F, S_{pre}, S_{im}, S_{post})$。其中，$L: S \to 2^{P}$ 用作 *U* 上状态的标签函数。

1. 将$\sigma \in \Delta(N,D)$对应自动机 U^{σ}中各元素依次加入 *U* 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 $s_0{}^{\sigma}$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 上的标签为空集，*S* 中其它状态的标签和 L^{σ}一样：即 $L = \{s_0 \mapsto \varnothing\} \cup \bigcup_{\sigma\in\Delta(N,D)} L^{\sigma}$
3. 分别合并 S_{pre} 和 S_{post} 中所有带有相同标签（通过 *L* 表示）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*。
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，同时相应的更新标签函数 *L*。

这样，通过上面的步骤，将按照算法 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 *U* 称为 BPN 模型 *N* 在顺序图 *D* 下的*非确定行为约束自动机*。

定理 2(非确定行为约束自动机 *U* 的正确性). $U = (\alpha, S, \delta, s_0, S_F, S_{pre}, S_{im}, S_{post})$是 BPN 模型 $N = (P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$在顺序图 $D=(O,E,M,G,V)$ 下的非确定行为约束自动机，即 *U* 接受 *N* 中所有带有 *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。

证明: 我们首先证明所有能被 U 接受的事件序列都是 N 的某个出现了 D 描绘的场景的运行的事件序列（“充分性”），接下来再证明 N 的所有出现了 D 描绘场景的运行的事件序列都可被 U 接受（“必要性”）。

“充分性” – 由路径和并发变迁的定义，若有片段 $\mu \xrightarrow{\tau} \mu'$，则 $\forall t \in \tau.t \in enabled(\mu)$。因为真像片段对应的事件序列是 D 的消息踪迹，又由 *mov* 的定义以及算法 2 中处理真像片段时加入新状态和边的规则，$\exists s_{im0} \in S_{im}.\delta^{\sigma}(s',*) = s_{im0} \Rightarrow s' \notin S_{im}$ 使得在 U 中 s_{im0} 可以依次经过其后的边处理 D 的消息踪迹上的所有事件。并且根据处理前像和后像片段时的算法，除了 s_{im0} 外，S_{im} 中的所有状态仅有来自 S_{im} 中其它状态的边。而因为 S_F^{σ} 是 $S_{post}{}^{\sigma}$的子集，所以 S_F 也是 S_{post} 的子集，又根据引理 1 所有状态从初始状态可达且所有状态皆可到达某终止状态，所以对于任何 U 接受的事件序列都有一段子序列，去除其中空事件之后就是 D 的消息踪迹。

这样，我们只用证明对于任何可被 U 接受的事件序列 *es* 都对应一个运行，即存在一个 N 的运行 r 满足 $trace(r) = es$。令 es_k ($k \le |es|$)是 *es* 的长度为 k 的前缀子序列。我们首先证明对于 *es* 的任何长度为 l ($0 < l \le |es|$)的前缀子序列 es_l，都有 N 的一个（部分）运行 r_l 满足 $trace(r_l) = es_l$。如果这个可证，因为对于任何终止状态 $s_F \in S_F$ 有 $\mu_F \subseteq L(s_F)$，这样和 *es* 对应的运行必然也是一个完全运行(full run)。首先令 $k=1$，不失一般性我们假设 es_1 对应 $s_0 \xrightarrow{\varepsilon} s_1 \xrightarrow{e_1} s_2$。根据算法 2，必存在 σ 的一个片段 ρ 使得 $L(s_1) = \mu_0$ 且 $L(s_2) = mov(\rho, \mu_0, e)$。故根据 *mov* 的定义，我们可得出必存在一个（部分）运行 r_1，这个运行从 $\mu_0 = L(s_1)$ 通过触发变迁 t (其中 $\lambda(t) = e$)到达 $L(s_2)$ 对应的标识，即 $trace(r_1) = e = es_1$。假设 es_k ($0 < k < |es|$)存在对应的（部分）运行 r_k 满足 $trace(r_k) = es_k$，设 $r_k = \mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} \mu_n$。接下来证明对于长度为 $k+1$ 的前缀序列，也存在对应运行 $trace(r_{k+1}) = es_{k+1}$。令 es_{k+1} 对应 $s_0 \xrightarrow{\varepsilon} s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \ldots \xrightarrow{e_k} s_k \xrightarrow{e_{k+1}} s_{k+1}$，则有 $s_0 \xrightarrow{\varepsilon} s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \ldots \xrightarrow{e_k} s_k$ 对应 es_k。因为 $L(s_{k+1}) \in mov(\rho, L(s_k), e_{k+1})$，根据路径和 *mov* 的定义，我们可以得到一定有一条运行片段从 $L(s_k)$ 中标识通过触发变迁 t (其中 $\lambda(t) = e_{k+1}$)到 $L(s_{k+1})$ 的标识。因此，我们证明了对于每个 *es*，都有 N 的一(部分)运行其事件序列和 *es* 相同。这样证明了“充分性”即所有能被 U 接受的事件序列都是 N 的某个出现了 D 描绘的场景的运行的事件序列。

“必要性” – 令 $\mu \xrightarrow{t} \mu'$ 是 N 的有一个 D 描绘场景出现的运行片段，那么对于任何 $p \in \mu - \mu'$ 以及 $p' \in \mu' - \mu$，有 $(p,t),(t,p') \in F$，而根据路径和并发变迁的定义，对于路径 $\sigma \in \Delta$ 的任何片段 $\mu_\sigma \xrightarrow{\tau} \mu_\sigma'$，$t \in \tau$ 当且仅当 $p \in \mu_\sigma$ 且 $p' \in \mu_\sigma'$。因此对于 U 的任何满足 $\mu \subseteq L(s)$ 的状态 s，如果 $\lambda(t) = \varepsilon$，根据 *mov* 的定义，必有状态 s'（ s' 可能与 s 相同）满足 $s' \in \delta^{\sigma}(s, \lambda(t))$ 并且 $\mu' \subseteq L(s')$。这个结论将在下面的证明中使用到。

下面我们用归纳法证明对于所有前缀子运行的所有出现了 D 描绘的场景的运行，对应的事件序列是可被 U 接受的事件序列的前缀。作为归纳基础，设长度为 1 的部分运行 $\mu_0 \xrightarrow{t_0} \mu_1$ 是某出现了 D 描绘场景的完整运行的前缀子运行。因为必定有状态 s 满足 $\mu_0 \subseteq L(s)$，所以如果 $\lambda(t_0) = \varepsilon$，对应空事件序列自然是某可被 U 接受的事件序列的子序列；否则，对应的事件序列是 $\langle \lambda(t_0) \rangle$ 也是能够被 U 接受的事件序列的前缀子序列。设长度为 k 部分运行 $r_k = \mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} \mu_k$ 是某出现了 D 描绘场景的完整运行的前缀子运行，假设其事件序列 $trace(r_k)$ 是能够被 U 接受的事件序列的前缀子序列。这样，也用归纳和上一段的结论可以证明由于 $\exists s \in S.\mu_0 \subseteq L(s)$，所以对于任何 k 有 $\exists s' \in S.\mu_k \subseteq L(s')$。因此，令长度为 $k+1$ 的部分运行为 $r_{k+1} = \mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} \mu_k \xrightarrow{t_k} \mu_{k+1}$，我们有 $\exists s, s' \in S\ .\ \mu_k \subseteq L(s) \wedge \mu_{k+1} \subseteq L(s') \wedge s' \in \delta^{\sigma}(s, \lambda(t_k))$。若 $\lambda(t_k) = \varepsilon$，则 $trace(r_{k+1}) = trace(r_k)$，由假设知 $trace(r_{k+1})$ 是能够被 U 接受的事件序列的前缀子序列；否则，因为存在标识 μ_{k+1} 和事件 $\lambda(t_k)$ 相应的状态和边，且由归纳假设 r_k 对应的事件序列可被 U 接受，故 $trace(r_{k+1})$ 也是一个能够被 U 接受的事件序列的前缀子序列。而由于完整运行的最后的标识包含终止标识，根据 $\Delta(N, D)$ 的定义，终止标

识是路径 $\sigma \in \Delta(N,D)$ 的最终标识的子集，所以 $\exists s_F \in S_F.\mu_{last} \subseteq L(s_F)$，其中 μ_{last} 是带有 D 描绘场景出现的完整运行的最后一个标识。也就是说，所有这样的运行对应的事件序列都可被 U 接受。

□

引理 1(U^σ 的性质). 令 $N=(P,T,F,\Lambda,\lambda,\mu_0,\mu_F)$ 是 BPN 模型，$D=(O,E,M,G,V)$ 是顺序图，U^σ=(α^σ, S^σ, δ^σ, $s_0{}^\sigma$, $S_F{}^\sigma$, $S_{pre}{}^\sigma$, $S_{im}{}^\sigma$, $S_{post}{}^\sigma$)为 $\Delta(N,D)$ 中路径 σ 的事件自动机。U^σ 有如下性质：(1) S^σ 中所有状态都从初始状态 s_0^σ 可达；并且(2)从 S^σ 中的每个状态都可到达 S_F^σ 某个终止状态。

证明：由于我们从 s_0^σ 构造 S^σ 中的当前状态的下一个状态，所以 S^σ 中所有状态都从初始状态 s_0^σ 可达。

下面我们证明性质(2)。首先我们证明 S_F^σ 不为空。对于 $\Delta(N,D)$ 中的每条路径 σ， μ_F 是路径最后的标识的子集，所以，一定有从 μ_0 中库所到 μ_F 中库所的 N 中流(flow)的闭包关系。因此必存在一个状态 s_F 满足 $\mu_F \subseteq L(s_F)$，即 $S_F^\sigma \neq \varnothing$。

下面用反证法证明该性质。假设存在状态子集 $S_{IF} \subset S^\sigma - S_F^\sigma$ 使得 S_{IF} 中的任一状态只有到 S_{IF} 中状态的边（因此 S_{IF} 中的状态不能到达某一终止状态），而 $S^\sigma - S_{IF}$ 中的状态或为终止状态或有一条边的下一个状态不在 S_{IF} 中，则 S_{IF} 中的状态均不可到达 S_F^σ 中的状态。下面证明不存在这样的 S_{IF}。因为性质(1)保证了所有的状态都从 s_0^σ 可达，且已证明 $S_F^\sigma \neq \varnothing$，故一定存在状态 $s_B \in S^\sigma$ 使得从 s_B 可达某终止状态并从 s_B 有一条到达 S_{IF} 中状态的边。令 $s_{Bn} \in S^\sigma - S_{IF}$，且从 s_B 有一条边到 s_{Bn}。这样，从 s_{Bn} 必可达某终止状态。此外，从 s_B 有边到达 S_{IF} 中的的某个状态 s_{BI}。可知 $(L(s_{Bn})-L(s_B))\cap(L(s_{BI})-L(s_B))=\varnothing$。

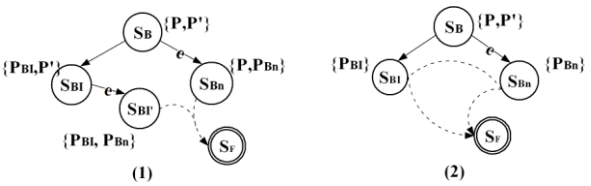


Fig. 11 Two cases in proof of Lemma 1

图6 引理 1 证明中两种情况的简单示例

接下来的证明分两种情况进行，首先证明如果 $(L(s_B)-L(s_{Bn}))\cap(L(s_B)-L(s_{BI}))=\varnothing$时 S_{IF} 不存在。根据 *mov* 的定义，对于任何事件 e 使得 $s_{Bn} \in \delta^\sigma(s_B,e)$，一定存在变迁 t 使得 $\lambda(t)=e$ 并且 $\exists p \in {}^\bullet t.\ p \in L(s_B)-L(s_{Bn})$。由于 $(L(s_{Bn})-L(s_B)) \cap (L(s_{BI})-L(s_B)) = \varnothing$，这样的库所 p 一定也在 $L(s_{BI})-L(s_B)$ 中。因此 $\exists s_{BI}' \in \delta^\sigma(s_{BI},e).\ L(s_{Bn})\cap L(s') \neq \varnothing$。根据定义 8，在 σ 中不会有冲突的变迁（即共享某个前驱库所的变迁），除非在这两个冲突变迁之间存在循环。因为 S_F^σ 中的状态从 s_{Bn} 可达，令 s_{Bn}' 是从 s_{Bn} 到 S_F^σ 中的状态的某条路径的下一个状态。那么 s_{Bn}' 从 s_{BI} 可达。则从 s_{BI} 可以沿着某事件序列到达 S_F^σ 中的状态。这与 S_{IF} 的条件矛盾。这个情况的示例如图 7(1)所示。

第二种情况，若 $(L(s_B)-L(s_{Bn})) \cap (L(s_B)-L(s_{BI})) \neq \varnothing$，则$\exists p \in L(s_B),\ p_{Bn} \in L(s_{Bn}),\ p_{BI} \in L(s_{BI}),\ t_{Bn}, t_{BI} \in p^\bullet.\ (t_{Bn},p_{Bn}),(t_{BI},p_{BI}) \in F$。根据并发变迁的定义，如果 $t_{Bn} \in \tau_{Bn}$且 $t_{BI} \in \tau_{BI}$ 那么 $\tau_{Bn} \neq \tau_{BI}$。不失一般性，假设 $\{t_{Bn}, t_{BI}\} = p^\bullet$。令 $\mu_B, \mu_{Bn}, \mu_{BI}$ 分别为 σ 中包含库所 p, p_{Bn}, p_{BI} 的标识。如果 τ_{BI} 在 τ_{Bn} 之前，那么就有 σ 的片段 $\mu_{BI} \xrightarrow{\tau_{BI}} \ldots \xrightarrow{\tau_k} \mu_B \xrightarrow{\tau_{Bn}} \mu_{Bn} \ldots$，而因为 $p \in L(s_B)$, $p_{Bn} \in L(s_{Bn})$ 且 $p_{BI} \in L(s_{BI})$，因此从 s_{BI} 可达某终止状态，这个与假设矛盾。如果 τ_{Bn} 在 τ_{BI} 之前，那么 $\mu_{Bn} \xrightarrow{\tau_{Bn}} \ldots \xrightarrow{\tau_k} \mu_B \xrightarrow{\tau_{BI}} \mu_{BI} \ldots$就是 σ 的片段，而因为 μ_F 一定也包含在 σ 的最后标识中，根据路径的定义，如果 $\mu_F \subseteq \mu_{BI}$，则由状态上标记库所的方法，有 $s_{BI} \in S_F$，这与假设矛盾；否则令 $\mu_B \xrightarrow{\tau_{BI}} \mu_{BI} \xrightarrow{\tau_m} \ldots \xrightarrow{\tau_{n-1}} \mu_n$ 是 σ 的后缀片段，根据构造 U^σ的算法，$\exists s \in S.\mu_n \subseteq L(s)$，所以 $s \in S_F$，而又因为 $p_{BI} \in L(s_{BI})$ 那么 s 从 s_{BI} 可达，这与假设矛盾。因此，无论 τ_{BI} 和 τ_{Bn} 在σ中以何种顺序出现，得到的结论都与假设矛盾。

这个情况的示例如图 7(2)所示。

这样我们证明了存在 S_{IF} 的假设不成立，所以 S_{IF} 为空集。此外，我们还证明了 $S_F^{\sigma} \neq \varnothing$。因此，可以说明，从 S^{σ} 中的任何一个状态，都可到达 S^{σ} 中的某个终止状态。

□

尽管按照上面方式构造的自动机 U 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），且这样的自动机是非确定的。因此，我们给出了对自动机 U 的化简，以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。在经过化简之后，我们得到的事件自动机 X，将作为最终的行为约束自动机。

为了防止化简影响自动机中与顺序图描绘场景相关的行为，我们首先对非确定行为约束自动机 $U = (\alpha, S, \delta, s_0, S_F, S_{pre}, S_{im}, S_{post})$上的顺序图描绘场景相关的事件进行改名。令 $imconnect \subseteq \delta$ 是 S_{im} 中状态之间事件非空的边，即 $imconnect = \{(s,e,s') \in \delta \mid s,s' \in S_{im} \wedge e \neq \varepsilon\}$。接下来，我们定义改名函数 $RN : imconnect \to \overline{\alpha}$，其中 $\overline{\alpha}$ 是一个事件名的集合 $\overline{\alpha} \cap \alpha = \varnothing$，并且对于 $imconnect$ 中的各个事件，在改名前后存在 1-1 映射关系，也就是说：$\forall (s,e,s') \in imconnect, \exists \overline{e} \in \overline{\alpha} \,.\, RN(s,e,s') = \overline{e}$，且对于任何$(s_1,e_1,s_1'),(s_2,e_2,s_2') \in imconnect$，若 $e_1 = e_2$ 则 $RN(s_1,e_1,s_1') = RN(s_2,e_2,s_2')$，否则 $RN(s_1,e_1,s_1') \neq RN(s_2,e_2,s_2')$。

设改名之后的自动机为 $U^1 = (\alpha^1, S, \delta^1, s_0, S_F, S_{pre}, S_{im}, S_{post})$，其中$\alpha^1$ 和δ^1 是根据 RN 改名之后的事件集合和带事件的边，即$\alpha^1 = \alpha \cup \text{ran } RN$，而$\delta^1 = \{(s, \overline{e}, t) \in S \times \overline{\alpha} \times S \mid \exists e \in \alpha . (s,e,t) \in \delta \wedge RN(s,e,t) = \overline{e}\}$。接下来，按照文献[1]中非确定有穷状态自动机确定化的方法，得到 U^1 确定化后的事件自动机 $X^1 = (\alpha^{X1}, S^X, \delta^{X1}, s_0^X, S_F^X, S_{pre}^X, S_{im}^X, S_{post}^X)$，其中令 $S_{im}^X = \{s \in S^X \mid \exists s' \in S^X,\ \overline{e} \in \overline{\alpha} . (s', \overline{e}, s) \in \delta^{X1}\}$。最后，由于 $imconnect$ 中事件改名前后存在 1-1 映射关系，我们将 X^1 中被改名事件改回原来的事件名，并更新自动机中的边。这样得到的事件自动机 $X=(\alpha^X, S^X, \delta^X, s_0^X, S_F^X, S_{pre}^X, S_{im}^X, S_{post}^X)$即 N 对应的服务在顺序图 D 下的*行为约束自动机*，其中$\alpha^X = \alpha - \{\varepsilon\}$，而 $\delta^X = \{(s,e,t) \in S^X \times \alpha^X \times S^X \mid \exists s_U, t_U \in S, \overline{e} \in \overline{\alpha} . (s_U, \overline{e}, t_U) \in \delta^{X1} \wedge RN(s_U,e,t_U) = \overline{e}\}$。

定理 3 行为约束自动机的正确性). $X=(\alpha^X, S^X, \delta^X, s_0^X, S_F^X, S_{pre}^X, S_{im}^X, S_{post}^X)$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 X 接受 N 中所有带有 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。

证明: 因为确定化得到的确定有穷状态自动机（DFA）接受与原先的非确定有穷状态自动机（NFA）完全相同的语言[1]，而在转化过程中仅涉及改名，且改名函数（RN）可保证改名前后的事件直接存在 1-1 映射，因此 X 接受和 U 相同的事件序列。由于定理 2 的保证，则 X 接受所有且仅接受 N 的带有 D 描绘场景出现的事件序列。

□

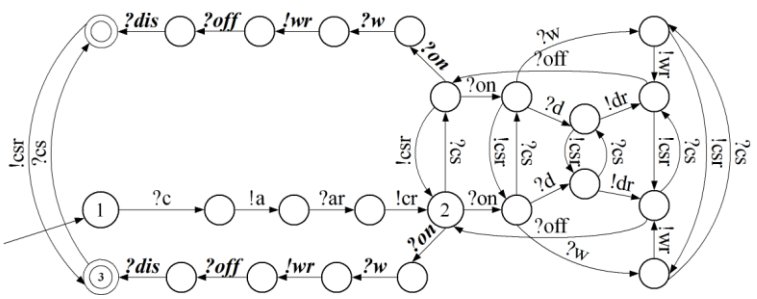


Fig. 6 Behavior constraint automaton of ATM Service under withdrawal scenario

图7 ATM 服务在取款场景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如图 6所示，其中 ATM 服务的 BPN 模型 N 如图 3所示，用户需求的取款场景顺序图 D 如图 1(B)所示。事件在自动机的边上标记，取款场景的行为用粗斜体表示。

4.3 运行时的行为调控

令 N 是 BPN 模型，D 是顺序图。针对顺序图 D 的行为调控需要在运行时完成。令 $X=(\alpha, S, \delta, s_0, S_F, S_{pre}, S_{im}, S_{post})$是 N 在顺序图 D 下的行为约束自动机。由于 X 接受的 N 的事件序列都带有顺序图 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，调控服务可以根据消息交互运行行为约束自动机 X，对输入给目标服务的消息进行过滤，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，区分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，并在服务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 $Next(Current,e) = \{s' \in S \mid \exists s \in Current. s' \in \delta(s,e)\}$，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 $Next(Current, e_m) \neq \emptyset$，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 $Expect = \{m_x \mid \exists s \in Current. \delta(s,e_x) \neq \emptyset\}$，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，如果当前消息可避免指定场景的出现，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，从而避免该场景在目标服务的此次执行中出现。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 $Next(Current, e_m) \cap S_{im} \neq \emptyset$，则告知用户为满足行为过滤的需求，应发送其它消息 $Expect = \{m_x \mid Next(Current, e_x) \cap S_{im} = \emptyset\}$，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 $Next(Current, e_m)$。

例如，对于 1.1 节中的两个消息交互序列 $ms1$ 和 $ms2$，如需抽取取款场景，调控服务从图 6中标号为 1 的初始状态开始运行行为约束自动机。$ms1$ 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 $ms2$，则在收到?disconnect 消息时，$Current$ 中为图 6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 实现了一个原型工具 BASIS（Behavior AnalySer for Interactive Services，交互式服务行为分析工具）来实现 web 服务的行为调控的自动化。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的 UML 顺序图。服务的 BPEL 行为规约被首先转化为 BPN 模型，接下来在该模型上针对顺序图进行行为分析。行为分析的结果被用来构造调控服务，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。在 BASIS 的每次执行完成后，用户将得到化简后的 BPEL 代码、BPN 模型、行为约束自动机以及构造的基于 BPEL 和 Java 的调控服务。接下来，用户可将调控服务部署在服务引擎上，之后就可以调用消息过滤服务，在运行时实现对目标服务的行为调控。ATM 服务的实例研究、BASIS 工具的详情和下载可访问网址：http://seg.nju.edu.cn/BASIS010/。

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。本文提出的服务行为调控的框架，通过监听、检查和转发的消息交互，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 BPEL-Petri

网模型（BPN 模型），用户需求的行为通过顺序图来描绘。行为分析中，我们提出了算法遍历 BPN 模型，获取目标服务 BPN 模型中所有符合顺序图的行为的集合，同时也克服了循环带来的无穷运行以及并发带来的状态空间爆炸等问题。行为分析的结果被用来构造行为约束自动机，这个自动机可以接受 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。我们构造了调控服务以在运行时，通过行为约束自动机，检查并过滤输入消息，完成对目标服务的行为调控。同时，我们设计了原型工具 BASIS 完成了基于场景规约的服务行为调控的自动化。

在将来的工作中，我们会考虑 BPEL 中的错误、补偿和中止处理流程的行为，并将数据流结合到行为调控的框架中来，以增强行为调控的能力，使得我们对服务行为的调控工作更为深入全面。此外，虽然现在的实例研究较为简单，今后的工作中我们会将行为调控的框架用于更加复杂的 SOA 应用中。

References:

[1] A. Aho , M. Lam, R. Sethi, J. Ullman. Compilers: Principles, Techniques, and Tools, 2nd Edition. Boston: Addison Wesley, 2006.

[2] M.H. ter Beek, A. Bucchiarone, S. Gnesi: Formal Methods for Service Composition. Annals of Mathematics, Computing & Teleinformatics. 2007, 1(5): 1-10.

[3] P. Bertoli, M. Pistore, and P. Traverso. Automated composition of web services via planning in asynchronous domains. Artificial Intelligence. 2010, 174(3-4). 316-361.

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

[5] M. Chen, X. Qiu,W. Xu, L.Wang, J. Zhao, and X. Li. UML activity diagram-based automatic test case generation for Java programs. The Computer Journal, 2009, 52(5):545–556,.

[6] S. Dustdar and W. Schreiner. A survey on web services composition. International Journal of Web and Grid Services. 2005, 1(1): 1-30.

[7] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In Proceedings of the 13th International Conference on World Wide Web, 2004. 621–630.

[8] Object Management Group. Unified Modeling Language, version 1.4.2, 2004. ISO/IEC 19501: http://www.omg.org/spec/UML/ISO/19501/PDF.

[9] X.-D. Li, J. Hu, L. Bu, J.-H. Zhao, and G.-L. Zheng. Consistency checking of concurrent models for scenario-based specifications. In Proceedings of the 12th International System Design Languages Forum, 2005. 298–312.

[10] N. Lohmann. A feature-complete Petri net semantics for WS-BPEL 2.0. In Proceedings of the 4th international conference on Web services and formal methods, 2007. 77-91.

[11] N. Lohnmann, P. Massuthe, and K. Wolf. Behavioral constraints for services. In Proceedings of 5th International Conference on Business Process Management, 2007. 271–287.

[12] A. Marconi, M. Pistore, and P. Traverso. Specifying data-flow requirements for the automated composition of web services. In Proceedings of 4th IEEE International Conference on Software Engineering and Formal Methods, 2006. 147–156.

[13] C. Ouyang, E. Verbeek, W.M.P. Aalst, S. Breutel, M. Dumas, and A. Hofstede. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming. 2007, 67(2-3):162–198.

[14] J.L. Peterson. Petri Nets 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] L. Yang, X. Liu, L. Wang, X. Chen, and X. Li. Scenario-based analysis and verification for web services message flows. Chinese Journal of Computers. 2009 32(9):1759–1772 (in Chinese with English abstract).

[17] X. Yi and K.J. Kochut. A CP-nets-based design and verification framework for web services composition. In Proceedings of IEEE International Conference on Web Services, 2004. 756–760.

附中文参考文献:

[16] 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,第 32 卷第 9 期.2009 年 9 月.1759-1772.