面向基于场景规约的Web服务消息流分析与验证

杨 璐 柳 溪 王林章 陈 鑫 李宣东

(南京大学计算机软件新技术国家重点实验室 南京 210093)

(南京大学计算机科学与技术系 南京 210093)

摘 要 采用UML顺序图构成基于场景的规约、WS-BPEL作为Web服务的描述语言,提出了一种面向基于场景规约对Web服务消息流进行分析与验证的方法:首先,对WS-BPEL消息流进行分析并将其自动抽象为基于Petri网的模型;同时,为了缩小状态空间、提高验证效率,在不影响消息交互顺序的前提下,对WS-BPEL源码和基于Petri网的模型分别进行化简,即面向基于场景规约将与验证无关的活动和元素删除;最后,通过遍历基于Petri网的模型以验证WS-BPEL消息流与基于场景的规约之间的一致性(消息交互顺序的存在/强制一致性).文中通过一个贯穿整个分析与验证过程的实例加以说明.该方法已经实现成为一个原型工具.

关键词 Web服务;基于场景的规约;消息交互一致性验证

中图法分类号 TP311 **DOI号**: 10.3724/SP.J.1016.2009.01759

Scenario-Based Analysis and Verification for Web Services Message Flows

YANG Lu LIU Xi WANG Lin-Zhang CHEN Xin LI Xuan-Dong

(*State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing* 210093)

(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

Abstract This paper purposes a scenario-based analysis and verification approach for the Web Services message flow, in which UML Sequence Diagrams are used to specify scenario-based specifications and WS-BPEL is used to describe Web Services designs. Firstly the authors analyze a WS-BPEL specification and automatically extract its message flow model expressed by a Petri net. At the same time, according to a given scenario-based specification the authors do the simplification for both the WS-BPEL source code and the Petri net model to reduce the state space for efficient verification, i.e. removing the activities and process elements which are not concerned with the verification against the scenario-based specification. Finally, the authors verify the consistency between the WS-BPEL message flow and the scenario-based specification (existential/mandatory consistency of message sequence) by traversing the Petri net model. A case study is given throughout the analysis and verification process to illustrate the approach. And a prototype tool is implemented to support this approach.

Keywords Web services; scenario-based specification; message interaction consistency checking

---

收稿日期:2009-04-17;最终修改稿收到日期:2009-08-08.本课题得到国家自然科学基金项目(60673125,90818022)、国家"八六三"高技术研究发展计划项目基金(2009AA01Z148)和江苏省基础研究计划项目基金(BK2007714)资助.**杨 璐**,女,1982年生,博士研究生,主要研究方向为软件工程、软件验证.E-mail:yanglu@seg.nju.edu.cn.**柳 溪**,男,1984年生,博士研究生,主要研究方向为软件工程.**王林章**,男,1973年生,博士,副教授,主要研究方向为模型驱动的软件测试与验证、软件测试自动化.**陈 鑫**,男,1975年生,博士,讲师,主要研究方向为软件工程、面向对象的技术、构件技术和形式化方法.**李宣东**,男,1963年生,教授,博士生导师,主要研究领域为软件建模与分析、软件测试与验证.E-mail:lxd@nju.edu.cn.

1 引 言

在开发和执行分布于网络上的各种业务流程时,Web 服务技术起着重要作用.对 Web 服务行为的分析与验证是 Web 服务技术相关研究领域中日益引起关注的一个问题.

Web 服务是由消息驱动的,本文重点关注 Web 服务之间的消息交互顺序,即对 Web 服务消息流进行分析,并在此基础上验证 Web 服务需求规约和设计规约之间消息交互顺序的一致性.

基于场景的规约(scenario-based specification)提供了一种直观的、可视化的方法来描述系统需求,适用于描述消息交互相关的 Web 服务需求.本文中基于场景的规约表达为 UML 顺序图[1].

由结构化信息标准促进组织(Organization for the Advancement of Structured Information Standards,OASIS)提出的 Web 服务业务流程执行语言(WS-BPEL)[2]是 Web 服务标准集中的一项事实标准.作为 Web 服务的一种设计规约语言,WS-BPEL 侧重于描述 Web 服务的行为和组合.

本文采用 UML 顺序图构成基于场景的规约、WS-BPEL 作为 Web 服务的描述语言,提出了一种面向基于场景规约对 Web 服务消息流进行分析与验证的方法:首先,对 WS-BPEL 消息流进行分析并将其自动抽象为基于 Petri 网[3]的模型;同时,为了缩小状态空间、提高验证效率,在不影响消息交互顺序的前提下,对 WS-BPEL 源码和基于 Petri 网的模型分别进行化简,即面向基于场景规约将与验证无关的活动和元素删除;最后,通过遍历基于 Petri 网的模型以验证 WS-BPEL 消息流与基于场景的规约之间的一致性(消息交互顺序的存在/强制一致性).

2 Web 服务的规约

2.1 Web 服务设计规约语言 WS-BPEL

WS-BPEL 是一种用于描述 Web 服务行为和组合的规约语言.WS-BPEL 基于 XML,有着较好的工业基础,已经成为了工业界的事实标准,并已由 OASIS 正式标准化.WS-BPEL 中包含了用于简单交互的结构(例如 Web 服务调用、操作数据、抛出异常或终止一个 Web 服务流程等),而这些简单结构能够被控制流结构(例如顺序、并行、条件分支选择等)组合到一起,从而创建出复杂的 Web 服务流程.WS-BPEL 规约中,这两种结构用基本活动(basic activity)和结构化活动(structured activity)来描述.

例 1. 用例场景:贷款核准服务 LAS.

我们以一个简单的贷款核准服务 LAS(Loan Approval Service)用例贯穿本文所关注的整个分析与验证过程,用以说明和示例本文的工作,该用例最初在 WS-BPEL2.0 规约[2]中提出.

该用例所涉及的场景如下:客户提出贷款申请,包括客户个人资料和申请的贷款金额等信息.LAS 得到这些信息后将执行贷款审核流程并返回审核结果,审核结果是一条"核准贷款"或"拒绝贷款"的消息.影响审核结果的因素包括客户申请的贷款金额和该客户的贷款风险评估等级.对贷款金额低于 10000,且贷款风险等级为"低"的客户,贷款申请通过一个简捷的流程被自动核准.对贷款金额高于 10000,或贷款风险等级为"中"或"高"的客户,则需要进行进一步的信用评估处理.LAS 可调用由另两个服务提供的功能来审核贷款申请:在针对低贷款金额的简捷流程中,风险评估服务(Risk Assessment Service,RAS)被用于对客户的贷款风险给出一个快速的评估;而当简捷流程不适用时,专家核准服务(Expert Approval Service,EAS)被用于进行信用评估,该服务可能需要一位贷款专家的参与.RAS 接收贷款申请消息,并返回风险等级消息.EAS 接收贷款申请消息,并返回批准结果消息来确认该申请是否被批准.该用例中假设客户的公司需要开发一个项目,提出的贷款申请需要由 LAS 来处理,并需要调用 EAS 来进行专家核准.

由于篇幅限制,该用例中的 WS-BPEL 源码在 SSCC 网站(http://seg.nju.edu.cn/SSCC/index.htm)中给出.

2.2 Web 服务需求规约:基于场景的规约

基于场景的规约能够描述具体的交互,且易为用户和领域专家所使用,是一种被广泛使用的需求描述手段.本文采用 UML 顺序图表达基于场景的规约,主要包括存在一致性规约(existential consistency specifications)和强制一致性规约(mandatory consistency specifications),其中强制一致性规约又可分为向前(forward)、向后(backward)和双向(bidirectional)强制一致性规约,这 4 种规约具体定义如下(见图 1):

(1)存在一致性规约要求顺序图 D 描述的被禁止场景不会在服务运行过程中出现;

(2) 向前强制一致性规约要求当顺序图 D_1 描述的场景出现在服务运行过程中时,顺序图 D_2 描述的场景必须紧跟着出现;

(3) 向后强制一致性规约要求当顺序图 D_1 描述的场景出现在服务运行过程中时,必须紧跟着顺序图 D_2 描述的场景;

(4) 双向强制一致性规约要求当顺序图 D_1 描述的场景出现在服务运行过程中、并且之后出现顺序图 D_2 描述的场景时,在两者之间必须出现顺序图 D_3 描述的场景.

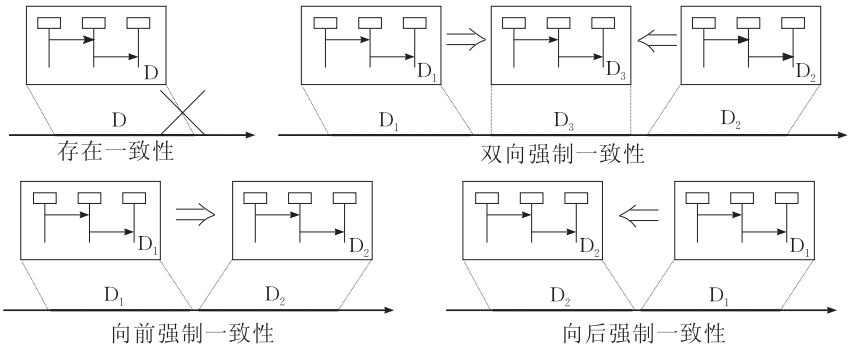

图 1　存在一致性和强制一致性规约

例 2.　基于场景的规约:贷款核准服务.

描述例 1 需求的基于场景规约如图 2 所示. 该用例中用于描述基于场景规约的 UML 顺序图 assessorUSD、approvalUSD 和 replyUSD 在 SSCC 网站中给出. 这 3 个 UML 顺序图既可用于存在一致性验证,也可用于强制一致性验证.

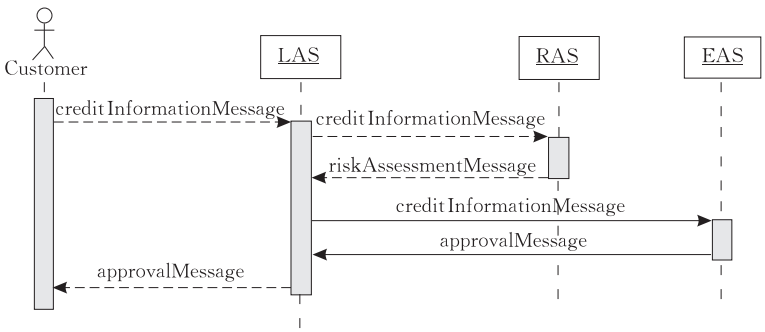

图 2　基于场景的规约:贷款核准服务的 UML 顺序图

2.3　WS-BPEL 消息流模型

本文基于 Petri 网构建 WS-BPEL 消息流的模型. WS-BPEL 规约中定义的每一个活动可被抽象为一个 Petri 网模式,若干这样的 Petri 网模式构成一个完整的基于 Petri 网的模型,这个模型被用于描述 WS-BPEL 消息流. 为了简化分析过程,本文使用单 token 的经典 Petri 网来对 WS-BPEL 消息流进行分析与建模. 经典 Petri 网的定义如下.

定义 1.　Petri 网. Petri 网 N 是一个四元组,$N=(P,T,F,\mu_0)$,其中

$P=\{p_1,p_2,\cdots,p_m\}$ 是一个库所节点(place)的有穷集;

$T=\{t_1,t_2,\cdots,t_n\}$ 是一个变迁节点(transition)的有穷集,且 $P\cap T=\varnothing$;

$F\subset(P\times T)\cup(T\times P)$ 是 Petri 网的流关系(flow relation);

$\mu_0\subset P$ 是 Petri 网的初始标识(initial marking).

标识(marking)μ 是 P 的任意一个子集. 对于任意一个变迁节点 t,${}^{\cdot}t=\{p\in P\mid(p,t)\in F\}$ 和 $t^{\cdot}=\{p\in P\mid(t,p)\in F\}$ 分别表示 t 的前置集(pre-set)和后置集(post-set). 如果 ${}^{\cdot}t\subseteq\mu$,在标识 μ 下变迁节点 t 为使能的(enabled),否则,t 为非使能的(disabled). $enabled(\mu)$ 表示 μ 下使能的变迁节点的集合.

Petri 网的行为用运行(run)表示,定义如下.

定义 2.　Petri 网的运行. Petri 网的一次运行 σ 是一个由标识节点和变迁节点组成的有穷或无穷序列,$\sigma=\mu_0\xrightarrow{t_0}\mu_1\xrightarrow{t_1}\cdots\xrightarrow{t_{n-1}}\mu_n\xrightarrow{t_n}\cdots$,其中 μ_0 是Petri 网的初始标识. 对于任意 $i(i\geq0)$,$t_i\in\text{enabled}(\mu_i)$;对于任意 i $(i\geq1)$,$\mu_i=(\mu_{i-1}-{}^{\cdot}t_{i-1})\cup t_{i-1}^{\cdot}$.

为了针对 WS-BPEL 消息流进行建模,我们引入 WS-BPEL Petri 网.

定义 3. WS-BPEL Petri 网(BPN). WS-BPEL Petri 网 BPN 是一个八元组,$BPN=(N, MP, L, \mathcal{L}, G, \mathcal{G}, A, \mathcal{A})$,其中

N 是一个 Petri 网;

$MP \subset P$ 是消息库所节点(message place)的集合;

$L=\{$"initial", "final", "abnormal", "exited", "disable"$\}$是标志(label)的有穷集;

$\mathcal{L}: P \rightarrow T$ 是标志函数(labeling function);

G 是卫式表达式(guard expression)的有穷集;

$\mathcal{G}: T \rightarrow G$ 是卫式函数(guarding function);

A 是赋值表达式(value assignment expression)的有穷集;

$\mathcal{A}: T \rightarrow 2^A$ 是赋值函数(assigning function).

WS-BPEL Petri 网基于经典 Petri 网 N 定义,并且定义了一个特殊的库所节点集合 MP 来表示消息交互的端点,称为消息库所节点. 标志函数 $\mathcal{L}$ 将标志附在库所节点上,来表示库所节点的接口类型. 卫式函数 $\mathcal{G}$ 将卫式表达式附在变迁节点上,来表示激活变迁节点的条件. 赋值函数 $\mathcal{A}$ 将赋值表达式附在变迁节点上,每个变迁节点可以附有多个赋值表达式. WS-BPEL Petri 网既可用于 WS-BPEL 消息流分析与建模,也可用于 WS-BPEL 数据流分析与建模.

本文重点关注由 UML 顺序图描述的基于场景的规约和由 BPN 描述的 WS-BPEL 消息流之间的一致性,因此变迁节点 t 可能标有一个表示 UML 顺序图中消息发送或接收的事件. 为此,本文定义了事件(event)、事件轨迹(event trace)和事件轨迹集合(event trace set).

定义 4. 事件. 如果变迁节点 t 的目标库所节点之一为消息库所节点 m,称为 t 发送一条消息,且 t 的事件定义为 $\varepsilon(t)=m!$. 如果变迁节点 t 的源库所节点之一为消息库所节点 m,称为 t 接收一条消息,且 t 的事件定义为 $\varepsilon(t)=m?$. 如果变迁节点 t 的源库所节点和目标库所节点中都没有消息库所节点,称为 t 既不发送消息也不接收消息.

定义 5. 事件轨迹. 事件轨迹定义为一个形式为 $\varepsilon(t_a)\ \varepsilon(t_b),\cdots,\varepsilon(t_z)$ 的事件序列,当且仅当

① $t_a, t_b, \cdots, t_z$ 在 BPN 的同一次运行中;

② $t_a, t_b, \cdots, t_z$ 在事件序列中的顺序和在 BPN 的运行中的顺序相同;

③ 在事件序列中的每个变迁节点都必须发送或接收消息;

④ 在 BPN 的运行中但不在事件序列中的每个变迁节点 t' 都必须既不发送消息也不接收消息;

⑤ 每个事件仅在序列中发生一次.

3 Web 服务消息流的分析、建模及验证

本文主要关注的问题是面向基于场景规约对 Web 服务消息流进行分析与验证,即验证消息交互顺序的存在/强制一致性. 其中基于场景的规约由 UML 顺序图构成、Web 服务由 WS-BPEL 描述.

针对上述问题,本节详细地给出了一种解决方法:首先,对 WS-BPEL 消息流进行分析并基于 Petri 网建模;同时,在不影响消息交互顺序的前提下,对 WS-BPEL 源码和基于 Petri 网的模型分别进行化简;最后,通过遍历基于 Petri 网的模型以验证 WS-BPEL 消息流与基于场景的规约之间的一致性.

该过程如图 3 所示. 首先,将用于描述 Web 服务的 WS-BPEL 源码作为输入;其次,对 WS-BPEL 源码进行化简,并在此基础上对 WS-BPEL 消息流进行分析并将其自动抽象为基于 Petri 网的模型;再次,对上一步中得到的基于 Petri 网模型进行化简得到化简后的基于 Petri 网模型;最后,针对输入的 UML 顺序图和化简后的基于 Petri 网模型,进行消息交互一致性的验证.

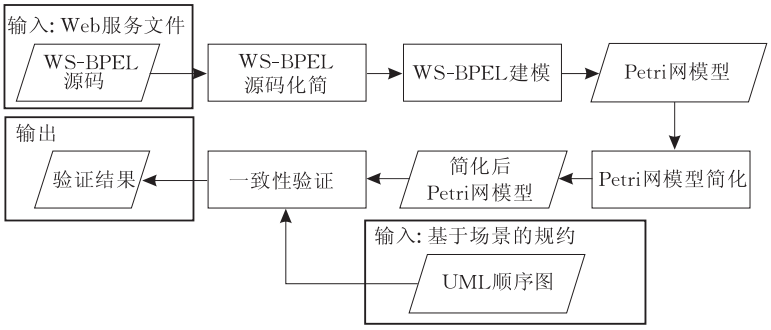

图 3 WS-BPEL 分析、建模和验证过程

3.1　WS-BPEL 消息流的分析与建模

WS-BPEL 用于描述 Web 服务的行为和组合,WS-BPEL 规约中包含了基本活动和结构化活动.

我们首先对 WS-BPEL 消息流进行分析并将其自动抽象为 *BPN* 模型. WS-BPEL 规约中定义的每一个活动可被抽象为一个 Petri 网模式,若干这样的 Petri 网模式构成一个完整的 *BPN* 模型,这个模型被用于描述 WS-BPEL 消息流. 为了缩小状态空间、提高验证效率,我们在建模过程中简化了与消息交互无关的活动(例如〈wait〉活动)以及与主要业务流程无关的元素(例如异常处理、补偿处理和终止处理,即 Fault Handling, Compensation Handling 和 Termination Handling,简称 FCT 元素),即针对基于场景规约的验证将无关的活动和元素删除.

在 Petri 网模式的图示中,虚线表示该 Petri 网模式的边界,边界上的库所节点给出了 Petri 网模式对外部的接口,这些接口被用于将不同的 Petri 网模式连接到一起.

3.1.1　WS-BPEL 基本活动的分析与建模

WS-BPEL 规约中定义了 10 种基本活动,基本活动描述了 Web 服务的基本行为:〈receive〉活动接收由其它 Web 服务流程发送的服务调用请求的消息;〈reply〉活动发送对接收到的消息的响应;〈invoke〉活动调用其它 Web 服务流程的一个单向(异步)或请求-响应(同步)的操作;〈wait〉活动使 Web 服务流程等待直到一个截止期限,或者等待一段特定的时间间隔;〈assign〉活动将数据从一个变量复制到另一个变量,也可以通过使用表达式更新变量值;〈empty〉活动什么都不做;〈exit〉活动中止 Web 服务流程实例,而且不调用 FCT 元素;〈throw〉活动在 Web 服务流程内部产生并抛出一个异常;〈rethrow〉活动将被异常处理捕获的异常向上层抛出.

〈invoke〉活动(单向/请求-响应)、〈receive〉活动、〈reply〉活动、〈assign〉活动、〈empty〉活动和〈exit〉活动的模式如图 4 所示. 〈wait〉活动、〈throw〉活动、〈rethrow〉活动和〈extensionActivity〉活动未在建模过程中给出.

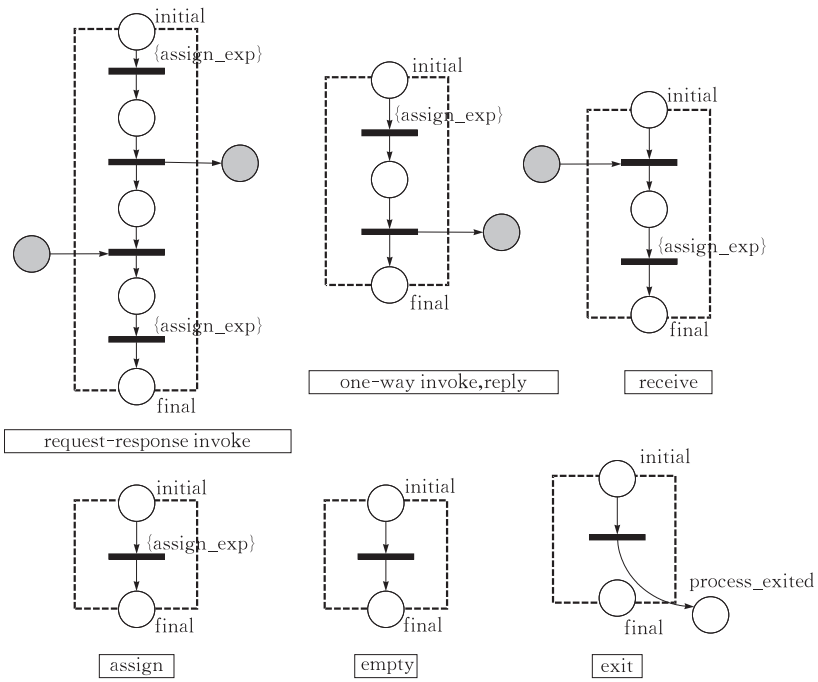

图 4　基本活动的模式

〈invoke〉活动(单向/请求-响应)、〈receive〉活动和〈reply〉活动与消息交互相关,即在 Web 服务流程中发送和接收消息. 因此在图 4 所示的这 3 种模式中,阴影表示的库所节点表示与消息交互相关的消息库所节点. 节点上附有消息四元组,包括该活动的 partnerLink、portType、operation 以及 message 属性.

3.1.2　WS-BPEL 结构化活动的分析与建模

WS-BPEL 规约中定义了 7 种结构化活动. 通过将基本活动连接起来并规定其运行方式,结构化活动构造出复杂的 Web 服务流程并给出其行为描述:〈sequence〉活动、〈if〉活动、〈while〉活动、〈repeatUntil〉活动和顺序〈forEach〉活动定义了基本活动间的顺序控制;〈flow〉活动和并行〈forEach〉

活动定义了基本活动间的并行以及同步控制；〈pick〉活动定义了由事件决定的选择控制.

〈sequence〉活动、〈if〉活动、〈while〉活动、〈repeatUntil〉活动、〈flow〉活动和〈pick〉活动的模式如图 5 所示.〈forEach〉活动的模式如图 6 所示.

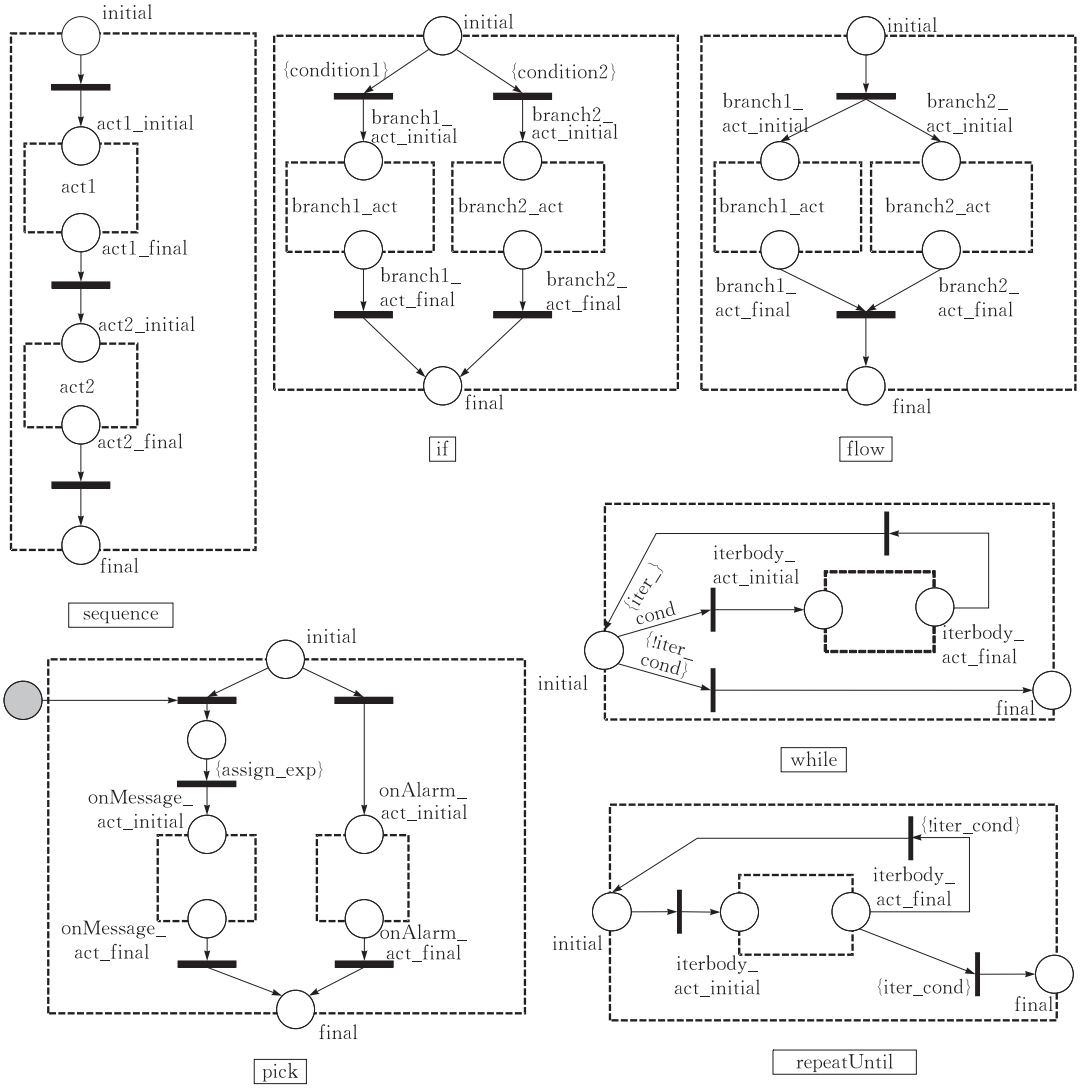

图 5 结构化活动的模式

在〈if〉、〈while〉、〈repeatUntil〉、〈forEach〉等活动的模式中，分支选择条件作为卫式(guard)附在变迁节点上.

在并行〈forEach〉活动中，$N+1$ 条内部〈scope〉分支并行执行（N = finalCounterValue - startCounterValue). 多数情况下 startCounterValue 和 finalCounterValue 的值不会被静态指定，因此我们可以任意指定一个整数作为 N 的值. N 个内部〈scope〉活动的模式组合在一起成为该并行〈forEach〉活动的模式. 在本文中并行行为的终止未被考虑，因此并行〈forEach〉活动的模式中对于 completionCondition 的检验被省略.

3.1.3 WS-BPEL 流程元素的分析与建模

我们需要建模的流程元素包括〈process〉元素、〈scope〉元素、eventHandlers 元素、以及〈flow〉活动中的 links 元素.〈process〉元素是 WS-BPEL 流程的顶层活动，即流程树定义的根节点；〈scope〉元素以分层方式将复杂流程划分为多个组织部分，并为活动提供了行为上下文.〈scope〉元素为嵌套在其中的活动提供故障处理、补偿处理、终止处理和事件处理功能，〈scope〉元素中可以包含多个 eventHandler、多个 faultHandler、一个 compensationHandler 以及一个 terminationHandler，其中 FCT 元素未在本文中讨论. 在并行执行的一组活动中，links元素指明

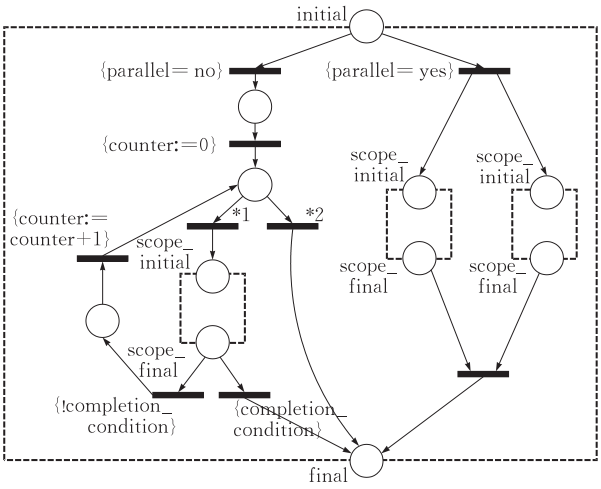

图 6　〈forEach〉活动的模式

执行顺序的约束.

〈scope〉元素的模式如图 7(a)所示.〈process〉元素的模式如图 7(b)所示.〈process〉和〈scope〉元素的语法和语义类似,因此两者的模式也很类似.但是,两者之间仍有区别,例如:compensationHandler 和 terminationHandler 不会附在〈process〉结构中、exited 接口仅用在〈process〉元素上.

links 元素的模式如图 8 所示. links 元素的语义是 WS-BPEL 行为的重要补充,链接的源活动的定义使得每个传出链接都对应一个〈source〉元素,链接的目标活动的定义使得每个传入链接都对应一个〈target〉元素.因为 links 元素必须是〈flow〉活动的属性,所以我们将 links 元素的模式放在〈flow〉活动的模式中理解,这里省略了〈flow〉活动中其余无关的部分.在该模式中,transitionCondition在源活

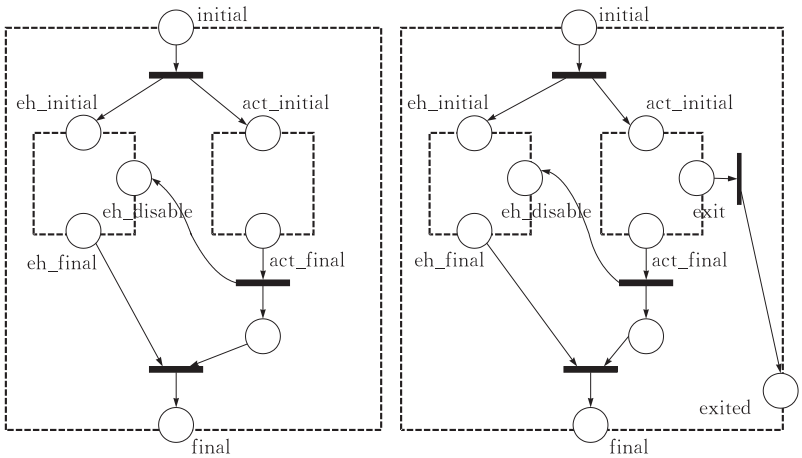

图 7　〈scope〉元素和〈process〉元素的模式

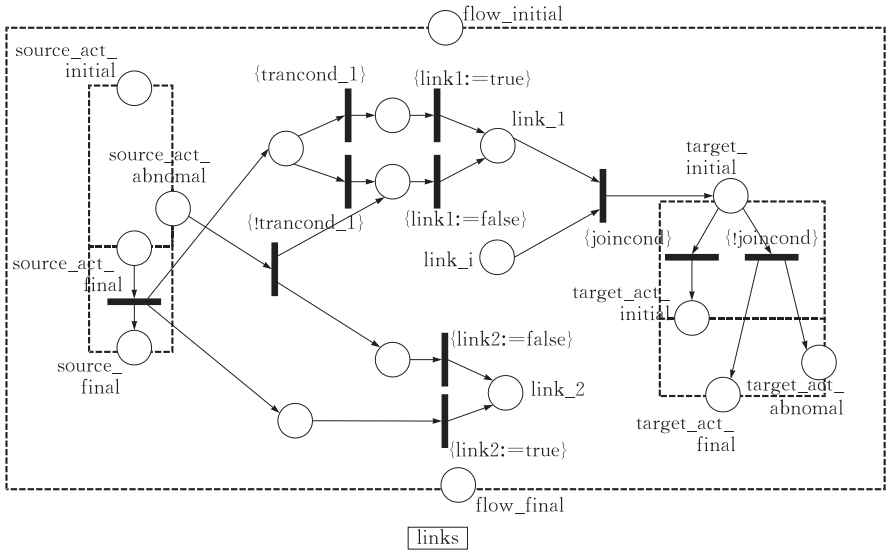

图 8　links 元素的模式

动结束之后被检验. 如果 transitionCondition 满足，则链接的值被设为 true，反之，链接的值被设为 false. 如果源活动未定义 transitionCondition，且源活动结束成功，则链接的值被设为 true，反之，源活动结束失败，链接的值被设为 false. joinCondition 在目标活动开始之前被检验. 如果 joinCondition 满足，则目标活动开始执行，反之，目标活动被跳过，整个流程异常中止(abnormal 接口被标志). 这里需要注意的是，如果一个链接的源活动异常中止，它的所有传出链接的输出值也被设为 false，用于“Dead Path Elimination”[2]. 此外，因为在本文中不考虑异常处理，所以对 suppressJoinFailure 的检验被省略.

eventHandlers 元素的模式如图 9 所示. 在 eventHandlers 元素的模式中，在 onAlarm 分支中 repeatEvery 属性被检验，以决定 onAlarm 分支是在一次执行后结束，还在一段给定时间中重复执行. 该模式接口上的 disable 接口被用于停止所有的 onEvent 分支和 onAlarm 分支. 当所有分支被停止后，eventHandlers 到达 final 接口.

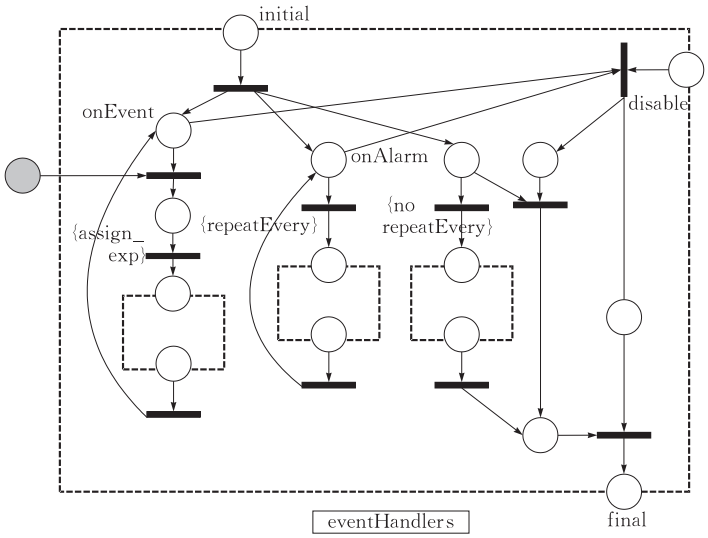

图 9 eventHandlers 元素的模式

本节中的分析与建模基于 WS-BPEL 的操作语义，并着重关注消息交互顺序. 从 WS-BPEL 消息流中自动抽取 BPN 模型的过程定义直观清晰，因此省略该建模过程的证明.

3.2 WS-BPEL 消息流模型的化简

本文重点关注基于场景规约和 WS-BPEL 消息流之间的一致性，因此我们着重关注消息交互相关的活动和元素，例如〈receive〉活动、〈reply〉活动、〈invoke〉活动、〈pick〉活动的 onMessage 分支、eventHandlers 元素的 onEvent 分支以及连接这些活动和元素的控制流. 消息交互无关的活动和元素被省略，例如〈assign〉活动、〈empty〉活动和〈wait〉活动，但保留了它们的链接依赖关系.

FCT 元素是 WS-BPEL 的主要特性之一，但 FCT 元素也加剧了验证中的状态空间爆炸问题. 因为我们着重关注 Web 服务的主要业务流程，所以也省略了 FCT 元素.

3.1 节给出的 Petri 网模式已经不包含上述 WS-BPEL 流程的动态语义，仅涉及 WS-BPEL 的主要业务流程以及消息交互相关的活动和元素，但经过分析与建模得到的 *BPN* 模型仍然存在状态空间过大的问题，可以进一步进行化简.

本节给出了两步化简规则：首先对 WS-BPEL 源码进行化简，并将化简过的 WS-BPEL 源码自动抽象为 *BPN* 模型；然后对该 *BPN* 模型再进行化简. 经过这两步化简，*BPN* 模型的状态空间被大大缩小.

3.2.1 WS-BPEL 源码的化简

对 WS-BPEL 源码的化简规则主要包括删除〈empty〉活动、动态语义相关的活动(例如〈wait〉活动)、FCT 元素以及没有内部活动的结构化活动. 此外，多个相连的〈assign〉活动被化简为一个. 对于 WS-BPEL 源码中的其余一些冗余结构也进行了化简.

对 WS-BPEL 源码的化简步骤如下.

(1) 用〈empty〉活动取代所有的〈wait〉活动、〈throw〉活动、〈rethrow〉活动和〈extensionAcitivty〉活动，并保留相应的链接依赖关系；

(2) 移除所有的 FCT 元素和〈invoke〉活动中的相应 FCT 元素定义,并移除所有 FCT 元素上的链接依赖关系;

(3) 如果〈scope〉元素中既没有 eventHandlers 元素,也没有变量定义,则用该〈scope〉元素的内部活动取代它,并且将〈scope〉元素上的链接依赖关系移到该内部活动上;

(4) 如果结构化活动的所有内部活动都是〈empty〉活动,则新建一个〈empty〉活动来代替该结构化活动,并将该结构化活动和其内部〈empty〉活动的链接依赖关系移到新建的〈empty〉活动上;

(5) 如果〈sequence〉活动的内部活动中存在一个〈empty〉活动,且该〈empty〉活动是链接的源活动,则将该链接依赖关系移到该〈empty〉活动的前驱活动上,并从〈sequence〉活动中移除该〈empty〉活动;如果该〈empty〉活动是链接的目标活动,则将该链接依赖关系移到该〈empty〉活动的后继活动上,并从〈sequence〉活动中移除该〈empty〉活动;

(6) 如果在〈sequence〉活动内部,两个或两个以上相连的〈assign〉活动顺序执行,且前一个〈assign〉活动的〈source〉元素定义中对 transitionCondition 的检验未涉及后一个〈assign〉活动的变量,则将后一个〈assign〉活动中的所有〈copy〉元素移到前一个〈assign〉活动中,并将后一个〈assign〉活动上的链接依赖关系移到前一个〈assign〉活动上,最后从〈sequence〉活动中移除后一个〈assign〉活动;

(7) 如果一个链接的源活动和目标活动都是〈empty〉活动,则移除该链接、源活动的相应〈source〉元素和目标活动的相应〈target〉元素;

(8) 如果 if 分支、elseif 分支、else 分支或 onAlarm 分支的内部活动是〈empty〉活动,且该〈empty〉活动没有〈source〉元素定义,则移除该分支及〈empty〉活动上的链接;如果该〈empty〉活动有〈source〉元素定义,则保留该分支和〈empty〉活动作为链接的源活动;

(9) 在〈flow〉活动内部,如果内部〈empty〉活动没有〈source〉元素定义,则移除该〈empty〉活动,并将该〈empty〉活动上的链接依赖关系移到〈flow〉活动上;如果内部〈empty〉活动有〈source〉元素定义,则保留该〈empty〉活动作为链接的源活动.

通过执行以上步骤,我们对 WS-BPEL 源码进行了化简.这里要注意,在规则 7 和规则 8 中,if 等分支或者〈flow〉活动的分支中的〈empty〉活动在某些情况下不会被移除,原因在于该〈empty〉活动需要作为链接的源活动存在,否则 WS-BPEL 源码中定义的作为控制流重要方面的链接依赖关系的信息可能会丢失.

3.2.2　BPN 模型的化简

经过 3.2.1 节对 WS-BPEL 源码的化简,本节对基于 Petri 网对 WS-BPEL 消息流自动建模的过程提供进一步的化简规则.

本文中为了避免状态空间爆炸问题省略了 FCT 元素,因此在 3.1 节定义的 Petri 网模式中,可以移除部分为 FCT 元素服务的库所节点和变迁节点,得到化简过的 Petri 网模式.接着,使用这些化简过的 Petri 网模式来对 WS-BPEL 消息流自动建模,并且得到的 BPN 模型还可以进一步进行化简.

在介绍对建模得到的 BPN 模型的化简规则之前,我们首先定义两组函数.

定义 6.　函数 $\mathcal{T}^P: P \to P.\ \mathcal{T}^P(p) = \bigcup_{t \in {}^\bullet p} t^\bullet$.

定义 7.　函数 $\mathcal{S}^P: P \to P.\ \mathcal{S}^P(p) = \bigcup_{t \in p^\bullet} {}^\bullet t$.

函数 $\mathcal{T}^P(p)$ 的结果为 ${}^\bullet p$ 中所有变迁节点的目标库所节点的集合.函数 $\mathcal{S}^P(p)$ 的结果为 $p^\bullet$ 中所有变迁节点的源库所节点的集合.

定义 8.　函数 $\mathcal{T}^T: T \to T.\ \mathcal{T}^T(t) = \bigcup_{p \in t^\bullet \wedge p \notin MP} p^\bullet$.

定义 9.　函数 $\mathcal{S}^T: T \to T.\ \mathcal{S}^T(t) = \bigcup_{p \in t^\bullet \wedge p \notin MP} {}^\bullet p$.

函数 $\mathcal{T}^T(t)$ 的结果为 $t^\bullet$ 中所有非消息库所节点的目标变迁节点的集合.函数 $\mathcal{S}^T(t)$ 的结果为 $t^\bullet$ 中所有非消息库所节点的源变迁节点的集合.

通过迭代执行以下步骤,我们对建模得到的 BPN 模型进行化简.该迭代过程的结束条件为 BPN 模型不再改变.其中,$P_{rm} \subset P$ 和 $T_{rm} \subset T$ 是每一次化简迭代步骤后要被移除的库所节点和变迁节点的集合,且都被初始化为 $\varnothing$.以下每条化简规则中符号 $\to$ 定义了化简过程,符号前为化简条件,符号后为化简后 BPN 模型的变化,这些变化由对 P_{rm},T_{rm} 和流关系的赋值来描述:

(1) $\forall p \in P, {}^\bullet p = p^\bullet = \varnothing \to P'_{rm} := P_{rm} \cup \{p\}$;

解释:步骤 1 移除所有的孤立库所节点,即既没有输入变迁节点也没有输出变迁节点的库所节点.例如,当整个 BPN 模型中没有〈exit〉活动时,〈process〉元素对应模式的 exited 接口为孤立库所节点.当一轮化简步骤执行结束之后,一个库所节点的输入变迁节点或输出变迁节点被移除,则该库所节点在下一轮化简步骤执行时可能成为孤立库所节点.

(2) $\forall p \in P, \mathcal{L}(p) = \text{“abnormal”} \wedge p^\bullet = \varnothing \to$
$P'_{rm} := P_{rm} \cup \{p\}$;

解释：如果一个没有输出变迁节点的库所节点作为 abnormal 接口存在，则该接口对应的活动不会有〈source〉元素定义，因此不需要验证该活动是否成功终止. 步骤 2 移除这个作为 abnormal 接口的库所节点.

(3) $\forall t\in T, t^{\cdot}=\varnothing \rightarrow T'_{\mathrm{rm}} := T_{\mathrm{rm}}\cup\{t\}$；

解释：步骤 3 将没有输出库所节点的变迁节点作为冗余节点移除.

(4) $\forall p\in P, t\in p^{\cdot}, {}^{\cdot}t\notin MP \wedge t^{\cdot}\notin MP \wedge |\mathcal{T}^{P}(p)|=1 \wedge |\mathcal{S}^{P}(p)|=1 \rightarrow P'_{\mathrm{rm}} := P_{\mathrm{rm}}\cup\{p_{\mathrm{next}}\} \wedge T'_{\mathrm{rm}} := T_{\mathrm{rm}}\cup p^{\cdot} \wedge F(p)' := F(p_{\mathrm{next}}) \wedge \forall t_{\mathrm{nxpre}}\in {}^{\cdot}p_{\mathrm{next}}, F(t_{\mathrm{nxpre}})' := F(t_{\mathrm{nxpre}})\cup\{p\}-\{p_{\mathrm{next}}\}$，其中 $\{p_{\mathrm{next}}\}=\mathcal{T}^{P}(p)$；

(5) $\forall t\in T, p\in t^{\cdot}, |\mathcal{T}^{T}(t)|=1 \wedge |\mathcal{S}^{T}(t)|=1 \rightarrow T'_{\mathrm{rm}} := T_{\mathrm{rm}}\cup\{t_{\mathrm{next}}\} \wedge \forall(p_m\in t^{\cdot} \wedge p_m\notin MP), P'_{\mathrm{rm}} := P_{\mathrm{rm}}\cup(t^{\cdot}-\cup p_m) \wedge F(t)' := F(t_{\mathrm{next}}) \wedge \forall p_{\mathrm{nxpre}}\in {}^{\cdot}t_{\mathrm{next}}-t^{\cdot}, F(p_{\mathrm{nxpre}})' := F(p_{\mathrm{nxpre}})\cup\{t\}-\{t_{\mathrm{next}}\}$，其中 $\{t_{\mathrm{next}}\}=\mathcal{T}^{T}(t)$；

解释：步骤 4 和步骤 5 化简如下结构：一个 Petri 网节点(库所节点 p 或变迁节点 t)的所有后继节点都仅有一个前驱节点和一个后继节点(如图 10(a)和 10(b)所示)，该结构可被化简为单个节点(p 或 t).

(6) $\forall p\in t^{\cdot} \wedge p\notin \mu_0, t\in T_{\mathrm{rm}} \wedge |{}^{\cdot}p|=1 \rightarrow P'_{\mathrm{rm}} := P_{\mathrm{rm}}\cup\{p\}$；

解释：如果一个库所节点的输入变迁节点被移除(如图 10(c)所示)，步骤 6 移除该库所节点.

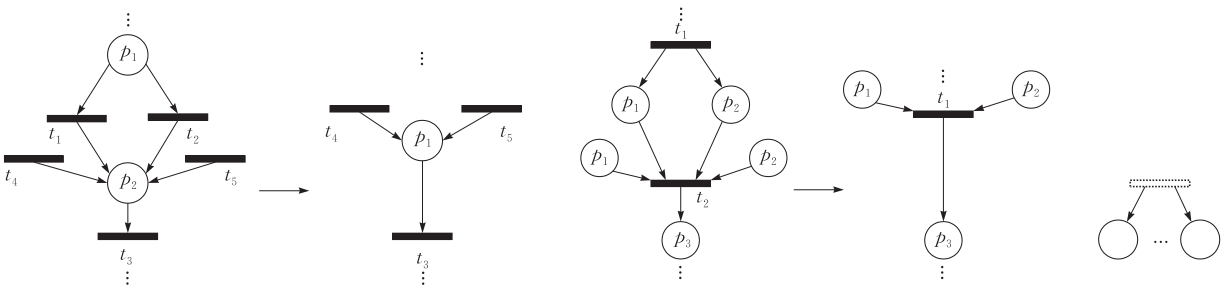

图 10 化简规则的模式

(7) $\forall p\in P_{\mathrm{rm}}, F(p) := \varnothing, P' := P-\{p\}$；$\forall t\in T_{\mathrm{rm}}, F(t) := \varnothing, T' := T-\{t\}$.

解释：在步骤 1 到步骤 6 中，库所节点和变迁节点不会被直接移除，而是分别记录在集合 P_{rm} 和 T_{rm} 中，步骤 7 从 BPN 模型中直接移除所有被记录的库所节点和变迁节点.

当所有步骤执行结束之后，返回步骤 1 并再次执行这些步骤，如果在一次执行中没有产生 BPN 模型的改变，则迭代过程终止，得到对 WS-BPEL 消息流的最终 BPN 模型.

本文提出的化简规则不会改变初始 BPN 模型的消息交互顺序，证明如下.

定义 10. 事件轨迹集合. WS-BPEL Petri 网 BPN 的事件轨迹集合 $TRACE_{BPN}$ 是 BPN 中所有事件轨迹的集合.

定理 1. $\forall trace\in TRACE_{\mathrm{orig}}$，其中 $TRACE_{\mathrm{orig}}$ 是初始 BPN 模型的事件轨迹集合 $\Rightarrow \exists trace'\in TRACE_{\mathrm{simp}}$，其中 $TRACE_{\mathrm{simp}}$ 是化简后的最终 BPN 模型的事件轨迹集合.

证明. 该定理通过对以上化简步骤的迭代执行次数进行归纳证明.

归纳开始于无迭代执行. 归纳基础：当上述化简步骤未迭代时，显见该定理满足.

归纳假设：假设当上述化简步骤迭代执行 k 次时，$\forall trace\in TRACE_{\mathrm{orig}} \Rightarrow \exists trace'\in TRACE_k$，其中 $TRACE_k$ 是化简步骤执行 k 次之后的 BPN 模型的事件轨迹集合.

现在我们证明当第 $k+1$ 次执行上述化简步骤之后，$\forall trace\in TRACE_{\mathrm{orig}} \Rightarrow \exists trace'\in TRACE_{k+1}$. 根据归纳假设，$\forall trace\in TRACE_{\mathrm{orig}} \Rightarrow \exists trace'\in TRACE_k$，因此，我们仅需证明 $\forall trace\in TRACE_k \Rightarrow \exists trace'\in TRACE_{k+1}$. 假设 $\exists trace\in TRACE_k$，$\forall trace'\in TRACE_{k+1}, trace'\neq trace$，那么 $trace$ 一定在化简中被修改了.

步骤(1)：孤立库所节点没有前驱和后继，因此，如果孤立库所节点不在初始标识中，则该库所节点不会出现在 BPN 的任何一个运行中；如果该库所节点在初始标识中，从 BPN 的任何运行和初始标识中删除该库所节点，删除后的 BPN 的运行依然合法. 因此删除孤立库所节点不会使得 $TRACE_k$ 和 $TRACE_{k+1}$ 不同.

步骤(2)：如果一个库所节点 $p^{\cdot}=\varnothing$，那么从

BPN 的任何运行中删除 p,删除后的 BPN 的运行依然合法. 因此删除 p 不会使得 $TRACE_k$ 和 $TRACE_{k+1}$不同.

步骤(3):如果一个变迁节点 $^{\cdot}t=\varnothing$,那么 t 不在 BPN 的任何运行中,$\varepsilon(t)$一定不在 $TRACE_k$ 的事件轨迹中. 因此删除 t 不会使得 $TRACE_k$ 和 $TRACE_{k+1}$不同.

步骤(4)(5):对于库所节点 p,若 $|T^P(p)|=|S^P(p)|=1$,以步骤(4)的方法,合并 p 与 p_{next}、删除相应的变迁节点和改变流关系,不会对 Petri 网 p 与 p_{next}之外的控制流造成影响;而且对于 $\forall t\in p^{\cdot}$,t 不收发消息,因此不会影响任何 BPN 的运行对应的事件轨迹. 类似的,对于变迁节点 t,如果 $|T^T(t)|=|S^T(t)|=1$,以步骤(5)的方法,合并 t 与 t_{next}并删除相应的库所节点和改变流关系,不会对 Petri 网 t 与 t_{next}之外的控制流造成影响;此外,如果 t 或 t_{next}收发消息,在步骤(5)中,t_{next}关联的消息库所都转为与 t 关联(通过调整相应的流关系),而与 t 关联的消息库所的流关系仍然保持,因此不会影响 BPN 的任何运行对应的事件轨迹. 综上,步骤(4)(5) 不会使得 $TRACE_k$和 $TRACE_{k+1}$不同.

步骤(6):对于一个库所节点 $|^{\cdot}p|=1$,假设 $t\in{}^{\cdot}p$,如果 t 根据前步,应当被移除,则 p 不会再出现在 BPN 的任何运行中,除非 $p\in\mu_0$. 因为前步不会造成 $TRACE_k$和 $TRACE_{k+1}$不同,故删除这样的 p 不会使得 $TRACE_k$和 $TRACE_{k+1}$不同.

步骤(7):准备删除的库所节点和变迁节点被实际删除. 因为前步不会造成 $TRACE_k$和 $TRACE_{k+1}$不同,故删除这样的库所节点和变迁节点不会使得 $TRACE_k$和 $TRACE_{k+1}$不同.

可见,上述 7 步均不会造成 $TRACE_k$ 和 $TRACE_{k+1}$ 不同,这与假设矛盾. 因此 $\forall trace\in TRACE_k\Rightarrow\exists trace'\in TRACE_{k+1}$.

综上所述,定理得证.　　证毕.

例 3.　分析、建模和化简:贷款核准服务.

贷款核准服务(LAS)的初始 WS-BPEL 源码在 WS-BPEL 2.0 规约[2]中给出,由于篇幅限制,该用例中经过源码化简步骤的 WS-BPEL 源码在 SSCC 网站中给出. 在化简后的 WS-BPEL 源码的基础上,贷款核准服务(LAS)的消息流被自动建模,并对得到的 BPN 模型进行化简. 经过这些步骤,状态空间由 77 个库所节点和 58 个变迁节点减少为 31 个库所节点和 18 个变迁节点,分析、建模与化简的结果如图 11 所示.

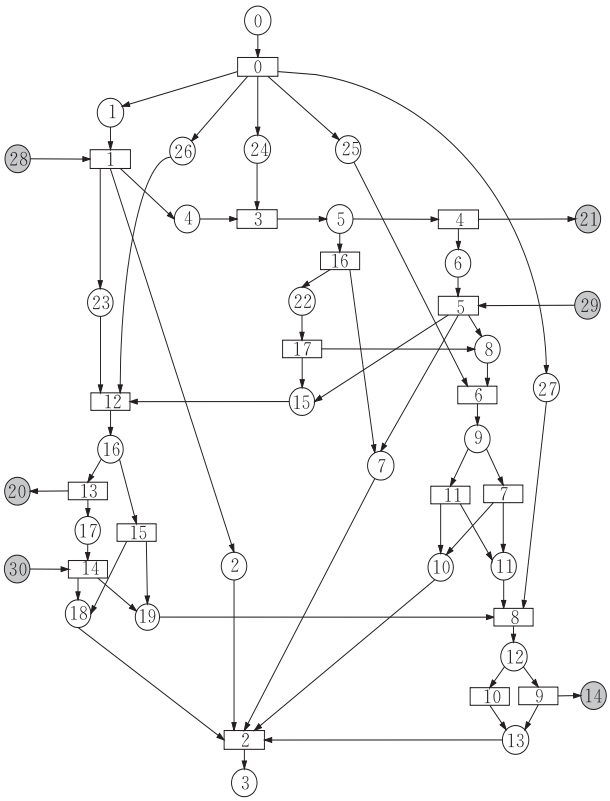

图 11　贷款核准服务:分析、建模与简化的结果

图 11 所示 BPN 模型中有 6 个由阴影标识的库所节点来表示消息交互:receive-to-assess, receive-to-approval, assess-to-setMessage, asses-to-approval, setMessage-to-reply, approval-to-reply. 每个库所节点上都附有一个消息四元组,包含活动的 partnerLink, portType, operation, message 属性. 如下所示:

(1) $f(t_1)=$custormer.loanServicePT.request.creditInformationMessage?,

(2) $f(t_4)=$assessor.riskAssessmentPT.check.creditInformationMessage!,

(3) $f(t_5)=$assessor.riskAssessmentPT.check.riskAssessmentMessage?,

(4) $f(t_9)=$custormer.loanServicePT.request.approvalMessage!,

(5) $f(t_{13})=$approver.loanApprovalPT.approve.creditInformationMessage!,

(6) $f(t_{14})=$approver.loanApprovalPT.approve.approvalMessage?

3.3　面向基于场景规约的 Web 服务消息流验证

经过对 WS-BPEL 消息流的分析、建模与化简之后,我们关注的问题就演变成为验证由 UML 顺序图描述的基于场景规约和由基于 Petri 网模型描

述的 WS-BPEL 消息流之间的一致性，这个问题可以基于文献[4]的工作加以解决. 在文献[4]中，我们给出了 Petri 网模型和消息顺序图(Message Sequence Charts，MSCs)之间的一致性验证算法，那些算法就是针对消息交互顺序的存在/强制一致性的. 因为 UML 顺序图与 MSCs 非常相似，文献[4]的算法略加修改就可用于解决本文所关注的验证问题.

4 原型工具

上文提出的 Web 服务消息流分析、建模和验证方法已经实现成为一个原型工具 SSCC(Service Scenario-based Consistency Checker)①，其图形用户界面如图 12 所示.

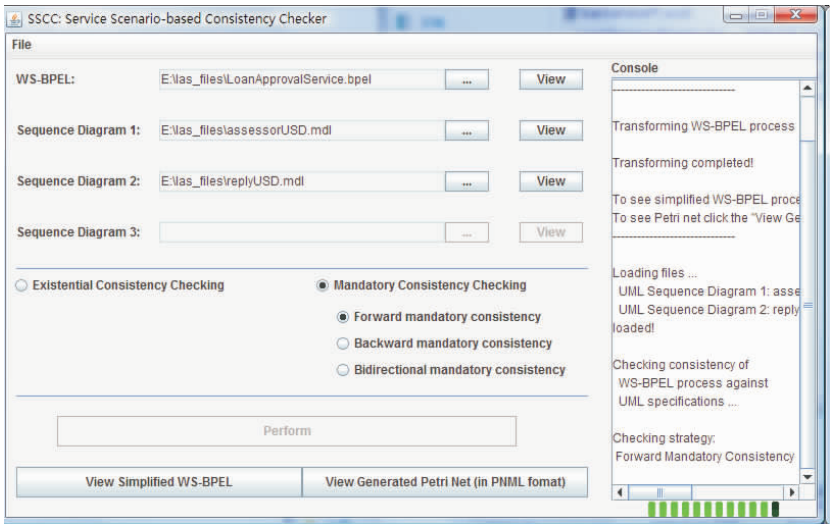

图 12 对服务的基于场景的验证工具：界面

SSCC 的输入为 WS-BPEL 源码文件和 Rational Rose UML 顺序图文件，输出为 *BPN* 模型和验证结果，其中 *BPN* 模型为 PNML(Petri net Modeling Language)格式.

SSCC 的使用步骤如下：首先，用户指定作为工具输入的 WS-BPEL 源码文件和 UML 顺序图文件，并选择所需进行的一致性验证的种类. 接着，SSCC 基于 *BPN* 对 WS-BPEL 消息流进行分析、自动建模并化简. 最后，SSCC 面向基于场景的规约进行指定的一致性验证，给出验证结果并分析 *BPN* 模型中的哪些路径与基于场景的规约相一致. 工具及其使用手册也在 SSCC 网站中同时给出. 用户还能使用 SSCC 查看其所有输入和输出文件.

例 4. 验证：贷款核准服务.

针对贷款核准服务实例，这里给出两组实验：实验 1 以 UML 顺序图 approvalUSD 为规约验证存在一致性(图 1)，其中 approvalUSD 作为 D，即验证在服务运行过程中是否出现由 D 描述的被禁止场景(调用专家核准服务 EAS 并得到响应)；实验 2 以 UML 顺序图 assessorUSD 和 replyUSD 为规约验证向前强制一致性，其中 assessorUSD 作为 D_1，replyUSD 作为 D_2，即验证当 D_1 描述的场景(调用风险评估服务 RAS 并得到响应)出现在服务运行过程中时，D_2 描述的场景(贷款核准服务 LAS 对客户的响应)是否紧跟着出现. 我们采用 SSCC 完成该用例中 *BPN* 模型(如图 11 所示)和上述基于场景的规约(如图 2 所示)之间的一致性验证.

验证结果如下所示，其中路径的表示形式为图 11 中变迁节点的 ID 序列.

实验 1 需要在模型中找到所有出现调用专家核准服务 EAS 并得到响应这一场景的路径，出现这一场景的部分路径如下：

(1) $t_0\hat{}t_1\hat{}t_3\hat{}t_4\hat{}t_5\hat{}t_{12}\hat{}t_{13}\hat{}t_{14}$；

(2) $t_0\hat{}t_1\hat{}t_3\hat{}t_{16}\hat{}t_{17}\hat{}t_{12}\hat{}t_{13}\hat{}t_{14}$；

(3) $t_0\hat{}t_1\hat{}t_3\hat{}t_4\hat{}t_5\hat{}t_6\hat{}t_7\hat{}t_{12}\hat{}t_{13}\hat{}t_{14}$.

实验 2 需要在模型中找到所有出现调用风险评估服务 RAS 并得到响应这一场景的路径，并验证该场景后是否紧跟着出现贷款核准服务 LAS 对客户的响应这一场景，满足向前强制一致性的部分路径如下：

(1) $t_0\hat{}t_1\hat{}t_3\hat{}t_4\hat{}t_5\hat{}t_6\hat{}t_7\hat{}t_{12}\hat{}t_{15}\hat{}t_8\hat{}t_9$；

(2) $t_0\hat{}t_1\hat{}t_3\hat{}t_4\hat{}t_5\hat{}t_6\hat{}t_{11}\hat{}t_{12}\hat{}t_{15}\hat{}t_8\hat{}t_9$；

(3) $t_0\hat{}t_1\hat{}t_3\hat{}t_4\hat{}t_5\hat{}t_{12}\hat{}t_{15}\hat{}t_6\hat{}t_7\hat{}t_8\hat{}t_9$.

① SSCC: Service Scenario-based Consistency Checker. http://seg.nju.edu.cn/SSCC/index.htm.

5　相关工作

WS-BPEL(及其早期版本 BPEL4WS,简称 BPEL)的形式化语义是近年来 Web 服务研究领域的重要研究方向.Fu 等人使用卫式自动机(guarded automata)来为 BPEL 建模[5-6],并实现了支撑工具 WSAT[7],其中卫式自动机被翻译为 Promela 模型,从而可以通过模型检验工具 SPIN 完成 Web 服务组合的验证工作,另外 XML 数据和 XPath 在 Promela 中的表示方式在文献[8]中进行了深入讨论.还有一些工作使用进程代数(process algebra)来形式化 BPEL 并进行验证:Salaün 等人使用 CCS (Calculus of Communicating Systems)为 BPEL 活动建模[9];He 使用 UTP(Unifying Theory of Programming)[10]给出 BPEL 的语义[11];Foster 等人使用 FSP (Finite State Process) 描述 BPEL 流程[12],并实现了工具 LTSA-WS 用于验证由 FSP 和 MSC 描述的 Web 服务组合的一致性[13].

Petri 网作为一种强有力的、针对并行的形式化模型,常被用于描述 BPEL 流程:Schmidt 等人探讨了从 BPEL 到 Petri 网的完全转换[14-15],并实现了工具 BPEL2PN[14];Martens 使用 Petri 网来验证 BPEL 流程的组合[16];Ouyang 等人实现工具 BPEL2PNML 来进行从 BPEL 控制流到 Petri 网的转换[17],该工具的输出为 PNML 格式,因此可被验证工具 WofBPEL 使用[18];Reisig 使用 business process nets(Petri 网的一种)来为 BPEL 建模[19];Verbeek 等人将 BPEL 转换为 workflow nets(Petri 网的一种)[20].除了上述使用经典 Petri 网的工作外,高阶 Petri 网也被用于验证 BPEL 流程的组合,例如有色 Petri 网[21].

本文的工作和上述工作的不同点在于:本文提出的方法面向基于场景规约,专门针对消息流的存在/强制一致性验证问题,因此不需要像上述工作一样给出 WS-BPEL 的全语义,而是使用 *BPN* 来对 WS-BPEL 消息流自动建模,并对这一步骤进行了化简,有效地缩小了模型的状态空间,使得验证算法得以有效的实现.

6　总　结

本文提出了一种面向基于场景规约对 Web 服务消息流进行分析与验证的方法,并实现了支持该方法的原型工具.

本文所关注的基于场景规约只能描述局部行为(没有考虑选择和循环).我们进一步的工作一方面是针对描述全局行为的基于场景规约扩展本文的工作,另一方面是将本文所提方法应用于大规模的实际用例.

参　考　文　献

[1] Object Management Group. Unified modeling language: Superstructure (version 2.0). OMG: Technical Report formal/05-07-04, 2005

[2] Alves A et al. Web services business process execution language (version 2.0). OASIS: Technical Report wsbpel-v2.0-OS, 2007

[3] Peterson J. Petri Nets Theory and the Modeling of Systems. NJ, USA: Prentice Hall, 1981

[4] Li X, Hu J, Bu L, Zhao J, Zheng G. Consistency checking of concurrent models for scenario-based specifications//Proceedings of the 12th International SDL Forum. Grimstad, Norway, 2005: 298-312

[5] Fu X. Formal Specification and Verification of Asynchronously Communicating Web Services [Ph. D. dissertation]. University of California, Santa Barbara, 2004

[6] Fu X, Bultan T, Su J. Analysis of interacting BPEL web services//Proceedings of the 13th International Conference on World Wide Web. New York, NY, USA, 2004: 621-630

[7] Fu X, Bultan T, Su J. WSAT: A tool for formal analysis of web services//Proceedings of the 16th International Conference on Computer Aided Verification. Boston, MA, USA, 2004: 510-514

[8] Fu X, Bultan T, Su J. Model checking XML manipulating software. SIGSOFT Software Engineering Notes, 2004, 29(4): 252-262

[9] Salaun G, Bordeaux L, Schaerf M. Describing and reasoning on Web services using process algebra//Proceedings of the IEEE International Conference on Web Services. San Diego, California, USA, 2004: 43

[10] Hoare C A R, He J. Unifying Theories of Programming. Hertfordshire: Prentice-Hall, 1998

[11] He J. UTP semantics for web services//Proceedings of the 6th International Conference on Integrated Formal Methods. Oxford, UK, 2007: 353-372

[12] Foster H, Uchitel S, Magee J, Kramer J. Model-based verification of Web service compositions//Proceedings of the 18th IEEE International Conference on Automated Software Engineering. Montreal, Canada, 2003: 152-161

[13] Foster H, Uchitel S, Magee J, Kramer J. LTSA-WS: A tool for model-based verification of Web service compositions and choreography//Proceedings of the 28th International Conference on Software Engineering. Shanghai, China, 2006: 771-774

[14] Hinz S, Schmidt K, Stahl C. Transforming BPEL to Petri nets//Proceedings of the 3rd International Conference on Business Process Management. Nancy, France, 2005: 220-235

[15] Schmidt K, Stahl C. A Petri net semantic for BPEL4WS-validation and application//Proceedings of the 11th Workshop on Algorithms and Tools for Petri Nets. Paderborn, Germany, 2004: 1-6

[16] Martens A. Analyzing web service based business processes//Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering. Edinburgh, UK, 2005: 19-33

[17] Ouyang C et al. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007, 67(2/3): 162-198

[18] Ouyang C et al. WofBPEL: A tool for automated analysis of BPEL processes//Proceedings of the 3rd International Conference on Service Oriented Computing. Amsterdam, The Netherlands, 2005: 484-489

[19] Reisig W. Modeling- and analysis techniques for Web services and business processes//Proceedings of the 7th International Conference on Formal Methods for Open Object-Based Distributed Systems. Athens, Greece, 2005: 243-258

[20] Verbeek E, Aalst W M P. Analyzing BPEL processes using Petri nets//Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management. Miami, FL, USA, 2005: 59-78

[21] Yi X, Kochut K J. A CP-nets-based design and verification framework for web services composition//Proceedings of the IEEE International Conference on Web Services. San Diego, California, USA, 2004: 756-760

YANG Lu, born in 1982, Ph. D. candidate. Her research interests focus on software engineering and software verification.

LIU Xi, born in 1984, Ph. D. candidate. His research interests focus on software engineering.

WANG Lin-Zhang, born in 1973, Ph. D.. His research interests focus on model driven software testing and verification, automatic software testing.

CHEN Xin, born in 1975, Ph. D.. His research interests focus on software engineering, object-oriented programming, component and formal methods.

LI Xuan-Dong, born in 1963, Ph. D.. His research interests focus on software modeling and analysis, software testing and verification.

Background

The Web Services technologies play an important role for the development and execution of business processes which are distributed over the network. The verification for Web Services behavior and composition becomes a problem that attracts more and more attention underlying Web Services. The Web Services Business Process Execution Language (WS-BPEL), which is a de facto standard in Web Services standard suite, can be used for behavior description and composition of Web Services. So in the past years many research groups discussed the verification for WS-BPEL (and the old version BPEL4WS), by using formal methods such as guarded automata, process algebra (e. g. CCS, UTP and FSP) and Petri net.

Be different with the above works, the group focuses on the temporal orders of message exchanges among processes. They use scenario-based specifications to offer an intuitive and visual way to describe requirements, and verify the consistency between WS-BPEL message flows and scenario-based specifications.

This work aims to propose a scenario-based analysis and verification approach for Web Services message flows. In the approach UML Sequence Diagrams are used to specify scenario-based specifications, and WS-BPEL is used to describe Web Service designs. According to a given scenario-based specification, the authors firstly analyze a WS-BPEL specification under verification and automatically extract its message flow model expressed by a Petri net. At the same time, the authors do the simplification for both the WS-BPEL source code and the Petri net model to reduce the state space for efficient verification, i. e. removing the activities and process elements which are not concerned with the verification against the scenario-based specification. Finally, the authors verify the consistency between the WS-BPEL message flow and the scenario-based specification (existential/mandatory consistency of message sequence) by traversing the Petri net model.

This project is supported by the National Natural Science Foundation of China under grant No. 60673125, and No. 90818022; the National High Technology Research and Development Program (863 program) of China under grant No. 2009AA01Z48; the Natural Science Foundation of Jiangsu Province under grant No. BK2007714.