第35卷 第8期 北京工业大学学报 Vol.35 No.8
2009年 8月 JOURNAL OF BEIJING UNIVERSITY OF TECHNOLOGY Aug. 2009

Web服务编排的并发事务逻辑模型

王 勇[1], 姜正涛[2], 侯亚荣[1], 方 娟[1], 毛国君[1]
(1.北京工业大学 计算机学院,北京 100124; 2.中国传媒大学 计算机学院,北京 100024)

摘 要: 为使得组织之间的服务编排建立在严格的形式化基础之上,基于并发事物逻辑建立了服务编排的模型,给出了从服务编排描述语言WS-CDL到并发事物逻辑的转换规则. 服务编排的并发事务逻辑模型建立在严格的形式化基础上,便于进行服务编排相关性质的验证. 最后通过一个实际的服务编排在并发事务逻辑中建模的例子,说明了模型建立的可行性.

关键词: Web服务; 并发事务逻辑; 服务编排
中图分类号: TP 393.09 **文献标志码**: A **文章编号**: 0254-0037(2009)08-1132-06

近年来,Web服务或者电子服务(下文简称服务)作为实现跨组织协作的一种体系结构和实现手段得到了业界和学术界的广泛关注,服务技术为网络环境下的资源共享和集成提供了基础. 在Web服务的使用范例中,业务应用提供的功能被包装为一个服务,该服务作为一个软件组件可以被应用程序或者其他服务通过一个开放的互联网协议栈访问,如HTML、XML、WSDL、SOAP、UDDI. 随着服务技术的快速发展,服务由最初的简单的、无状态的形式发展为复杂的、组合的形式;由研究、实验性的形式发展为成熟的、商业化的形式;由组织内的服务应用环境发展为跨越组织域的服务组合环境. 服务只有通过组合成为更大粒度的服务,才能充分发挥服务的潜力和作用[1].

服务组合包含了2个方面[1-3]:一个方面称为服务编制(web service orchestration,简称为WSO),侧重组织内部的可执行业务流程,该流程可以在消息级别上同组织内部或者组织外部的服务进行交互,包含了组织的业务流程和任务的执行次序,可以通过一个过程模型来刻画;另一个方面称为服务编排(web service choreography,简称为WSC),侧重组织之间交互的可见行为,即各个交互的服务之间公共消息的交互次序. 服务编排侧重组织之间交互的可见行为,面临的是一个广泛分布、动态、自治、异构的网络环境,保障服务编排的正确性以及相关特性的验证问题显得尤为重要. 形式化方法是一种有效的解决方法,通过严格的形式化模型及其相应的验证工具可以确保服务编排的正确性.

2001—2003年,业界和研究人员研究重点放在了制定服务编排描述语言上. 其中比较有影响力的有Sun公司联合其他公司先后推出WSCI语言[4]和W3C的WS-CDL语言[5],其中WS-CDL V1.0版本已经成为W3C的候选推荐标准. WSCI和WS-CDL没有建立在严格的形式化模型的基础上,研究人员开始研究服务编排的形式化模型,其中相当一部分的工作在为服务编排描述语言提供形式化基础[6-8],这些形式化的工作具有基于转移系统(如Petri网、进程代数等)的语义,为描述语言提供了形式化语义和验证能力.

1 服务编排描述语言WS-CDL概述

WS-CDL是一种基于XML的语言,用一种全局的观点来描述合作者之间的对等协作关系,这种协作关系表现为经过认同的执行顺序和规则的集合. WS-CDL由以下元素构成.

1) 角色类型、关系类型和参与者类型用来定义协作者之间的关系:在编排中信息总是在信任域内或

收稿日期: 2007-10-11.
基金项目: 国家"九七三"重点基础研究发展规划项目基金资助项目(2007CB311100);北京工业大学博上科研启动基金资助项目(52007013200704).
作者简介: 王 勇(1974—),男,山东临朐人,讲师.

信任域之间的参与者之间进行交换，角色之间的交互表现在参与者上，并由关系进行约束. 参与者类型把相同逻辑实体实现的可见行为进行了分组；而角色类型列举了一个参与者类型可能表现的所有潜在的可见行为；关系类型表明了使得协作成功的彼此之间需要承担的义务；

2) 信息类型、变量和标识用来定义协作中的数据：变量描述了可见行为中的信息交换的内容；标识是变量中一部分的别名；而信息类型用来定义变量或者标识所代表的信息类型；

3) 编排用来定义交互的参与者之间的协作：编排生命线用来表达一个协作的进程；异常块定义了编排发生异常时发生的行为；编排终结块用来确认一个已经结束的编排效果；

4) 通道类型用来定义参与者之间的协作点；

5) 工作单元用来定义一个编排获得进展必须遵循的约束；

6) 活动和次序结构用来定义一个编排需要执行的行为.

WS-CDL 1.0 版本已经成为 W3C 的候选推荐标准.

2 并发事务逻辑概述

并发事务逻辑(concurrent transaction logic，简称 CTR)是对经典谓词逻辑的扩展，增加了对于状态转移的支持.

同经典逻辑一样，形如 $p(t_1,\cdots,t_n)$ 的表达式称为原子公式，这里 p 是谓词符号，而 t_i 为功能项. 更为复杂的公式可以通过连接词和量词构造，除去经典逻辑中的 $\exists$，$\forall$，$\neg$，$\wedge$，$\vee$ 等连接词和量词以外，CTR 增加了 2 个连接词：$\otimes$表示顺序连接，$|$表示并行连接. 此外，并发事务逻辑还增加了 2 个模态运算符：$\diamond$表示执行的可能性，e 表示隔离执行.

CTR 的语义是数据库状态和路径(状态的有限序列)的集合. CTR 公式的真值决定于某个路径而不是某个状态，也就是说 CTR 公式 ϕ 在某个路径 $\langle s_1,\cdots,s_n\rangle$ 上为真，意味着 ϕ 从 s_1 状态开始执行，依次经过状态 $s_2,\cdots,s_i,\cdots$，最终在状态 s_n 运行终止. 基于这样的语义，给出 CTR 的连接词的含义.

1) $\phi\wedge\psi$：ϕ 与 ψ 沿着相同的路径执行，可以用来对服务编排中的约束进行建模，如 ϕ 为某个活动，而 ψ 为伴随活动 ϕ 的约束；

2) $\phi\vee\psi$：执行 ϕ 或者执行 ψ，可以用来对服务编排中的选择执行结构建模；

3) $\neg\phi$：ϕ 不可能存在合法的执行；

4) $\phi\otimes\psi$：执行完 ϕ 之后执行 ψ，可以用来对服务编排中的顺序执行结构建模；

5) $\phi|\psi$：ϕ 和 ψ 并行执行，可以用来对服务编排中的并行执行结构建模；

6) $\diamond\phi$：检查在当前状态 ϕ 是否是可执行的；

7) $e\phi$：隔离执行 ϕ.

3 用并发事务逻辑对服务编排建模

1) 原子活动与执行约束

WS-CDL 的 6 种类型的原子活动(interaction 活动、perform 活动、assign 活动、silentAction 活动、noAction 活动以及 finalize 活动)在 CTR 中用原子公式表示. 定义的 perform 类型的原子活动为：

```
〈perform name="creditForA" choreographyName="CoordCreditAuthorization"
        choreographyInstanceId="'creditForA'〉
〈/perform〉
```

在 CTR 中用一个原子公式表示，记为 creditForA.

其他类型的原子公式的处理与上面 Invoke 活动的处理类似.

活动定义中出现的执行条件等执行约束的处理与原子活动的处理相同，把它们转换为 CTR 中的原子公式.

2) sequence 活动

sequence 活动表示所包含的子活动的顺序执行，在 CTR 中对于顺序执行采用连接词 ⊗，定义的 sequence 活动为：

```
〈sequence〉
    〈interaction name="processGoodCredit"
        channelVariable="goodCredit-channel" operation="doCredit"〉
      …
    〈/interaction〉
    ……
    〈interaction name="processBadCredit"
        channelVariable="badCredit-channel" operation="doBadCredit"〉
      …
    〈/interaction〉
〈/sequence〉
```

在 CTR 中表示为 processGoodCredit⊗… ⊗processBadCredit.

3) choice 活动

choice 活动表示所包含的子活动的选择执行，在 CTR 中对于选择执行采用连接词 ∨，定义的 choice 活动为：

```
〈choice〉
    〈interaction name="processGoodCredit"
        channelVariable="goodCredit-channel" operation="doCredit"〉
      …
    〈/interaction〉
    ……
    〈interaction name="processBadCredit"
        channelVariable="badCredit-channel" operation="doBadCredit"〉
      …
    〈/interaction〉
〈/choice〉
```

在 CTR 中表示为 processGoodCredit ∨ … ∨ processBadCredit.

4) parallel 活动

parallel 活动表示所包含的活动的并行执行，在 CTR 中对于并行执行采用连接词 |. 定义的 parallel 活动为：

```
〈parallel〉
    〈interaction name="processGoodCredit"
        channelVariable="goodCredit-channel" operation="doCredit"〉
      …
    〈/interaction〉
    ……
    〈interaction name="processBC" channelVariable="badCredit-c" operation="doBCt"〉
        … …
    〈/interaction〉
〈/parallel〉
```

在 CTR 中表示为 processGoodCredit | … | processBadCredit.

5) workunit 活动

workunit 描述了编排获得进展必须遵循的约束，可以表示包含活动的选择执行或者循环执行，由 guard 属性和 repeat 属性来执行. 定义 workunit 活动为：

```
〈workunit  name="StockCheck"
    guard="cdl:getVariable('StockQuantity','','/Product/Qty', 'tns:Retailer')〉10)"
    block="false" 〉
    〈interaction name="badCreditInteraction"  channelVariable="tns:CreditResponder"
                    operation="creditDenied"〉
          〈participate relationshipType="CreditReqCreditResp"
              fromRoleTypeRef="tns:Responder"
              toRoleTypeRef="tns:CreditRequestor"/〉
    〈/interaction〉
〈/workunit〉
```

在 CTR 中的表示为(guard⊗badCreditInteraction) ∨ (¬guard).

对于循环执行的 workunit 活动：

```
〈workunit  name="POProcess"
    repeat="cdl:isVariableAvailable('POAcknowledgement','','','tns:Customer')"
    block="true"〉
    〈interaction name="drI" channelVariable="tns:CR" operation="drawDown"〉
    〈/interaction〉
〈/workunit〉
```

在 CTR 中的表示为

POProcess←((repeat⊗drawdownInteraction⊗POProcess) ∨ (¬repeat)).

通过上面的转换关系，WS-CDL 中的结构可以转换为 CTR 中的公式，这样就可以利用 CTR 的性质进行相关的推理和验证.

4　一个服务编排的例子

在 WS-CDL 1.0 规范中给出了一个服务编排的例子，大致结构为：

```
〈choreography name="CreditDecider"〉
    〈parallel〉
        〈perform name="creditForA"
            choreographyName="CoordinatedCreditAuthorization"
            choreographyInstanceId="'creditForA'"〉
            〈! -- bind such that this does the business for A --〉
        〈/perform〉
        〈perform name="creditForB"
            choreographyName="CoordinatedCreditAuthorization"
            choreographyInstanceId="'creditForB'"〉
            〈! -- bind such that this does the business for B --〉
        〈/perform〉
    〈/parallel〉
    〈workunit name="chooseA"
        guard="cdl:getVariable('Chosen','','','Broker') = 'A'" 〉
```

```
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForA'"
                finalizerName="drawDown"/〉
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForB'"
                finalizerName="replenish"/〉
    〈/workunit〉
    〈workunit name="chooseB"
        guard="cdl:getVariable('Chosen','','','Broker')='B'" 〉
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForB'"
                finalizerName="drawDown"/〉
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForA'"
                finalizerName="replenish"/〉
    〈/workunit〉
    〈workunit name="chooseNeither"
        guard="cdl:getVariable('Chosen','','','Broker')='0'"〉
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForA'"
                finalizerName="replenish"/〉
        〈finalize choreographyName="CoordinatedCreditAuthorization"
                choreographyInstanceId="'creditForB'"
                finalizerName="replenish"/〉
    〈/workunit〉
〈/choreography〉
```

整个服务编排在CTR中的表示为：

$(creditForA|creditForB)\otimes(chooseA \vee chooseB \vee chooseNeither)$.

而chooseA可以表示为：

$(guard\otimes drawDown\otimes replenish) \vee (\neg guard)$.

chooseB, chooseNeither的进一步表示与chooseA类似.

5 结束语

服务编排面临广泛分布、动态、自治、异构的网络环境，设计阶段的正确性等性质的验证是一个重要环节. 为了便于进行相关性质的验证，基于并发事务逻辑作为形式化手段，建立了服务编排的形式化模型，并给出了从服务编排描述语言WS-CDL到并发事务逻辑的转换规则. 并发事务逻辑是一个有效的形式化工具，服务编排中的活动等控制结构在并发事务逻辑中都有直观的表示，可以利用并发事务逻辑的推理机制进行并发事务逻辑相关性质的验证.

参考文献：

[1] OASIS Web Services Business Process Execution Language (WSBPEL) TC. Business process execution language for web services[S/OL]. America: OASIS, 2006[2007-10-01]. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[2] PELZ C. Web services orchestration and choreography[J]. IEEE Computer, 2003, 36(8): 46-52.

[3] PELZ C. Web services orchestration and choreography, a look at WSCI and WS-BPEL[S/OL]. America: HP, 2003[2007-10-01]. http://devresource.hp.com/drc/technical_articles/wsOrchestration.pdf.

[4] W3C. WSCI: web service choreography interface (WSCI) 1.0[S/OL]. America: W3C, 2002[2007-10-01]. http://www.w3.org/TR/wsci.

[5] W3C. WS-CDL: web services choreography description language version 1.0[S/OL]. America: W3C, 2006[2007-10-01]. http://www.w3.org/TR/ws-cdl-10/.

[6] YANG H, ZHAO X P, QIU Z Y, et al. A formal model for web service choreography description language[C]// Proc of the IEEE International Conference on Web Services 2006. Chicago, USA: IEEE, 2006: 192-198.

[7] BROGI A, CANAL C, PIMENTEL E, et al. Formalizing web services choreographies[C]// Proc of 1st International Workshop on Web Services and Formal Methods. Pisa Haly: IEEE Computer Society Press, 2004: 181-188.

[8] BUSI N, GORRIERI R, GUIDI C, et al. Towards a formal framework for Choreography[R/OL]. America: IEEE Computer Society Press, 2003[2007-10-01]. http://www.cs.unibo.it/lucchi/papers/dmc.pdf.

Web Service Choreography Model Based on Concurrent Transaction Logic

WANG Yong[1], JIANG Zheng-tao[2], HOU Ya-rong[1], FANG Juan[2], MAO Guo-jun[2]

(1. School of Computer Science & Engineering, Beijing University of Technology, 100124 Beijing, China;
2. School of Computer Science, Communication University of China, 100024 Beijing, China)

Abstract: Web service choreography among organizations should be established on the basis of formal model. This paper establishes the model of web service choreography based on concurrent transaction logic and the translation rules from WS-CDL to concurrent transaction logic are given. The concurrent transaction model of web service choreography is established on the basis of formal methods and is easy to verify properties of web service choreography. Finally, an actual web service choreography model based on concurrent transaction logic is illustrated.

Key words: web services; concurrent transaction logic; service choreography

(责任编辑　郑筱梅)