BPEL应用程序验证模型研究

闻 晓 张为群 杨 阳 黄 娟
(西南大学计算机与信息科学学院 软件学院 重庆 400715)

摘 要 在Web服务应用中,BPEL是一种基于流程的描述业务行为的语言。为了验证以BPEL构建的应用程序是否满足某些性质,提出了BVM模型来表达应用程序的语义,然后运用模型检测方法进行形式化验证。实验结果表明,以上方法在设计阶段能有效地发现并排除错误。

关键词 Web服务组合,BPEL,有限自动机,形式化验证
中图法分类号 TP311 **文献标识码** A

Verification Model for BPEL Application

WEN Xiao ZHANG Wei-qun YANG Yang HUANG Juan
(College of Computer and Information Science & College of Software, South West University, Chongqing 400715, China)

Abstract Business Process Execution Language (BPEL) is a flow-based language for specifying business process behavior in Web service applications. In order to verify whether a given BPEL-based application conforms to some characteristics, this paper proposed the BVM model to capture the semantic of the application. Then model checking methods were used to verify the system in a formalizing way. The experimental results show that the proposed method can detect and remove bugs effectively during design phase.

Keywords Web service composition, BPEL, Finite automaton, Formal verification

随着Web服务的深入发展,越来越多的基于Web服务的业务需要组合后以流程的方式来执行。BPEL[1](Business Process Execution Language)作为涌现出来的业务执行描述语言的一种,为基于Web服务组合的程序提供了详尽地描述通信与协同行为的能力。尽管如此,BPEL程序代码往往冗长而复杂,面临死锁等与运行相关的问题。此外,BPEL不是一种可执行的程序语言,仍然存在二义性、不一致性和不完备性。于是尽力排除以上缺陷,并利用形式化方法验证BPEL应用程序成为研究的热点。

基于模型检测[2]的方法是验证BPEL应用程序的途径之一。Jana Keohler[3]提出了对面向业务专家的流程模型进行转换的必要性。Shin Nakajima[4]将BPEL程序转换为有限自动机模型,在SPIN中验证了BPEL程序中流程的可达性以及程序运行中应当满足的性质。Zheng Yongyan[5]等人提出的WSA模型捕获了BPEL中的操作语义,给出了生成测试用例的框架。由此可见,以上研究都只针对BPEL的一个子集,未能表达BPEL的全部语义。

补偿是BPEL语义中的重要组成部分,是指业务流程中因某个Web服务终止而施加到其它相关联Web服务的行为。补偿在实际流程中体现了系统的可靠性。本文提出了一种验证BPEL应用程序的模型BVM(BPEL Verification Model),旨在清晰地表达包括拥有补偿行为的BPEL应用程序,并进一步用模型检测的方法验证。这样就扩充了验证BPEL应用程序的范围。

1 基于扩展的有限自动机的模型BVM

下文首先定义将BPEL转换后的基于扩展的有限自动机的模型以及补偿部分在模型中的语义,然后简要地叙述转换的过程。

1.1 扩展的有限自动机模型——BVM

定义1 $BVM=\{Q,\Sigma,\Delta,\delta,q_0,F,V,E\}$。

其中Q表示状态集合,$q_0\in Q$是初始状态,$F\subseteq Q$是终止状态集合。Σ是有穷符号集合。Δ是输出符号集合。V是BPEL中变量的集合。$E::=\Sigma^*\mid\wp(V)\mid\Delta^*$($\wp(V)$表示$V$的幂集)。$\delta$是转移函数,它将$Q\times E$映射到$Q$。

定义2 $V=V_B\cup V_L\cup V_P\cup V_H$。

V_B表示BPEL中由〈variables〉定义的变量的集合。V_L是布尔型变量的集合,表示〈flow〉中的〈link〉是否存在。V_P是布尔型变量的集合,表示switch和while语句中的条件表达式的真值。如果S1,S2是BPEL中的两个作用域,S1调用S2,则S1和S2之间存在父子关系,且S1是S2的父亲。V_H是布尔型变量的集合,表示BPEL中作用域(Scope)之间是否存在父子关系。

Web服务组合中的流程节点之间的关系在BVM中用状

到稿日期:2008-11-10 本文受到重庆市自然科学基金重点项目"软件测试技术和方法研究"(CSTC,2006BA2003)支持。
闻 晓(1980-),男,工程师,硕士研究生,研究方向为软件工程、Web软件测试,E-mail:birdtalk@163.com;张为群(1950-),男,教授,硕士生导师,CCF会员,主要研究领域为形式化软件工程、软件测试;杨 阳(1982-),男,工程师,硕士研究生,研究方向为传感器网络、分布式控制系统、计算机通信;黄 娟(1983-),女,硕士研究生,研究方向为软件工程、Web软件测试。

态之间的转移关系来表示。E 代表了转移关系中两类重要动作，即消息传递和变量赋值。前者表示流程中的消息被成功地传送，后者表示前者触发的相关变量的一系列赋值动作。δ 函数包含的转移关系 $Q\times E\times Q$ 表示由状态 $q\in Q$ 转移到状态 $q'\in Q$，且 E 代表的动作同时发生。

1.2 补偿在 BVM 中的体现

BPEL 是以 XML 为基础的基于流程的语言。作用域是 BPEL 的基本组成部分，下列代码对此做了举例说明。

```
〈process〉
    〈scope〉
        〈faultHandlers〉
        Activities
        〈/faultHandlers〉
        〈compensationHandlers〉
        Activities
        〈/compensationHandlers〉
        Activities
    〈/scope〉
〈/process〉
```

作用域中的活动(Activities)可以划分为功能不同的两个部分：基本活动和结构活动。基本活动是描述流程中 Web 服务元动作，不能再做进一步的拆分，它们包括 receive，reply，assign，invoke，throw，empty 和 wait。结构活动是描述流程中 Web 服务组合的顺序和转移，包括 sequence，switch，while，flow 和 pick。宏观上来看，BPEL 是在结构活动的组织下形成具有一定优先顺序的作用域。微观上来看，BPEL 是由基本活动组成的元动作序列。

BPEL 中的补偿行为由多个活动组成，旨在消除流程中某一个活动因故放弃而带来的后果。BPEL 允许分别在〈faulthanders〉和〈compensationhandlers〉中调用补偿活动。一般地，设 *Scope* 代表 BPEL 中的作用域，则

$$Scope=CH\cup FH\cup Act$$

其中 CH 代表〈compensationhandlers〉的代码，FH 代表〈faulthanders〉的代码，Act 代表余下的代码，如图 1 所示。当触发条件满足时，CH 和 FH 部分就分别执行补偿行为。有时候，补偿行为可能还需要引用其它作用域，图中假设这一过程涉及的作用域只有两个。由此可见，BPEL 中的补偿行为由作用域之间的调用和作用域内部的活动组成。

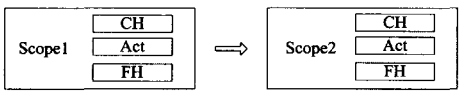

图 1 BPEL 中补偿行为调用结构图

如前所述，$V=V_B\cup V_L\cup V_P\cup V_H$，得

$$E::=\sum^*|\wp(V_P\cup V_L\cup V_H)|\wp(V_B)|\Delta^*$$

其中 $\wp(V_P\cup V_L\cup V_H)$ 是转移条件 G，$\wp(V_B)$ 表示可能发生的赋值操作。于是在转移函数 δ 中，如果 G 被满足，则 $q\to q'(q,q'\in Q)$，亦即在状态 $q\in Q$ 下执行赋值操作后转移到状态 $q'\in Q$。以上的转移条件 G 描述了 BPEL 中的下列活动关系：

1)基本活动的转移条件始终为真，直接进入下一个 Web 服务；

2)结构活动的转移条件根据语句的条件表达式真值决定；

3)作用域之间的转移条件依赖父子关系。如果所有的子作用域都执行完毕，则父作用域才开始执行自己的活动。

BVM 在表示补偿行为时，首先根据父子关系确定作用域之间的调用顺序，然后在作用域内部根据转移条件的 a,b 来完成操作。

图 2 中的 BVM 表示了一个包含补偿的作用域。路径 $s_0\to s_1\to s_2\to s_7$ 表示作用域正常执行。路径 $s_0\to s_1\to s_4\to s_5\to s_6$ 表示作用域因异常发生而执行的补偿行为。路径 $s_0\to s_1\to s_2\to s_3\to s_7$ 表示作用域因某个流程被撤销而执行的补偿行为。

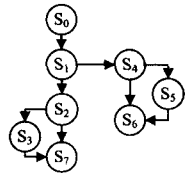

图 2 表示包含补偿作用域的 BVM

1.3 BPEL 转换为 BVM 的算法

相比而言，BVM 比 BPEL 能更容易地描述复杂业务结构，也为后面的模型检测提供了便利。BVM 是按照 BPEL 的基本结构自上而下转换得来。转换算法如下：

1)若 BPEL 中有 n 个作用域，设 $S=\bigcup_{i=0}^{n}S_i$ 是 BPEL 中作用域的集合。

2)$i=i+1$。

3)将 S_i 中的每个活动用 BVM 的状态 $q\in Q$ 表示。

4)对于 S_i 的基本活动，它们的行为分为消息传递和变量赋值，则由 BVM 定义中的 $E\subseteq\sum^*\cup\wp(V)\cup\Delta^*$ 来表示，并得到 $\delta\subseteq Q\times E\times Q$。

5)对于 S_i 的结构活动，它们的行为是根据 BPEL 中的条件表达式的真值表来对基本活动进行调用。于是将条件表达式改写成为 $E\subseteq\sum^*\cup\wp(V_P\cup V_L\cup V_H)\cup(\wp(V_B)\cup\Delta^*)$ 中的转移条件 G，剩下的转换同 4)。

6)若 $i<n+1$，则回到第二步。

7)结束。

BPEL 中的操作语义包含在作用域的活动当中。从以上算法可以看出，BVM 对 BPEL 中每一个作用域的活动都做了分析。于是，该算法转换得出的 BVM 没有改变原先 BPEL 中活动的行为。

2 模型检测 BPEL 的框架

模型检测是一种验证计算机系统或程序是否满足需求的形式化方法。该方法的步骤如下：

1)利用适当的模型 M 表示系统(在模型检测器中 M 是某一类别的语言)。

2)利用公式 ϕ 表示描述待验证性质的规范。

3)运行模型检测器判断计算模型 M 是否满足 ϕ[6]。

研究表明，模型检测特别适合验证交互的进程中与并行通信相关的性质。在 Web 服务方面，这些性质主要包括安全和活性两个方面[7]。前者判定系统是否实现了预定功能，或者没有意外的发生。后者判定系统的运行是否进入不可结束的状态。

下面简要说明 BPEL 应用程序中待验证的性质。

2.1 待验证的性质的表示

安全性质偏重于判断系统未来是否在某种情况下达到某个状态，而活性性质偏重于验证未来任何情况下都能达到某个状态。表示这些性质的公式的真与假不是固定不变的。时态逻辑可以用来表示这些待验证的性质。时态逻辑包含若干状态，公式的真假随着状态变化而变化。线性时间逻辑把未来看成状态序列，逻辑公式的满足性问题需要判断从某个状态出发的所有序列。而分支时间逻辑把未来看成分支树，能够表达一条序列是否存在的断言。

2.2 实验验证

为了验证 BVM 的可行性，本文进行了下列实验以及结果分析。图 3 是含有补偿行为的一个 BPEL 的 BVM。实验中使用的模型检测器是 NuSMV[8]。NuSMV 以描述系统的模型和时态逻辑组成的代码作为输入。若性质被满足，它产生输出结果"真"，否则产生一个不被满足的反例。实验中的模型描述了补偿行为从开始执行到结束的过程。

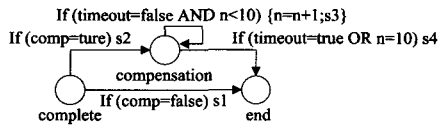

图 3 实验中的 BVM

实验的第一步是用 NuSMV 规定的语言来表达 BVM 和相关性质。部分代码如下所示，主要定义了模型的基本结构和待验证的性质。

```
MODULE main
VAR
state:{complete,compensation,end };
n:0..10;
timeout:Boolean;
comp:Boolean;
behavior:{s1,s2,s3,s4};
ASSIGN
init(state):= complete;
init(n):= 0;
init(behavior):=s1;
SPEC
AG(state=compensation->EF(state=end))
```

模型中含有 3 个状态 complete，compensation 和 end，共同组成 BVM 中的 S。$timeout, comp \in V_p$，分别表示网络等待超时和是否存在补偿行为。$n \in V_B$，表示补偿调用者的计数器(实验中假定上限值为 10)。

实验的第二步是运行 NuSMV。结果显示模型满足图 3 中的性质。如果更换将要检测的性质为 AG (state = complete -> EF (state=compensation))，则 NuSMV 将返回以下信息：

```
-- specification AG (state = complete -> EF state = compensation) is false
-- as demonstrated by the following execution sequence
Trace Description:CTL Counterexample
Trace Type:Counterexample
-> State:1.1 <-
  state = complete
  n = 0
  timeout = 0
  comp = 0
  behavior = s1
```

从以上实验可以看出，BVM 在 NuSMV 中能够检测出性质是否满足，并能给出不满足时存在的反例。

结束语 本文讨论了将 BPEL 应用程序转换为 BVM 的过程，对下一步的验证过程做了初步讨论。BPEL 应用程序的结构复杂，需要一种途径来理清逻辑框架。此外，BPEL 主要面向需求领域的业务专家，不能很好地应用到计算机自动化验证方面。BVM 在以上两个方面作了有益的尝试。将来的工作是将该 BVM 自动转换为模型检测器的输入语言(描述待验证系统的语言)，再运行检测器验证是否满足特定的性质。在此过程中，状态爆炸问题可能使得验证过程不能结束，于是有待于对 BVM 进一步优化。

参 考 文 献

[1] Curbera F, et al. Business Process Execution Language for Web Services. Version 1.1. BEA, IBM, Microsoft, SAP AG and Siebel Systems. May 2003

[2] Clarke E, Grumberg O, Peled D. Model Checking. MIT Press, December 1999

[3] Koehler J, Tirenni G, Kumaran S. From business process model to consistent implementation: A case for formal verification methods// Proc. 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC). 2002:96-106

[4] Nakajima S. Model-checking Behavioral Specification of BPEL Applications. Electronic Notes in Theoretical Computer Science, 2006,151(2):89-105

[5] Zheng Y, Krause P. An automatic test generation framework for BPEL web services. Technical Report. England: University of Surrey, 2006

[6] Huth M, Ryan M. 面向计算机科学的数理逻辑：系统建模与推理[M]. 第二版. 北京：机械工业出版社，Spinger，2007：115

[7] Sharygina N, Kröning D. Model Checking with Abstraction for Web Services. Test and Analysis of Web Services, Spinger, 2007:121-145

[8] Cimatti A, Clarke E, Giunchiglia F, et al. NUSMV: A New Symbolic Model Verifier // Proceedings Eleventh Conference on Computer-Aided Verification (CAV'99), number 1633 in LNCS. Springer, 1999:495-499

(上接第 154 页)

[7] Roshan K, Sandhu R. Conceptual Foundations for a Model of Task-based Authorizations [C] // IEEE Computer Security Foundations Workshop. 1994

[8] 葛声，孙瑛霖，杜宗霞. 基于角色的协作关系建模研究[J]. 计算机机工程与应用，2003(3)：14-19

[9] Crook R, Ince D, Nuseibeh B. Modelling access policies using roles in requirements engineering[J]. Information and software technology, 2003(45):979-991

[10] Liu Kecheng, Sun Lily, Dix A, et al. Norm Based Agency for Designing Collaborative Systems[J]. Information Systems Journal, 2001(11):229-247

[11] Kuan Peipei, Rarunasekera S, Sterling L. Improving Goal and Role Oriented Analysis for Agent Based Systems[C]//Proceedings of the 2005 Australian software engineering conference. 2005