基于赋时有色 Petri 网的 Web 组合服务建模

王玉英[1,2] 陈 平[1]

(西安电子科技大学软件工程研究所 西安 710071)[1] (西安建筑科技大学理学院 西安 710055)[2]

摘 要 面向 Web 服务的业务流程执行语言 BPEL 本身缺乏健壮的语义,构建 Web 组合服务又是一种易于出错的任务。基于有色赋时 Petri 网,提出了从 BPEL 描述的 Web 组合服务流程到有色赋时 Petri 网模型的转换方法。在转换过程中考虑了 BPEL 活动的执行方式和执行环境,如时间、环境属性等,给出了更为精确的模型,为更好地使用工具和现有方法验证和测试 Web 组合服务奠定了基础。并给出了这种转换方法的应用实例。

关键词 组合服务,BPEL,模型,流程验证

中图法分类号 TP311 **文献标识码** A

Models of Web Services Composition Based on Timed Color Petri Nets

WANG Yu-ying[1,2] CHEN Ping[1]

(Software Engineering Institute, Xidian University, Xi'an 710071, China)[1]

(School of Science, Xi'an Technology and Architecture University, Xi'an 710055, China)[2]

Abstract BPEL is often used to describe the composition of Web Services, but it is lack of sound formal semantic. Web services were prone errors. Based on Timed Color Petri Net, transitions from Web Services which was described using BPEL to Timed Color Petri Net models were proposed, while BPEL activities execute ways and environments were considered. The models we get are more exacter and can be used to verify and test Web Service. An instance of this transition were given.

Keywords Web Service, BPEL, Model, Process verify

面向 Web 服务的业务流程执行语言 WSBPEL(也称为 BPEL 或 BPEL4WS,下面简称为 BPEL)是一种使用 Web 服务定义和执行业务流程的语言。从 2002 年 6 月其首版发布以来,许多大公司加入了 BPEL 技术委员会,包括 Adobe, Hewlett-Packard, NEC, Oracle 和 Sun 公司,研究人员也对 BPEL 做了很多研究工作[1]。由于存在并发、补偿处理、死路径删除等复杂特征[2],BPEL 流程容易出错。另外,BPEL 流程中可以通过调用的方式使用 Web 服务的一些有价值的资源。因此,需要保证 BPEL 流程的正确性。测试是发现不正确行为的一种有效方法,使用验证技术和工具探查不正确行为也是一种有益的方法。目前人们开发了一些不同的验证技术和工具。常用的验证技术都是基于验证对象的模型基础上的。本文主要讨论 BPEL 建模技术。

目前常用的 BPEL 建模技术主要有:基于 Petri 网的建模(包括有色 Petri 网)、进程代数、抽象状态机、自动机等。本文主要研究基于赋时有色 Petri 网的 Web 组合服务建模。

1 研究现状

Christian Stahl[1]最早在他的博士论文中提出了用 Petri 网描述 BPEL 的语义,接着有很多文献讨论 BPEL 到 Petri 网或者有色网(CP-nets)的映射问题。但是这些讨论基本上都局限在 BPEL 的结构上[2-5]。文献[5]中将基本活动看作原子操作,模型中考虑了基本活动被跳过的情况。文献[1,4]中考虑了原子活动的内部执行情况,所以建造的模型比较复杂。

原有工作存在的问题:

(1)没有考虑在〈sequence〉结构中,前后相邻的两个活动之间的关系和用〈link〉连接的两个活动之间关系的区别。

(2)没有对〈flow〉结构内部做详细考虑,〈flow〉结构内部活动的执行受〈link〉的 joincondition 和 suppressJoinFailure 属性的影响,也受 Dead-Path-Elimination 的影响,结构非常复杂。

(3)没有考虑到 BPEL 中 Activity 所处的环境问题。不同的环境中,每个 Activity 会有不同的表现。

(4)没有考虑时间概念。

本文在解决上述问题的基础上给出从 BPEL 描述的 Web 组合服务流程到 TCP-nets 模型的转换方法。文中将基本活动看作原子操作,只考虑操作是否成功执行,不考虑引起错误的原因。因为本文在假设每个单独的服务都是正确的前提下,主要解决的问题是 Web 服务组合的建模问题,为 Web 服务组合的验证和测试奠定了基础。

本文图中的模型是使用丹麦 Aarhus 大学开发的 CPN Tool 工具[6]生成的,颜色集 STATE 是赋时 bool 类型,t 表示

到稿日期:2009-11-19 返修日期:2010-01-05 本文受国家十一五国防预研项目(513060601),校基础研究基金(A12035)资助。

王玉英(1964-),博士生,副教授,主要研究方向为逆向工程、Web 服务、软件测试,E-mail:yinyin632@sina.com;陈 平(1953-),男,博士生导师,主要研究方向为软件工程。

bool 类型常量 true，模型中使用它作为资源流动，表示一个 activity 的状态。颜色集 MSG 为 STRING 类型，变量 msg 为 MSG 类型。

2 基本活动建模

基本活动可以看作是原子的，即没有内部结构。

2.1 Empty activity(空活动)

这个活动在其它活动的 CP-nets 中用于构造控制流。其模型如图 1 所示。

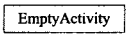

图 1 Empty Activity 的 CP-net

2.2 Wait activity(等待活动)

图 2 为〈wait〉的模型图。图中 deadline 和 duration 都是常数，图 2(a)中的@deadline 表示到时刻 deadline 时继续执行，图 2(b)中的@+duration 表示等待一段给定的时间 duration，然后继续执行。这个 activity 是与模型时间密切相关的，它们的存在是本文采用有色赋时 Petri 网(简称 TCP-nets)模型的主要原因。由于 TCP-nets 模型不能直接表达在某一时刻激发 transiton，因此我们定义了一个函数 OK()，用这个函数值作为防守条件控制 transition 的执行时间，使它在时刻 deadline 或者超过时刻 deadline 开始执行。函数 OK() 被定义为：

OK() = ModelTime. cmp (ModelTime. fromInt (deadline), time())

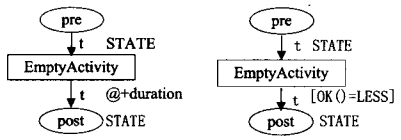

图 2

这里使用了 CPN 的 simulator functions 中的函数 time()，它返回模型执行过程中的当前时间，ModelTime. fromInt(t) 将整数表示的时间 t 转换为 time 类型[8]，ModelTime. cmp ($t1$,$t2$) 用于比较时间 $t1$ 和 $t2$ 的大小，返回 LESS，GREATER 或 EQUAL，如果 $t1<t2$，则返回 LESS。

EBEL 中只有〈wait〉活动涉及到时间因素，其他活动模型严格意义上讲没有涉及到时间因素。对于〈wait〉活动，用变迁的防守函数实现了时间约束，每个库所的时间戳均为 0，这里严格意义上说是使用了 TCP-net 建模。在后面的模型中我们只谈使用 CP-net 建造模型。对于 BPEL 服务组合模型，可以说是使用全局时间变量和 CP-net 建模。

2.3 Assign Activity & Validate Activity & Throw Activity & Exit Activity(赋值，确认，抛出，退出)

这几个活动结构简单，完成的任务单一。〈Assgin〉和〈validate〉是对系统中变量的值进行赋值和验证，〈throw〉和〈exit〉是在错误发生的时候抛出错误和终止进程。图 3 是 Assign Activity 的 CP-nets 模型图，Validate Activity，Throw Activity 和 Exit Activity 的 CP-nets 与此类似。

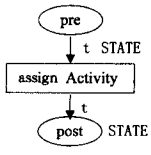

图 3 Assign Activity 的 CP-net

2.4 Receive Activity & Reply Activity & Invoke Activity & Rethrow Activity(接收，回答，调用，再抛出)

BPEL 中这 4 个活动都涉及到和其他活动的交互。图中的库所 incoming 和 outgoing 分别表示接收到的消息和发送出去的消息。有两种类型的〈invoke〉活动，一种类型是 Request-response，另一种是 One-way invocation。前者既有输入变量也有输出变量，在它的 CP-net 模型中有一个输出库所和一个输入库所；后者只有输入变量，在它的 CP-net 模型中只有一个输入库所。〈Rethrow〉执行后给出错误的名字、类型等信息。图 4 中给出的是 Receive Activity 和 Invoke Activity 的 CP-net 模型图，Reply Activity 和 Rethrow Activity 的模型图与 Receive Activity 的类似。

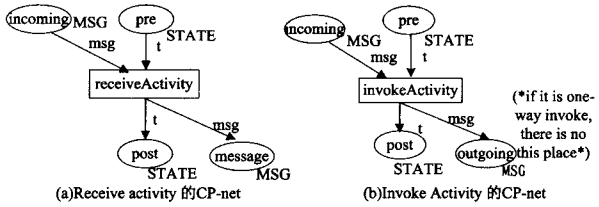

图 4

3 结构化活动建模(Struct Activities)

结构化的活动规定了一组活动发生的顺序，描述了创建业务流程的基本活动组成的结构，这些结构表达了涉及业务协议的流程实例间的控制形式、数据流程、故障和外部事件的处理以及消息交换的协调。BPEL 的结构化活动包括：顺序控制由〈sequence〉，〈switch〉和〈while〉组成；活动之间的并发和同步由〈flow〉组成；基于外部事件的不确定的选择由〈pick〉组成；可以递归地使用结构化的活动。

3.1 Sequence Activity(顺序结构)

一个〈Sequence〉活动可以包含一个或多个顺序执行的活动，图 5 中给出了包含两个活动的模型。事实上这种活动可以嵌套，从而形成了任意个活动组成的〈Sequence〉活动。当最后一个包含的活动执行完后，整个〈Sequence〉活动执行完毕。

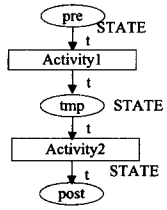

图 5 Sequence Activity 的 CP-net

将每个活动分别建模，然后根据执行顺序，将前一个活动的 post 库所和下一个活动的 pre 库所合并成中间库所 tmp，〈Sequence〉活动的开始库所为内含的第一个活动的开始库所，〈Sequence〉活动的结束库所为内含的最后一个活动的结束库所，这样就形成了〈Sequence〉活动模型，如图 5 所示。

3.2 If Activity(顺序结构)

〈If〉活动中包含了一个或几个条件性选择执行的活动。

在〈If〉活动模型中使用一个〈empty〉活动构建了条件执行结构，利用变迁上的防守函数来控制活动的执行，如图 6 所示。

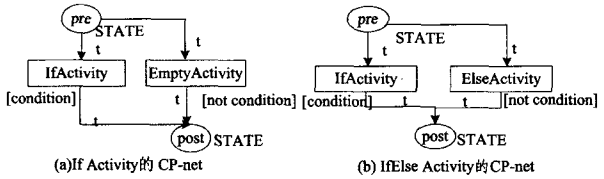

图 6

3.3 Repeat Activity, While Activity, For Each(循环结构)

〈While〉活动中所包含的活动在"条件为真"时重复执行，每次执行前求解这个条件表达式，所以在模型中将这个表达式作为防守函数，如图 7 所示。

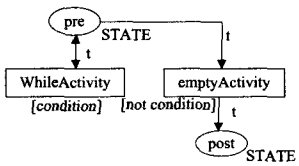

图 7 While Activity 的 CP-net

在〈Repeat〉活动模型中使用两个〈empty〉活动和一个临时的库所构建了循环执行结构。库所 temp 表示活动刚刚执行完，托肯 t 的传送启动下一次循环或者结束循环。Repeat Activity 和 While Activity 的模型图如图 8 所示。

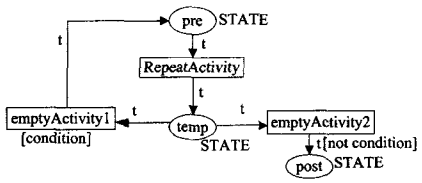

图 8 Repeat Activity 的 CP-net

〈ForEach〉中所包含的活动执行固定次数 N，当其属性 paralle 值为"no"时，这 N 次执行是串行的，paralle 值为"yes"时，这 N 次执行是并行的。

在〈ForEach〉的模型中，开始库所的颜色为 INT × STATE，初始标记为 $N\times t$，其中分量 N 表示 ForEach 所包含的活动的执行次数。当属性 paralle 值为"no"时，模型中变迁 ForEachActivity 的输入为(n,t)，输出为$(n-1,t)$，防卫函数为$[n>0]$，这样确保了这个变迁被激发 N 次，即这个变迁一次执行结束后开始下一次执行。当属性 paralle 值为"yes"时，模型中变迁 ForEachActivity 的输入为 t 和 k，输出为 $k-1$；通过一个防卫函数为$[n>0]$的〈empty〉活动，变迁 ForEachActivity不停地接收托肯 t，而且只接受 N 个，这样确保了这个变迁并行地执行 N 次；利用一个防卫函数为$[k=0$ orelse $n=0]$的〈empty〉活动确保了 N 小于等于零时整个活动结束。

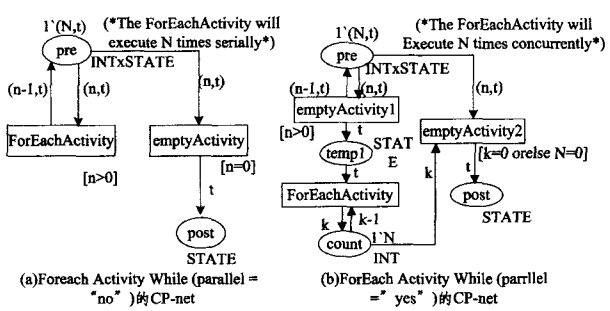

图 9

当属性 paralle 值为"no"时，〈ForEach〉活动中包含的活动 ForEachActivity 的开始库所和结束库所都合并到模型中的开始库所 pre 中；当属性 paralle 值为"yes"时，〈ForEach〉活动中包含的活动 ForEachActivity 的开始库所合并到模型中的 tmp1 中，结束库所都合并到模型中的库所 count 中，如图 9 所示。

3.4 Pick Activity(选取结构)

〈Pick〉活动提供基于外部事件的不确定的选择机制。外部事件有两种：onMessage 和 onAlarm。〈Pick〉中至少包含一个 onMessage 事件。在 onAlarm 事件中，for 表达式指定一个时间间隔 duration，在此时间间隔后 onAlarm 事件发出信号；另一个可选的表达式 until 指定 onAlarm 事件的触发时刻，如图 10 所示。

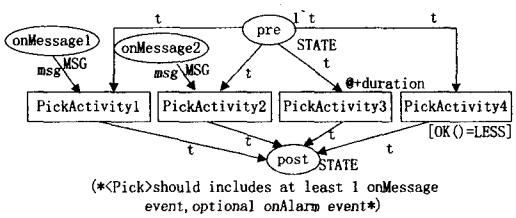

图 10 Pick Activity with 2 onMessage Events and 2 onAlarm Events 的 CP-net

3.5 Flow Activity(并发和同步流程结构)

〈flow〉活动是所有活动中最为复杂的活动，它提供内嵌活动的并发和同步控制。当内嵌的活动都执行完，这个〈flow〉活动执行完毕。〈flow〉里面涉及到的元素有〈link〉，〈transitionCondition〉，〈joinCondition〉等，还有一个非常重要的属性〈suppressJoinFailure〉。由于篇幅关系，我们将在另一篇论文中专门讨论〈flow〉活动的模型，其中讨论分为 4 个方面：

不含有〈link〉的〈flow〉活动；

〈link〉的基本模型；

〈link〉序列建模及控制流中的死路径删除问题；

〈Flow〉中控制依赖的传递性。

4 模型分析与实例

4.1 模型分析

从以上的建模过程中可以看到，所建立的模型具有以下特点：

1. 每个模型含有一个 pre 库所和一个 post 库所，它们分别表示活动的起始状态和结束状态。当 post 库所值为真值时，表明活动正常结束；当 post 库所值为假值时，表明活动非正常结束。

2. 库所颜色有限，以 BOOL 型(STATE 色)和字符串型(MSG 色)为主，还有整型。库所颜色决定了弧表达式的类型。

3. 弧表达式包含 BOOL 类型时，弧的方向表明了 BPEL 中的流程走向；弧表达式包含 MSG 类型时，弧的方向表明了 BPEL 中信息的传送，即数据的访问。

4. 在只考虑流程的控制流时，可以去掉模型中的所有 MSG 类型的库所，这样不影响结果。

4.2 层次化建模

BPEL 采用 XML Schema 编写，从形式上它定义了用于组成复杂业务流程交互的基础和结构化活动，指定了业务流程怎样使用 Web 服务达到其目的，以及由业务流程提供的 Web 服务。BPEL 是一种以 XML 来描述企业内部流程的语言，使原本建立在不同产品上的商业流程也能像 Web 服务一样可以跨平台互通。我们可以根据 XML 语言层次化特性，层次化地构建 BPEL 语言所描述的 Web 服务组合。构建方法将另外撰文论述。

4.3 模型实例

《服务的业务流程执行语言》即规范中定义了一个名为 Purchase Order[7] 的流程服务案例，利用本文所述的建模算法，对这个例子进行建模，得到模型如图 11 所示。

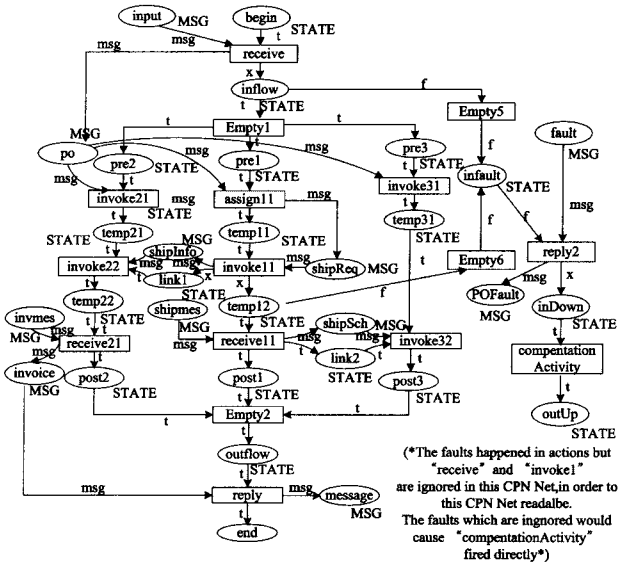

图 11 Purch aseOrder 的 CP-net(子网展开后的)

结束语 本文在总结前人所作工作的基础上，描述了基于赋时有色 Petri 网的 Web 组合服务建模方法。这种方法从执行环境、时间等方面更好地描述了 BPEL 流程，可以在此基础上进一步分析和验证 BPEL 流程。本文创新点在于使用赋时有色 Petri 网更为准确地模拟了 BPEL 流程，考虑了时间等环境因素。未来的研究方向在于对 BPEL 模型从数据流和控制流两个方面进行分析验证。

参 考 文 献

[1] Stahl C. A Petri Net Semantics for BPEL[R]. Technical Report 188, Humboldt-Universität zu Berlin, Institut für Informatik, 2005

[2] Forster H, Uchitel S, Magee J, et al. Model-based verification of Web service composition[C]// Proceedings of 18th IEEE International Conference on Automated Software Engineering. Montreal, Canada: IEEE Computer Society, October 2003: 152-161

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

[4] Kang Hui, Yang Xiuli, Yuan Sinmiao. Modeling and Verification of Web Services Composition based on CPN[C]// 2007 IFIP International Conference on Network and Parallel Computing Workshops(NPC 2007). Lialian, China, September 2007

[5] Ouyang C, Verbeek E, van der Aalst W, et al. Formal semantics and analysis of control flow in WS-BPEL[J]. Science of Computer Programming, 2007, 67(2/3): 162-198

[6] http://wiki.daimi.au.dk/cpntools/cpntools.wiki. 2009

[7] OASIS Standard, Web Services Business Process Execution Language Version 2.0[S]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf, 2008-5-22

[8] http://wiki.daimi.au.dk/cpntools-help/simulator_functions.wi-ki?cmd=get&anchor=Simulator+functions. 2008-11-24

(上接第 126 页)

[9] Ghiassi-Farrokhfal Y, Pakravan M R. Cross-layer flooding for sensor networks without location information [C]// IEEE International Conference on Mobile Adhoc and Sensor Systems Conference. Washington, DC. USA, 2005: 110-114

[10] Wang X, Yin J, Zhang Q, et al. A cross-layer approach for efficient flooding in wireless sensor networks [C]// 2005 IEEE Wireless Communications and Networking Conference. New Orleans, LA USA, 2005: 1812-1817

[11] Srivastava V, Motani M. Cross-layer design: A survey and the road ahead [J]. IEEE Communications Magazine, 2005, 43(12): 112-119

[12] Stemm M, Katz R H. Measuring and reducing energy consumption of network interfaces in hand-held devices [J]. IEICE Transactions on Communications, 1997, 80(8): 1125-1131

[13] Chen B, Jamieson K, Balakrishnan H, et al. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks [J]. Wireless Networks, 2002, 8(5): 481-494

[14] De P, Liu Y, Das S K. An epidemic theoretic framework for evaluating broadcast protocols in wireless sensor networks[C]// IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems(MASS 2007). Pisa, Italy, 2007: 1-9

[15] Daley D J, Kendall D G. Epidemics and rumours [J]. Nature, 1964, 204(4963): 1118

[16] Moreno Y, Nekovee M, Pacheco A F. Dynamics of rumor spreading in complex networks [J]. Physical Review E, 2004, 69(6): 66130

[17] Lu F, Chia L T, Tay K L, et al. Nbgossip: An energy-efficient gossip algorithm for wireless sensor networks [J]. Journal of Computer Science and Technology, 2008, 23(3): 426-437