

基于 XYZ/ADL 的异步 Web 服务组合描述与验证

石慧娟¹ 戎 玫² 张广泉^{1,3} 朱稷涵¹

(苏州大学计算机科学与技术学院 苏州 215006)¹ (暨南大学深圳旅游学院 深圳 518053)²

(中国科学院软件研究所计算机科学国家重点实验室 北京 100080)³

摘 要 以 Web 服务组合为研究对象,重点讨论了服务组合中异步通信行为和时间属性的形式化描述和验证。首先,从软件体系结构角度分析 Web 服务组合,采用基于时序逻辑的 XYZ/ADL 描述 Web 服务的交互行为和时间属性;然后,提出一种符合模型检测工具 UPPAAL 规约的时间异步通信模型 TACM;最后,实现了 XYZ/RE 通信命令到 TACM 的映射,利用 UPPAAL 验证了服务组合系统异步通信行为的正确性。

关键词 Web 服务组合,XYZ/ADL,异步通信,时间属性,模型检测

中图分类号 TP311 **文献标识码** A

Description and Verification of Asynchronous Web Service Composition Based on XYZ/ADL

SHI Hui-juan¹ RONG Mei² ZHANG Guang-quan^{1,3} ZHU Ji-han¹

(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)¹

(Shenzhen Tourism College, Jinan University, Shenzhen 518053, China)²

(State Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing 100080, China)³

Abstract Concerned with Web service composition, this paper especially emphasized the formal description and verification of asynchronous communication behaviors and timed properties. Firstly, analyzing Web service composition from software architecture, the interactive behaviors and timed properties were described by XYZ/ADL based on temporal logic language. Secondly, timed asynchronous communication model (TACM) which accords with the specification of model checker UPPAAL was proposed. Finally, based on the transition from XYZ/RE communication command sentences to TACM, the correctness of asynchronous communication behaviors of the service composition system was verified by UPPAAL.

Keywords Web service composition, XYZ/ADL, Asynchronous communication, Timed properties, Model checking

1 引言

面向服务体系架构(Service Oriented Architecture, SOA)是蓬勃发展的一种软件架构理念,具有良好的可重用性、松耦合性、互操作性和平台无关性。Web 服务作为目前广泛应用的分布式计算技术,已成为实现 SOA 架构的主流方式。随着电子商务的快速发展,单一的 Web 服务已不能满足日益复杂的业务需求,由此应运而生的 Web 服务组合成为了软件服务领域的研究热点。

Web 服务组合是指由各个小粒度的 Web 服务相互之间通信和协作来实现大粒度的服务功能,而 Web 服务通信又分为同步和异步两种方式。现有的一些工作^[1-3]大部分是建立在同步通信基础上的,而 Web 服务的松耦合性要求其具有异步通信的特征,使得现有研究成果在实际应用中具有一定的

局限性。为了克服这些缺陷,有关 Web 服务的异步通信研究引起了人们的关注,并取得了一些进展。文献[4]对 Web 服务的同步和异步两种通信方式进行了比较,详细地阐述了它们之间的区别和联系,提出了自上而下和自下而上两种异步交互模型。文献[5]使用回调机制研究 Web 服务的异步通信,提出了一种 Web 服务可靠交互模型。文献[6]在 π 演算的基础上讨论了 Web 服务组合问题,提出了基于异步 π 演算的面向服务计算模型。然而,上述这些研究对 Web 服务异步交互的行为属性(如死锁、安全性、活性)和时间属性的分析验证问题,很少提出有效可行的方法。此外,我们在文献[7]中对 Web 服务组合的时间属性进行了验证,但未考虑异步通信情况。

针对上述问题,本文给出了解决方案。首先,采用基于时序逻辑的软件体系结构描述语言 XYZ/ADL 准确描述 Web

到稿日期:2010-12-29 返修日期:2011-05-06 本文受国家自然科学基金(60973149),江苏省自然科学基金(BK2011281),中国科学院计算机科学国家重点实验室开放课题(SYSKF0908),江苏省高校自然科学研究项目(08KJB520010),苏州大学国家级大学生创新性实验计划项目(101028524)资助。

石慧娟(1986—),女,硕士生,主要研究方向为软件服务与形式化方法,E-mail:shi_huijuan@163.com;戎玫 博士,副教授,主要研究方向为软件工程与网络分布计算;张广泉 博士,教授,CCF 高级会员,主要研究方向为服务计算、软件体系结构与形式化方法、网络与分布式计算等;朱稷涵 男,本科生。

服务交互的行为和时间属性;然后,提出一种可直接使用模型检测器 UPPAAL 验证的时间异步通信模型 TACM(Timed Asynchronous Communication Model);最后,将 Web 服务组合中的 XYZ/RE 通信命令转化为 TACM,利用 UPPAAL 来验证 Web 服务异步交互的正确性。相对于其他方法,本方法减少了系统模型到检测器可接受的描述形式之间的转换,简化了验证过程。

2 基于 XYZ/ADL 的 Web 服务组合描述

结合 Web 服务组合和软件体系结构(SA)领域的研究可以发现,它们之间存在相似之处:单个 Web 服务可以看作构件,Web 服务之间的交互规则可以对应到连接件,Web 服务组合的总体布局则可以映射为配置,这为从软件体系结构角度研究 Web 服务组合问题提供了依据和支持。采用基于时序逻辑的软件体系结构描述语言 XYZ/ADL 描述 Web 服务组合,便于进行形式化验证。

2.1 XYZ/E 及其扩展

XYZ/E 是一种可执行的时序逻辑语言,其基本单元是条件元,有两种形式:

$$LB=y \wedge R \Rightarrow \$Ov=e \wedge \$OLB=z \quad (1)$$

$$LB=y \wedge R \Rightarrow @(Q \wedge LB=z) \quad (2)$$

式中,“ $\Rightarrow$ ”表示蕴含; R, Q 是一阶逻辑公式,分别称为条件部分和动作部分; LB 为标号控制变量, y 和 z 分别表示定义标号和转出标号;符号 $@$ 为一阶逻辑算子,可以是下一时刻算子 $\$O$ 或最终时刻算子 $\langle \rangle$ 。式(1)定义了相邻状态之间的转换关系,用于描述动态语义;式(2)表示程序抽象规范,用于描述静态语义。

XYZ/ADL 是在 XYZ/E 基础上扩展得到的软件体系结构描述语言,能够在统一的时序逻辑框架下表示从形式规范到可执行程序不同层次的系统描述。一个完整的构件、连接件及配置的 XYZ/ADL 描述分别如下:

```
%COMPONENT COM==[
  %PORT P: Record(%CHN A; DATATYPE1;%CHN B;
  DATATYPE2)
  %PROPERTY==[...]
  %COMPUTATION==[...]
  %CONNECTOR CON==[
  %ROLE SourceData==OUT(DT1,v1);
  %ROLE SinkData==IN(DT2,v2);
  %GLUE==[...]
  %ATTACHMENTS==[ComIns.Port # ConIns.Role;Com-
  Ins.Port # # Port;...]
```

其中,构件的内部规范用一个 XYZ/E 单元表示不同抽象层次的行为,如果涉及到时间约束,则可以使用 XYZ/RE 来描述。XYZ/RE 是对 XYZ/E 进行实时扩展得到的,也就是对时序算子 $\$O, \langle \rangle, [], \$U, \$W$ 加以扩充,使之表示出这些算子的实时下限(即 l)和实时上限(即 u)的信息^[8]。XYZ/RE 条件元的两种基本形式如下:

$$LB=L0 \wedge R \Rightarrow \$O\{l, u\}(Q \wedge LB=L1)$$

$$LB=L0 \wedge R \Rightarrow @\{l, u\}(Q \wedge LB=L1)$$

2.2 Web 服务组合描述

组合 Web 服务的行为是通过几个子构件的连接来表示的,COMPUTATION 描述了 Web 服务构件的组合方式。以下列出几种常见的组合方式^[9]。

(1)顺序组合。顺序组合是指按业务逻辑依次执行 Web 服务构件。如图 1 所示,依次执行 ServiceA, ServiceB 就得到组合服务 ServiceC,即 ServiceA 的输入和 ServiceB 的输出就是 ServiceC 的输入、输出。其语法定义为

$$LB=L0 \Rightarrow \$O(In? ServiceA, input \wedge LB=L1)$$

$$LB=L1 \wedge ServiceA \Rightarrow \$O(ServiceB \wedge LB=L2)$$

$$LB=L2 \Rightarrow \$O(Out! ServiceB, output \wedge LB=L3)$$

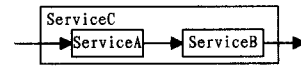


图 1 顺序组合

(2)选择组合。选择组合是指根据条件选择所要执行的 Web 服务构件。如图 2 所示,如果 Con 为真,ServiceC 即为 ServiceA,反之 ServiceC 为 ServiceB。其语法定义为

$$LB=L0 \wedge Con \Rightarrow \$O(In? ServiceA, input | In? ServiceB, input) \wedge \$OLB=L1$$

$$LB=L1 \wedge Con \Rightarrow \$O(ServiceA \wedge LB=Next | ServiceB \wedge LB=Next)$$

$$LB=L2 \wedge Con \Rightarrow \$O(Out! ServiceA, output | Out! ServiceB, output) \wedge \$OLB=L3$$

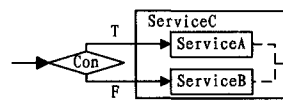


图 2 选择组合

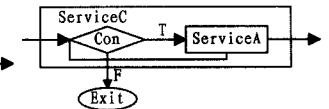


图 3 循环组合

(3)循环组合。循环组合是指在满足条件的前提下重复执行某一个 Web 服务构件。如图 3 所示,ServiceC 在满足条件 Con 时循环执行 ServiceA, ServiceA 的输入和若干次执行 ServiceA 后 ServiceA 的输出为 ServiceC 的输入、输出。其语法定义为

$$LB=L0 \wedge Con \Rightarrow \$O(In? ServiceA, input \wedge LB=L1)$$

$$* [LB=L1 \wedge Con \Rightarrow (\$OLB=Next | \$OLB=Exit)]$$

$$LB=Label\{ | ServiceA | \} \$OLB=L1]$$

$$LB=L2 \Rightarrow \$O(Out! ServiceA, output \wedge LB=L3)$$

(4)并发组合。并发结构是指同时执行多个(两个以上) Web 服务构件。本文讨论两种类型的并发结构,如图 4 所示。图 4(a)表示 ServiceC 由并发执行的 ServiceA 和 ServiceB 组成。ServiceA 和 ServiceB 的输入为 ServiceC 的输入,ServiceA 加上 ServiceB 的输出为 ServiceC 的输出。其语法定义为

$$LB=L0 \Rightarrow \$O(In? ServiceA, input \wedge In? ServiceB, input \wedge LB=L1)$$

$$LB=L1 \Rightarrow [| [ServiceA, ServiceB] \wedge \$OLB=L2$$

$$LB=L2 \Rightarrow \$O(Out? (ServiceA, output \wedge ServiceB, output) \wedge LB=L3)$$

图 4(b)表示 ServiceC 由并发执行的 ServiceA 和 ServiceB 组成。ServiceA 和 ServiceB 的输入为 ServiceC 的输入,条件 Con 决定 ServiceC 的输出(ServiceA 或 ServiceB 的输

出)。其语法定义为

$$\begin{aligned} LB=L0 &\Rightarrow \$O(In? ServiceA, input \wedge In? ServiceB, input \\ &\quad \wedge LB=L1) \\ LB=L1 &\Rightarrow |[ServiceA, ServiceB] \wedge \$OLB=L2 \\ LB=L2 \wedge Con &\Rightarrow \$ (Out? ServiceA, output | Out? ServiceB, output) \wedge \$OLB=L3 \end{aligned}$$

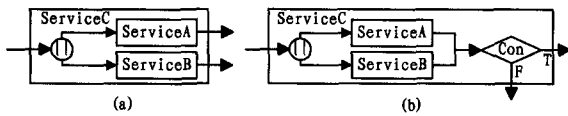


图4 并发组合

3 Web 服务异步通信分析与建模

在实际的 Web 服务组合中,服务间的交互是以异步通信方式进行的,而且异步通信也是构建健壮的 Web 服务的必要条件^[4]。下面将分析 Web 服务异步通信中出现的问题,并考虑相关的时间约束,给出 Web 服务异步通信的形式化模型。

3.1 Web 服务异步通信分析

Web 服务通过传递消息的方式进行交互,包括发送和接收消息,分别用! m 和? m 表示。为了表示 Web 服务的时间约束,可以引用时间自动机中的标准时钟,它的值随时间的流逝而增大。状态迁移是由使能条件和重置时钟集合来标记的,其中使能条件是关于时钟的关系表达式,其也称为时钟约束。

图5是一个 Web 服务异步通信的例子。为了确保服务的正确交互,顺利完成 Web 服务的异步通信,可以为每个服务引入一个消息队列。发送方发送消息时,可先将消息暂存在接收方的消息队列中,而接收方接收消息时,只需从它的消息队列中取出相应的消息。这里假设消息队列是无界的,消息可以以任意的顺序从队列中取出。

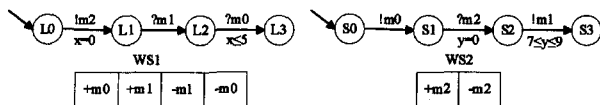


图5 Web 服务异步通信示例

在不考虑时间约束的情况下,分析 WS1 和 WS2 的交互过程。首先,WS1 发送消息 $m2$,消息 $m2$ 被插入 WS2 的消息队列中(图中“+”表示消息被插入队列)。然后,WS2 向 WS1 发送消息 $m0$,其接收消息 $m2$ 时,只需从消息队列中取出 $m2$ (图中“-”表示消息被取出),之后继续发送消息 $m1$ 。最后,WS1 依次执行接收消息 $m1$ 和 $m0$ 的动作,顺利地完成任务,并且各方的消息队列都变为空。

在实际的 Web 服务组合中都会涉及到对时间的限制,所以时间属性是服务交互的一个重要属性^[10]。如果考虑图中所表示的时间约束,两个服务之间的通信情况如下:WS1 发送完消息 $m2$ 后将时钟 x 置为 0,然后接收消息 $m1$ 。WS2 在接收消息 $m2$ 后将时钟 y 置为 0,同时在 7~9 个单位时间内发送消息 $m1$ 。故 WS1 接收到 $m1$ 时 $x \geq 7$,这与之后在 $x \leq 5$ 的条件下接收消息 $m0$ 冲突,致使 WS1 在状态 L2 无法继续进行迁移而处于死锁状态。

3.2 时间异步通信模型 TACM

为了对 Web 服务异步通信进行分析建模,先将时钟和消息均抽象为不同类型的变量,并将它们的值初始化为 0。其中,时钟是连续型变量,随时间的流逝而自动增加,并可以重新赋值为 0。消息是离散型变量,其值只能取 0 或 1。如果是发送消息,则消息被插入到消息队列,同时将相应消息变量的值置为 1。如果是接收消息,则要检查该消息是否已经在消息队列中(即判断消息变量的值是否等于 1)。若在队列中,则取出消息并将该消息变量的值置 0;否则,无法继续进行状态迁移。根据上述分析,下面给出 Web 服务异步通信模型的定义。

定义 1(时间异步通信模型) 时间异步通信模型 TACM 是一个七元组 (S, s_0, F, C, M, A, T) ,其中:

- S 是有穷状态集合;
- $s_0 \in S$ 是初始状态;
- $F \subseteq S$ 是终止状态集合;
- C 是时钟变量集合;
- M 是消息变量集合;
- $A: M \rightarrow \{?, !\}$ 是一个标记函数,为每一个消息变量指定一个接受(?)或发送(!)动作;

• $T \subseteq S \times g(C, M) \times u(C, M) \times S$ 是状态迁移集合, (s, g, u, s') 表示如果在状态 s 满足条件 g ,则要执行动作 u 来更新时钟和消息变量的值,同时迁移到状态 s' 。

$g(C, M)$ 是关于时钟变量和消息变量的约束条件,其定义如下:

$g(C, M) = \text{true} \mid x \sim v \mid y = 1 \mid \phi_1 \wedge \phi_2$, 其中, $\sim \in \{\leq, <, =, !, >, \geq\}$, $x \in C, y \in M, \phi_1, \phi_2 \in g(C, M)$, $v \in \mathbb{R}^+$ 是非负实数常量。

$u(C, M)$ 是更新时钟和消息变量的动作集合,其定义如下:

$u(C, M) = x := n \mid y := 0 \mid y := 1 \mid \phi_1 \wedge \phi_2$, 其中, $x \in C, y \in M, n \in \mathbb{N}$ 是自然数, $\phi_1, \phi_2 \in g(C, M)$ 。

定义 2(TACM 语义) 时间异步通信模型 TACM = (S, s_0, F, C, M, A, T) 的语义可以定义为一个标记迁移系统 $L_m = (L, L_0, \rightarrow)$, 其中:

• $L \subseteq S \times V_C \times V_M$ 是 L_m 的状态集合, $V_C \subseteq \mathbb{R}^+$ 是时钟变量值的集合, $V_M \subseteq \{0, 1\}$ 是消息变量值的集合;

• $L_0 = (s_0, v_0, \gamma_0)$ 是初始状态,对于 $\forall x \in C, v_0(x) = 0$, $\forall y \in M, \gamma_0(y) = 0$;

• $\rightarrow$ 是迁移关系,有两种形式:

①时间迁移: $(s, v, \gamma) \xrightarrow{d} (s, v+d, \gamma)$, 其中 $d \in \mathbb{R}^+$ 是时间增量;

②位置迁移: $(s, v, \gamma) \xrightarrow{m} (s', v', \gamma')$, 其中 $m \in M$, 当且仅当存在迁移 (s, g, u, s') 使得 $v(C) \vdash g \wedge \gamma(M) \vdash u$, 执行动作 u 后, $v(C) = v'(C)$, $\gamma(M) = \gamma'(M)$ 。

• 如果 $A(m) = ?$, 则 $g(C, M)$ 中消息变量的约束条件是 $m = 1$, 即判断消息 m 是否在消息队列中, $u(C, M)$ 中更新消息变量的动作是 $m := 0$, 即从队列中取出消息 m ;

• 如果 $A(m) = !$, 则 $g(C, M)$ 中没有关于消息变量的约

束条件, $u(C, M)$ 中更新消息变量的动作是 $m_i = 1$, 即将消息 m 插入队列中。

4 异步 Web 服务组合模型检测

模型检测是一种验证系统正确性的方法, 它的基本思想是: 将系统行为表示为状态迁移模型 M , 系统性质由时序逻辑公式 F 描述。这样, “系统是否具有期望的性质” 就转化为数学问题 “状态迁移模型 M 是否是公式 F 的一个模型”, 用公式表示为 $M \models F$ 。目前各具特色的模型检测工具 (SPIN, NuSMV, UPPAAL 等) 已得到广泛应用。本文选择 UPPAAL 作为验证工具, 主要基于两方面考虑: 第一, UPPAAL 可以验证 Web 服务组合中的时间属性; 第二, TACM 可以直接作为 UPPAAL 的输入而不需要做任何转换。

基于 XYZ/ADL 的异步 Web 服务组合的验证过程如下: 首先, 使用 XYZ/ADL 描述 Web 服务组合; 然后, 将服务组合 XYZ/ADL 描述中的 XYZ/RE 通信命令转换为符合 UPPAAL 规约的时间异步通信模型 TACM; 最后, 将系统待验证的性质表示为 CTL 公式, 把 TACM 和 CTL 公式输入模型检测工具 UPPAAL, 实现相关性质的验证。

Web 服务组合的 XYZ/ADL 描述中, 服务的交互行为是由 XYZ/E 单元表示的, 相关的时间约束则由 XYZ/RE 描述。由于 Web 服务交互的本质是信息的传递, 因此 Web 服务组合中用得最多的就是通信命令, 即输入和输出语句, 故只需实现 XYZ/RE 中输入、输出语句至 TACM 的转换。在 XYZ/RE 中, $ch?x$ 和 $ch!z$ 分别表示输入、输出谓词, 可以出现在条件部分 R 或动作部分 Q 中。其中, 通道 ch 是输入、输出的载体, “?” 表示该通道执行的是输入动作, “!” 表示通道执行的是输出动作, 参量 x 和 z 一般可省略。下面给出 XYZ/RE 的输入、输出语句至 TACM 的映射规则, 如表 1 所列。

表 1 XYZ/RE 通信命令及对应的 TACM

XYZ/RE	TACM
输入语句: $LB=L1 \wedge R1 \wedge ch? \Rightarrow \$O\{l, u\}$ ($Q1 \wedge \$OLB=L2$)	$(L1) \xrightarrow{R1, l \leq x \wedge ch=1} Q1, ch=0 (L2)$
输出语句: $LB=L3 \wedge R2 \wedge ch! \Rightarrow \$O\{l, u\}$ ($Q2 \wedge \$OLB=L4$)	$(L3) \xrightarrow{R2, l \leq x \wedge ch=1} Q2, ch=1 (L4)$

5 实例分析

本节以组合服务 BuyBook 为例, 按照上述方法对其进行分析, 从而进一步验证本文所提方法的可行性。BuyBook 由 Customer, BookShop 和 Bank 3 个服务组合而成, 三方交互的过程及时间约束如下: Customer 向 BookShop 发送 searchBook 请求, BookShop 查询要订购的书, 如果有库存则在 5 个单位时间内依次返回书号 bookID 和书价 bookPrice, 否则返回 outStock 信息。Customer 先接收 bookPrice 消息, 然后在 2 个单位时间内向 Bank 发送查询余额的请求 balanceInquire, 之后接收 bookID。Bank 处理请求并在 3~5 个单位时间内返回余额信息 balance, Customer 在接收到 balance 信息后判断账户余额是否充足, 以便决定是否继续服务。如果继续, 则在 5~10 个单位时间内向 Bank 发送支付信息 payInfo, 否则向 BookShop 发送信息 cancel 以取消服务。Bank 在收到 payInfo

后完成转账操作, 并在 3 个单位时间内向 BookShop 发送消息 payConfirm, 通知 BookShop 购买者已支付账单, 最后 BookShop 在 4 个单位时间内向 Customer 发送发货单信息 invoice。该服务组合是一个涉及时间约束的异步通信的例子, 下面给出交互行为的 XYZ/ADL 描述代码。

```
%COMPONENT Customer==[
LB=S0 ! searchBook=>$OLB=S1;
LB=S1 ! outStock=>$OLB=S2;
LB=S1 ! bookPrice=>$O(x=0 & LB=S3);
LB=S3($O{0,2})(! balanceInquire & LB=S4);
LB=S4 ! bookID=>$OLB=S5;
LB=S5 ! balance=>$O(x=0 & LB=S6);
LB=S6 ! cancel=>$O{5,10}(LB=S7);
LB=S6 ! payInfo=>$O{5,10}(x=0 & LB=S8);
LB=S8 ! invoice=>$OLB=S9;]
%COMPONENT BookShop==[
LB=L0 ! searchBook=>$O(y=0 & LB=L1);
LB=L1 ! outStock=>$O{0,5}(LB=L2);
LB=L1 ! bookID=>$O{0,5}(LB=L3);
LB=L3 ! bookPrice=>$OLB=L4;
LB=L4 ! cancel=>$OLB=L5;
LB=L4 ! payConfirm=>$O(y=0 & LB=L6);
LB=L6=>$O{0,4}(! invoice & LB=L7);]
%COMPONENT Bank==[
LB=M0 ! balanceInquire=>$O(z=0 & LB=M1);
LB=M1=>$O{3,5}(! balance & LB=M2);
LB=M2 ! payInfo=>$O(z=0 & LB=M3);
LB=M3=>$O{0,3}(! payConfirm & LB=M4);]
```

根据 XYZ/RE 输入、输出语句至 TACM 的映射规则, 可以得到服务 Customer, BookShop 和 Bank 的 TACM 模型, 如图 6 所示。其中, Customer 中的 S2, S7 和 S9 是终止状态; BookShop 的终止状态是 L2, L5 和 L7; M2 和 M4 是 Bank 的终止状态, 它们均以绿色标出。

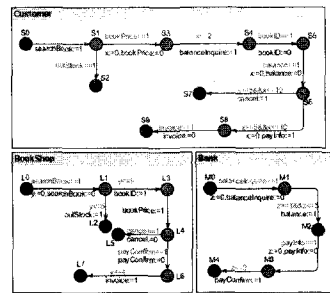


图 6 BuyBook 各交互方的 TACM 模型

下面将利用模型检测工具 UPPAAL 分析验证服务交互过程中是否出现死锁现象, 以及组合 Web 服务的业务功能是否符合系统需求, 即服务组合系统的死锁、安全性、活性, 验证结果如图 7 所示。

(1) 死锁: Web 服务交互过程中进入某个状态 (非终止状态), 该状态无法满足迁移条件, 致使交互不能继续进行。在 TACM 中, 无死锁的判断条件是最终到达终止状态并且消息队列为空 (即所有的消息变量的值均为 0), 其 CTL 公式表示为

$E\langle \rangle (Customer, S2 \text{ and } BookShop, L2) \text{ or } (Customer, S7$

and BookShop, L5 and Bank, M2) or (Customer, S9 and BookShop, L7 and Bank, M4) and (searchBook == 0 and bookPrice == 0 and balanceInquire == 0 and bookID == 0 and balance == 0 and payInfo == 0 and invoice == 0 and cancel == 0 and outStock == 0 and payConfirm == 0)

该性质验证结果为真,说明交互过程中未出现时间冲突、信息丢失等错误。

(2)安全性:“坏事情永远都不会发生”。在该实例中用户期望“Customer 在未付款的情况下收到了发货单信息 invoice”永远不会发生,其 CTL 公式表示为

$A \square \text{not}(\text{Customer}, S7 \text{ and } \text{Customer}, S9)$

(3)活性:“好事情最终会发生”。在该实例中用户期望“在完成付款后的 10 个时间单位内收到发货单”最终会发生,其 CTL 公式表示为

$A \langle \rangle \text{Customer}, S8 \text{ imply } \text{Customer}, S9 \text{ and } x < 10$

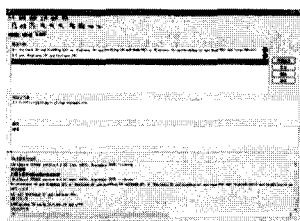


图7 性质验证结果

结束语 Web 服务组合是面向服务计算领域的研究热点之一,而异步通信是服务组合过程中信息交互的重要特征。针对现有研究成果中较少考虑服务组合中的时间属性及异步通信问题,本文采用 XYZ/ADL 描述 Web 服务组合,将其转化为符合实时模型检测工具 UPPAAL 输入规约的 TACM 模型,利用 UPPAAL 对服务组合中的异步通信行为进行了验证。下一步我们将改进和完善时间异步通信模型 TACM,使其应用于更多的异步通信场景。另外,还将考虑如何捕捉和处理 Web 服务交互中出现的各种异常,以保证 Web 服务组

合的可靠性。

参考文献

- [1] Pistore M, Roveri M, Busetta P. Requirements-driven Verification of Web Services[C]//Proc. of the 1st International Workshop on Web Services and Formal Methods (WSFM2004). Pisa, Italy, 2004; 95-108
- [2] 雷丽晖,段振华.一种基于扩展有限自动机验证组合 Web 服务的方法[J]. 软件学报, 2007, 18(12): 2980-2990
- [3] Foster H, et al. Model-based Verification of Web Service Compositions[C]//Proc. of the 18th IEEE International Conference on Automated Software Engineering (ASE 2003). Montreal, Canada, 2003; 152-161
- [4] Fu X, Bultan T, Su J. Synchronizability of conversations among web services [J]. IEEE Transactions on Software Engineering, 2005, 31(12): 1042-1055
- [5] Pallickara S, Fox G, Pallickara S L. An Analysis of Reliable Delivery Specifications for Web Services[C]//Proc. of the 10th International Conference on Information Technology, Coding and Computing (ITCC 2005). Las Vegas, USA, 2005; 360-365
- [6] Vieira H T, Caires L, Seco J C. The conversation calculus: A model of service oriented computation[C]//Proc. of the 17th European Symposium on Programming (ESOP 2008). Budapest, Hungary, LNCS 4960, 2008; 269-283
- [7] 何亚丽,戎玫,张广泉.一种基于 UPPAAL 的 Web 服务组合模型检测方法[J]. 计算机科学, 2010, 37(11): 122-125
- [8] 唐稚松,等.时序逻辑程序设计与软件工程[M]. 北京:科学出版社, 2002
- [9] 魏慧,戎玫,张广泉.一种基于体系结构的 Web 服务组合描述方法[J]. 计算机工程与科学, 2008, 30(12): 5-8
- [10] Guermouche N, Godart C. Timed Model Checking Based Approach for Web Services Analysis[C]//Proc. of the 7th IEEE International Conference on Web Services (ICWS 2009). Los Angeles, USA, 2009; 213-221

(上接第 120 页)

请求需要的传输带宽资源有所不同。d)数据项具有不同的时效性,过期数据项会被节点删除,以释放资源。e)用户查询分布的不均衡,这将导致数据项具有不同的被请求频率,高请求频率会使数据项成为热点,从而增加节点的处理负载。本文针对层次化的 P2P 网络拓扑结构提出了一种负载均衡算法。仿真结果表明,本算法可以依据节点能力的不同,确保负载在各个节点上的公平分布。

参考文献

- [1] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications[C]//Proceedings of ACM Sigcomm. Aug 2001; 149-160
- [2] Lian Jie, Naik K, Agnew G B. A Framework for Evaluating the Performance of Cluster Algorithms for Hierarchical Networks [J]. IEEE/ACM Transactions on Networking, 2007, 15(6): 1478-1489
- [3] Mirzaei A, Rahmati M. A Novel Hierarchical-Clustering-Combination Scheme Based on Fuzzy-Similarity Relations[J]. IEEE Transactions on Fuzzy Systems, 2010, 18(1): 27-39

- [4] Rao A, Lakshminarayanan K, Surana S, et al. Load Balancing in Structured P2P Systems[J]. Lecture Notes in Computer Science, 2003, 2735: 68-79
- [5] Godfrey P B, Lakshminarayanan K, Surana S, et al. Load balancing in dynamic structured P2P systems[C]//Proceedings of IEEE Infocom. Mar 2004; 2253-2262
- [6] Zhu Ying-wu, Hu Yi-ming. Efficient, proximity-aware load balancing for DHT based P2P systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2005, 16(4): 349-361
- [7] Rieche S, Vinh B T, Wehrle K. Range Queries and Load Balancing in a Hierarchically Structured P2P System[C]//Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN). Oct 2008; 28-35
- [8] Baumgart I, Heep B, Krause S. OverSim: A scalable and flexible overlay framework for simulation and real network applications [C]//Proceedings of IEEE Ninth International Conference on Peer-to-Peer Computing, Seattle, Washington, USA, 2009; 87-88