基于 XYZ/ADL 的 Web 服务组合描述与验证

张广泉[1,2],戎 玫[3],朱雪阳[2],何亚丽[1],石慧娟[1]
(1.苏州大学计算机科学与技术学院,江苏苏州 215006; 2.中国科学院计算机科学国家重点实验室,北京 100080;
3.暨南大学深圳旅游学院,广东深圳 518053)

摘 要: Web 服务组合是当前 Web 服务领域的一个研究热点,目前已有一些相关的描述与验证方法,本文从软件体系结构角度研究 Web 服务组合描述与验证方法.基于软件体系结构描述语言 XYZ/ADL 和精化检验/模型检测方法,提出了一种 Web 服务组合的描述与验证方法.XYZ/ADL 是时序逻辑语言 XYZ/E 的扩展,考虑到多数 Web 服务具有实时特征,采用 XYZ/E 的实时扩展语言 XYZ/RE 表示系统应满足的时间约束.针对 Web 服务组合系统,根据 XYZ/RE 到时间自动机的映射规则将系统描述转换为对应的时间自动机,分别采用精化检验和模型检测两种技术验证 Web 服务组合的正确性;最后通过两个实例分析分别阐述了上述方法的可行性和有效性.

关键词: Web 服务组合; XYZ/ADL; XYZ/RE; 时间自动机; 精化检验; 模型检测
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2011) 3A-086-08

Specification and Verification of Web Service Composition Based on XYZ/ADL

ZHANG Guang-quan[1,2], RONG Mei[3], ZHU Xue-yang[2], HE Ya-Li[1], SHI Hui-juan[1]
(1. *School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu* 215006, *China*;
2. *State Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing* 100080, *China*;
3. *Shenzhen Tourism College, Jinan University, Shenzhen, Guangdong* 518053, *China*)

Abstract: Web service composition is the hotspot in the field of Web services. Many methods are proposed to describe and verify its correctness. This paper researches specification and verification of Web services composition from software architecture. Refinement checking and model checking are two important formal verification methods. This paper explores the problem of Web service composition based on both software architecture description language XYZ/ADL and formal verification, then proposes a specification and verification method of Web service composition. XYZ/ADL is the extension of the temporal logic language XYZ/E. Considering most Web service with real time characteristics, we can use XYZ/RE which is the real time extension of XYZ/E to express time constraints of the system. For systems with time constraints, we transform the system description to corresponding timed automata according to the mapping rules, then use refinement checking and model checking to verify the correctness of web service composition. Experiments demonstrate feasibility and validity of above idea.

Key words: Web services composition; XYZ/ADL; XYZ/RE; timed automata; refinement checking; model checking

1 引言

近年来,Web 服务组合已成为 Web 服务领域的一个研究热点,目前已有一些相关的描述与验证方法,根据软件体系结构领域已有的研究工作,我们发现服务组合与软件体系结构之间存在一些内在的关联,例如可以将服务看作是构件、服务之间的交互看作是连接件、服务之间的交互的实例联系看作是配置,可以利用软件体系结构已有的研究成果如体系结构描述语言(ADL)等为 Web 服务组合的描述和验证提供支持.

本文提出了一种 Web 服务组合验证框架,将形式验证技术与软件体系结构相结合,描述与验证 Web 服务组合问题,从而能够在实际运行之前就发现系统中的漏洞,避免软件系统投入运行以后产生更大的损失,降低软件开发成本.

形式验证是一种基于已建立的形式规约、对系统相关特性进行数学分析和证明的技术.它采用一种形式化模型来刻画系统,然后分析该模型是否符合各种系统规

收稿日期:2010-10-27;修回日期:2011-02-26
基金项目:国家自然科学基金(No.60973149);中国科学院计算机科学国家重点实验室开放课题(No.SYSKF0908);江苏省高校自然科学研究项目(No.08KJB520010)

约.系统规约通常采用时序逻辑公式表示,也可以由系统描述语言抽象表示.精化检验和模型检测是目前两种重要的形式验证技术[1].在本文使用的精化检验方法中,系统的行为及其应满足的性质均采用时间自动机模型表示,通过判断行为自动机与性质自动机的乘积所接受的语言是否为空,来判断系统的行为是否满足性质规约.模型检测(model checking)是另一种验证系统正确性的方法[2],由美国的 E.M.Clarke、E.A.Emerson 和法国的 J.Sifakis 于 20 世纪 80 年代初分别提出(他们三位也因此荣获 2007 年图灵奖).模型检测的基本思想是:将系统行为表示为一个状态迁移模型 M,系统性质由时序逻辑公式 ϕ 描述,这样"系统是否满足所期望的性质"就可以转化为"状态迁移模型 M 是否满足时序逻辑公式 ϕ".

本文分别采用精化检验(图 1 中沿虚线部分)和模型检测(图 1 中沿中间有间隔点的虚线)两种方法研究基于软件体系结构描述语言 XYZ/ADL 的 Web 服务组合描述与验证问题,总体框架如图 1 所示.

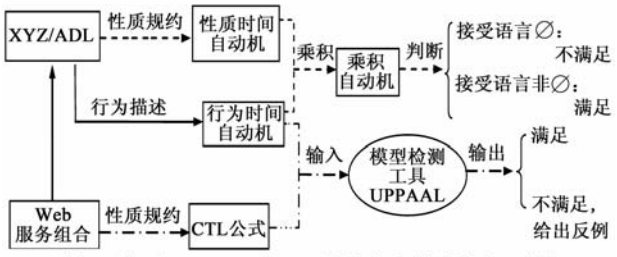


图1　基于XYZ/ADL的Web服务组合描述与验证框架

2　基于 XYZ/ADL 的 Web 服务组合描述方法

XYZ/ADL 是一种基于可执行时序逻辑语言 XYZ/E 的软件体系结构描述语言[3],它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的不同抽象层次的系统描述,既可以表示软件体系结构的动态语义,又可以表示其静态语义,弥补了已有 ADL 的不足,采用 XYZ/ADL 对 Web 服务及其组合进行描述是本文的工作基础之一.

2.1　XYZ/RE 及 XYZ/ADL 简介

XYZ/E 是中国科学院软件研究所唐稚松先生提出的一种可执行时序逻辑语言,其基本单元是条件元,有两种形式:

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

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

其中,R、Q 是一阶逻辑公式,R 是条件部分,Q 和 $\$Ov = e$ 为动作部分,LB 是控制变量,y 和 z 分别是转入和转出控制标号,$\Rightarrow$ 表示逻辑蕴涵.式(1)中条件元定义了系统相邻状态之间的转换关系;式(2)中条件元中的符号 @可以是下一时刻算子$ O 或最终算子 < >,表示程序抽象规范;它们的语义即为此时序逻辑公式的语义.

将 XYZ/E 中的时序算子$O, < >, [], $U, $W 进行扩充,使之表示出这些算子的实时下限(即 l)和实时上限(即 u)的信息,即得到 XYZ/RE.这里所谓算子的实时下限是指该算子从可能执行到该算子实际执行开始之间所需要的最短时间,实时上限是指该算子从可能执行到该算子全过程实际执行完毕之间所允许的最长时间,从而得到如下相应形式的实时算子:$\$O\{l,u\}$, $< >\{l,u\}$, $[]\{l,u\}$, $\$U\{l,u\}$, $\$W\{l,u\}$.

上述两种基本形式的 XYZ/E 加入实时语义后形式如下:

$LB = y \wedge R \Rightarrow \$O\{l,u\}(v = e \wedge LB = z)$

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

表示并行语句及选择语句的实时性时,形式分别如下:

$\| \{l, u\}[$ ProsInstaNm1 (Par1); …; ProsInstaNmk (Park)$]$

!! $\{l,u\}[$Cond1 | > ExeAct1, …, Condk | > ExeActk$]$

XYZ/RE 中的循环语句表示如下:

$*\{l,u\}[LB = \text{Entrylabel} \wedge p \Rightarrow (\$O\ LB = \text{NEXT} | \$O\ LB = \text{EXIT})$;

$LB = \text{label}\{|R|\}\$O\ LB = \text{Entrylabel}]$

实时分情形语句:

$?\ \{l,u\}[LB = \text{Entrylabel} \wedge p1 \Rightarrow \$O\ LB = z1$
$\quad | p2 \Rightarrow \$O\ LB = z2$
$\quad \cdots\cdots$
$\quad | pk \Rightarrow \$O\ LB = zk;$
$LB = z1\{|Q1|\}\ \$O\ LB = \text{EXIT};$
$LB = z2\{|Q2|\}\ \$O\ LB = \text{EXIT};$
……
$LB = zk\{|Qk|\}\ \$O\ LB = \text{EXIT}]$

实时 S 条件语句:$LB = y \wedge R \Rightarrow \$O\{l,u\}(Q1 \wedge LB = \text{NEXT} | Q2 \wedge LB = \text{NEXT})$

实时等待语句:$LB = y \wedge R \Rightarrow M\ \$U\{l,u\}(N \wedge LB = \text{NEXT})$

XYZ/ADL 是在 XYZ/E 的基础上扩展而来的一种体系结构描述语言,支持软件体系结构中构件、连接件以及配置的描述,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(*DT*1, *v*1);
%ROLE SinkData = = IN(*DT*2, *v*2);
%GLUE = =[…]]

%ATTACHMENTS = =[
Comlns.Port # Conln s.Role;
Comlns.Port # # Port; …]

下文我们采用体系结构描述语言 XYZ/ADL 对 Web 服务组合系统进行高层规范化描述,针对 Web 服务组合的特点,采用 XYZ/ADL 定义服务组件(简单 Web 组件和复合 Web 组件)及连接件.

2.2 简单 Web 组件

Web 组件用于描述组合服务中某个 Web 服务的构造实体.每个简单 Web 组件通过接口定义服务之间的数据流信息,接口至少包含一个端口,通过端口描述说明 Web 组件与外部环境的交互行为,端口由一组消息及端口行为组成.

消息表示 Web 组件需要的信息或者提供给其他 Web 组件的信息,语法定义如下:

%TYPE
Message = RECORD(Parameter1:DataType;
……
ParameterN:DataType;)

其中 ParameterN 为消息参数名,DataType 为参数类型.

端口有两种端口类型,即请求操作的输入端口以及提供操作的输出端口,其定义如下:

%PORTTYPE DataIn(DataType, vn) = = DataType;
□[*LB* = Start ⇒ $ ODataIn? *vn* ∧ $ O*LB* = *L*1;
LB = *L*1 ∧ ~(*vn* = EOF ⇒ $ ODataIn? *vn* ∧ $ O*LB* = *L*1;
LB = *L*1 ∧ (*vn* = EOF) ⇒ $ O*LB* = STOP]
%PORTTYPE
DataOut(DataType, *vn*) = = DataType;
□[*LB* = Start ⇒ $ O*LB* = *L*1;
LB = *L*1 ∧ ~(*vn* = EOF) ⇒ $ ODataOut! *vn* ∧ $ O*LB* = *L*1;
LB = *L*1 ∧ (*vn* = EOF) ⇒ $ ODataOut! EOF ∧ $ O*LB* = STOP]

DataIn 端口用于不断接受来自外部环境的数据 *vn*,直到系统关闭为止.DataOut 端口用于向外界传输数据 *vn*,直到系统关闭为止.

通过上述定义,Web 组件端口定义如下:

%PORT PortName = = DataX(DataType, *vn*)
/* DataX 为 DataIn 或 DataOut */

PROPERTY 抽象描述组件功能,定义如下:

%PROPERTY = =[operation1 ∧ … ∧ operationN]

在 Web 服务组合中,COMPUTATION 抽象地描述了性质描述中 Web 组件功能是如何完成的,定义如下:

%COMPUTATION = =[Computation Specification]

其中,入口标号 *LB* = operation1 表示 operation1 操作的开始,直到出口标号为 *LB* = End 表示 operation1 操作的结束.

一个 Web 组件不仅包含业务本身操作,而且还包含参与组合流程的操作,因此本文引入了 MYPORT 端口用于区别这两种操作.如果完成一个抽象功能涉及的端口中有 MYPORT 端口,则为业务本身操作.

2.3 连接件

连接件是用于建立 Web 组件组合交互规则的构造实体,描述各组件之间是如何进行交互的,它由角色和交互协议组合而成.角色规定了挂接在连接件上的交互方的单个接口的消息和行为,其描述与 Web 组件端口类似,定义如下:

%ROLE RoleName = = DataX(DataType, *vn*)
/* DataX 为 DataIn 或 DataOut */

在 Web 服务组合描述中,一个连接件有两个角色,一个是用于接收 Web 组件传输数据的角色,另一个是组件用于接收数据的角色.交互协议用于描述这两个角色是如何在一起协作工作的.

2.4 复合 Web 组件

复合 Web 组件是由简单 Web 组件或其他复合 Web 组件组合而成的.与简单 Web 组件相比,复合 Web 组件的接口描述与之相同,但在复合 Web 组件内部多了一个内部结构,用来描述 Web 组件是如何进行组合的.在描述复合组件时,要用配置来表示复合组件内子组件、连接件的总体布局.配置通常由一组粘结和绑定操作组成,其中粘结操作用于表示组件端口与连接件角色的连接关系,这里的端口指 PORT 端口,而绑定操作用于表示将子组件的端口作为该复合组件的端口,其中绑定输入端口为 MYPORT 端口,输出端口为 PORT 端口.在描述组合过程中,通过 COMPOSITION 描述该复合 Web 组件是由哪些 Web 组件组合而成以及这些组件之间交互的连接件;通过 ATTACHMENTS 描述了哪个组件参与何种交互实现 Web 组件与连接件的绑定或粘结操作.复合 Web 组件的行为是通过几个子组件的连接来表示的,由 COMPUTATION 描述 Web 组件是如何组合的,以下列出了 Web 组件常见的几种组合方式:

(1)顺序组合

顺序组合是指 Web 组件按业务逻辑依次执行,只有等前一个 Web 组件活动完成后方可执行下一个组件,如图 2 所示.组合服务 ServiceC 由 ServiceA 和 ServiceB 组成,依次执行 ServiceA 、ServiceB.ServiceA 的输入和 ServiceB 的输出分别作为 ServiceC 的输入和输出.其

语法定义为:

$LB = L0 \Rightarrow \$\ O(\text{In?}\ \text{ServiceA.input} \wedge LB = L1)$;

$LB = L1 \wedge \text{ServiceA} \Rightarrow \$\ O(\text{ServiceB} \wedge LB = L2)$;

$LB = L2 \Rightarrow \$\ O(\text{Out!}\ \text{ServiceB.output} \wedge LB = L3)$;

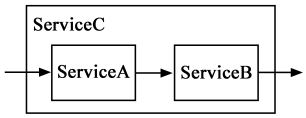


图2　顺序组合

(2)选择组合

选择组合是指根据条件选择要执行的 Web 组件,如图 3 所示.如果 Con 为真,ServiceC 即为 ServiceA,反之,ServiceC 为 ServiceB.其语法定义为:

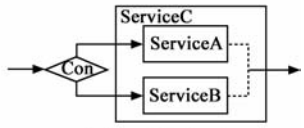


图3　选择组合

$LB = L0 \wedge \text{Con} \Rightarrow \$\ (\text{In?}\ \text{ServiceA.input} \mid \text{In?}\ \text{ServiceB.input}) \wedge \$\ O LB = L1$;

$LB = L1 \wedge \text{Con} \Rightarrow \$\ O(\text{ServiceA} \wedge LB = \text{Next} \mid \text{ServiceB} \wedge LB = \text{Next})$;

$LB = L2 \wedge \text{Con} \Rightarrow \$\ (\text{Out!}\ \text{ServiceA.output} \mid \text{Out!}\ \text{ServiceB.output}) \wedge \$\ O LB = L3$;

(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 组件.如图 4 所示,ServiceC 为在满足条件 Con 的情况下若干次执行 ServiceA,ServiceA 的输入为 ServiceC 的输入,在若干次执行 ServiceA 后 ServiceA 的输出为 ServiceC 的输出.其语法定义为:

$LB = L0 \wedge \text{Con} \Rightarrow \$\ O(\text{In?}\ \text{ServiceA.input} \wedge LB = L1)$;

$*[\ LB = L1 \wedge \text{Con} \Rightarrow (\$\ O LB = \text{NEXT} \mid \$\ O LB = \text{EXIT})$;

$LB = \text{Label}\{\mid \text{ServiceA} \mid\}\ \$\ O LB = L1\ ]$

$LB = L2 \Rightarrow \$\ O(\text{Out!}\ \text{ServiceA.output} \wedge LB = L3)$;

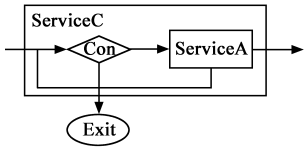


图4　循环组合

(4)并发组合

并发结构是指同时执行多个(两个以上)Web 组件,这些组件是平行执行的.这里,有两种类型的并发结构,如图 5 所示.

图 5(*a*)表示组合服务 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:

$LB = L0 \Rightarrow \$\ O(\text{In?}\ \text{ServiceA.input} \wedge \text{In?}\ \text{ServiceB.input} \wedge LB = L1)$;

$LB = L1 \Rightarrow \|\ [\text{ServiceA};\text{ServiceB}] \wedge \$\ O LB = L2$;

$LB = L2 \Rightarrow \$\ O(\text{Out?}\ (\text{ServiceA.output} \wedge \text{ServiceB.output}) \wedge LB = L3)$;

图 5(*b*)与(*a*)不同的是,根据 Con 决定 ServiceC 的输出(ServiceA 或 ServiceB 的输出).其语法定义为:

$LB = L0 \Rightarrow \$\ O(\text{In?}\ \text{ServiceA.input} \wedge \text{In?}\ \text{ServiceB.input} \wedge LB = L1)$;

$LB = L1 \Rightarrow \|\ [\text{ServiceA};\text{ServiceB}] \wedge \$\ O LB = L2$;

$LB = L2 \wedge \text{Con}\ \$\ (\text{Out?}\ \text{ServiceA.output} \mid \text{Out?}\ \text{ServiceB.output}) \wedge \$\ O LB = L3$;

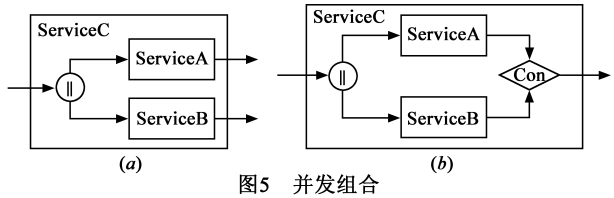


图5　并发组合

3　XYZ/RE 到时间自动机的映射

由于多数 Web 服务具有实时特性,本文采用 XYZ/E 的实时扩展语言 XYZ/RE 来表示 Web 服务组合系统中应满足的相关时间约束.下面先给出时间自动机的相关定义,然后给出 XYZ/RE 到时间自动机的映射规则.

3.1　时间自动机定义

定义 3.1　一个时间自动机(timed automata,TA)是一个六元组 $< L, L0, \Sigma, X, I, E >$,其中:

L:有穷状态集合;

$L0$:初始状态,且 $L0 \in L$;

Σ:有穷符号集合;

X:有穷时钟集合;

I:是一个映射,它给 L 中的每个状态指定 $\varphi(X)$中的一个时钟约束;

E:是一个转换集合,且 $E \subseteq L \times \varphi(X) \times \Sigma \times 2^X \times L$. $< l, \varphi, a, \gamma, l' >$ 表示输入符号 a 时,从状态 l 到状态 l' 的转换,φ 是 X 上的一个时钟约束,它在转换发生时被满足,$\gamma \subseteq X$ 是在该转换发生时复位零值的时钟集合.

时间自动机 TA 的语义可以通过一个状态迁移系统 S_{TA}来定义.S_{TA}的一个状态是一个二元组(l, v),l 是 TA 的一个状态,v 是满足 $I(l)$的一个时钟解释.如果 l 是 TA 的一个初始状态,且对于所有的时钟 x,$v(x) = 0$,则状态(l, v) 是一个初始状态.转换分为以下两种:

(1)因时间流逝的转换.对一个状态(l, v)和一个实数值时间增量 $\delta \geqslant 0$,如果对于所有 $0 \leqslant \delta' \leqslant \delta$,$v + \delta'$ 满足 $I(l)$,那么$(l, v) \rightarrow (l, v + \delta)$.

(2)因状态改变的转换.对于一个状态(l, v)和一个转换 $< l, \varphi, a, \gamma, l' >$,如果 v 满足 φ,那么$(l, v) \rightarrow (l', v[\gamma = 0])$.

定义 3.2 设 $TA_i = < L_i, L0_i, \Sigma_i, X_i, I_i, E_i > (i = 1,2)$是两个时间自动机,且 $X_1 \cap X_2 = \varnothing$,则乘积自动机 $A1 \times A2$ 是如下定义的一个时间自动机 $A1 \times A2 = < L_1 \times L_2, L_{01} \times L_{02}, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, I, E >$:

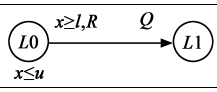
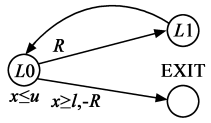
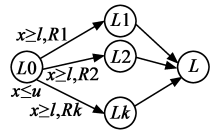
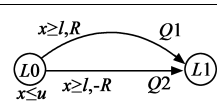
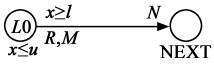
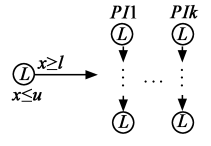
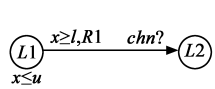
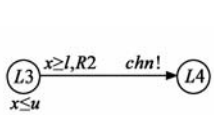
(1)$I(l_1, l_2) = I(l_1) \wedge I(l_2)$;

(2)对于 $a \in \Sigma_1 \cap \Sigma_2$,若 $< l, \varphi, a, \gamma, l' > \in E_i$,则 E 包含 $< (l_1, l_2), \varphi_1 \wedge \varphi_2, a, \gamma_1 \cup \gamma_2, (l_1', l_2') >$;

(3)对于 $a \in \Sigma_1 - \Sigma_2$,若 $< l, \varphi, a, \gamma, l' > \in E1$,对任意 $l_2 \in L_2$,则 E 包含 $< (l_1, l_2), \varphi_1, a, \gamma_1, (l_1', l_2') >$;

(4)对于 $a \in \Sigma_2 - \Sigma_1$,情况与 3 类似.

表 1 XYZ/RE 句型、表达式及对应时间自动机

句型及表达式	对应时间自动机
基本条件元: $LB = L0 \wedge R \Rightarrow \$\ O\{l,u\}(Q \wedge LB = L1)$	
基本条件元: $LB = L0 \wedge R \Rightarrow @\{l,u\}(Q \wedge LB = L1)$	同实时等待语句
循环语句: $*\{l,u\}[LB = L0 \wedge$ $R \Rightarrow \$\ OLB = NEXT \mid \$\ OLB = EXIT;$ $LB = L1\{\mid S\mid\}\ \$\ OLB = L0]$	
实时情形语句: $?\ \{l,u\}[LB = L0 \wedge R1 \Rightarrow \$\ OLB = L1$ $\mid R2 \Rightarrow \$\ OLB = L2$ …… $\mid R_k \Rightarrow \$\ OLB = Lk;$ $LB = L1\{\mid Q1\mid\}\ \$\ OLB = EXIT;$ $LB = L2\{\mid Q2\mid\}\ \$\ OLB = EXIT;$ …… $LB = Lk\{\mid Qk\mid\}\ \$\ OLB = EXIT]$	
实时 S 条件语句: $LB = L0 \wedge R \Rightarrow \$\ O\{l,u\}(Q1 \wedge LB = L1 \mid Q2 \wedge LB = L1)$	
实时等待语句: $LB = L0 \wedge R \Rightarrow M\ \$U\{l,u\}(N \wedge LB = NEXT)$	
并发组合语句: $\parallel \{l,u\}[ProsInstaNmi(par1);\cdots;$ $ProsInstaNmk(park)]$	
输入语句: $LB = L1 \wedge R1 \Rightarrow \$\ O\{l,u\}chn?\ x1 \wedge$ $\$\ OLB = L2$	
输出语句: $LB = L3 \wedge R2 \Rightarrow \$\ O\{l,u\}chn!\ x2 \wedge$ $\$\ OLB = L4$	

3.2 XYZ/RE 到时间自动机的映射

文献[5]将时间自动机中一个位置上的时间局限在$\{0,u\}$内,因此无法对转移发生最短时间为 l 时的情形给出更具体的映射规则,l 指算子从可能执行到实际执行开始之间所需要的最短时间,它在时间自动机中应该映射为转移发生的必要条件,而不是充分条件.这在实际应用中是有一定的局限性的,本小节对文献[5]工作做进一步的补充,使时间自动机在其中一个位置上时间的允许范围限制在$\{l,u\}$内,而不局限在$\{0,u\}$内.表 1 给出了 XYZ/RE 到时间自动机的映射规则.

4 基于 XYZ/ADL 的 Web 服务组合精化检验与模型检测

本节将给出使用精化检验和模型检测验证 Web 服务组合的相关步骤.

4.1 基于 XYZ/ADL 的 Web 服务组合精化检验

在本文的精化检验中,首先将组合 Web 服务系统的行为及其应满足的性质均采用时间自动机模型描述,然后对二者作乘积运算,通过判断乘积自动机的接受语言是否为空,来判断系统行为模型是否满足性质规约.随着 Web 服务逐渐成为 Internet 环境下实现分布式应用的强有力工具,多数情况下 Web 服务都具有实时特征.本文从已有的体系结构描述语言 XYZ/ADL 出发,使用 XYZ/ADL 描述具有时间约束的 Web 服务组合,将其中涉及到组合 Web 服务时间特性的描述部分即实时 XYZ/E(XYZ/RE)程序映射到一个时间自动机,采用精化检验的方法来验证 Web 服务组合系统的正确性,具体步骤如下:

(1)将系统中组合 Web 服务的行为用 XYZ/RE 来表示,继而转换为时间自动机,即行为时间自动机;

(2)将待验证组合 Web 服务系统性质用 XYZ/RE 来描述,再转换为时间自动机,即性质时间自动机;

(3)求行为时间自动机与性质时间自动机的乘积得到乘积自动机;

(4)验证行为自动机与性质自动机的乘积自动机所接受的语言是否为空,如果不为空,则说明系统行为满足系统性质,即组合服务满足用户的需求;反之,则不满足.

4.2 基于 XYZ/ADL 的 Web 服务组合模型检测

目前国际上已开发出可自动验证的模型检测工具,在一定程度上大大节省了人力劳动,也使系统的可靠性得到大幅度提高.和精化检验不同的是,基于 XYZ/ADL 的 Web 服务组合的模型检测,组合后系统应满足的性质由 CTL 时序逻辑公式表示,将表示系统行为的行为时间自动机和性质规约 CTL 时序逻辑公式作为模型检测工具 UPPAAL 的输入,从而实现对 Web 服务

组合系统的死锁、安全性和活性的验证,具体步骤如下:

(1)使用软件体系结构描述语言 XYZ/ADL 对 Web 服务及其组合进行描述.

(2)将其中对有时间约束的 Web 服务组合描述的 XYZ/RE 部分映射至时间自动机,单个实时 Web 服务映射至单个时间自动机,多个实时 Web 服务的组合映射至多个时间自动机的并发组合.

(3)将多个时间自动机作为模型检测工具 UPPAAL 的输入,组合后系统应满足的性质由 CTL 公式来表示,利用前驱风格的状态空间搜索算法[6]实现 Web 服务组合系统的死锁、安全性和活性的验证.

5　实例分析

根据前文提出的 Web 服务组合验证框架,本节结合两个例子,进一步阐述该验证框架的可行性和有效性.

5.1　网上旅游购票服务系统的描述与精化检验

以文献[7]提出的网上购票服务系统为例,若该组合服务交互中时间约束如下:假设 Traveler 在满足时钟约束 x≤10 时发送 Inquire 信息给 TravelAgent,TravelAgent 在执行完查询操作后,在 3 ~ 5 个时间单位之内将线路及票务信息 Result 发送给 Traveler,Traveler 在接收到 TravelAgent 发送的 Result 信息后,在 5 ~ 10 个时间单位内做出决定,如果接受 TravelAgent 的线路,则在接受 SeatsReserved 消息后的 20 个时间单位之内给 Ticketcenter 发送 BookSeat 信息确认订票(ConfirmTicket),否则票务预定作废.

首先分析该系统的行为,在 TravelAgent 没有返回 Result 时认为 Result = 0, Accept = 0 表示客户不满意 TravelAgent 反馈的 Result 信息,决定放弃出行,Accept = 1 表示客户同意接受 Result 信息并进入 Confirm 状态确认订票.对应 XYZ/RE 描述如下:

LB = idle ∧ Inquire = 1 ∧ Result = 0 ⇒ $ O{0,10} LB = waiting;

LB = waiting ∧ Result = 1 ∧ Accept = 0 ⇒ $ O{5,10} LB = end;

LB = waiting ∧ Result = 1 ∧ Accept = 1 ⇒ $ O{5,10} LB = confirm;

将上述 XYZ/RE 代码转换得到的行为时间自动机如图 6 所示(边上有小箭头的节点代表初始节点,环形节点代表时间自动机的接受状态,下同).

下面分析系统性质,要求如下:系统成功接收到 Inquire 信息后 Inquire = 1,并必会将查询结果 Result 在 3 ~ 5 个单位时间内发送给 Traveler.对应的 XYZ/RE 描述如下:

LB = s1 ∧ true ⇒ $ O{0,10} LB = $s1$;

LB = $s1$ ∧ Inquire = 1 ∧ Result = 0 ⇒ $ O{3,5}(LB = $s2$ ∧ Result = 1);

将上述 XYZ/RE 代码进行转换得到的性质时间自动机如图 7 所示.

将上述两个时间自动机进行乘积运算,得到的乘积时间自动机如图 8 所示.由图中可知:从初始状态(idle,$s1$)开始,存在两条可以到达可接受节点的执行路径,因此可知,该乘积自动机接受的语言不为空,即该组合 Web 服务系统行为满足系统规约.

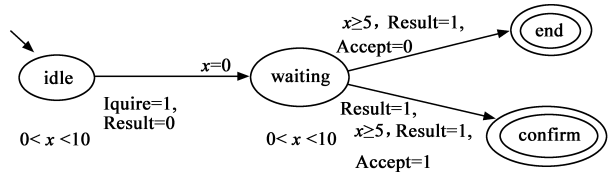


图6　购票系统行为时间自动机

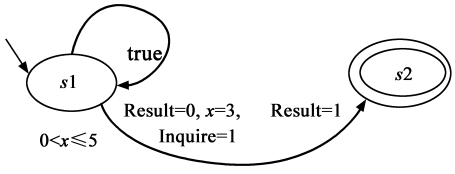


图7　购票系统性质时间自动机

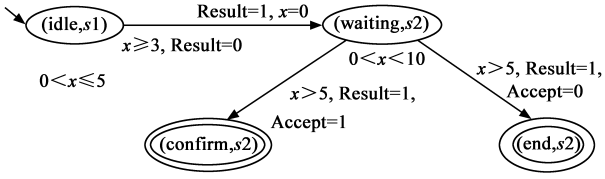


图8　行为时间自动机与性质时间自动机的乘积

5.2　股票分析服务系统的描述与模型检测

本小节以文献[8]提出的股票分析服务系统(Stock Analysis Service,SAS)为例,各交互端交互过程如图 9 所示.若该服务组合交互中的时间约束如下:Investor 在该服务启动 5 个时间单位内发送一个 regist 请求给 StockBroker,StockBroker 在接收到该请求 10 个时间单位内决定接受或拒绝该请求.若接受,则 ResearchDept 在接收到 StockBroker 发送的 request 消息后 7 个时间单位内对该股票作出分析并将分析结果 report 发送给 Investor.Investor 收到 report 后的 3 个时间单位内决定继续或取消该服务.StockBroker 在接收到 ack 消息请求后,在 5 个时间单位内反馈 bill 消息给 Investor,若接收到的是 cancel

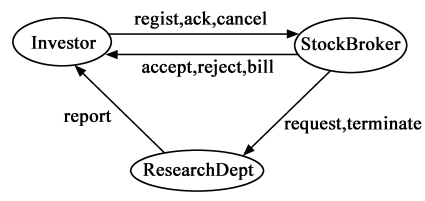


图9　SAS系统各端交互图

消息,则仍是在 5 个时间单位内发送 terminate 消息给 ResearchDept 结束整个服务.

以下使用 XYZ/ADL 分别对 Investor, StockBroker 和 ResearchDept 三个交互端的交互行为描述如下:

%COMPONENT Investor = =[

$LB = I0 \wedge \text{regist!} \Rightarrow \{0,5\}\ \$\ OLB = I1;$

$LB = I1 \wedge \text{reject?} \Rightarrow \$\ OLB = I0;$

$LB = I1 \wedge \text{accept?} \Rightarrow \$\ OLB = I2;$

$LB = I2 \wedge \text{report?} \Rightarrow \$\ O(LB = I3 \wedge x = 0);$

$LB = I3 \wedge \text{ack!} \Rightarrow \{0,3\}\ \$\ OLB = I4;$

$LB = I3 \wedge \text{cancel!} \Rightarrow \{0,3\}\ \$\ OLB = I5;$

$LB = I4 \wedge \text{bill!} \Rightarrow \$\ OLB = I0;$

$LB = I5 \Rightarrow \$\ OLB = I0;$]

%COMPONENT StockBroker = =[

$LB = S0 \wedge \text{regist?} \Rightarrow \$\ OLB = S1;$

$LB = S1 \wedge \text{accept!} \Rightarrow \{0,10\}\$\ OLB = S2;$

$LB = S2 \wedge \text{request!} \Rightarrow \$\ OLB = S3;$

$LB = S3 \wedge \text{ack?} \Rightarrow \$\ O(LB = S4 \wedge y = 0);$

$LB = S4 \wedge \text{bill!} \Rightarrow \{0,5\}\ \$\ OLB = S0;$

$LB = S3 \wedge \text{cancel?} \Rightarrow \$\ O(LB = S5 \wedge y = 0);$

$LB = S5 \wedge \text{terminate!} \Rightarrow \{0,5\}\ \$\ OLB = S0;$]

%COMPONENT ResearchDept = =[

$LB = R0 \wedge \text{request?} \Rightarrow \$\ O(LB = R1 \wedge z = 0);$

$LB = R0 \wedge \text{terminate?} \Rightarrow \$\ OLB = R0;$

$LB = R1 \wedge \text{report!} \Rightarrow \{0,7\}\ \$\ OLB = R0;$]

根据我们在文献[5]中提出的映射规则,将上述各个 Web 服务的 XYZ/ADL 描述中的实时描述部分 XYZ/RE 转换得到的时间自动机如图 10 所示,将此三个模型

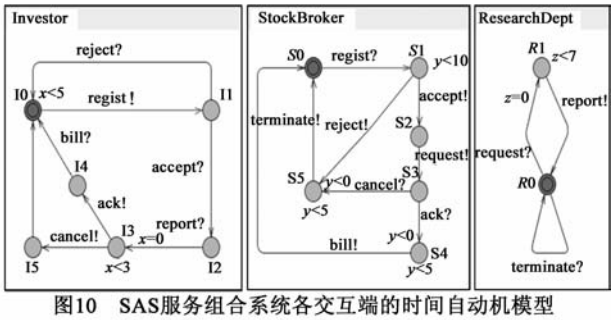


图10 SAS服务组合系统各交互端的时间自动机模型

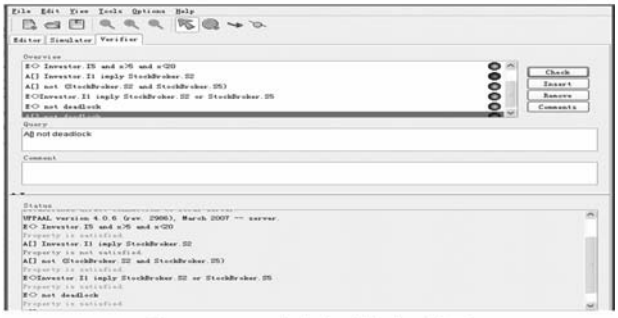


图11 SAS服务组合系统验证结果

输入到模型检测工具 UPPAAL 中,死锁、安全性、活性的验证结果如图 11 所示,分别用 CTL 公式表达如下:

(1)死锁:E< > not deadlock,验证结果为真,即该组合系统不存在死锁.

(2)安全性:A[ ] not (StockBroker.*S*2 and StockBroker.*S*5),验证结果为真,即 StockBroker 不可能同时到达接受和拒绝 regist 请求的状态.

(3)活性:E〈〉Investor.*I*1 imply StockBroker.*S*2 or StockBroker.*S*5,验证结果为真,即 Investor 若发出 regist 请求则 StockBroker 可能接受该请求,也可能拒绝该请求.

E< > Investor.I5 and $x > 5$ and $x < 20$,验证结果为真,即 Investor 可能在 5 ~ 20 个单位时间内完成从发出 regist 请求到取消结束服务的操作.

6 相关工作介绍与比较

当前,从软件体系结构角度研究 Web 服务组合的相关工作还处于探索阶段[9],其中文献[10]提出了需要通过挖掘发现 SOA 中基本要素来从软件体系结构的角度研究 Web 服务组合及其验证的观点,并且指出了组合服务体系结构的基本元素.文献[11]提出了一种专门描述 Web 服务组合系统的体系结构描述语言 WSC/ADL,但与现有的软件体系结构描述语言相比,该语言还不够成熟,也未给出如何对 Web 服务组合进行验证.

与上述工作相比,本文从软件体系结构角度出发,并考虑到 Web 服务组合中的实时特征,提出了一种基于软件体系结构描述语言 XYZ/ADL 和形式化方法结合的 Web 服务组合描述与验证框架,在采用 XYZ/ADL 描述 Web 服务组合的基础上,将其中的实时描述部分 XYZ/RE 转换至时间自动机,分别通过精化检验和模型检测两种技术对 Web 服务组合系统进行验证,从而在一定程度上避免实际运行后修正错误带来的昂贵代价.

7 结束语

精化检验与模型检测作为两种重要的形式验证技术,目前已在硬件电路设计、通信协议等领域中取得了广泛应用.本文采用这两种形式化验证技术与软件体系结构方法相结合描述和验证 Web 服务组合,在一定程度上保证了组合服务的质量;从相关工具支持来看,模型检测的自动化程度较高,而从性质规约的方式来看,CTL 公式的表达能力没有时间自动机的表达能力强,因此,将精化检验和模型检测两种技术结合,可取长补短.由于模型检测是对状态空间的穷尽搜索,无法避免状态爆炸问题,因此,如何对服务组合模型进行抽象约简,采用新的模型检测技术,以缓解状态爆炸问题、提高验证效率则是我们下一步要研究的内容之一;

另外,Web 服务中涉及到多种复杂机制如异常处理及补偿机制等,进一步的分析验证工作还将考虑这些问题.

参考文献

[1] 文艳军,王戟,齐志昌.并发反应式系统的组合模型检验与组合精化检验.软件学报[J].2007,18(6):1270 – 1281.
Wen Y J, Wang J, Qi ZC. Compositional model checking and compositional refinement checking of concurrent reactive systems[J]. Journal of Software, 2007, 18(6): 1270 – 1281. (in Chinese)

[2] 林惠民,张文辉.模型检测:理论、方法与应用[J].电子学报,2002,30(12A):1907 – 1912.
Lin H M, Zhang W H. Model checking: theories, techniques and applications[J]. Acta Electronica Sinica, 2002, 30(12A): 1907 – 1912. (in Chinese)

[3] 朱雪阳,唐稚松.基于时序逻辑的软件体系结构描述语言 XYZ/ADL[J].软件学报,2003,14(4):714 – 720.
Zhu X Y, Tang Z S. A temporal logic-based software architecture description language XYZ/ADL[J]. Journal of Software, 2003, 14(4): 714 – 720. (in Chinese)

[4] Zhang G Q, Rong M, Wang L, et al. Modeling and analysis for Web services composition based on dynamic software architecture[A]. Proceedings of the 6th International Symposium on Web Information Systems and Applications (WISA2009)[C]. Xuzhou, China: IEEE Computer Society, 2009. 122 – 125.

[5] Zhang G Q, Rong M, He Y L, et al. A refinement checking method of web services composition[A]. 5th IEEE Int'l Workshop on Service-Oriented System Engineering (SOSE2010)[C]. Nanjing, China: IEEE Computer Society, 2010. 103 – 106.

[6] Behrmann G, David A, Larsen KG. A tutorial on UPPAAL [A]. Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT2004)[C]. LNCS 3185, Springer-Verlag, 2004. 200 – 236.

[7] Diaz G, Pardo J, Cambronero M, et al. Automatic translation of WS-CDL choreographies to timed automata[A]. Int'l Workshop on Web Services and Formal Methods (WS-FM 2005)[C]. LNCS 3670, 2005. 230 – 242.

[8] Fu X, Bultan T, Su JW. Model checking interactions of composite web services[N]. UCSB Computer Science Department Technical Report, 2004.

[9] Vinoski S. WS-Nonexistent Standards[J]. IEEE Internet Computing, 2004, 8(6): 94 – 96.

[10] Milanovic N, Malek M. Architectural support for automatic service composition[A]. Proc IEEE Int'l Conf. Service Computing (SCC2005)[C]. Washington. DC. USA: IEEE Computer Society, 2005. 133 – 140.

[11] 杨鑫,陈俊亮.WSC/ADL:Web Services 组合系统体系结构描述语言[J].软件学报,2006,17(5):1182 – 1194.
Yang X, Chen J L. WSC/ADL: An architecture description language for web services composition system[J]. Journal of Software, 2003, 14(4): 714 – 720. (in Chinese)

作者简介

张广泉　男,1965 年生于江苏连云港.苏州大学计算机科学与技术学院教授、博士/博士后、CCF 高级会员.研究方向为软件工程与形式化方法、服务计算、网络与分布式计算等.
E-mail:gqzhang@suda.edu.cn

戎　玫　女,1966 年生于重庆.副教授、博士.研究方向为软件体系结构与 SOA、形式化方法与协议分析等.

朱雪阳　女,1971 年生于福建莆田.博士.研究方向为软件体系结构与形式化方法等.

何亚丽　女,1983 年生于河南漯河.硕士.研究方向为 Web 服务与形式化方法.

石慧娟　女,1986 年生于河南濮阳.硕士研究生.研究方向为 Web 服务与形式化方法.