Web服务行为一致性与相容性判定

殷昱煜,李 莹,邓水光,尹建伟
(浙江大学计算机学院,浙江杭州 310027)

摘 要: Web服务行为相容性和一致性是保证服务组装的正确性和可靠性的关键.然而,如何有效快捷的对其判定是一大难点.在深入分析服务行为特点的基础上,引入Martin-Löf类型论(简称:MTT),并针对其在服务动态行为描述上的不足进行有益的扩展.然后,基于扩展后MTT的相关理论,给出服务行为相容性和一致性的证明规则.最后,通过实例给出服务行为相容性和一致性判定过程,并分析其复杂度.

关键词: Web服务行为; 相容性; 一致性
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2009) 03-0433-06

Determining on Consistency and Compatibility of Web Services Behavior

YIN Yu-yu, LI Ying, DENG Shui-guang, YIN Jian-wei
(*Institute of Computer Science and Technology, ZheJiang University, Hangzhou, Zhejiang* 310027, *China*)

Abstract: Web services behavior is the key aspect to compatibility and consistency of web services, which can ensure correctness and reliability in web services choreography. But few methods can conduct the trade-off between expressiveness and amenability of efficient verification, this paper gives a better answer to solve the problem. It constructs services behavioral type discipline based on extended the Martin-Löf's type theory (for short, MTT) which supports a type-theoretic formulation of services behavior structured patterns, so that services behavior in a distributed system can be verified by type checking. Then, the type rules for subtype, duality, compatibility and consistency of web services behavior are discussed. The deductions are given to show that how to verify the consistency between behavior of vendor and behavior of vendor-s. Finally, the complexity of the deduction is given.

Key words: web services behavior; consistency; compatibility

1 引言

Web服务作为互联网中的一种新的计算资源形态,在电子商务、企业应用集成等领域扮演着越来越重要的角色.目前,Web服务模型已从最初的黑盒式接口模型,如WSDL模型,发展为具有多视图的模型,如OWL-S (OWL Web Ontology Language for Services)和WSMO (Web Service Modeling Ontology)等.后者包含用于描述Web服务各个侧面的多个视图,如OWL-S既包括用于注册和发现的服务概要信息(Service Profile),也包含描述服务内部实现细节的流程模型.随着新型模型的出现,许多研究者指出,服务不仅包含静态的语法、语义等信息,还包含内部控制流、数据流、交互协议、状态变迁等动态行为属性[1].这一观点伴随服务组合技术的发展越发受到重视.而服务行为一致性和相容性是保证服务组合可靠性和正确性的关键.目前,服务行为一致性和相容性的判定已得到关注,但这些工作大多基于Petri网[2]或者自动机理论[3]展开,一旦服务行为、服务间的交互过程变得错综复杂,这些方法都将出现空间爆炸,从而导致计算和判定的复杂度增大.为此,本文引入MTT这种形式化方法,借鉴其证明即程序(proof-as-programming)的思想来完成Web服务行为一致性和相容性的分析及验证,可有效减少验证过程的复杂度.

本文的结构如下:第二部分,分析服务行为结构;第三部分,对MTT进行扩展,并给出服务行为的构造方法;第四部分,提出服务相容性与一致性的基本规则.并给出它们的推演过程,而后分析验证过程的复杂度;第五部分,介绍本文的相关工作;第六部分,给出结论并展望下一步工作.

2 Web服务行为分析

服务外部接口可理解为消息的输入和输出,而服务内部的逻辑流程可理解为消息驱动下的状态转移.本文给出商品贩卖服务(*Vendor*)的例子,以此来分析服务

收稿日期:2008-05-05;修回日期:2008-10-10
基金项目:国家自然科学基金(No.60703042);国家863高技术研究发展计划(No.2006AA01Z171,2007AA01Z124);国家科技支撑计划重大项目(No.2006BAH02A01);浙江省自然科学基金(No.Y106045)

行为,如图 1. 图中“ + ”和“- ”分别表示输入和输出消息. 该服务的业务流程为:在接收到买家的登录系统请求消息后验证买家身份,如认证不通过则返回登录不成功的消息,否则,登录成功,买家可以选购商品;在这之后会有三种情况发生:其一,商品以所提供的价格被卖出,流程结束;其二,商品没被卖出,流程结束;其三,服务收到一个低于本身报价的价格,此时又产生两种情况:买家以该价格买到商品,流程结束;该价格低于商品的最低价格,流程被终止.

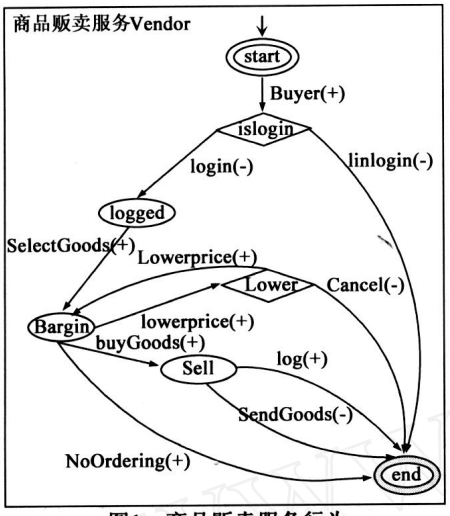


图1 商品贩卖服务行为

由此例可以看出,服务的内部流程逻辑包含如下几种主要的结构:顺序结构,如登录和选购商品是有先后顺序的;内部选择结构,如接收到买家的登录系统请求消息后验证买家的身份,则出现两种情况,如果认证不通过则返回登录不成功消息,否则,登录成功. 那么,内部选择结构是服务根据外部提供的信息而自动做出的选择,是一种主动的选择结构. 外部选择结构,如在买家选择商品时可能出现三种情况,买家不买商品并退出;买家按默认的价格购买商品并退出;买家不满意默认的价格开始还价. 那么,外部选择结构是外部信息直接导致服务做出一种选择. 因此,外部选择结构是一种被动的选择结构;递归结构,如本例中的商品议价操作是一种递归结构,买家可不断的出价直到最低价格.

3 类型论与 Web 服务行为

MTT 作为一种形式化方法,其特点是将程序的构造过程看作命题的证明过程. 它作为程序构造的理论,能够同时表示程序的规格说明和程序本身. 而且应用证明规则能够完成从规格说明到一个正确的程序的构造以及能检验一个给定程序具有某个性质. 因篇幅所限,MTT 的详细介绍参见文献[4].

3.1 MTT 的扩展

在以前的工作中[5~7]对服务行为兼容性进行研究,并使用 MTT 解决服务匹配问题. 故引入 MTT 到 Web 服务技术有助于解决其一些关键问题,比如:服务描述、服务匹配、服务组装及验证等. 但也发现 它虽可对 Web 服务的静态属性,如:功能、接口等,进行准确表达,但不能对 Web 服务的交互行为进行描述. 因此,如何扩展 MTT 使其支持 Web 服务的一致性和相容性研究是一个重要问题.

本文将序列作为构造服务行为的一个基本结构,其可以表示一个或多个数据类型或对象的以某种顺序输入或输出. MTT 并没有给出序列的概念. 本节提出序列的概念,序列中各种元素是具有顺序关系的,而且其中的元素是有重复性的.

为了形成序列集合,首先给定 n 个不同的集合 $A_1\mathrm{Set}, \cdots, A_n\mathrm{Set}$,然后我们引入三个新常元:相关度(arity)为 $0 \otimes \cdots \otimes 0 \multimap 0$ 的常元 Seq,相关度为 0 的 nil 以及相关度为 $0 \otimes 0 \multimap 0$ 的 cons. 我们将按照一个元素在某序列的首次出现来决定该元素在序列中的位置. 下面给出形成序列集的规则.

形成规则 $\dfrac{A_1\mathrm{Set}, \cdots, A_n\mathrm{Set}}{\mathrm{Seq}(A_1, \cdots, A_n)\ \mathrm{Set}}$

通过形成规则可以得到序列集 $\mathrm{Seq}(A_1, \cdots, A_n)\ \mathrm{Set}$,记为 $\mathrm{Seq}(A_1, \cdots, A_n)$.

为了能够在构造序列时使用中置操作,定义:$l.\ a \equiv \mathrm{cons}(l, a)$,中置操作 cons 表示在一个序列的队尾加入一个元素.

引入规则 $\dfrac{\mathrm{nil} \in \mathrm{Seq}(A_1, \cdots, A_n),\quad a \in A_1, \cdots, A_n \quad l \in \mathrm{Seq}(A_1, \cdots, A_n)}{l.\ a \in \mathrm{Seq}(A_1, \cdots, A_n)}$

引入相关度为 $0 \otimes 0 \otimes (0 \otimes 0 \otimes 0 \multimap 0) \multimap 0$ 的原始非典则常元 Seqrec 去表示序列集上的递归. 表达式 $\mathrm{Seqrec}(l, c, e)$ 被计算如下:

(1) 首先计算 l 的值;

(2) 若 l 的值是 nil,则 $\mathrm{Seqrec}(l, c, e)$ 的值是 c 的值. 若 l 的值是 $l_{1.a}$,则 $\mathrm{Seqrec}(l, c, e)$ 的值是 $e(l_1, a, \mathrm{Seq}(l_1, c, e))$ 的值.

消去规则

$$\frac{l \in \mathrm{Seq}(A_1, \cdots, A_n)\quad C(v)\ \mathrm{set}[v \in \mathrm{Seq}(A_1, \cdots, A_n)]\quad c \in C(\mathrm{nil})\quad e(x, y, z) \in C(x.\ y)[x \in \mathrm{Seq}(A_1, \cdots, A_n)], y \in A_1 \cup \cdots \cup A_n, z \in C(x)}{\mathrm{Seqrec}(l, c, e) \in C(l)}$$

证明 设 $l \in \mathrm{Seq}(A_1, \cdots, A_n)$, $c \in C(\mathrm{nil})$, $e(x, y, z) \in C(x.\ y)[x \in \mathrm{Seq}(A_1, \cdots, A_n), y \in A_1 \cup \cdots \cup A_n, z \in C(x)]$.

(1) 若 l 的值是 nil,则根据 Seqrec 的计算规则,Seqrec(l, c, e) 的值为 c 的值,由第二个前提知,此值为 C(nil) 的典则元. 故 $C(l) = C(\text{nil})$,则 Seqrec(l, c, e) 的值是 $C(l)$ 的典则元.

(2) 若 l 的值是 $x.y$,其中 $x \in \text{Seq}(A_1, \cdots, A_n)$, $y \in A_1 \cup \cdots \cup A_n$,则 Seqrec($l$, c, e) 的值为

$$e(x, y, \text{Seq}(x, c, e)) \qquad (a)$$

的值. 那么现在要证明 Seqrec(x, c, e) $\in C(x)$. 若它得证,则从第三个前提知,式(a) 的值是 $C(x)$ 的典则元,故其也是 $C(l)$ 的典则元. 为了证明 Seqrec(x, c, e) $\in C(x)$,先求 x 的值再计算 Seqrec(x, c, e). x 的值为 nil 或 $x_1.y_1$,其中 $x_1 \in \text{Seq}(A_1, \cdots, A_n)$, $y_1 \in A_1 \cup \cdots \cup A_n$.

(1) 若 x 的值是 nil,则同 1 理可得 Seqrec(x, c, e) 的值是 $C(x)$ 的典则元.

(2) 若 x 的值是 $x_1.y_1$,则同 2 理可得 Seqrec(x_1, c, e) 的值是 $C(x_1)$ 的典则元.

所有 $x \in \text{Seq}(A_1, \cdots, A_n)$ 是由中置操作 cons 作用有穷次而得,故此证明过程将终止于 x 的值是 nil. □

最后,本文给出相等性规则,Seqrec 的计算规则可以对其论证,过程类似于消去规则,故略.

相等性规则 1

$$\frac{l \in \text{Seq}(A_1, \cdots, A_n)\quad C(v)\,\text{set}[v \in \text{Seq}(A_1, \cdots, A_n)]\quad c \in C(\text{nil})\quad e(x, y, z) \in C(x.y)[x \in \text{Seq}(A_1, \cdots, A_n)], y \in A_1 \cup \cdots \cup A_n, z \in C(x)}{\text{Seqrec}(\text{nil}, c, e) = c \in C(\text{nil})}$$

相等性规则 2

$$\frac{a \in A_1, \cdots, A_n\quad l \in \text{Seq}(A_1, \cdots, A_n)\quad C(v)\,\text{set}[v \in \text{Seq}(A_1, \cdots, A_n)]\quad c \in C(\text{nil})\quad e(x, y, z) \in C(x.y)[x \in \text{Seq}(A_1, \cdots, A_n)], y \in A_1 \cup \cdots \cup A_n, z \in C(x)}{\text{Seqrec}(l.a, c, e) = e(l, a, \text{Seqrec}(l, c, e)) \in C(l.a)}$$

3.2 用 MTT 构造服务行为

3.2.1 消息构造

服务需要接收消息以触发服务内部流程的运转. 而消息是由一个或多个参数组成的序列,参数可以属于不同的数据类型,比如:简单数据类型 Int, String 等,复杂数据类型,比如对象. 故先给出数据类型的定义.

简单数据类型. 简单数据类型可以用枚举集合来构造,由其形成规则得到简单数据类型:{bool, int, string, ⋯} set,记为:Stype.

复杂数据类型. 复杂数据类型可以由简单数据类型构造,由列表集合的形成规则得到:

$$\frac{\text{Stype}}{\text{List(Stype) Set}}$$

,故复杂数据类型为 List(Stype) Set,记为:Ctype.

消息序列集. 基于序列集的形成规则,给出消息的构造:

$$\frac{\text{Stype set}\quad \text{Ctype set}}{\text{Seq(Stype, Ctype) Set}}$$

则消息序列集为 Seq(Stype, Ctype) Set,简写为 Seq(Stype, Ctype).

在消息序列集中,引入相关度为 0 ⊗0 →0 的常元 ↓ 和 ↑,分别表示接收消息操作和发送操作操作,那么,接收消息序列可以表示为 $\downarrow l[l \in \text{Seq(Stype, Ctype)}]$,而发送消息序列可以表示为 $\uparrow l[l \in \text{Seq(Stype, Ctype)}]$.

3.2.2 服务行为构造

第一节分析了服务行为结构. 顺序结构可以直接通过序列集构造. 选择结构中的元素特点与 MTT 中枚举集合的元素关系类似,同时 MTT 也给出了枚举集合的选择子,选择子表达式的作用是根据元素的位置得到集合中的相应元素,故可利用它来表达选择结构,其表达式为 $\text{case}_{\{i_1, \cdots, i_n\}}(a, b_1, \cdots, b_n)$,这里 $\text{case}_{\{i_1, \cdots, i_n\}}$ 是相关度(arity) 为 0 ⊗ ⋯ ⊗0 →0 的常元. 选择子表达式为的意义为 case a of $i_1 \Rightarrow b_1 | \cdots | i_n \Rightarrow b_n$,其计算方式:如果 $a = i_k$,那么 $\text{case}_{\{i_1, \cdots, i_n\}}(a, b_1, \cdots, b_n) = b_k$. 但无法区分内部选择结构和外部选择结构两种行为. 因此,本文在枚举集合中引入相关度为 0 ⊗ ⋯ ⊗0 →0 的常元 ⊕和 +,分别表示内部选择子和外部选择子,其表达式分别为 $\oplus\text{case}_{\{i_1, \cdots, i_n\}}(a, b_1, \cdots, b_n)$ 和 $+\,\text{case}_{\{i_1, \cdots, i_n\}}(a, b_1, \cdots, b_n)$,通常我们不写出它们的下标,因为下标可以通过上下文确定,故它们的表达式可简写为 $\oplus\text{case}(a, b_1, \cdots, b_n)$ 和 $\text{case}(a, b_1, \cdots, b_n)$. 它们的计算方法同选择子表达式为 $\text{case}(a, b_1, \cdots, b_n)$. 通过 ⊕和 + 构造服务内部行为中的内部选择结构和外部选择结构.

递归结构是描述服务内部业务流程不可或缺的结构. 在序列集中引入相关度为 0 ⊗0 →0 非典则常元 rec 来表示递归行为和相关度为 0 的典则常元 t. 表达式 rec(t, A) 表示对象进行 A 规定的操作,如果遇到 t,则再次进行 A 规定的操作. 同时引入相关度为 0 ⊗0 →((0 ⊗0) ⊗0) 的非典则常元 unfold 去表示 rec(t, A) 的展开,其计算方法为:$\text{unfold}(\text{rec}(t, A)) = T(\text{rec}(t, A))/t)$. 之后,引入两个相关度为 0 的终结符 ⊥和 τ,分别表达结束和空操作.

服务行为构造语法:

$$A = \{l[l \in \text{Seq}(A)],\ \oplus\{a, t_1, \cdots, t_n\}[t_1, \cdots, t_n \in A], +\{a, t_1, \cdots, t_n\}$$

$[t_1, \cdots, t_n \in A]$,

$\uparrow l\,[l \in \mathrm{Seq}(\mathrm{Stype}, \mathrm{Ctype})]$, $\downarrow l\,[l \in \mathrm{Seq}(\mathrm{Stype}, \mathrm{Ctype})]$,

$\mathrm{Rec}(t, A)$, $\perp, t, \tau\}$

表 1 服务行为的替换规则

规则 1(序列替换)	$t, s \in \mathrm{Seq}(A)$ $t = l_1.\ a[l_1 \in \mathrm{Seq}(A), a \in A]$ $s = l_2.\ b[l_2 \in \mathrm{Seq}(A), b \in A]$ $a \prec b$ $\underline{\mathrm{Seqrec}(l_1, c, e) \prec \mathrm{Seqrec}(l_2, c, e)}$ $t \prec s$
规则 2(内部选择替换)	$\underline{t_i \prec s_i}$ $\oplus(a, t_i)\,[a, t_i \in A] \prec \oplus(a, s_i)\,[a, s_i \in A]$
规则 3(外部选择替换)	$\underline{t_i \prec s_i}$ $+(a, t_i)\,[a, t_i \in A] + (a, s_i)\,[a, s_i \in A]$
规则 4(接收序列替换)	$t = l_1.\ a[l_1 \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype}), a \in \mathrm{Ctype} \cup \mathrm{Stype}]$ $s = l_2.\ b[l_2 \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype}), b \in \mathrm{Ctype} \cup \mathrm{Stype}]$ $a \prec b$ $\underline{\mathrm{seqrec}(l_1, c, e) \prec \mathrm{seqrec}(l_1, c, e)}$ $\downarrow t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] \prec \downarrow s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]$
规则 5(发送序列替换)	$t = l_1.\ a[l_1 \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype}), a \in \mathrm{Ctype} \cup \mathrm{Stype}]$ $s = l_2.\ b[l_2 \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype}), b \in \mathrm{Ctype} \cup \mathrm{Stype}]$ $a \prec b$ $\underline{\mathrm{seqrec}(l_1, c, e) \prec \mathrm{seqrec}(l_1, c, e)}$ $\uparrow t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] \prec \uparrow s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]$
规则 6(递归结构 L 替换)	$s \in \mathrm{Seq}(A)$ $\underline{\mathrm{unfold}(\mathrm{rec}(t, T)) \prec s}$ $\mathrm{rec}(t, A) \prec s$
规则 7(递归结构 R 替换)	$s \in \mathrm{Seq}(A)$ $\underline{s \prec \mathrm{unfold}(\mathrm{rec}(t, T))}$ $s \prec \mathrm{rec}(t, A)$
规则 8(空操作替换)	$A \prec \tau$

服务 Vendor 的行为:

$\mathrm{Vendor} = (\downarrow(\mathrm{String}, \mathrm{int})\,[(\mathrm{string}, \mathrm{int}) \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]$,

$\oplus(a, (\uparrow(\mathrm{true})\,[\mathrm{true} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], T[T\mathrm{Seq}(A)]), (\uparrow(\mathrm{false})\,[\mathrm{false} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \perp))\,[(\uparrow(\mathrm{true})\,[\mathrm{true} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], T[T \in \mathrm{Seq}(A)]) \in \mathrm{Seq}(A), (\uparrow(\mathrm{false})\,[\mathrm{false} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \perp) \in \mathrm{Seq}(A)]$

$T = (\downarrow(\mathrm{String}, \mathrm{float})\,[(\mathrm{string}, \mathrm{float}) \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]$,

$T\,[T' \in \mathrm{Seq}(A)])\,[(\downarrow(\mathrm{String}, \mathrm{float})\,[(\mathrm{string}, \mathrm{float}) \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], T\,[T' \in \mathrm{Seq}(A)]) \in \mathrm{Seq}(A)]$

$T' = \mathrm{rec}(t, R)$

$R = +(a, (\uparrow(\mathrm{string})\,[\mathrm{string} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \perp), \perp, B)\,[(\uparrow(\mathrm{string})\,[\mathrm{string} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \perp) \in \mathrm{Seq}(A), B \in \mathrm{Seq}(A)]$

$B = \oplus(a, (\downarrow(\mathrm{float})\,[\mathrm{float} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \uparrow(\mathrm{string})\,[\mathrm{string} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], \perp), \perp, S)\,[(\uparrow(\mathrm{string})\,[\mathrm{string} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], T[T \in \mathrm{Seq}(A)]) \in \mathrm{Seq}(A), S \in \mathrm{Seq}(A)]$

$S = +(a, (\downarrow(\mathrm{float})\,[\mathrm{float} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], t), \perp)\,[(\downarrow(\mathrm{float})\,[\mathrm{float} \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})], t) \in \mathrm{Seq}(A)]$

4 服务行为一致性与相容性判定

服务行为的相容性是指当两个服务组合起来以后,它们是否能够正常交互;服务行为的一致性是指一个服务能否替换另一个服务,以取代其继续进行工作.本节将给出服务行为的相容性与一致性的判定规则.

4.1 服务行为相容性与一致性判定规则

若要判断两个服务行为是否具有一致性或相容性,首先需要判断两个服务行为序列中相同位置的结构所表达的原子结构所表达的行为是否具有一致性或相容性.本文引入符号"$\prec$"表示替换,如:表达式 $t \prec s$ 表示 t 替换 s.服务行为各种结构的替换规则,如表 1.

定义 1(行为一致性) 设 $T_i, S_i \in \mathrm{Seq}(A)$,若 $T_i \prec S_i$,则 T_i 可替换 S_i 记作 $T_i \cong S_i$.

给出行为相容性的定义,本文引入单个行为的对偶,并给出了如下的对偶规则,如表 2.

表 2 对偶规则

接收	$\underline{t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] \prec s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]}$ $\overline{\downarrow t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] = \uparrow s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]}$
发送	$\underline{t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] \prec s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]}$ $\overline{\uparrow t[t \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})] = \downarrow s[s \in \mathrm{Seq}(\mathrm{Ctype}, \mathrm{Stype})]}$
内部选择	$\underline{t_i = s_i[t_i, s_i \in \mathrm{Seq}(A)]}$ $\overline{\oplus(a, t_i)\,[t_i \in \mathrm{Seq}(A)] = +(a, s_i)\,[s_i \in \mathrm{Seq}(A)]}$
外部选择	$\underline{t_i = s_i[t_i, s_i \in \mathrm{Seq}(A)]}$ $\overline{+(a, t_i)\,[t_i \in \mathrm{Seq}(A)] = \oplus(a, s_i)\,[s_i \in \mathrm{Seq}(A)]}$
递归-L	$\underline{s \in \mathrm{Seq}(A)\ \ s = \overline{\mathrm{unfold}(\mathrm{rec}(t, T))}}$ $\overline{\mathrm{rec}(t, T) = s}$
递归-R	$\underline{s \in \mathrm{Seq}(A)\ \ s = \mathrm{unfold}(\mathrm{rec}(t, T))}$ $\overline{s = \mathrm{rec}(t, A)}$

定义 2(行为相容性)

设 $T_i, S_i \in \mathrm{Seq}(A)$,若 $T_i \prec U_i, S_i = U_i$,则 T_i 与 S_i 兼容,记作 $T_i \approx S_i$.

4.2 服务行为推演实例

考虑如下场景,在 Vendor-s 和 Client 组成的复合服务 S 中,如果 Vendor-s 突然停止运行,那么将导致 S 不可用,故需要找到可以接替 Vendor-s 工作的服务,并与 Client 兼容,使服务 S 正常运行.假设存在 Vendor,该服务与 Vendor-s 的区别在于其具备议价功能.如图 2 描述

了 Vendor-*s* 和 Client 的交互.

Vendor-*s* 行为形式化表达:

Vendor-*s* = (↓(string ,int) [(string ,int) ∈Seq(Ctype , Stype)] , ⊕(*a* , (↑(true) [true ∈Seq(Ctype ,Stype)] , T_1[T_1 ∈Seq(*A*)]) , (↑(false) [false ∈Seq(Ctype , Stype)] , ⊥)) [(↑(true) [true ∈Seq(Ctype ,Stype)] , T_1[T_1 ∈Seq(*A*)]) ∈Seq(*A*) , (↑(false) [false ∈Seq (Ctype ,Stype)] , ⊥) Seq(*A*)]

T_1 = (↓(string ,float) [(string ,float) ∈Seq(Ctype , Stype)] , S_1[S_1 ∈Seq(*A*)])

S_1 = + (*a* , (↑(string) [string ∈Seq (Ctype , Stype)] , ⊥) , ⊥) [(↑(string) [string ∈Seq (Ctype ,Stype)] , ⊥) ∈Seq(*A*)]

Client 行为形式化表达:

Client = (↑(string ,int) [(string ,int) ∈Seq(Ctype ,Stype)] , + (*a* , (↓(true) [true ∈Seq(Ctype , Stype)] , *S* [*S* ∈Seq (*A*)]) , (↓(false) [false ∈Seq(Ctype , Stype)] , ⊥)) [(↓ (true) [true ∈Seq(Ctype , Stype)] , *s* [*s* ∈Seq(*A*)]) ∈Seq (*A*) , (↓(false) [false ∈Seq(Ctype , Stype)] , ⊥) ∈Seq (*A*)]

S = (↑(string ,float) [(string ,float) ∈Seq(Ctype ,Stype)] , *s* [*s* ′∈Seq(*A*)])

S ′= ⊕(*a* , (↓(string) [string ∈Seq(Ctype ,Stype)] , ⊥) , ⊥) [(↓(string) [string ∈Seq(Ctype , Stype)] , ⊥) ∈ Seq(*A*)]

4.2.1　Vendor 和 Vendor-s 行为一致性证明

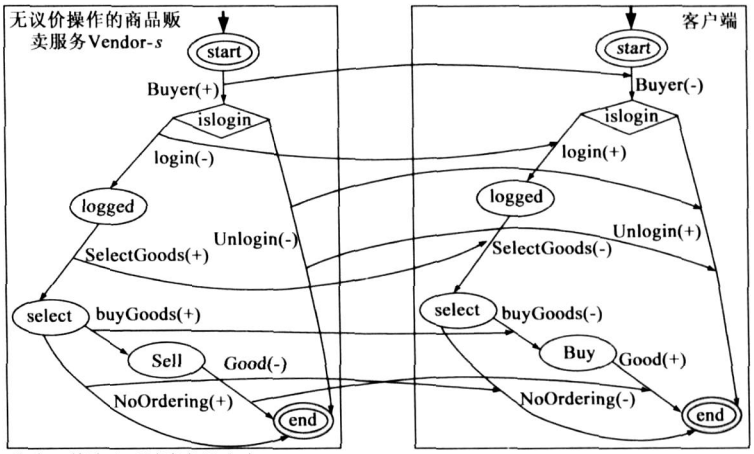
根据定义 1,判断 Vendor 和 Vendor-*s* 的一致性,即要证 Vendor ≺Vendor-*s* [Vendor , Vendor-*s* ∈Seq(*A*)] 成立. 根据规则 1 ,得到 Seqrec(Vendor , *c* , *e*) ≺Seqrec(Vendor-*s* , *c* , *e*) [Vendor ,Vendor-*s* ∈Seq(*A*)] ,然后依次使用规则 1 ,2 ,1 ,3 和 6 可得 unfold(*T* [*T* ′∈Seq(*A*)]) ≺ S_1 [S_1 ∈Seq(*A*)]. 经过递归计算得 *R*[*R* ∈Seq(*A*)]) ≺ S_1 [S_1 ∈Seq(*A*)]. 最后,可通过规则 3 和 8 得证.

由此得 Seqrec (Vendor , *c* , *e*) ≺Seqrec (Vendor-*s* , *c* , *e*) ,从而命题得证. □

4.2.2　Vendor 和 Client 行为相容性证明

由定义 2,需证明 Vendor≈Client[Vendor ,Client ∈Seq (*A*)]成立,即找到一个 *U*,其满足 Vendor ≺*U*[Vendor , *U* ∈Seq(*A*)] 和 *U* = Client [Vendor , *U* ∈Seq(*A*)]. 故求 Client 服务的对偶,则发现:Vendor-*s* = Clinet ,又因 4.2.1 节中已证明了命题 Vendor ≺Vendor-*s* ,故 Vendor 就是满足条件的 *U*. 则命题 Vendor≈Client[Vendor ,Client ∈Seq (*A*)]成立. □

以上两个证明过程可以用工具 Coq[8]实现. 限于篇幅,故不做详述.



注意:箭头表示消息传递方向

图2　服务Vendor-*s*与Client间的交互

4.3　证明过程复杂度分析

本文给出基于子类型理论的服务行为替换规则和基于类型对偶的相容性规则. 对于子类型和类型对偶判定都是可判定性问题. Brandt 和 Henglein[9]使用 Co-Inductive 方法得到了类型等价(类型对偶可看作判定类型等价)和子类型判定的证明,其复杂度为 $O(n^2)$. 对于本文的判定服务行为相容性的方法而言,需要先判断行为是否对偶,复杂度为 $O(n^2)$,然后证明行为的一致性,复杂度为 $O(n^2)$. 故本文的判定服务行为相容性和一致性方法的复杂度为 $O(n^2)$.

5　相关工作

服务行为一致性和相容性是保证服务组合可靠性和正确性的重要问题. 目前,一些形式化方法已被使用. 李宣东教授等[3]用接口自动机为构件的行为建模. 以上基于 Petri 网和自动机的方法在服务行为的描述上采用图形方式较为直观,但当服务行为和服务间的交互过程变得复杂时,这种方式容易出现状态空间爆炸,其计算和验证的复杂度也随即增加. 吕建教授等[10]提出用 Petri 网对服务行为和服务质量进行统一建模的框架. 因此,有研究人员提出采用进程代数的方法(如通信系统演算 CCS、通信顺序进程 CSP 和 Pi 演算)来刻画服务行为[2,5,11]. 另外,北京大学裘宗燕教授等[12]引入类型系统来验证 Web 服务组合描述语言(WS-CDL)所描述的服务组装需求规格的正确性. 国防科技大学陈火旺教授等[13]提出一门新的面向构件语言 SAJ ,把构件 ,端口 ,连接子等软件体系结构概念引入到 SAJ 中. 复旦大学钱乐秋教授等[14]根据构件交互过程,借鉴 Pi 演算的类型系统和进程构造方法,提出构件交互的类型系统和基于交互的构件模型.

6　结束语

服务行为相容性和一致性是保证服务组合质量和

健壮性的重要方面.本文在分析服务行为特点的基础上,通过对 MTT 的扩展,给出了服务行为相容性和一致性的判定方法.本文特色主要体现在三个方面:(1)引入 MTT 作为 Web 服务的数学基础,并对 MTT 进行扩展,使其具有较强的服务行为表达能力;这样在正确表达服务行为的同时,避免了状态空间爆炸的缺陷;(2)基于扩展 MTT 及类型的特点,提出服务行为的相容性和一致性判定规则,使其验证复杂度降低.这对于服务的动态组合与动态演化正确性判定具有重要意义.

本文的工作在将来主要集中在:本文虽对 MTT 进行扩展但只能对两个服务间的行为的相容性和一致性的验证,不能研究多构件间的行为,因此对 MTT 进行扩展的研究还需继续.

参考文献:

[1] Bultan T, Fu X, Hull R, Su J. Conversation specification: A new approach to design and analysis of E-service composition[A]. In: Proceedings of the 12th International World Wide Web Conference [C]. Budapest, HUNGARY: Springer-Verlag, 2003. 403 - 410.

[2] Brogi A, Canal C, Pimentel E, Vallecillo A. Formalizing web service choreographies [J]. Electronic Notes in Theoretical Computer Science, 2004, 105(12): 73 - 49.

[3] 张岩,胡军,于笑丰,张天,李宣东,郑国梁. 场景驱动的构件行为抽取[J]. 软件学报, 2007, 18 (1): 50 - 61.
Hu Jun, Zhang Yan, Yu Xiao-fei, Zhang Tian, Li Xuan-dong, Zheng Guo-liang. Scenario-driven component behavior derivation[J]. Journal of Software, 2007, 18(1): 50 - 61. (in Chinese)

[4] Bengt Nordstrom, Kent Petersson, Jan M. Smith. Programming in Martin-Löf Type Theory: An Introduction[M]. Oxford: Oxford University Press, 1999.

[5] 邓水光,李莹,吴健,邝砾,吴朝晖. Web 服务行为兼容性的判定与计算[J]. 软件学报, 2007, 12(18): 3001 - 3014.
DENG Shui-guang, Li Ying, WU Jian, KUANG Li, WU Zhao-hui. Determination and Computation of Behavioral Compatibility for Web Services[J]. Journal of Software, 2007, 12(18): 3001 - 3014. (in Chinese)

[6] YuYu Yin, Ying Li, Shuiguang Deng, Kuangli, Jian Wu, XuXue Sun, Jian Jiang, ZhaoHui Wu. Automating service matchmaking using type theory[A]. Proceedings of the 2007 International Conference on Services Computing[C]. Salt Lake City, Utah, USA: Springer-Verlag, 2007. 723 - 724.

[7] Zhiwei Chen, Jian Wu, Shuiguang Deng, Ying Li, Zhaohui Wu. Describing and verifying web service using type theory [A]. Proceedings of the 10th International Conference Computer Supported Cooperative Work in Design[C]. NanJing, China: Springer-Verlag, 2006. 1 - 5.

[8] The Coq Development Team. The Coq Proof Assistant Reference Manual (Version 8.1) [EB/OL]. http://coq.inria.fr/V8.1pl3/refman/index.html, 2006.

[9] Brandt M, Henglein F. Coinductive axiomatization of recursive type equality and subtyping[J]. Fundamenta Informaticae, 1998, 33(4): 309 - 338.

[10] 胡昊,殷琴,吕建. 虚拟计算环境中服务行为与质量的一致性[J]. 软件学报, 2007, 18(8): 1943 - 1957.
HU Hao, YIN Qin, LÜJian. Service behavior and quality consistency in virtualized computing environment[J]. Journal of Software, 2007, 18(8): 1943 - 1957. (in Chinese)

[11] Bordeaux L, Salaün G. Using process algebra for Web services: Early results and perspectives[A]. Proceedings of the 5th VLDB Workshop on Technologies for E-Services[C]. Toronto: Springer-Verlag, 2004. 54 - 68.

[12] Hongli Yang, Xiangpeng Zhao, Zongyan Qiu, Geguang Pu, Shuling Wang. A formal model for web service choreography language (WS-CDL)[A]. Proceedings of the 5th International Conference on Web Services[C]. Chicago, USA: Springer-Verlag, 2006. 893 - 894.

[13] 陈波,谭庆平,李舟军,陈火旺. 一门新的面向构件语言[J]. 电子学报, 2006, 34(S1): 130 - 134.
CHEN Po, TAN Qing-ping, LI Zhou-jun, CHEN Huo-Wang. A new component-oriented programming language[J]. Acta Electronica Sinica, 2006, 34(S1): 130 - 134. (in Chinese)

[14] 龚洪泉,赵文耘,徐如志,钱乐秋. 基于 Pi 演算的构件演化研究[J]. 电子学报, 2004, 32(S1): 242 - 246.
GONG Hong-quan, ZHAO Wen-gen, XU Ruo-zhi, QIAN Le-qiu. A research on Pi-calculus based component evolution[J]. Acta Electronica Sinica, 2004, 32(S1): 242 - 246. (in Chinese)

作者简介:

殷昱煜 男,1980 年出生于新疆乌鲁木齐,浙江大学计算机学院,博士生.主要研究领域:Web 服务技术、中间件技术、构件技术、形式化方法.
联系电话:13083961587.
Email: yinyuyu@zju.edu.cn

李 莹 男,1973 年出生于浙江衢州,博士,浙江大学计算机学院,副教授.主要研究领域:Web 服务技术、中间件技术、构件技术、软件体系结构、程序理论及形式化方法.本文的通信作者.
Email: cnliying@zju.edu.cn