描述语义Web服务的带类型的OWL-S

刘 超[1,2],眭跃飞[1],曹存根[1]
LIU Chao[1,2], SUI Yuefei[1], CAO Cungen[1]
1.中国科学院 计算技术研究所 智能信息处理重点实验室,北京 100190
2.中国科学院 研究生院,北京 100049
1.Key Lab of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing 100190, China
2.Graduate University of Chinese Academy of Sciences, Beijing 100049, China

LIU Chao, SUI Yuefei, CAO Cungen. Features of typed OWL-S for semantic Web services. Computer Engineering and Applications, 2012, 48(22):58-64.

Abstract: Semantic Web services make the Web services computer-interpretable by using markup technology, enabling users and software agents to discover, invoke and compose Web resources. The current works and the existing problems of the semantic Web services are analyzed in this paper, and a typed first-order dynamic logic is proposed to resolve these problems. This paper presents an approach to formalizing the relationships between the types, objects and concepts being used for markup properties and relations of two Web services when two services are composed together. The approach also provides effective supports for service composition verification.
Key words: semantic Web service; OWL-S; type; first-order dynamic logic; service validity

摘 要: 语义Web服务利用语义Web中的语义标注技术使得Web服务可以被计算机所理解,从而实现Web服务的发现、调用、组合等的自动化。分析了目前语义Web服务的研究现状和存在的问题,从类型标注的角度出发,用一种带类型的OWL-S来形式化语义Web服务,并且形式地陈述了标注两个被组合的服务的属性和关系的类型、对象、本体概念之间的联系,从而为服务组合提供了一种形式化的验证方法。
关键词: 语义Web服务; OWL-S; 类型; 一阶动态逻辑; 服务验证
文章编号: 1002-8331(2012)22-0058-07 **文献标识码:** A **中图分类号:** TP182

1 引言

在语义Web服务[1-7]的相关研究工作中,以Web服务的组合、验证[8-9]为目的,已经有大量的研究人员和研究机构对其进行了广泛的研究,其中对Web服务的形式化以及组合验证的相关方法包括:自动机理论;进程代数(π演算)和Petri网。其中,由于Petri网采用了基于状态的形式化建模方法,具有直观的图形表示方法、严格的形式化语义,而且提供了系统分析验证的手段,因此作为一种形式化方法被广泛地使用到Web服务组合的形式化的描述和验证中[10]。文献[11]采用有限状态自动机对Web服务进行建模,验证了两个服务之间协作的正确性。π演算[12]能够描述服务的动态行为,刻画服务交互行为,因而也被用于语义Web服务的组合和验证的任务。基于Petri网或者自动机的方法虽对服务组合进行描述的时候比较直观,但是随着服务的流程规模变大,服务数量变多,会引起组合爆炸,因此这两种方法的复杂度会变得很高;相比之下,π演算虽然适合刻画服务组合的动态行为,但是缺乏直观的工具支持。

类型和类型系统在计算机科学,尤其是程序设计语言中被广泛地研究和应用,类型系统可以用于确保系统按照隐式或者显示的说明方式运行,并自

基金项目: 国家自然科学基金(No.60573063,No.60573064,No.60773059,No.61035004);国家社会科学基金(No.10AYY003)。
作者简介: 刘超(1988—),男,硕士生,主研方向:描述逻辑,Web服务,人工智能;眭跃飞(1963—),男,博士,研究员,主研方向:人工智能,大规模知识获取的理论基础;曹存根(1964—),男,博士,研究员,主研方向:人工智能,大规模知识获取。
收稿日期: 2012-03-05 **修回日期:** 2012-05-09
DOI:10.3778/j.issn.1002-8331.2012.22.013

动检测程序由类型错误而引起的语义错误，从而避免不规范行为的产生[13]。用类型标注程序，可以验证程序在类型层次上的正确性。同样地，用面向对象中的对象去标记程序中的符号，可以验证程序在对象层次上的正确性；用本体[14]中的概念去标注程序中的符号，可以在本体的层面验证程序的正确性。如同在各个层面标注程序中的符号，可以用类型系统中的类型、面向对象中的对象、本体中的概念描述服务中的属性和关系的取值，用以标注整个服务框架。

受类型理论[15]在程序设计以及验证中的启发，本文将类型用于对语义Web服务的描述中，用一种带类型的OWL-S框架OWL-S^{ty}来形式化Web服务，从而使得每个语义Web服务及其说明都带有类型，并且在服务组合的时候遵循相应的类型约束，从而为语义Web服务的组合提供了一种形式化验证方法，用以排除不合法的组合方式。如服务组合是服务中的一个研究问题一样，若两个服务能够组合，则用于标注两个服务的属性和关系的类型、对象、本体概念之间应该有某种联系，如何将这种关系形式地陈述出来，并利用这种形式化方法验证两个Web服务的组合的合法性是本文的主要目的。

2　带类型的一阶语言$\mathscr{L}$

一个逻辑系统通常由语言(language)、语法(syntax)、语义(semantics)和推导系统(deductive system)组成。本章首先给出描述语义Web服务的带类型的逻辑语言的符号以及构成良定公式的语法规则及其语义，本文暂不考虑系统的推导规则。

2.1　语言

带类型的一阶逻辑语言$\mathscr{L}$包含下列符号：

可数个常量符号：$c_1, c_2, \cdots$用来描述在Web服务中出现的对象；在本文中用元语言中的符号c，d，$\cdots$，来表示常量符号。

可数个变元符号：$x_1, x_2, \cdots$用来描述在Web服务的输入输出中出现的变量。

对于每个$k>0$，可数个k元谓词符号，用来描述服务所涉及的对象之间的关系：

$$\begin{matrix} P_0^1 & P_1^1 & P_2^1 & \cdots \\ P_0^2 & P_1^2 & P_2^2 & \cdots \\ P_0^3 & P_1^3 & P_2^3 & \cdots \\ \vdots & \vdots & \vdots & \end{matrix}$$

对于每一个P_n^k，k表示谓词的元数，n表示该谓词是在k元谓词中第n个谓词。在本文中，用元语言中的符号P，Q，$\cdots$来表示谓词符号。

特殊的二元谓词符号=，描述表示对象之间的相等关系。

对于每个$k>0$，可数个k元函数符号：

$$\begin{matrix} f_0^1 & f_1^1 & f_2^1 & \cdots \\ f_0^2 & f_1^2 & f_2^2 & \cdots \\ f_0^3 & f_1^3 & f_2^3 & \cdots \\ \vdots & \vdots & \vdots & \end{matrix}$$

在本文中用元语言中的符号$f, g, \cdots$来表示函数符号。

逻辑联结词：$\neg, \wedge, \vee, \rightarrow$；

量词符号：$\forall$；

标点符号：(　)；

类型构造符：$\mapsto, \times, \otimes, \cup$

2.2　语法

语法给出形式的规则来形成良定的公式用以区别用语言符号随意组成的字符串。本节首先给出语言$\mathscr{L}$的项以及公式等构造的规则，这些规则定义了通常的一阶逻辑的项和公式。随后在此基础之上定义了$\mathscr{L}$的一阶语言的类型部分，并给出了类型之间的运算规则。

定义1(Term($\mathscr{L}$))　$\mathscr{L}$的项定义如下：

$$t::=x|c|f(t_1, t_2, \cdots, t_n)$$

其中，x为变元符号，c为常量符号，f为n元函数符号，$t_1, t_2, \cdots, t_n$表示项。

在描述服务时，简单对象可以用变元符号和常量符号来表示，而这些简单对象可以构造出复杂的对象，用函数符号来描述这种构造复杂对象的构造子。

定义2(Form($\mathscr{L}$))　$\mathscr{L}$的公式定义如下：

$$\phi, \psi::=P(t_1, t_2, \cdots, t_n)|t=t'|\forall x\varphi|\neg\varphi|\varphi\rightarrow\psi|\varphi\wedge\psi|\varphi\vee\psi$$

其中，P为n元谓词符号，$t, t', t_1, t_2, \cdots, t_n$为项，并且$\varphi, \psi$为公式在元语言中的表示。此外，可以引入形如$\exists x\varphi$的公式，表示$\neg\forall x\neg\varphi$。

服务的一个重要特征是由一些断言性的知识来说明服务在执行之前所必需满足的条件或者服务执行之后产生的结果，用一阶的逻辑公式来描述不同对象之间所满足的关系用以说明这种断言性的知识。

定义3(Type($\mathscr{L}$))　$\mathscr{L}$的类型定义如下：

$$\tau::=\text{bool}|\text{b}|\tau_1\times\tau_2|\tau_1\mapsto\tau_2|\tau_1\cup\tau_2$$

其中，bool为布尔类型，b为基本类型。

在上述定义基础上，下面在通常的一阶逻辑的之上引入类型部分，用以说明语言$\mathscr{L}$的各个组成部分的类型定义，从而在形成$\mathscr{L}$的项和公式的时候，除了要考虑公式的递归定义，还要考虑具体的类型约束的限制。具体的类型定义如下所述：

定义4 常量符号,变元符号,函数符号和谓词符号的类型定义:

(1)每一个变量与常量都关联到一个基本类型上,且用Type(x)来表示 x 的类型。

(2)如果 $(t_1, t_2, \cdots, t_n)$ 为 n 元组,并且 $t_1{:}\tau_1, t_2{:}\tau_2, \cdots, t_n{:}\tau_n$,则 $(t_1, t_2, \cdots, t_n){:}\tau_1\times\tau_2\times\cdots\times\tau_n$。

(3)对于所有的 $n\geq 1$,n 元函数符号 f 的类型为:$f{:}\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\tau$。

(4)对于所有的 $n\geq 1$,n 元谓词符号 P 的类型为:$P{:}\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\text{bool}$。

(5)对于特殊的二元谓词 =,其类型为:$=:\tau_1\times\tau_2\mapsto\text{bool}$。

定义5(逻辑联结词的类型定义)

$\neg$:bool$\mapsto$bool

$\vee$:bool$\times$bool$\mapsto$bool

$\wedge$:bool$\times$bool$\mapsto$bool

$\rightarrow$:bool$\times$bool$\mapsto$bool

定义6(量词的类型定义)

$\forall x$:bool$\mapsto$bool

$\mathscr{L}$的任何一个项的类型都可以由如下的类型规则给出:

$x{:}\tau$, 如果Type(x)=τ。

$c{:}\tau$, 如果Type(c)=τ。

如果 n 元函数的类型为 $f{:}\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\tau$,并且 $(t_1, t_2, \cdots, t_n){:}\tau_1\times\tau_2\times\cdots\times\tau_n$,则项 $f(t_1, t_2, \cdots, t_n){:}\tau$。

$\mathscr{L}$的任何一个公式的类型都可以由如下的类型规则给出:

如果 n 元谓词的类型为 $P{:}\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\text{bool}$,并且 $(t_1, t_2, \cdots, t_n){:}\tau_1\times\tau_2\times\cdots\times\tau_n$,则原子公式的类型为 $P(t_1, t_2, \cdots, t_n)$:bool。

如果原子公式的形式为:$t=t'$,并且 $t{:}\tau, t'{:}\tau', =:\tau\times\tau'$ 则 $t=t'$:bool。

如果公式的形式为:$\forall x\varphi$,并且 $\forall x$:bool$\mapsto$bool,φ:bool,则 $\forall x\varphi$:bool。

如果公式是由逻辑连接词构成的复合公式,则公式的类型为定义5中相应的类型规则的运算结果。

2.3 语义

带有类型的一阶语言 $\mathscr{L}$ 在一阶逻辑的基础之上引入了类型的概念,从而使得语义模型除了在说明对象之间的关系的性质之外,还需要对对象之间的类型匹配作出相关的说明。

定义7 带有类型的一阶逻辑 $\mathscr{L}$ 的一个带有类型的模型 $\mathfrak{M}$ 是一个二元组 $(U^T, \Im)$,其中:

(1) U^T 为一个由带某个类型的个体组成的非空集合,作为 $\mathfrak{M}$ 的带类型的论域。

(2)函数 $\Im$ 是一个解释函数,满足如下条件:

①对于任意的常量符号 c,$\Im(c)\in U^T$,并且要求若 c 的类型为 τ,则在论域 U^T 中,$\Im(c)$ 的类型也必须为 τ;为方便起见,记为:$c^{\Im}{:}\tau$。

②对任意的 n 元函数符号 f,$\Im(f){:}U^n\rightarrow U$,并且要求若 f 的类型为 $\sigma\mapsto\tau$,则在 U^T 上的函数 $\Im(f)$ 的类型也必须为 $\sigma\mapsto\tau$,记为:$f^{\Im}{:}\sigma\mapsto\tau$。

③对任意的 n 元谓词符号 P,$P\subseteq U^n$,并且要求若 P 的类型为 $\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\text{bool}$,则在 U^T 上的 n 元谓词 $\Im(P)$ 的类型也必须为 $\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\text{bool}$ 记为:$P^{\Im}{:}\tau_1\times\tau_2\times\cdots\times\tau_n\mapsto\text{bool}$。

定义8(带有类型的赋值函数定义)

一个赋值函数 v 是变元符号集合到模型论域上的一个函数,使得对于任意的变元符号 x,$v(x)\in U^T$,并且要求若 x 的类型为 τ,则在论域 U^T 中,$v(x)$ 的类型也必须为 τ。

$\mathscr{L}$中的项的带类型的赋值:

任意给一个项 t,项 t 在赋值 v 和解释 $\Im$ 的作用下的取值和类型为[1]:

$$t^{\Im,v}=\begin{cases}\Im(c){:}\sigma & \text{如果}t=c\text{并且}c{:}\sigma\\ v(x){:}\sigma & \text{如果}t=x\text{并且}x{:}\sigma\\ f^{\Im}(t_1^{\Im,v}, t_2^{\Im,v}, \cdots, t_n^{\Im,v}){:}\sigma & \text{如果}t=f(t_1, t_2, \cdots, t_n)\text{并且}t{:}\sigma\end{cases}$$

最后引入公式的可满足性。

$\mathscr{L}$中的公式的可满足定义:

任意给定一个 $\mathscr{L}$ 的公式,相对于模型 $\mathfrak{M}$ 和赋值 v,用 $(\mathfrak{M}, v)\models\varphi$ 表示公式 φ 在模型 $\mathfrak{M}$ 和赋值 v 下成立,根据 φ 的结构归纳定义如下:

(1) $(\mathfrak{M}, v)\models P(t_1, t_2, \cdots, t_n)$ 当且仅当 $(t_1^{\Im,v}, t_2^{\Im,v}, \cdots, t_n^{\Im,v})\in\Im(P)$。

(2) $(\mathfrak{M}, v)\models t=t'$ 当且仅当 $t^{\Im,v}=t'^{\Im,v}$。[2]

(3) $(\mathfrak{M}, v)\models\neg\varphi$ 当且仅当 φ 在模型 $\mathfrak{M}$ 和赋值 v 下不成立。

(4) $(\mathfrak{M}, v)\models\varphi\wedge\psi$ 当且仅当 $(\mathfrak{M}, v)\models\varphi$ 并且 $(\mathfrak{M}, v)\models\psi$。

(5) $(\mathfrak{M}, v)\models\varphi\vee\psi$ 当且仅当 $(\mathfrak{M}, v)\models\varphi$ 或者 $(\mathfrak{M}, v)\models\psi$。

(6) $(\mathfrak{M}, v)\models\varphi\rightarrow\psi$ 当且仅当 $(\mathfrak{M}, v)\models\psi$ 或者 φ

[1]注意此处的=符号仅仅表示通常的等于符号,并非是语言 $\mathscr{L}$ 的符号

[2]在此定义中,=的第一次出现表示语言 $\mathscr{L}$ 中的符号,第二次出现表示论域中的恒等关系。

在模型$\mathfrak{M}$和赋值v下不成立。

（7）$(\mathfrak{M},v)\models\forall x\varphi$当且仅当对任意$a\in U^T$，$(M,v)\models\varphi[x/a]$。

3　带类型的一阶动态语言$L^{\mathfrak{A}}$

在针对语义Web服务的研究工作中，最有代表性的是OWL-S。OWL-S是一个用OWL语言描述的Web服务本体，分别从ServiceProfile，ProcessModel和ServiceGrounding等三个方面对Web服务进行了刻画。OWL-S本身并没有明确对Web服务动态特性的推理；另外OWL-S所应用的本体语言OWL是建立在描述逻辑的基础之上的。描述逻辑可以有效地表示关于静态领域的知识，但是对于具有动态特征的知识如服务和动作却缺乏相应的表示机制。另一方面，虽然OWL-S为了表示动态的特性，引入了服务的前提条件和属性，但是这些属性仅仅作为OWL所构建本体的普通的属性来处理，从而缺乏相关的动态的语义支持。

因此，希望可以针对上述问题，将一阶动态逻辑（PDL）的概念引入带类型的一阶逻辑，一方面可以基于类型对静态领域知识进行类型说明，另一方面可以将上述知识作为背景，在此基础上对关于动作的知识进行描述和刻画。然而，PDL是用来描述程序动态性质的逻辑，并且假定程序都是以自然数作为输入的，因此如果仅仅对PDL中的程序做类型的描述，那么类型系统只会包括简单的自然数类型，而显而易见服务中涉及的对象的类型并不只是自然数的类型，因此需要对PDL的类型进行扩充。

3.1　语言

为$\mathscr{L}$引入动作的概念，并将$\mathscr{L}$扩充为可以描述动作的语言$L^{\mathfrak{A}}$，$L^{\mathfrak{A}}$的符号集合如下：

包括$\mathscr{L}$的所有符号；并且包括

动作构造符：:=，?，∪，;　，*

3.2　语法

定义9（$L^{\mathfrak{A}}$的项和公式）

令$L^{\mathfrak{A}}$的项定义同L的项定义，即Term$(L^{\mathfrak{A}})$=Term$(\mathscr{L})$；并令$L^{\mathfrak{A}}$的公式包括$\mathscr{L}$的所有公式，并且除此之外，增加形式为$[p]\psi$的公式，其中ψ为$L^{\mathfrak{A}}$中的公式并且不能含有量词符号，p为动作（见定义10），为简便起见，用Form$(L^{\mathfrak{A}})$来表示$L^{\mathfrak{A}}$的公式集。

定义10（Process$(L^{\mathfrak{A}})$）　$L^{\mathfrak{A}}$的动作定义如下：

$$p,p'::=x:=t|\varphi(x)?|p\cup p'|p;p'|p^*$$

其中，x为$L^{\mathfrak{A}}$中的变元，$t\in$Term$(L^{\mathfrak{A}})$，$\varphi\in$Form$(L^{\mathfrak{A}})$。动作构造符“;”用来描述两个动作之间时间先后执行的概念，“∪”描述两个动作之间选择，“*”描述动作的迭代，“$\varphi(x)$?”描述对对象x是否具有性质φ做测试的动作。

定义11（动作的类型规则）

若动作的形式为$x:=t$，则

$$\frac{x:\sigma,\quad t:\tau}{x:=t:\sigma\mapsto\tau}$$

若动作的形式为$\varphi(x)?$，则

$$\frac{x:\tau}{\varphi(x)?:\tau\mapsto\text{bool}}$$

若动作的形式为$p;p'$，则

$$\frac{p:\tau_1\mapsto\sigma_1,p':\sigma_1\mapsto\tau_2}{p;p':\tau_1\mapsto\tau_2}$$

若动作的形式为$p\cup p'$，则

$$\frac{p:\tau_1\mapsto\sigma_1,p':\tau_2\mapsto\sigma_2}{p\cup p':\tau_1\cup\tau_2\mapsto\sigma_1\cup\sigma_2}$$

若动作的形式为p^*，则

$$\frac{p:\tau\mapsto\tau}{p^*:\tau\mapsto\tau}$$

令$L^{\mathfrak{A}}$中项和公式的类型定义同$\mathscr{L}$中的类型定义，但是针对于新增的形式为$[p]\psi$的公式，有如下类型定义：

定义12

$[]:(\tau\mapsto\sigma)\times\text{bool}\mapsto\text{bool}$

于是有，如果公式的形式为：$[p]\psi$，则

$$\frac{[]:(\tau\mapsto\sigma)\times\text{bool}\mapsto\text{bool},p:\tau\mapsto\sigma,\psi:\text{bool}}{[p]\psi:\text{bool}}$$

3.3　语义

为方便讨论，令$S^{\mathfrak{A}}$为上述赋值函数的集合，并且若u为一个赋值，则定义$u[x/a]$为一个新的赋值通过如下的方法得到：

$$u[x/a](y)=\begin{cases}u(y) & \text{如果}y\neq x\\ a & \text{否则}\end{cases}$$

定义13（模型定义）

$L^{\mathfrak{A}}$中的公式带类型的模型$\mathfrak{M}^{\mathfrak{A}}$为一个二元组$(U^T,\mathfrak{I}^{\mathfrak{A}})$，其中$U^T$同定义9中带类型的论域，$\mathfrak{I}^{\mathfrak{A}}$为解释函数包括函数$\mathfrak{I}$并做如下扩充：

若动作的形式为$x:=t$，则

$$\mathfrak{I}^{\mathfrak{A}}(x:=t)=\{(u,u[x/t^{\mathfrak{I}^{\mathfrak{A}},u}])|u\in S^{\mathfrak{A}}\}$$

若动作的形式为$\varphi(x)?$，则

$$\mathfrak{I}^{\mathfrak{A}}(\varphi(x)?)=\{(u,u)|(\mathfrak{M}^{\mathfrak{A}},u)\models\varphi(x)\}$$

若动作的形式为$p;p'$，则

$$\mathfrak{I}^{\mathfrak{A}}(p;p')=\mathfrak{I}^{\mathfrak{A}}(p)\circ\mathfrak{I}^{\mathfrak{A}}(p')$$

若动作的形式为$p\cup p'$，则

$\mathfrak{I}^{\mathfrak{A}}(p\cup p')=\mathfrak{I}^{\mathfrak{A}}(p)\cup\mathfrak{I}^{\mathfrak{A}}(p')$

若动作的形式为 p^*，则

$\mathfrak{I}^{\mathfrak{A}}(p^*)=\mathfrak{I}^{\mathfrak{A}}(p)^*$

其中，$\cup$ 的第一次出现为语言中的符号，第二次出现表示集合的并；* 的第一次出现表示语言的符号，第二次出现表示集合的传递闭包。

需要注意的是，由于赋值函数是有类型说明的，因此在如上可达关系的构造和复合过程中，需要遵守之前定义的类型的约束，使得符号在解释之前的类型说明同符号在解释到论域中之后同论域中的相应部分的类型说明和定义相同。

进行如上扩充解释函数之后，可以定义对新增公式 $[p]\psi$ 的可满足性定义：$(\mathfrak{M}^{\mathfrak{A}},u)\models[p]\psi$ 当且仅当对任意的 v，若 $(u,v)\in\mathfrak{I}^{A}(p)$，则 $(\mathfrak{M}^{\mathfrak{A}},v)\models\psi$。

如果 Σ 是公式的集合，$\mathfrak{M}^{\mathfrak{A}}$ 是模型，并且对于 Σ 中的任何一个公式 φ，有 $\mathfrak{M}^{\mathfrak{A}}\models\varphi$，那么称 $\mathfrak{M}^{\mathfrak{A}}$ 满足 Σ，记为 $\mathfrak{M}^{\mathfrak{A}}\models\Sigma$。

若满足公式集合 Σ 的每个解释 $\mathfrak{I}^{\mathfrak{A}}$ 和赋值 v 都满足公式 ψ，则称 ψ 为 Σ 的逻辑推论，记为 $\Sigma\models\psi$。

4 对语义Web服务的形式化

本章首先以OWL-S为参考来考察OWL-S中描述的语义Web服务。然后根据语义Web服务的特点，以带类型的一阶动态逻辑语言 $L^{\mathfrak{A}}$ 为基础对语义Web服务进行形式化，最后应用上述形式化的结果，对服务之间的自动组合进行简要的讨论。

4.1 基于OWL-S的语义Web服务

OWL-S以本体为核心来描述Web服务，使其具有语义。在OWL-S的研究框架中，一个Service的本体由三个部分构成：ServiceProfile，ServiceModel和ServiceGrouding。其中，ServiceProfile描述了服务能够做什么，它提供了一系列的概念来描述服务的功能；ServiceModel描述了服务的实现方法，并给出了服务的内部流程；ServiceGrouding描述了Web服务的具体访问细节。

这里并不考虑一个Web服务如何被具体实现以及一个描述Web服务的上层本体是如何映射到下层的具体的协议和标准的，因此本文暂不考虑Service-Grouding的形式化问题。同样，也不考虑关于一个Web服务的非功能的信息，从而本文将主要集中于对一个语义Web服务的功能属性的形式化以及两个语义Web服务的组合的关系的形式化。

由于Web服务的组合是若干个分布式的服务根据一定的业务逻辑临时组合而成的，一般而言，组合中服务本身具有异构性，从而使得服务组合具有一定程度的不可靠性。两个被组合的服务之间除了要考虑组合的语法层次的合法性，如前驱Web服务的输出格式和数量同后继服务的输入格式和数量等语法信息一致；而且要考虑两个Web服务的语义之间的一致性，如参数类别，前提条件之间的可满足性关系等等。如若两个被组合的服务之间的语义不一致，则将这两个服务组合起来会造成执行的语义错误，从而带来损失。因此，为确保服务组合能够正常的执行并且满足组合的目的，需要在组合而成的Web服务执行之前，对其进行验证。

4.2 基于 $L^{\mathfrak{A}}$ 的形式化

针对ServiceModel的上述特点，本节在 $L^{\mathfrak{A}}$ 的基础之上对语义Web服务的主要属性进行形式化，定义带类型的框架 OWL-$S^{\mathfrak{A}}$。由于服务的组合涉及到服务内部的组合和服务之间的组合，根据OWL-S对服务内部组合描述的框架（复合服务），这种复合的服务具有程序的特征（如并发，迭代等）。本文的主要目的是讨论服务之间的组合，因而本文将这种复合服务形式化为一种具有程序特征的原子服务。一般的原子过程由输入、输出、前提条件以及结果4个部分组成。输入和输出体现了服务请求者和服务发布者之间的信息交互；前提条件和结果体现了服务在执行前后状态的改变，前提条件描述了服务在执行之前所必需被满足的条件；结果描述了服务被执行之后将会产生的影响，因此可以对原子过程做如下形式化：

定义 14（原子服务 S_a）以下两类服务是原子服务：

（1）程序原子服务 p，其中 p 为 $L^{\mathfrak{A}}$ 中的动作；p 的前提条件和结果分别为 φ 和 ψ，φ 和 ψ 为 $\mathscr{L}$ 中的公式。

（2）一般的原子服务由如下部分组成：

①用来表示输入参数的列表 I：$I_1,I_2,\cdots,I_n$，其中 I_i 为逻辑语言 $L^{\mathfrak{A}}$ 中的变元符号。

②用来表示输出参数的列表 O：$O_1,O_2,\cdots,O_n$，其中 O_i 为逻辑语言 $L^{\mathfrak{A}}$ 中的变元符号。

③前提条件 φ，φ 为 L 中的逻辑公式。

④结果 ψ，ψ 为 L 中的逻辑公式。

定义 15 服务是按照以下规则构成的有穷长度

[3]符号。表示关系的复合：设 X,Y,Z 是三个集合，R 是 X 到 Y 的关系，S 是 Y 到 Z 的关系，则 R 和 S 的复合表示为：$R\circ S=\{<x,z>|x\in R\wedge z\in S\wedge(\exists y)(y\in Y\wedge<x,y>\in R\wedge<y,z>\in S)\}$

的符号串：

（1）每个原子服务是服务。

（2）若 S_1 和 S_2 是服务，则 $S_1 \bullet S_2$ 是服务。

以上只是对OWL-S复合过程中的控制结构进行了形式化，但是还未对OWL-S复合过程中的数据流进行处理。本文利用类型之间的约束和匹配的规则对OWL-S复合过程的数据流进行建模，因此需要先对服务的参数列表进行类型的说明，进而对原子服务进行类型说明，然后在此基础之上，根据复合服务的构造符和类型的运算法则，得到复合服务的类型说明。

定义16（参数列表的类型）

若参数列表为：$(Para_1, Para_2, \cdots, Para_n)$，则：

$$\frac{Para_1{:}T_1, Para_2{:}T_2, \cdots, Para_n{:}T_n}{(Para_1, Para_2, \cdots, Para_n){:}T_1 \times T_2 \times \cdots \times T_n}$$

定义17（服务S的类型）

若S为程序原子服务，则：

$$\frac{\varphi{:}\mathrm{bool}, \psi{:}\mathrm{bool}}{S{:}(\tau \mapsto \sigma) \otimes (\mathrm{bool} \mapsto \mathrm{bool})} \quad p{:}\tau \mapsto \sigma$$

若S为一般的原子服务，则：

$$\frac{I{:}\tau, O{:}\sigma, \varphi{:}\mathrm{bool}, \psi{:}\mathrm{bool}}{S{:}(\tau \mapsto \sigma) \otimes (\mathrm{bool} \mapsto \mathrm{bool})}$$

若服务的形式为 $S_1 \bullet S_2$，则：

$$\frac{S_1{:}(\tau_1 \mapsto \sigma_1) \otimes (\mathrm{bool} \mapsto \mathrm{bool}), S_2{:}(\sigma_2 \mapsto \tau_2) \otimes (\mathrm{bool} \mapsto \mathrm{bool})}{S_1 \bullet S_2{:}(\tau_1 \mapsto \tau_2) \otimes (\mathrm{bool} \mapsto \mathrm{bool})}, \sigma_1 = \sigma_2$$

其中，p 为 S 的动作，I 为 S 的输入参数列表，O 为 S 的输出参数列表，φ 为前提条件，ψ 为结果。$\otimes$ 为描述服务信息交互的类型和状态改变的类型之间的类型构造算子。

定义18 语义Web服务S的一个带有类型的模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 是一个二元组 $(U_S^T, \mathfrak{S}_S^T)$，其中：

（1）U_S^T 为一个由带某个类型的个体组成的非空集合，作为 $\mathfrak{M}_S^{\mathfrak{A}}$ 的带类型的论域。

（2）$\mathfrak{S}_S^T$ 是一个服务状态空间，是赋值的集合，$\mathfrak{S}_S^T$ 上的二元关系 $\mathfrak{R}_S^T$ 为服务的可达关系。

定义19（服务的可满足性）

（1）若服务的形式为程序原子服务，存在模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 和赋值 u，且在该模型和赋值下：$\varphi \models [p]\psi$，则称模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 满足服务S，记为 $(\mathfrak{M}_S^{\mathfrak{A}}, u) \models S$。

（2）若服务的形式为一般的原子服务S，给定模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 下满足如下约束：

①S中的输入输出参数列表中的参数的类型同前提条件 φ 和结果 ψ 中的相关变量类型一致。

②存在赋值 u 和 v，满足 $(u, v) \in \mathfrak{R}_S^T$，$(\mathfrak{M}_S^{\mathfrak{A}}, u) \models \varphi$ 并且 $(\mathfrak{M}_S^{\mathfrak{A}}, v) \models \psi$。

则称模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 满足服务S，记为 $(\mathfrak{M}_S^{\mathfrak{A}}, u) \models S$。

（3）若服务的形式为 $S_1 \bullet S_2$，并且在模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 中，有赋值 u，满足如下约束：

① $(\mathfrak{M}_S^{\mathfrak{A}}, u) \models S_1$。

②在模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 和赋值 u 下，$\psi_1 \models \varphi_2$，其中 ψ_1 为 S_1 的结果，φ_2 为 S_2 的前提条件。

③ $(\mathfrak{M}_S^{\mathfrak{A}}, v) \models \psi_2$，其中 ψ_2 为 S_2 的结果，$(u, v) \in \mathfrak{R}_{S_1}^T \circ \mathfrak{R}_{S_2}^T$。

④Type $(\mathfrak{I}^{\mathfrak{A}}(O_1))$ 同Type $(\mathfrak{I}^{\mathfrak{A}}(I_2))$ 相同。

其中 O_1 为 S_1 的输出参数，I_2 是 S_2 的输入参数。则称模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 和赋值 u 满足服务 $S_1 \bullet S_2$，记为 $(\mathfrak{M}_S^{\mathfrak{A}}, u) \models S_1 \bullet S_2$。

若在模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 下，对于任意的赋值 u 都有 $(\mathfrak{M}_S^{\mathfrak{A}}, u) \models S$，则称模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 满足服务 S，记为 $\mathfrak{M}_S^{\mathfrak{A}} \models S$。

考虑一个具体的网上购鞋的Web服务。该服务由两个服务组合而成，组合的前驱服务 S_1 为查询鞋码的服务，组合的后继服务 S_2 为根据鞋码购鞋的服务。其中，S_1 的输入为脚码；S_2 在执行之前要求用户的信用卡有效。

利用定义14对以上两个Web服务进行带类型的OWL-S框架OWL-$S^{\mathfrak{A}}$ 形式化：

（1）S_1 的带类型的框架为：

```
define atomic service S1
( Inputs:(feetsize hasType CM),
  Outputs:(shoessize hasType ZH),
  Preconditions:(true hasType bool),
  Effects:(true hasType bool)
)
```

其中，CM为长度类型，ZH表示鞋码为中国码类型。从而根据定义17可以得到 S_1 的类型：

$$S_1{:}(CM \mapsto ZH) \otimes (\mathrm{bool} \mapsto \mathrm{bool})$$

（2）S_2 的带类型的框架为：

```
define atomic service S2
( Inputs:(shoessize hasType US),
  Outputs:(output hasType OrderShippedAcknowledgment),
  Preconditions:(Creditcard(x, y) ^Validation(y) hasType bool),
  Effects:(DebtAdd(x) hasType bool)
)
```

其中，US表示输入的鞋码为美国码类型，该例中用到的谓词定义如下：

Validation(x):x有效;

DebtAdd(x):x的债务增加;

Creditcrad(x,y):x拥有信用卡y;

同理可得 S_2 的类型:

S_2:($US \mapsto OrderShippedAcknowledgment$) $\otimes$ (bool $\mapsto$ bool)

虽然可以构造适当的模型 $\mathfrak{M}_S^{\mathfrak{A}}$ 和赋值 u 使得 $true \models Credit\,\mathrm{card}(x,y) \wedge Validation(y)$,但是由于 S_1 的输出类型ZH和的 S_2 输入类型US不匹配,所以这两个服务的服务组合不是合法的,从而排除了因为鞋码单位不一致而发生的购买错误。

5 结束语

本文给出了一个描述Web服务的带类型的OWL-S框架OWL-$S^{\mathfrak{A}}$ 用来形式化Web服务。本文从类型标注的角度出发,用一种带类型的OWL-S来形式化语义Web服务,并且形式地陈述了标注两个服务组合的属性和关系的类型、对象、本体概念之间的联系;并且OWL-$S^{\mathfrak{A}}$ 中的类型规则可以用来对服务的自组合进行类型检查和排错。

下一步的工作包含对框架中语言 $L^{\mathfrak{A}}$ 的逻辑分析:建立推导系统并且分析可靠性和完备性,以及设计推理的算法,从而为语义Web服务的自动发现和自动组合提供完整的支持。

参考文献:

[1] McIlraith S A,Son T C,Zeng H.Semantic web services[J].IEEE on Intelligent Systems,2001,16(2):46-53.

[2] Roman D,Keller U,Lausen H.Web service modeling ontology[J].Applied Ontology,2005,1(1):77-106.

[3] Sycara K,Paolucci M,Ankolekar A.Automated discovery, interaction and composition of semantic web services[J].Web Semantics:Science,Services and Agents on the World Wide Web,2011,1(1).

[4] Fensel D,Facca F M,Simperl E.Semantic Web services[M].New York:Springer,2011.

[5] Martin D,Burstein M,Hobbs J,et al.OWL-S:Semantic markup for web services[J].W3C Member Submission,2004,22.

[6] Horrocks I,Patel-Schneider P F,Boley H,et al.SWRL:A semantic web rule language combining OWL and RuleML[Z].W3C Member Submission,2004.

[7] Martin D,Burstein M,Mcdermott D,et al.Bringing semantics to web services with owl-s[J].World Wide Web,2007,10(3):243-277.

[8] 邓水光.Web服务自动组合与形式化验证的研究[D].杭州:浙江大学,2007.

[9] 王杰生,李舟军,李梦君.用描述逻辑进行语义Web服务组合[J].软件学报,2008,19(4).

[10] Hamadi R,Benatallah B.A Petri net-based model for web service composion[C]//Proceeding of the Fourtheenth Australasian Database Conference on Database Technologies,2003.

[11] Wombacher A,Fankhauser P,Mahleko B,et al.Matchmaking for business processed based on choreographies[J].International Journal of Web Services,2004.

[12] 廖军,谭浩,刘锦德.基于Pi演算的Web服务组合的描述和验证[J].计算机学报,2005,28(4):635-643.

[13] Pierce B C.Types and programming languages[M].MIT:The MIT Press,2002.

[14] Guarino N,Oberle D,Staab S.What Is an Ontology?[M]//Staab S,Studer R.Handbook on Ontologies,2009:1-17.

[15] Cardelli L,Wegner P.On understanding types,data abstraction,and polymorphism[J].ACM Computing Surveys(CSUR),1985,17(4):471-523.

(上接44页)

[2] Fedorenko R P.The speed of convergence of one iterative process[J].USSR Comp Math Math Phys,1964,4(3):227-235.

[3] Bakhvalov N S.On the convergence of a relaxation method with natural constraints on the elliptic operator[J].USSR Comp Math Math Phys,1966,6:101-135.

[4] Brandt A.Multi-level adaptive technique(MLAT) for fast numerical solution to boundary value problems,82-8[R].Berlin:Springer,1973.

[5] Trottenberg U,Oosterlee C W,Schuller A.Multigrid[M].New York:Academic Press,2001:34-62.

[6] Elman H C,Ernst O G,O'Leary D P.A mulitigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations[J].SIAM J Sci Comput,2001,23:1290-1314.