文章编号: 1000-5862(2016)06-0648-05

基于流演算的语义 Web 服务组合研究

孙爱玲
(南通大学现代教育技术中心,江苏 南通　226019)

摘要: 针对现有语义 Web 服务组合方法未考虑到互联网环境的动态性及 Web 服务的随机性问题,提出采用流演算理论对语义 Web 服务组合进行研究. 首先将 Web 服务的输入、输出、前提和效应映射为基于流演算的动作形式化描述; 然后定义将 OWL-S 中的原子过程和复合过程向流演算转换的规则; 再根据提供的 Web 服务组合目标,利用形式化地推理来得出 Web 服务的组合序列,从而能动态地形成正确有效的 Web 服务组合方案. 最后,通过一个会议安排实例验证上述理论,结果表明该方法是可行的.

关键词: Web 服务组合; 语义 Web; 流演算; Web 服务; OWL-S
中图分类号: TP 393　　文献标志码: A　　DOI: 10.16357/j.cnki.issn1000-5862.2016.06.21

0　引言

随着互联网技术的迅速发展和应用需求的多样化,越来越多的企业都将其业务功能和流程封装成标准的 Web 服务发布出去,为基于 Internet 的应用开发提供了丰富的资源[1]. 但单个 Web 服务提供的功能是有限的,更多的是需要将这些 Web 服务组合起来,从而提供更为强大的服务功能.

当前,许多专家学者对 Web 服务组合进行了大量研究,提出的方法大致有: 基于工作流[2-5]、基于形式化方法[6-8]、基于人工智能规划[9-10]等.

在基于工作流的 Web 服务组合系统中,需要领域专家事先定义好一些 Web 服务组合框架,因此它更适合静态 Web 服务组合,对动态的 Web 服务组合支持不足. 基于形式化方法是利用数学方法和形式化工具对 Web 服务组合方法进行研究,但在实用性和通用性方面存在不足. 基于人工智能规划是将 Web 服务组合问题转换成人工智能中的规划问题,其基本思想是: 将 Web 服务看成人工智能中的动作,并对这些动作形式化,然后根据构造的目标,借助于公理系统,推理出 Web 服务的组合序列,动态地形成服务组合方案,能保证规划结果的正确性和完整性. 可见,基于人工智能的方法能考虑到互联网环境的动态性和 Web 服务组合的随机性问题.

因此,本文选取流演算[10]作为 Web 服务组合语义的形式化描述方法,在此基础上展开形式化推理,对 Web 服务进行动态组合.

1　流演算概述

流演算是人工智能中的一个带等词的二阶多型语言[11],可方便地形式化表示动态变化的世界,其基本思想是将动态世界的变化视为动作序列的执行.

1.1　基本概念

流演算使用动作、情景、流、状态和知识概念来刻画动态世界.

(i) 动作和情景. "动作"指与动态世界的交互,它可以改变动态世界的环境,也可以改变知识状态,通常用函数表示. 例如, $I_{\text{nquiryCarInfo}}(p_1,p_2,d)$ 表示查询日期为 d,从 p_1 到 p_2 的汽车信息. 谓词 $P_{\text{oss}}(a,s)$ 描述了情景 s 下动作 a 执行的前提条件. "情景"指动作执行的历史,即一组动作序列,常用一阶谓词表示. S_0 表示初始情景,即系统中没有做任何动作. 函数 $D_o(a,s)$ 表示情景 s 下动作 a 执行后产生的后继情景. 比如,$D_o(I_{\text{nquiryCarInfo}}(p_1,p_2,d),s)$ 表示在情景 s 下执行"查询日期 d 从 p_1 地到 p_2 地的汽车信息"后而产生的情景. 又如,情景 $D_o(R_{\text{eserveCar}}(c_{\text{Num}},p_1,p_2,d,t),D_o(I_{\text{nquiryCarInfo}}(p_1,p_2,d),s))$ 是由动作序列 $[I_{\text{nquiryCarInfo}}(p_1,p_2,d),R_{\text{eserveCar}}(c_{\text{Num}},p_1,p_2,d,t)]$ 组成,表示查询日期 d 从 p_1 到 p_2 的汽车信息、订购日期 d 车次为 c_{Num} 的汽车票的一个历史.

(ii) 流和状态. "流"是指在动作执行后值会发生变化的单个原子属性,常用项表示. 流分为关系流

收稿日期: 2016-01-15
基金项目: 国家自然科学基金(61171132)资助项目.
作者简介: 孙爱玲(1972-),女,湖北鄂州人,实验师,主要从事管理信息系统的开发与人工智能的研究.

和函数流，其中值为真或假的称为关系流. 比如，$H_{\text{asMoney}}(b_{\text{ankCard}})$ 表示银行卡是否有钱(若有钱，值为真; 若没钱，值为假); $C_{\text{Num}}(c_{\text{ar}})$ 表示汽车的车次号(假设为 KK6062)，是个函数流. “状态”是由成立的流组成的集合. 二元函数“o”: 映射 2 个状态为一个状态. 如项 $B_{\text{ankCardNumber}}$(“6225212300113942”) o $H_{\text{asMoney}}(b_{\text{ankCard}})$ 表示卡号为 6225212300113942 的银行卡里有钱. 常量 φ 表示空状态，即此时没有流成立. 此外，$H_{\text{olds}}(f,z)$ 表示流 f 在状态 z 下成立.

(iii) 知识. 在人工智能领域中，有类动作执行之后改变的是知识，这类动作称为“产生知识的动作”，比如，感知动作. 根据可能世界语义，知识流 $K_{\text{State}}(s,z)$ 表示 z 是情景 s 下的一个可能状态. 如果流 f 在情景 s 下的所有可能状态下(不)成立，那么就知道它(不)成立，用 $K_{\text{nows}}(f,s)$ ($K_{\text{nows}}(\neg f,s)$) 表示. 即 $K_{\text{nows}}(f,s) = (\forall z)(K_{\text{State}}(s,z) \supset H_{\text{olds}}(f,z))$.

此外，宏 K_{nowsVal} 用来表示知道流 f 中的参数值，定义为 $K_{\text{nowsVal}}(\vec{x}_1,f,s) = (\exists \vec{x}_1)(K_{\text{nows}}(f,s))$.

1.2　公理系统

流演算的公理系统包括 2 种: (i) 与领域无关公理，刻画了系统在任一领域都需要遵守的规则; (ii) 与领域相关公理，包括初始状态公理、前提条件公理、知识更新公理、惟一名公理等.

1.2.1　与领域无关公理　$zo\varphi = z$，φ 为空状态; $zoz = z$; $z_1 o z_2 = z_2 o z_1$; $(z_1 o z_2) o z_3 = z_1 o (z_2 o z_3)$; $H_{\text{olds}}(f,f_1 o z) \supset f = f_1 \vee H_{\text{olds}}(f,z)$; $\neg H_{\text{olds}}(f,\varphi)$; $H_{\text{olds}}(f,g) \supset f = g$;

$(\forall f)(H_{\text{olds}}(f,z_1) \equiv H_{\text{olds}}(f,z_2)) \supset z_1 = z_2$; $(\forall P)(\exists z)(\forall f)(H_{\text{olds}}(f,z) \equiv P(f))$，其中 P 是一个属于流型的二阶谓词变量.

1.2.2　与领域相关公理　(i) 初始状态公理. 初始状态公理刻画了系统最初未做任何动作时的状态.

(ii) 前提条件公理. 前提条件公理描述了动作能执行的前提条件. 每个动作都会有一条前提条件公理，形如 $P_{oss}(A(\vec{x}),z) \equiv \Pi(z)$，其中 A 为动作，$\vec{x}$ 和 z 为自由变元，$\Pi(z)$ 为状态公式.

(iii) 知识更新公理. 知识更新公理刻画了动作执行后的状态变化，由物理效应和认知效应 2 部分组成，形如

$$P_{oss}(A(\vec{x}),s) \supset (\exists \vec{y})(\forall z')(K_{State}(D_o(A(\vec{x}),s),z') \equiv (\exists z)(K_{State}(s,z) \wedge \psi(z',z) \wedge \Phi(z',D_o(A(\vec{x}),s))))$$

，其中，物理效应 $\psi(z',z)$ 刻画了动作执行后动态世界发生的实际变化，形如 $(\exists \vec{y}_1)(\Delta_1(z) \wedge z' = z - v_1^- + v_1^+) \vee \cdots \vee (\exists \vec{y}_n)(\Delta_n(z) \wedge z' = z - v_n^- + v_n^+)$，其中，$\Delta_i(z)$ 为状态公式($1 \leqslant i \leqslant n$ 且 $n \geqslant 1$)，$\vec{y}_i$ 和 z 为自由变元; v_i^+、v_i^- 为正负效应.

认知效应 $\Phi(z',D_o(A(\vec{x}),s))$ 描述了动作执行后知识方面发生的变化. 形如 $[\Phi_1(z') \equiv \Phi_1(D_o(A(\vec{x}),s))] \wedge \cdots \wedge [\Phi_k(z') \equiv \Phi_k(D_o(\vec{x}),s))] \wedge H_{\text{olds}}(F_1(\vec{t_1}),z) \wedge H_{\text{olds}}(F_1(\vec{t_1}),D_o(A(\vec{x}),s)) \wedge \cdots \wedge H_{\text{olds}}(F_l(\vec{t_l}),z) \wedge H_{\text{olds}}(F_l(\vec{t_l}),D_o(A(\vec{x}),s)) \wedge \Phi(\vec{x},\vec{y})$，其中 $\Phi_i(z')$ 为状态公式($1 \leqslant i \leqslant k$，$k \geqslant 1$)，$F_j(\vec{t_j})$ 为流($1 \leqslant j \leqslant l$，$l \geqslant 1$)，$\Phi(\vec{x},\vec{y})$ 为不包括状态的辅助公式，$\vec{x}$、$\vec{y}$ 为自由变元.

(iv) 惟一名公理. 函数名相同而参数不同或函数名不同的流都属于不同的流

$$\bigwedge_{i=1}^{n}\bigwedge_{j=1,j\neq i}^{n} H_i(\vec{x}) \neq H_j(\vec{y}) \wedge \bigwedge_{i=1}^{n}(H_i(\vec{x}) = H_i(\vec{y})) \rightarrow \vec{x} = \vec{y}).$$

2　OWL-S 概述

OWL-S(Web Ontology Language for Service，Web 服务本体语言) 是一种基于 OWL[12-16] 的 Web 服务描述语言. 它包含 3 个服务本体: Service Profile，Service Model 和 Service Grounding. Service Profile 描述了服务做什么，服务提供者可以用它来描述 Web 服务，用户也可以通过它来请求 Web 服务. Service Model 描述服务怎么做，即指服务提供者用来描述服务的内部流程. 一个服务通常称为一个过程，过程分为原子过程、复合过程和简单过程. Service Grounding 描述怎么访问服务，包含协议、端口、消息格式等.

OWL-S 通常使用 IOPRs(Inputs，Outputs，Preconditions 和 Results) 来刻画 Web 服务的功能，其中 Inputs 为服务的输入集合，Outputs 为服务的输出集合，Preconditions 为服务的前提条件，Results 为服务产生的效应.

3　用流演算进行语义 Web 服务组合描述

为了实现用流演算理论来描述语义 Web 服务组合，需要对 OWL-S 中的 IOPRs 概念进行描述和转换，还需要对服务的流程进行描述和转换.

3.1 概念的描述与转换

根据表 1,OWL-S 的输入和前提都被映射为流演算的前提,其中将 OWL-S 输入转换来的前提称为认知前提,而将 OWL-S 前提转换而来的前提称为物理前提.

表 1 OWL-S 的概念向流演算转换的规则

OWL-S 的概念	流演算的概念	实现
输入	知识前提	前提条件公理
输出	知识效应	知识更新公理
前提	物理前提	前提条件公理
效应	物理效应	知识更新公理

如 $R_{\text{eserveCar}}(c_{\text{Num}}, p_1, p_2, d, t, z) \equiv H_{\text{olds}}(A_{\text{vailable}}(T_{\text{icket}}), z) \wedge K_{\text{nowsVal}}(c_{\text{Num}}, T_{\text{icket}}, z) \wedge K_{\text{nowsVal}}(p_1, T_{\text{icket}}, z) \wedge K_{\text{nowsVal}}(p_2, T_{\text{icket}}, z) \wedge K_{\text{nowsVal}}(d, T_{\text{icket}}, z) \wedge K_{\text{nowsVal}}(t, T_{\text{icket}}, z)$,其中流 $A_{\text{vailable}}(T_{\text{icket}})$ 表示有票,其它参数根据输入获得.

类似地,OWL-S 的输出和效应都被映射为流演算的效应.

3.2 过程的描述与转换

3.2.1 原子过程描述及向流演算转换规则 OWL-S 中的原子过程是个不可再分的过程,对应于流演算中的动作. 例如订汽车票,在流演算中可以形式化表示为动作 $R_{\text{eserveCar}}(c_{\text{Num}}, p_1, p_2, d, t)$.

3.2.2 复合过程描述及与流演算转换的规则 复合过程一般由原子过程或小的复合子过程按照一定的结构组合而成,其中的结构有: 顺序、选择、循环、并行、随机选择、重复执行等. 而流演算也支持相似的结构,因此将 OWL-S 的复合过程转变成流演算则变得容易.

为了描述其语义,引入了 2 个谓词 $F_{\text{inal}}(\delta, z)$ 和 $T_{\text{rans}}(\delta, z, \delta', z', h')$. $F_{\text{inal}}(\delta, z)$ 表示程序 δ 能合法地终止于状态 z. $T_{\text{rans}}(\delta, z, \delta', z', h')$ 表示 δ 在 z 下执行一步(一个基本动作或测试动作),则 z 变化到 z',并剩下 δ' 未执行,h' 是从 z 到 z' 执行的动作.

OWL-S 的复合过程与流演算的转换规则如下.

(i) 顺序. 假设 C 是由 2 个顺序子过程 C_1 和 C_2 组成的复合过程,则相应的流演算表示成 $\delta_{C_1}; \delta_{C_2}$,表示只有当 δ_{C_1} 执行完后 δ_{C_2} 才能执行,这里 δ_{C_1} 和 δ_{C_2} 为复合动作.

$$T_{\text{rans}}(\delta_{C_1}; \delta_{C_2}, z, \delta', z', h') \equiv \exists r, \delta'. ((r; \delta_{C_2}) \wedge T_{\text{rans}}(\delta_{C_1}, z, r, z', h') \vee F_{\text{inal}}(\delta_{C_1}, z) \wedge T_{\text{rans}}(\delta_{C_2}, z, \delta', z', h')),$$

即若动作 δ_{C_1} 可以执行一步变成 r,状态从 z 变成 z',那么顺序动作 $\delta_{C_1}; \delta_{C_2}$ 经过一次 T_{rans} 后就剩下 $r; \delta_{C_2}$,δ_{C_2} 并没有被执行,h' 就是一个简单的动作列表,或为空或为一个基本动作,因为一次 T_{rans} 最多执行一个基本动作. 或者是 δ_{C_1} 已经合法地终止于状态 z,即 δ_{C_1} 在状态 z 下没有可执行的基本动作,那么就对 δ_{C_2} 进行 T_{rans},过程同 δ_{C_1} 类似.

(ii) 选择. 假设 C 是 OWL-S 中含有选择结构的复合过程,其判断条件为 φ,则相应的流演算表示成 if φ then δ_{C_1} else δ_{C_2} endif,即

$$T_{\text{rans}}(\text{if } \varphi \text{ then } \delta_{C_1} \text{ else } \delta_{C_2} \text{ endif}, z, \delta', z', h') \equiv \varphi[z] \wedge T_{\text{rans}}(\delta_{C_1}, z, \delta', z', h') \vee \neg\varphi[z] \wedge T_{\text{rans}}(\delta_{C_2}, z, \delta', z', h'),$$

表示若条件 φ 成立,则执行 δ_{C_1},否则执行 δ_{C_2},其中 $\varphi[z]$ 为状态 z 下的条件表达式.

(iii) 循环. 假设 C 是 OWL-S 中含有循环结构的复合过程,其循环判断条件为 φ. 其含义是,若满足条件 φ,则重复执行过程 C_1,否则循环结束. 相应的流演算表示成 while φ do δ_{C_1},即 $T_{\text{rans}}(\text{while } \varphi \text{ do } \delta_{C_1}, z, \delta', z', h') \equiv \exists r. (\delta' = r; \text{while } \varphi \text{ do } \delta) \wedge T_{\text{rans}}(\delta_{C_1}, z, r, z', h')$.

(iv) 并行. 假设 C 是包含 2 个并行子过程 C_1 和 C_2 的复合过程,相应的流演算表示为 $\delta_{C_1} \parallel \delta_{C_2}$,即

$$T_{\text{rans}}(\delta_{C_1} \parallel \delta_{C_2}, z, \delta', z', h') \equiv \exists r. (\delta' = r \parallel \delta_{C_2}) \wedge T_{\text{rans}}(\delta_{C_1}, z, r, z', h') \vee \exists r. (\delta' = \delta_{C_1} \parallel r) \wedge T_{\text{rans}}(\delta_{C_2}, z, r, z', h').$$

如果动作 δ_{C_1} 可以执行一步变成 r,状态从 z 变化到 z',那么并行动作 $\delta_{C_1} \parallel \delta_{C_2}$ 经过 1 次 T_{rans} 后就剩下 $r \parallel \delta_{C_2}$. 或者,如果 δ_{C_1} 在状态 z 下没有可执行的基本动作,那么就对 δ_{C_2} 进行 T_{rans},过程同 δ_{C_1} 类似.

(v) 随机选择. 假设 C 是 OWL-S 的复合过程,包含两个子过程 C_1 和 C_2,它们是随机选择的关系,则相应的流演算表示为 $\delta_{C_1} | \delta_{C_2}$,即 $T_{\text{rans}}(\delta_{C_1} | \delta_{C_2}, z, \delta', z', h') \equiv T_{\text{rans}}(\delta_{C_1}, z, \delta', z', h') \vee T_{\text{rans}}(\delta_{C_2}, z, \delta', z', h')$,2 个动作 δ_{C_1} 和 δ_{C_2} 均可使状态从 z 到达 z'.

(vi) 重复执行. $T_{\text{rans}}(\delta^*, z, \delta', z', h') \equiv \exists r. (\delta' = r; \delta^*) \wedge T_{\text{rans}}(\delta, z, r, z', h')$ 即该动作是被重复执行的,可以执行 0 次或多次.

4 实例与分析

假设某用户需要从 P_1 到 P_2 城市去参加会议,希望会议日程安排系统能为他自动地安排行程,包括交通、住宿等,其流程见图 1.

此流程包含的 Web 服务有: (i) 估算两地需要花费的时间. (ii) 查询汽车信息; 并订购汽车票. (iii) 查询火车信息; 并订购火车票. (iv) 查询飞机

信息; 并订购飞机票. (v) 查询酒店信息; 并订酒店.

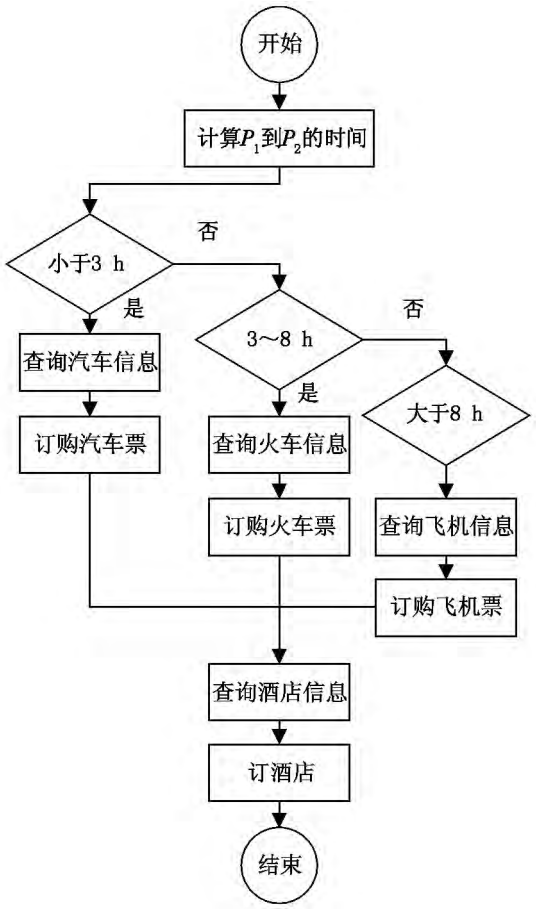

图 1　会议日程安排流程图

流程中包含了顺序和选择 2 种结构,其中需要根据距离而选择坐汽车、火车还是飞机的交通方式,其余是顺序序列,都必须执行. 下面根据流演算语言的动作理论,从动作、流、初始状态公理、前提条件公理、知识更新公理、复杂动作等方面对 Web 服务进行语义描述.

1) 动作. 将上述的 Web 服务定义为流演算中的动作有: (i) $I_{\text{nquiryTime}}(p_1,p_2,t_{\text{ime}})$,估算 p_1 到 p_2 需要的时间; (ii) $I_{\text{nquiryCarInfo}}(p_1,p_2,d)$,查询日期为 d,p_1 到 p_2 地的汽车信息, 返回汽车班次的信息; (iii) $R_{\text{eserveCar}}(c_{\text{Num}},p_1,p_2,d,t)$,订购车号为 c_{Num},从 p_1 到 p_2 地, 日期为 d, 时间为 t 的汽车票; (iv) $I_{\text{nquiryTrainInfo}}(p_1,p_2,d)$: 查询日期为 d,p_1 到 p_2 地的火车信息,返回火车车次信息; (v) $R_{\text{eserveTrain}}(t_{\text{Num}},p_1,p_2,d,t)$: 订购车次为 t_{Num},从 p_1 到 p_2 地,日期为 d,时间为 t 的火车票; (vi) $I_{\text{nquiryFlightInfo}}(p_1,p_2,d)$: 查询日期为 d,从 p_1 到 p_2 地的飞机信息,返回飞机的航班信息; (vii) $R_{\text{eserveFlight}}(a_{\text{Num}},f_{\text{Num}},p_1,p_2,d,t)$: 订购航空公司为 a_{Num},航班号为 f_{Num},日期为 d,从 p_1 到 p_2 地的机票; (viii) $I_{\text{nquiryHotelInfo}}(p_2,d)$: 查询日期为 d,p_2 地的酒店,返回酒店信息; (ix) $R_{\text{eserveHotel}}(h_{\text{Name}},p_2,d_1,d_2)$: 预定 p_2 地酒店名为 h_{Name} 的一个房间,时间从 d_1 到 d_2.

2) 流. 为了实现系统中状态的变化是通过流的增减而实现变化的. 例如,在订购火车票的 Web 服务中, 定义了流: $C_{\text{ity}}(p_1)$,$C_{\text{ity}}(p_2)$,$P_{\text{ersonName}}(p_n)$,$D_{\text{ate}}(d)$,$C_{\text{arTicket}}(c_{\text{Num}},p_1,p_2,p_{\text{rize}},s_{\text{eatNum}},d,t,b_{\text{cNum}})$,$T_{\text{ime}}(t)$,$B_{\text{ankCard}}(b_{\text{cNum}})$ 等. 类似的,还需要为 $R_{\text{eserveCar}}$,$R_{\text{eserveFlight}}$ 和 $R_{\text{eserveHotel}}$ 的前提和效应定义相应的流.

3) 初始状态. $K_{\text{state}}(s,z_0) \equiv (z^0 = C_{\text{ity}}(\text{“beijing”}) \ o\ C_{\text{ity}}(\text{“nantong”}) \ o\ D_{\text{ate}}(\text{“2014-10-17”}) \ o\ P_{\text{ersonName}}(\text{“yang”}) \ o\ B_{\text{ankCard}}(\text{“6225212300113942”}) \ o\ z)$.

4) 动作前提公理. 每个动作都对应有一个前提条件公理. 以服务 $I_{\text{nquiryCarInfo}}$ 为例,其前提条件包括己知出发地、目的地和出发日期,以及银行卡有效. 将 Web 服务执行的条件定义为 $P_{\text{oss}}(\text{I}_{\text{nquiryCarInfo}}(p_1,p_2,d),z) \equiv K_{\text{nowsVal}}(p_1,C_{\text{arTicket}}(c_{\text{Num}},p_1,p_2,p_{\text{rize}},s_{\text{eatNum}},d,t,b_{\text{cNum}}),z) \wedge K_{\text{nowsVal}}(p_2,C_{\text{arTicket}}(c_{\text{Num}},p_1,p_2,p_{\text{rize}},s_{\text{eatNum}},d,t,b_{\text{cNum}}),z) \wedge K_{\text{nowsVal}}(d,C_{\text{arTicket}}(c_{\text{Num}},p_1,p_2,p_{\text{rize}},s_{\text{eatNum}},d,t,b_{\text{cNum}}),z) \wedge H_{\text{olds}}(B_{\text{ankCarValid}},z)$. 其它原子服务类似,此略.

5) 知识更新公理. 将 Web 服务执行的结果定义为知识更新公理,定义动作的执行对流值的改变情况. 如 $I_{\text{nquiryCarInfo}}$ 执行后会使得流 C_{Num} 的值变为真.

$P_{\text{oss}}(\text{I}_{\text{nquiryFlightInfo}}(p_1,p_2,d),z) \rightarrow K_{\text{nowsVal}}(c_{\text{Num}},C_{\text{arTicket}}(c_{\text{Num}},p_1,p_2,p_{\text{rize}},s_{\text{eatNum}},d,t,b_{\text{cNum}}),z)$.

6) 过程. 最后将此任务表示为由 9 个原子 Web 服务组成的过程:

Proc Manageschedule
$I_{\text{nquiryTime}}(p_1,p_2,t_{\text{ime}})$;
if $t_{\text{ime}} < 3$ then
　$I_{\text{nquiryCarInfo}}(p_1,p_2,d)$;
　$R_{\text{eserveCar}}(c_{\text{Num}},p_1,p_2,d,t)$
else if　$d_{\text{iwtance}} < 8$ then
　$I_{\text{nquiryTrainInfo}}(p_1,p_2,d)$;
　$R_{\text{eserveTrain}}(t_{\text{Num}},p_1,p_2,d,t)$
else
　$I_{\text{nquiryFlightInfo}}(p_1,p_2,d)$;
　$R_{\text{eserveFlight}}(a_{\text{Num}},f_{\text{Num}},p_1,p_2,d,t)$
endif
　$I_{\text{nquiryHotelInfo}}(p_2,d)$;
　$R_{\text{eserveHotel}}(h_{\text{Name}},p2,d_1,d_2)$
EndProc.

以上是基于流演算完成的对会议行程安排系统中 Web 服务组合的语义描述.

5 结束语

由于流演算具有较强的动态系统描述能力,非常适合描述动态的 Web 服务组合. 因此,利用流演算将 Web 服务与动态世界中的动作和推理进行有机地结合,将 Web 服务看作动作,通过描述动作的前提条件公理和知识更新公理来实现环境状态的变化,从而能够描述动态的 Web 服务组合问题. 最后,通过一个会议日程安排实例说明描述上述理论具有一定的可行性. 通过这方面的研究,可以为未来开发网络环境中的复杂的 Web 服务组合研究提供新思路.

6 参考文献

[1] 崔华,应时,袁文杰,等. 语义 Web 服务组合综述 [J]. 计算机科学,2010,37(5):21-25.

[2] Charif Y,Sabouret N. An overview of semantic web services composition approaches [J]. Electronic Notes in Theoretical Computer Science,2006,146(1):33-41.

[3] 苑迎春,王克俭,韩宪忠,等. 基于工作流的 Web 服务组合多视图模型 [J]. 计算机集成制作系统,2010,16(1):30-36.

[4] 徐萌,陈俊亮,彭泳,等. 基于服务关系的服务本体生成 [J]. 软件学报,2007,19(4):545-556.

[5] 刘祥. 基于工作流和 Agent 的 Web 服务柔性组合技术研究及应用 [D]. 南京:南京理工大学,2013.

[6] 史忠植,常亮. 基于动态描述逻辑的语义 Web 服务推理 [J]. 计算机学报,2008,31(9):1599-1611.

[7] 王杰生,李舟军,李梦军. 用描述逻辑进行语义 Web 服务组合 [J]. 软件学报,2008,4(19):967-980.

[8] Broy M,Kruger I H,et al. A formal model of service [EB/OL]. [2015-10-17]. https://www.researchgate.net/publication/220403835_A_formal_model_of_services.

[9] 江卓. 基于智能规划的自适应动态 Web 服务组合研究 [D]. 重庆:重庆大学,2015.

[10] 贾静兰. 语义 Web 服务自动组合方法 [D]. 武汉:华中师范大学,2015.

[11] 杨爱琴. 基于流演算的多 Agent 通信动作的研究 [J]. 计算机工程与设计,2010,31(1):221-224.

[12] Martin D,Paolucci M,Mcilraith S,et al. Bring semantics to Web services with OWL-S [J]. World Wide Web,2007(10):243-277.

[13] 耿耘,蒋严冰,郭岩,等. 基于组合验证的 Web 页面抽取算法研究 [J]. 江西师范大学学报:自然科学版,2013,37(3):142-147.

[14] 魏丹丹,邱乐兴. 基于 Web3.0 的网络教学个性化服务平台构建 [J]. 江西师范大学学报:自然科学版,2013,37(6):584-586.

[15] 张晓丹,李静,张秋霞,等. 语义 Web 本体语言 OWL2 研究 [J]. 电子设计工程,2015,23(16):28-31.

[16] 何涛,覃国蓉,梁永生,等. 基于语义 Web 的软件需求验证工具研究 [C]. International Conference on Intelligent Computation and Induwtrial Application,2011.

The Research on Semantic Web Services Composition Based on Fluent Calculus

SUN Ailing

(Modern Educational Technology Centre,Nantong University,Nantong Jiangsu 226019,China)

Abstract: According to not considering the dynamics of the Internet environment and randomness of Web service during the existing semantic Web service composition methods,the theory of fluent calculus is proposed to research the semantic Web services composition. First of all,the Web service input,output,precondition and result and so on are mapped for the action formalization description based on fluent calculus. Then the rules are defined from the atomic processes and composite processes of OWL-S to fluent calculus. Then,a combined sequence of Web service is derived by formally reasoning with compositing Web services as the goal. So the correct and effective service composition schemes are dynamically formed. Finally,a session arrangement example is applied to validate the above theory. Result shows that it is feasible.

Key words: web services composition; semantic web; fluent calculus; web service; ontology web language for services

(责任编辑:冉小晓)