doi:10.3772/j.issn.1002-0470.2015.02.002

基于动态描述逻辑的 Web 自动化服务组合①

张建华②[*,**] 田东平[*,**] 岳金朋[*,**] 张 博[*,***]

([*]中国科学院计算技术研究所智能信息重点实验室 北京 100190)

([**]中国科学院大学 北京 100049)

([***]中国矿业大学计算机学院 徐州 221116)

摘 要 研究了动态描述逻辑(DDL)在 Web 服务组合中的应用。基于目标规划以及动态描述逻辑给出了 Web 服务组合算法。该算法考虑了客户的个人偏好,扩大了 Web 服务的应用范围。它使用 DDL 进行 Web 服务组合刻画,因为 DDL 将具有相似功能的服务划为一个动作集合,因而能够实现对语义 Web 中的动态知识和静态知识的统一描述,克服描述逻辑缺乏对动态知识描述的不足。该算法能够利用目标规划给出主体的动作执行序列,以有效实现 Web 服务的自动组合。最后通过具体实例验证了该算法的可行性。

关键词 Web 服务组合, 语义 Web, 描述逻辑, 动态描述逻辑(DDL)

0 引 言

随着 Web 服务类型、数量的爆炸式增长,Web 服务从局部发展到了全球,从集中式发展到分布式,变成了一种新的分布式服务计算模型[1]。然而单个 Web 服务的功能有限,如何协调和组织多个 Web 服务来构造新的 Web 服务以提高服务组件的重用性和利用率,成为 Web 服务领域最具有挑战性的问题之一。目前 Web 服务组合可以分为三类。第一类是基于工作流的方法,如 BPEL4WS[2]、e-Flow[3]及 METEROR-S[4]。BPEL4WS 是由 IBM、微软和 BEA 公司共同提出的一种 Web 服务语言,通过一个流程将不同的 Web 服务组合起来。BPEL4WS 提供了统一的建模和定义工具,但在实际操作中需要人工参与,难以发挥 Web 服务的自适应特点。e-Flow 由 HP 实验室发布,对 Web 服务组合过程中的自适应性提供了一定的动态支持。虽然基于工作流的 Web 服务组合以半自动化方式实现,但是需要大量的领域专家参与,灵活性不强。第二类是基于人工智能规划的方法。范贵生等[5]采用 Petri 网对 Web 服务组合的过程进行描述,其优点是较为直观,但是随着服务组合规模的增大,容易引起状态空间爆炸问题。肖芳雄等[6]提出了一种时间概率代价进程代数。虽然该方法可以有效地对 Web 服务组合进行描述和验证,但缺乏对 Web 服务组合成本进行建模和分析的能力。Salam 等[7]基于业务流程执行语言(business process execution language, BPEL)提出了一个通用的架构,该架构可以在进程代数和 Web 服务间进行映射,并且举例说明了建模方法和推理工具。第三种是基于语义的方法。语义 Web 服务组合中比较有影响力的语言有:OWL-S[8],WSML[9],SWSO[10]和 WSDL-S[11]。但这些语言需要事先确定服务之间的关系,降低了服务的组合效率[12]。

针对描述能力和推理的复杂性,研究者开始使用描述逻辑[13,14]作为 Web 服务的描述语言。然而,描述逻辑的语义模型只包含 Web 服务的静态的信息,不能够对 Web 服务过程中的动态知识进行刻

① 973 计划(2013CB329502),国家自然科学基金(61035003, 60933004, 61202212, 61072085),863 计划(2012AA011003), 国家科技支撑计划(2012BA107B02)和中国信息安全测评中心(CNITSEC-KY-2012-006/1)资助项目。

② 男,1982 年生,博士;研究方向:认知科学,Web 服务组合;联系人,E-mail: zhangjh@ics.ict.ac.cn

(收稿日期:2013-08-12)

画。针对此问题,本文以动态描述逻辑(dynamic description logic,DDL)动作推理为基础,对语义 Web 中的动态知识和静态知识进行统一的刻画。在此基础上,借助动态描述逻辑中关于动作的推理功能,实现相关信息的推理,进而提出基于 DDL 的 Web 服务组合算法,实现了 Web 服务的自动组合。

1 动态描述逻辑

描述逻辑是基于对象的形式化知识表示,它吸取了 KL-ONE 的主要思想,是一阶谓词逻辑的一个可判定子集。描述逻辑具有合适定义的语义,且具有很强的表达能力。一个描述逻辑系统主要分为四个基本组成部分:表示概念和关系的构造集;TBox 包含断言;ABox 实例断言;TBox 和 ABox 上的推理机制[15]。

动态描述逻辑是在传统描述逻辑的基础上扩充得到的。描述逻辑是一个家族,本节以最小的描述逻辑 ALC 为基础。

1.1 语法

动态描述逻辑中有三个基本元素:概念、关系和动作。简单的概念、关系和动作通过构造算子,可以构造出复杂的概念、关系和动作。通常描述逻辑中包含以下算子:交(∩)、并(∪)、非(¬)、存在量词(∃)和全称量词(∀)。通过对基本的 ALC 添加算子,可以构建更复杂的描述逻辑。DDL 有多种动作算子,如:合成(;)、反复(*)、选择(?)等。在原子动作上添加动作算子,构成了复杂的动作,极大地增强了动态描述逻辑的表达能力。

在一个动态描述逻辑系统中,包含有概念的集合 N_C,关系的集合 N_R,个体集合 N_I,和动作的集合 N_A。

动态描述逻辑的概念定义如下:

$$C, D ::= \bot \mid A \mid \{u\} \mid \neg C \mid C \cup D \mid \forall R.C \tag{1}$$

其中,A 表示原子概念(概念名),C、D 表示概念(概念描述),u 表示个体名,R 表示关系名。

用 N_C 表示动态描述逻辑的所有概念名的集合,N_R 表示动态描述逻辑的所有关系名的集合。

动态描述逻辑的公式定义如下:

$$\phi, \psi ::= C(u) \mid R(u,v) \mid \neg \phi \mid \phi \vee \psi \mid < \pi > \phi \tag{2}$$

其中,u, v 表示个体名,C 表示概念,R 表示关系名,π 表示动作。将形如 $R(u, v)$、$\neg R(u, v)$,以及 $C(u)$ 和 $\neg C(u)$ 的公式都称为简单公式。

动态描述逻辑的 TBox TB 是由概念定义式和概念包含公理 $C \subseteq D$ 构成的有限集合,其中 C 和 D 是动态描述逻辑的概念。如果 $C \subseteq D$ 和 $D \subseteq C$,则记为 $C \equiv D$,称为概念等价公理。

给定某个 TBox,将 $\alpha \equiv (P, E)$ 称为一个原子动作定义式,其中:

- α 为所定义的动作名,$\alpha \in N_A$;
- P 是由简单公式组成的有限集合,表示动作在执行前需要满足的条件;
- E 是由简单公式组成的有限集合,表示动作执行后的效果;
- P 和 E 之间的关系,对于任一的 $\phi \in E$,都有 $\phi^{\neg} \in P$。

利用动态描述逻辑的动作构造算子,多个原子动作可以构造更加复杂的动作。动态描述逻辑的复杂动作可以根据以下规则生成:

$$\pi, \pi' ::= \alpha \mid \phi? \mid \pi \cup \pi' \mid \pi; \pi' \mid \pi^* \tag{3}$$

其中,α 表示原子动作,ϕ 表示公式。

其中,$\phi?$ 表示测试;$\pi \cup \pi'$ 表示选择;$\pi; \pi'$ 表示顺序;π^* 表示迭代。

动态描述逻辑还可以添加模态算子,这使得我们可以方便地断言可达世界的性质。$< \pi > \varphi$ 表示可达世界中至少有一个可以满足公式 φ,$\neg [\pi] \neg \varphi$ 表示断言所有 π 都满足公式 φ。

每个带参数的原子动作定义 $\alpha(v_1,\cdots,v_n) \equiv (P,E)$ 都描述了一类原子动作。对于 $(v_1,\cdots,v_n)$,相应的代入 n 个个体名称 $p_1,\cdots,p_n$ 之后,将得到一个原子动作定义式,$\alpha(p_1,\cdots,p_n) \equiv (P[v_1/p_1,\cdots, v_n/p_n], E[v_1/p_1,\cdots,v_n/p_n])$,其中 $P[v_1/p_1,\cdots, v_n/p_n]$ 和 $E[v_1/p_1,\cdots,v_n/p_n]$ 是将在 P 和 E 中出现的 $v_1,\cdots,v_n$ 用 $p_1,\cdots,p_n$ 进行替代后得到的公式集合。

1.2 语义

动态描述逻辑在传统描述逻辑的基础之上引入了动态维,使得语义表现为由多个可能世界构成的可能世界空间。动态描述逻辑将概念解释为一定论域的子集,将关系解释为该论域上的二元关系。

表 1 DDL 语义

Constructor	Semantic
Atomic Concept	$A^I \subseteq \Delta^I$
Atomic Relation	$PI \subseteq \Delta^I X \Delta^I$
Top	Δ^I
Bot	ϕ
Conjunction Concept	$C^I \cap D^I$
Disjunction Concept	$C^I \cup D^I$
Negation Concept	$\Delta^I - C^I$
Exist Restriction	$\{x \mid \exists y,(x,y) \in R^I \wedge y \in C^I\}$
Value Restriction	$\{x \mid \forall y,(x,y) \in R^I \Rightarrow y \in C^I\}$
Atomic Action	$\{< u,v > \mid u,v \in W, u \rightarrow_\alpha^\gamma v\}$
Sequence Action	$\{< u,v > \mid u,v,w \in W, u \rightarrow_\alpha^\gamma w \wedge w \rightarrow_\beta^{\gamma'} v\}$
Choice Action	$\{< u,v > \mid u,v \in W, u \rightarrow_\alpha^\gamma v \vee u \rightarrow_\beta^{\gamma'} v\}$
Iteration Action	$\{< u,v > \mid u,v \in W, u \rightarrow_\alpha^{\gamma'} v \vee u \rightarrow_{\alpha\mid\alpha}^{\gamma''} v \vee u \rightarrow_{\alpha\mid\alpha\mid\alpha}^{\gamma''} v \cdots\}$
Test Action	$\{< u,u > \mid u \in W, u \mid = \varphi\}$

描述逻辑形式上为一个三元组 $M = (\Delta, W, I)$。Δ 作为论域,W 是由可能世界组成的组合,I 对 W 中的每个可能世界 w 赋予一个解释函数 $I(w) = (\Delta, \cdot^{I(w)})$。解释函数把每个原子概念 $A \in N_C$ 映射到 Δ^I 的子集,把每个关系 $R \in N_R$ 映射到 $\Delta^I \times \Delta^I$ 的子集,而把每个个体名 p_i 解释为 Δ^I 中的某个个体 $p_i^{I(w)}$,且满足对于 W 中的任一可能世界 w' 都有 $p_i^{I(w)} = p_i^{I(w')}$;由于 p_i 的解释与可能世界无关,因此也将 $p_i^{I(w)}$ 简记为 p_i^I。

1.3 相关推理任务

推理的目的是用来发现服务的序列,推理服务由一个个的推理动作连接而成。在 DDL 中 TBox 界定论域,ABox 刻画当前环境。Badder[16] 给出了衡量服务的两个概念,可执行性和投影。

- 可执行性

在由 TBox 和 ABox 刻画的环境中,π 是可执行的,即对于任意满足 TBox 和 Abox 的动态描述逻辑解释有 (I,W),且 $(\pi)^{I,W} \neq \phi$。

动作作用在环境中,会引起环境的变化。判断某公式在执行后的环境中是否成立是动作的投影问题。

- 投影

在由 TBox 和 ABox 刻画的环境中,公式 φ 是动作 π 执行后的结果,对于任意满足 TBox 和 ABox 解释 (I,W),存在 $I' \in W$,且 $(I,I') \in \pi^I$,则有 $(I', W) \mid = \varphi$。

根据公式的定义,我们可以进一步对动作的相关属性进行刻画:

(1) π 在当前的环境中可以执行,当且仅当 $\mathrm{Conj(ABox)} \rightarrow [\ \pi\ ]\mathrm{true}$。

(2) π 在当前的环境中实现 φ,当且仅当 $\mathrm{Conj(ABox)} \rightarrow [\pi]\varphi$。

2 语义 Web 服务自动组合

根据 Web 服务的不同属性,可以将 Web 服务分为功能性描述和非功能性描述。Web 服务的功能性描述主要是针对服务的组合和发现。与 DDL 的动作相类似,Web 服务也是通过规划服务执行前后环境变化来进行刻画的。在本文中,我们将 Web 服务中的知识分为两部分,即静态知识和动态知识。静态知识包括当前的环境以及用户的需求,通过 TBox 和 ABox 进行刻画;动态知识为 Web 服务的功能,通过动作 ActBox 进行刻画。具有相似功能的 Web 服务放置到一个集合当中,集合中的所有元素具有相同的动作名称。同一个集合内部服务的主要区别在于非功能性属性。

服务的匹配问题之所以可以通过 DDL 的推理来解决,主要是基于以下两点:

(1) 在表现形式上,服务与动态描述逻辑中的动作有相似之处;

(2) 在计算的过程中,服务匹配问题可以规约为 DDL 公式的可满足性问题。

2.1 Web 自动组合算法预备定义

定义 1 服务之间无关。在环境 TBox 和 ABox

下,对于任意模型 I、I' 和服务 S_i、S_j,如果存在 $I \xrightarrow{s_i;s_j} I'$,都有 $I \xrightarrow{s_j;s_i} I'$,则称服务 S_i 和服务 S_j 不相关。

定义 2 服务之间相关。在环境 TBox 和 ABox 下,对于任意模型 I、I' 和服务 S_i、S_j,如果存在 $I \xrightarrow{s_i;s_j} I'$,没有有 $I \xrightarrow{s_j;s_i} I'$,则称服务 S_i 和服务 S_j 相关,服务的执行具有先后顺序约束。

定义 3 服务之间等价。在环境 TBox 和 ABox 下,对于任意的模型 I 和 I',如果存在服务 S_i 使得 $I \xrightarrow{s_i} I'$,存在服务 S_j 使得 $I \xrightarrow{s_j} I'$,则称 S_i 和 S_j 等价。

定义 4 一个 Web 服务功能性描述是一个 DLL 的原子动作。

$\alpha_i = \mathrm{des}(ws_i)$,$\alpha_i$ 表示 ws_i 服务集合功能性描述。$\alpha_i = (P,E)$,其中 P 为前提公式集,表示动作执行前必须满足的前提条件,也就是服务执行时需要满足的必要条件;E 为结果公式集,表示动作执行后的结果,在服务中表示服务执行后的结果。$ws_i = \mathrm{service}(\alpha_i)$,$ws_i$ 表示动作 α_i 所代表的服务集合。

定义 5 假设 WS 是所有的 Web 服务,根据 Web 服务的功能属性,对其进行划分,结果为 $WS = \{ws_1, ws_2, \cdots, ws_n\}$,其中 n 是对 WS 划分的个数。

每个 Web 服务子集对应一个 DDL 动作。同一个 Web 服务子集中的服务功能性属性相同,区别主要在于非功能性属性。

定义 6 系统的状态是一个不含语义冲突的 Abox。

随着 Web 服务组合的进行,系统中的状态随之进行动态改变。

定义 7 系统中所有的服务都有一个动作名称,具有相同动作名称的服务组成一个集合,所有的动作组成一个动作集合存储在 ActBox 中。

同一个集合中的不同服务之间具有相同的功能性属性,区别只是在于非功能性属性。

定义 8 服务组合。一个服务组合可以用一个个五元组进行表示:$< T, A, Act, G, WS >$。其中

(1) T 是该领域的概念集合;

(2) A 是包含概念集合的所有断言集合,同时也是整个系统的初始状态;

(3) Act 是系统中动作的集合;

(4) G 是所有目标状态的集合;

(5) WS 是所有系统服务的集合。

例:对于订电影票的一个过程:输入电影的名称及其放映时间地点,输入客户的信用卡账号,输出订票的信息(confirm)。订电影票的动作可以抽象为:BookMovieTicket ≡ ({MovieName($v1$), Time($v2$), Location($v3$), Client($v4$), own($v4$,$v1$)}, {Confirm($v5$)})。

BookMovieTicket 动作包含在服务集合 WS(BookMovieTicket)中。

2.2 语义 Web 服务组合算法

对于目标公式 g,我们有:

- g 在动作集合的结果状态中存在;
- 动作序列可以顺序且成功的执行;
- 根据用户的非功能性条件,选择出最终的动作。

算法 1: Plan

```
输入:ABox Env,TBox T, ActBox Act,目标 g。
输出:Web 服务组合成功实现输出实现队列,否则返回
      null,任务失败。
1. p = < >//初始化动作序列
2. if(¬ (Env)→[p]true 不可满足){
3.    if(¬ (Env)→[p]g 不可满足)
4.          return p;
5.     else{
6.        对目标 g 进行层次分解,直到不能分解为止,
          建立目标 g 的目标树;
7.        对目标树进行搜索得到子目标序列,g1,g2,
          …,gn;
8.    for CurG = g1 to gn{
9.      if(¬ (Env)→[act]true 不可满足){
10.    if(¬ (Env)→[act]CurG 不可满足){
11.              p = p∪act;}//end if CurG
12.          else{
13.              return null;}//end else
14.           }//end if true
15.         }//end for
16.        }//end else
17.}//end if
18 return p;
```

算法 1 需要输入的信息是系统的环境状态 ABox、系统的界定论域 TBox、系统的动作集合 ActBox 以及系统的目标 g。算法首先通过 tableau 算法验证当前的环境是否满足目标公式,如果当前的系统状态已经满足了目标公式,则不再需要进行动作序列的规划,直接返回空动作序列。如果当前的环境并没有实现目标公式,则对目标进行分解。对于目标 g 的分解,一直将目标分解到不能再继续分解的子目标为止。这样,我们专注在子目标的求解上。目标分解完毕后,对子目标进行排序,建立目标 g 的所有子目标,存入目标求解队列。

对子目标进行求解,将完成每个子目标的动作添加到 p 队列中。当所有的子目标求解完毕之后,返回动作序列。在计算的过程中,如果有任何一个子目标不能进行求解,目标求解失败,返回 null。

算法 2: Web service composition

输入:ABox Env,TBox T,Web 服务划分集合 WS,目标公式 g。
输出:成功返回目标 g 的组合动作,失败返回 null。

```
1. res = < >//初始化结果
2. for(i = 0;i < |WS|;i + + ){
3.    act = act ∪ des(ws_i);
4. }//end action set initialization
5. res = Plan(Env,T,WS,g);
6. if(res = = < >){
7.   当前环境满足目标,无需进行动作规划;
8. }//end if 目标自然满足,无需动作
9. else if(res = = null){
10.    系统不能产生目标 g 的动作规划:
11. }//end else if
12. else{
13. wsc = < >;//web service initialization
14. for(i = 0;i < |res|;i + + ){
15.   取 res 中的第 i 个动作 a_i;
16.   根据客户的非功能性需求,ws = service(a_i);
17.   wsc = <wsc,ws >;//向 wsc 队列添加 ws
18.   }//end for
19.}//end else
```

算法 2 首先初始化动作队列 res 为空。紧接着算法 2 调用算法 1,将结果存储在 res 中。然后算法 2 对返回的结果进行判断。如果算法返回的结果是 < >,说明系统当前的状态不需要采取任何动作就能够满足目标 g;如果算法返回的是 null,这说明该算法不能够完成客户的请求,Web 组合失败;如果算法 1 成功返回动作序列,则按照客户的非功能性需求,如时间、金钱等从提供相关动作的服务中选择,最后将这些服务组合在一起完成了客户的需求。

算法中只对目标 g 分解了一次,所以,即使存在 Web 服务组合可以组合的情况的,存在不能求解的情况。从这个角度说,我们的算法是不完全的。Web 服务组合过程中,除了服务的功能性需求外,还有非功能性的需求。为此,为了更好地满足客户的 Web 服务请求,系统中的 Web 服务资源越多,越能够满足客户的需求。

3 案例研究

为了进一步说明服务组合的过程,此节给出一个 Web 自动服务组合的案例。

假定一个旅行者要从北京到海南岛去旅游,希望在提供旅游服务的网站上完成整个旅行安排。客户的要求如下:提交旅游计划的时间,系统返回需要乘坐的交通工具和预订的酒店信息。用户希望输入时间和个人喜好后,系统给出乘车路线并预订酒店房间。顾客可以乘坐火车也可以乘坐飞机。

在这个案例中,用户需求涉及到 3 个 Web 服务。预订火车 W1 服务,预订飞机 W2 服务,预订酒店 W3 服务。但是这些服务分布在不同的服务器中,任何单个的服务器都不能满足客户的需要。

为了简单说明组合多个服务的推理过程,下面给出单个 Web 服务集合的精简描述,但是不考虑单个 Web 服务的细节。

预订火车服务:

w11 = (P11,E11);

P11 = {Departure _ data(d)};

E11 = {Airplane _ Ticket,Arrive _ date(d)};

预订飞机服务:

w21 = {P21,E21};

P21 = {Departure _ data(d)};

E21 = {Train _ Ticket,Arrive _ date(d)};

预订酒店服务:

w31 = {P31,E31};

P31 = {Arrive _ date(d)};

E31 = {hotel _ booking};

根据服务的类型,系统中将相同类型的服务组合在一个集合中,同一集合中的服务区别主要在非功能属性上。即 $ws1 = \{w11, w12, \cdots w1n\}$,其中 n 为 $ws1$ 中服务的个数,$ws2 = \{w21, w22, \cdots, w2m\}$,其中 m 为 $ws2$ 中服务的个数,$ws3 = \{w31, w32, \cdots, w3k\}$,其中 k 为 $ws3$ 中服务的个数。将 3 种类型的 Web 服务封装成 3 种类型的 action。

以上 3 种类型的服务都定义在无环的 TBox 中,$T = \{\text{Traffic \_ tool} = \text{Airplane \_ Ticket} \cup \text{Train \_ Ticket}\}$。

假设客户具有个人偏好,对时间没有要求,而希望花费的代价最小。添加个人喜好动作,action = {ϕ, Min _ cost}。目标 g = {Traffic _ tool, hotel _ booking}。系统将目标 g 分解为 2 个子目标,$g1$ = Traffic _ tool 和 $g2$ = hotel _ booking。根据子目标执行顺序,$g1$ 子目标优先于 $g2$ 执行。目标 g 分解到不能再分解的子目标之后,就可以从集合中选择相应的 Web 服务。在本例中,根据客户输入的时间可以从 Web 服务中选择乘坐铁路还是乘坐飞机。

利用上述算法,在服务计算阶段就可以根据实际应用的语义信息定义各个服务部件需要满足的前提条件并以逻辑的形式表示。在选择出的旅游出行工具后,按着客户的个人喜好在后选的出行工具中选择价格最便宜的。出行工具确定后,然后就确定了达到的日期,从而可以进一步给客户预订放假。

经过上述执行 Web 服务组合后,客户就可以获知出行的交通工具以及预订的酒店信息。

4 结 论

本文提出了一种基于动态描述逻辑的(DDL) Web 服务组合算法。该算法将原子服务描述为 DDL 中的原子动作,充分利用了 Web 组合过程中所涉及到的动态知识以及静态知识。通过将 Web 服务封装为动作,利用目标规划算法,给出主体的动作执行序列,实现 Web 服务的自动组合,有效提高了自动组合能力。接下来的主要工作是进一步拓展 Web 服务的非功能性描述,在满足 Web 服务功能的基础之上给出满足用户偏好的 Web 服务组合,以及研究如何采用 DDL 更加有效的表示动作。

参考文献

[1] 邱莉榕,史忠植,林芬等. 基于主体的语义 Web 服务自动组合研究. 计算机研究与发展,2007,44(4):643-650

[2] Mandell D J, Mcilraith S A. Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In: Proceedings of the Semantic Web-ISWC2003 . Berlin Heidelberg :Springer,2003. 227-241

[3] Casati F, Ilnicki S, Jin L, et al. Adaptive and dynamic service composition in eFlow. In:Proceedings of the 12th International Conference on Advanced Information Systems Engineering, Stockholm, Sweden, 2000. 13-31

[4] Patil A A, Oundhakar S A, Sheth A P, et al. Meteor-s web service annotation framework. In: Proceedings of the 13th International Conference on World Wide Web, New York, USA, 2004. 553-562

[5] 范贵生,虞慧群,陈丽琼等. 基于 Petri 网的服务组合故障诊断与处理. 软件学报,2010,21(2):231-247

[6] 肖芳雄,李燕,黄志球等. 基于时间概率代价进程代数的 Web 服务组合建模和分析. 计算机学报. ,2012,35(5): 918-936

[7] Salaun G, Bordeaux L, Schaerf M. Describing and reasoning on web services using process algebra. *International Journal of Business Process Integration and Management*, 2006, 1(2): 116-128

[8] Martin D, Burstein M, Mcdermott D, et al. Bringing semantics to web services with OWL-S. *World Wide Web*, 2007, 10(3): 243-277

[9] De Bruijn J, Lausen H, Polleres A, et al. The web service modeling language WSML: An overview. The Semantic Web: Research and Applications. Berlin Heidelberg : Springer, 2006. 590-604

[10] Battle S, Bernstein A, Boley H, et al. Semantic web services ontology (swso). http://www.w3.org/Submission/SWSF-SWSO: W3C. 2005

[11] Akkiraju R, Farrell J, Miller J A, et al. Web service se-

mantics-wsdl-s. http://www. w3. org/Submission/WS-DL-S: W3C. 2005

[12] 邱爽,王亚东,刘永壮. 基于语义 Web 服务内部流程接口匹配的服务组合算法. 高技术通讯,2012,22(6): 596-603

[13] 刘思培,刘大有,齐红等. 基于描述逻辑规则的语义 Web 服务组合. 计算机研究与发展,2011,48(5): 831-840

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

[15] 史忠植,董明楷,蒋运承等. 语义 Web 的逻辑基础. 中国科学: E 辑,2004,34(10): 1123-1138

[16] Baader F, Lutz C, Milicic M, et al. A description logic based approach to reasoning about web services. In: Proceedings of the WWW 2005 Workshop on Web Service Semantics, Chiba City ,Japan, 2005. 10-14

Automatic composition of Web services based on dynamic description logic

Zhang Jianhua[*,**], Tian Dongping[*,**], Yue Jinpeng[*,**], Zhang Bo[*,***]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116)

Abstract

This study focused attention on the application of the dynamic description logic (DDL) to the Web service composition, and gave a new algorithm for Web service composition based on dynamic description logic and goal planning. The algorithm considers users' preference to expand the application range of Web service. It uses the DDL to depict Web service composition because the DDL puts the services with the same function into an action set, so the unified representation of the static knowledge and dynamic knowledge in semantic Webs can be realized to overcome the description logic's shortcoming of lack of dynamic knowledge description. Meanwhile, it uses the goal planning to give the main part's sequence of action execution to achieve effective automatic composition of Web services. The algorithm's feasibility was verified by a simple example.

Key words: Web service composition, semantic Web, description logic, dynamic description logic(DDL)