

基于完备回溯树的语义 Web 服务自动组合

李瑞宁,周竹荣

(西南大学 计算机与信息科学学院,重庆 400715)

(zhouzr@swu.edu.cn)

摘 要:基于语义的 Web 服务自动组合方法具有较高的效率及自动化程度,能更好地解决复杂的服务组合问题。提出一种基于完备回溯树的语义 Web 服务自动组合方法(CBT_ASWSC),该方法为 Web 服务引入语义以实现对象间的语义转化并将搜索空间受限于完备回溯树中,在加快 Web 服务组合效率的同时提高了 Web 服务组合的成功率。

关键词:语义 Web 服务;服务组合;完备回溯树

中图分类号: TP311 **文献标志码:** A

Automatic semantic Web services composition based on complete backward tree

LI Rui-ning, ZHOU Zhu-rong

(College of Computer and Information Science, Southwest University, Chongqing 400715, China)

Abstract: Compared to the traditional ways of composing Web services, semantic-based ways of composing Web services automatically are more effective and automatic. An approach based on complete backward tree to compose semantic Web services automatically was proposed. It not only added the semantics to Web services, but also had a smaller search space limited to the complete backward tree. Experimental results show that this method improves both the efficiency and the successful ratio of Web services composition.

Key words: semantic Web services; services composition; Complete Backward Tree (CBT)

0 引言

传统的 Web 服务组合方法有:基于工作流的组合^[1]、基于人工智能规划的自动服务组合^[2]、基于中间件的服务组合^[3]和基于图搜索的自动服务组合^[4]。这些方法存在的问题^{[4]1897}有:1)自动化程度不高且效率较低;2)方法的复杂度较高,不易实现;3)分析手段较为固化,不能适应需求的动态变化;4)服务关系图的构建时间开销很大。

为了解决以上问题,文献[5]提出一种新的基于语义的 Web 服务自动组合方法,将语义引入 Web 服务,在服务发现中应用本体推理,以实现服务的自动发现、组合且具有较高的效率。但在 Web 服务数量大且关系复杂时,构建和遍历服务本体关系图将耗费大量时间,从而影响该类方法的实际可用性。在传统的 Web 服务组合方法中,效率较高的是文献[4]提出的基于回溯树的 Web 服务自动组合方法,但该方法对 Web 服务的定义缺乏语义,服务组合的成功率不高。

本文提出一种基于完备回溯树的语义 Web 服务自动组合方法(CBT_ASWSC)。该方法将基于图搜索与基于语义 Web 的自动服务组合方法融为一体,利用本体来标注 Web 服务的语义信息,通过计算本体概念间的语义相似度,为用户请求的每一个输出对象及时建立完备回溯树,并在回溯树的构建过程中选取最佳有效生成路径,最后将路径合成为可运行的流程服务。该方法为 Web 服务引入语义,并将搜索空间受限于完备回溯树中,提高了 Web 服务自动组合的成功率及效率。

1 相关理论

1.1 语义 Web 服务与 Web 服务组合

语义 Web 服务 (semantic Web services) 是语义网研究的

重要领域,也是 Web 服务的发展趋势^[6]。语义 Web 服务的目标是通过 Web 服务的语义描述使机器和人都能一致地理解和使用 Web 服务,并实现 Web 服务的自动发现、组合、互操作和监控。

由于目前对 Web 服务组合尚未有统一的定义,文献[7]综合众多观点,给出了一个较好的定义:Web 服务组合是指当单个 Web 服务无法满足用户需求时,将若干 Web 服务进行有机合成,以形成大粒度的具有内部流程逻辑的组合服务,并通过执行组合服务而达到业务目标的过程。作为 SOA 开发软件应用、实现业务过程的核心方法和技术,Web 服务组合并不是一个孤立的问题,它包含了 Web 服务发现、Web 服务组合、Web 服务组合验证、Web 服务执行与监控、Web 服务组合安全和事物管理等关键问题,这些问题共同构成了 Web 服务组合的研究框架。

1.2 语义相似度计算

本体概念间的语义相似度计算是基于语义的服务匹配问题中服务与请求的相似度计算的基础。目前,学术界提出的计算方法可被划分为两大类:基于信息共享量的计算方法和基于树的计算方法。其中,基于树的计算方法将所有的本体概念构建成一棵概念树,利用两个概念在概念树中的最短路径、概念在概念树中的层高以及所在层高的节点密度等作为量度指标。如文献[8]在综合考虑这些指标之后,提出了两个概念间相似度的计算公式:

$$\text{SimCC}(C_1, C_2) = \begin{cases} e^{-al} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & C_1 = C_2 \\ 1, & C_1 \neq C_2 \end{cases} \quad (1)$$

其中, l 表示两个概念间的最短路径长度; h 表示两个概念在

树中最近的相同父辈概念在树中的高度; $\alpha, \beta \geq 0$ 是调节因子,据文献[8]中的实验,当 $\alpha = 0.2, \beta = 0.8$ 时效果最佳。式(1)表明,两个概念的相似度随 l 递减,随 h 递增。该方法通用性较好,容易实现,因此本文选用该方法作为两个本体概念间的语义相似度的计算方法。

2 基于完备回溯树的语义 Web 服务自动组合

CBT_ASWSC 方法对文献[4]的组合方法作了以下改进:1)基于语义 Web 服务规则库构建完备回溯树(Complete Backward Tree, CBT)^{[4]1900},CBT 的节点所包含的输入、输出对象均与特定的本体概念相关联;2)通过计算本体概念间的

语义相似度完成 CBT 的构建,实现对象间的语义转化;3)基于语义完成对象间的匹配;4)将 CBT 的生成与最佳有效生成路径(Optimal Valid Generation Path, OVGP)^{[4]1900}的选取结合在一起。

2.1 CBT_ASWSC 方法模型

CBT_ASWSC 方法模型如图1所示。左侧为利用本体来标注 Web 服务的语义信息,是本文的后续工作;中间深色模块为通过计算本体概念间的语义相似度,为用户请求的每一个输出对象即时建立 CBT,并在 CBT 的构建过程中选取 OVGP,是本文的重点;右侧为将路径合成为可运行的流程服务^{[4]1904}。

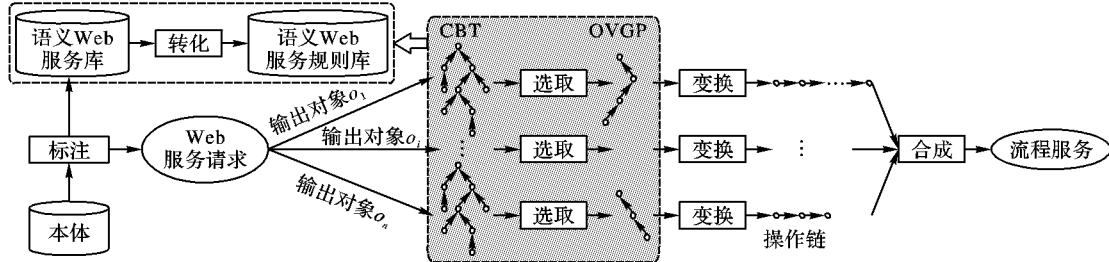


图1 CBT_ASWSC 方法模型

2.2 语义 Web 服务与语义 Web 服务规则库

对于 Web 服务库 $WS = \{ws_1, ws_2, \dots, ws_n\}$ 来说,一个 Web 服务可以被看做一个原子的、不可分解的流程,即黑盒视图^[9]。在这种视图下 Web 服务等同一组操作的集合,所有内部过程被屏蔽,对外暴露的只有它的输入和输出接口。因此,可抽取 OWL-S 服务概述的接口描述部分,将 Web 服务抽象为如下形式。

定义1 语义 Web 服务。语义 Web 服务是对 Web 服务所支持的操作的抽象描述,可表示为一个二元组 $ws = (N, P)$,其中: N 为服务的名称,是服务的唯一标识; $P = \{p_1, p_2, \dots, p_n\}$ 为该服务的操作集合。操作又可表示为一个二元组 $p = (I, O)$,其中, $I = \{i_1, i_2, \dots, i_k\}$ 是操作接收的输入对象集合; $O = \{o_1, o_2, \dots, o_l\}$ 是操作产生的输出对象集合。每个输入、输出对象都用本体进行标注。

操作是 Web 服务的基本功能实体,Web 服务组合最终体现在服务间操作的组合。每个 ws 的操作都可被抽象为一个从操作接收的输入对象到操作产生的输出对象的产生式规则^{[4]1897},记为 $r = (O_s, O_d, p)$ 。将 ws 内部每一个操作转化为若干 r 后, $ws = (N, P)$ 可视为规则的集合,而语义 Web 服务库可视为语义 Web 服务规则库。

定义2 语义 Web 服务规则库。语义 Web 服务库 $WS = \{ws_1, ws_2, \dots, ws_n\}$ 对应的语义 Web 服务规则库为 $R_{WS} = (I, O, R)$,其中:1) $I = \bigcup_{ws \in WS} \left\{ \bigcup_{p \in ws.P} p.I \right\}$ 是 R_{WS} 接受的所有输入对象,每个对象都用本体进行标注($ws.P$ 为 ws 中的操作集合, $p.I$ 为操作 p 的输入对象集合);2) $O = \bigcup_{ws \in WS} \left\{ \bigcup_{p \in ws.P} p.O \right\}$ 是 R_{WS} 能产生的所有输出对象,每个对象都用本体进行标注($p.O$ 为操作 p 的输出对象集合);3) $R = \bigcup_{ws \in WS} \left\{ \bigcup_{p \in ws.P} \varphi(p) \right\}$ 是 R_{WS} 中的所有规则($\varphi(p)$ 为操作 p 的产生式规则集合)。

为了实现 CBT_ASWSC,必须完成从 WS 到 R_{WS} 的自动转化^{[4]1898},即依次遍历 WS 中的每一个 ws ,针对其操作的输出分别生成一条 r 。

2.3 CBT_ASWSC

基于黑盒视图的服务发现策略通常认为,判定一个服务

能否满足服务请求,首先需要检测该服务能否提供请求所需的所有输出;如果能,则再检验服务请求能否提供调用该服务所需要的所有输入。如若服务库中不存在满足用户请求的服务,则需要组合原子服务来满足服务请求。

定义3 Web 服务请求。Web 服务请求是对用户请求的接口的抽象描述,可表示为一个二元组 $r_q = \{I', O'\}$,其中:1) $I' = \{i'_1, i'_2, \dots, i'_m\}$ 是用户可提供的输入对象集合;2) $O' = \{o'_1, o'_2, \dots, o'_n\}$ 是用户请求的目标输出对象集合。每个输入、输出对象都用本体进行标注。

给定一个 $R_{WS} = (I, O, R)$ 和一个 $r_q = \{I', O'\}$, CBT_ASWSC 方法分2步完成:1)完备回溯树的自动生成及最佳有效生成路径的选取;2)合成最佳有效生成路径。

2.4 CBT 的自动生成及 OVGP 的选取

给定一个 $R_{WS} = (I, O, R)$ 和 $r_q = \{I', O'\}$ 的一个输出对象 $o \in O'$,可通过 CBTC 算法^{[4]1902}的改进算法——S_CBTC&OVGPS算法自动创建输出对象 o 的 CBT $t_o^* = (V, \chi, E)$,并在创建过程中即时判断 OVGS,选取 OVGP。在 CBT_ASWSC 方法中我们设定一个阈值 T_s 来判断两个对象是否语义匹配,即,若两个对象通过式(1)计算得到的相似度值高于 T_s ,则认为它们是语义匹配的,否则认为不匹配。文献[10]给出三种启发式的方式来确定阈值的取值,分别为固定相似值法、差值法和 n 百分比法。究竟使用那一种阈值,目前还只能依赖于直觉。我们选择固定相似值法^[10],即阈值 T_s 为一个由领域专家指定的固定相似值, $T_s = C$ 。

算法1 S_CBTC&OVGPS

输入: A Semantic Web Service-Rule-Repository $R_{WS} = (I, O, R)$ and an output object $o \in O'$;

输出: OVGP $P_{v \rightarrow r} = \langle e_i, \dots, e_j, \dots, e_k \rangle$ and OVGS $g = (P_{v \rightarrow r}, v, \chi(v))$ for o .

- 1) Set $tQueue$ as a first-in-first-out queue of node; $tNode$ as a node; o as the root of t_o^* ;
- 2) Set $s1 = s2 = 0$ as integer; Flag as boolean;
- 3) $V.add(o)$; //把根节点加入集合 V 中
- 4) $EnQueue(tQueue, o)$; //根节点插入队列 $tQueue$ 的尾部
- 5) While($!tQueue.empty()$) {

```

6) DeQueue (tQueue, tNode);
   // 删除队头节点并把它赋予 tNode
7) For each  $o^x \in (\chi(tNode) \cap O)$  {
   // 逐一计算  $o^x$  中的对象与 tNode 的语义相似度值并与
    $T_s$  比较, 找出最佳匹配。
8) For each  $o \in O$  {
9) if ( $simCC(o^x, o) > s1$ )
   {  $s1 = simCC(o^x, o); o^s = o; \}$ 
10) if ( $s1 \geq T_s$ ) Set  $O^{tNode} = (O^{tNode} \cup o^s) \cap O$ ;
11) else Return false; // 找不到最佳匹配, 算法结束
12) For each  $o' \in O^{tNode}$  {
13) Set  $l = \langle e_1, e_2, \dots, e_n \rangle$  as the path from tNode to the root;
14) For each  $r \in R$  {
15) If ( $simCC(r, O_d, o') \geq T_s$  &&  $r, O_s \cap [\bigcup_{e_i \in l'} e_i, O_d] = \emptyset$ ) {
16) Create a new node cNode;
17) Set  $\chi(cNode) = (\chi(tNode) - o') \cup r, O_s$ ;
   For each  $o^c \in \chi(cNode)$  {
   // 判断 cNode 是否是 OGS
18) For each  $i \in l'$ 
19) if ( $simCC(o^c, i) > s2$ )  $s2 = simCC(o^c, i)$ ;
20) if ( $s2 < T_s$ ) Set Flag = 1;
   else Set Flag = 0;
21) if (Flag) { // cNode 不是 OGS
22) EnQueue (tQueue, cNode);
   V. add (cNode);
23) Create an edge  $e = (cNode, tNode, r)$ ;
   E. add (e);
24) else { // cNode 是 OGS
25) Return OVGP  $P_{cNode \rightarrow r} = \langle e_i, \dots, e_j, \dots, e_k \rangle$ ;
26) Return OVGS  $g = (P_{cNode \rightarrow r}, cNode, \chi(cNode))$ ;
27) } } } } }

```

算法 S_CBTC&OVGPS 采用递归方式以从根节点自顶向下、从左到右逐层添加节点和有向边的方式构造 o 的 CBT, 从根节点开始将每个节点 V 包含的对象与 R_{ws} 中的输出对象集合 O 进行语义相似度匹配, 选取满足 T_s 的 r 来构造 CBT 及确定新增节点的对象集合; 每新增一个节点, 就通过条件 $\chi(V) \subseteq I'$ 判断该节点是否为有效生成源, 若不是, 继续构造 CBT; 否则, 该节点即为 o 的 OVGS (因为由该节点确定的 OVGP 的长度一定是最短的), 算法终止, 停止构造 CBT。

算法 S_CBTC&OVGPS 总是可以在有限递归次数后停机, 因为: 1) R_{ws} 中的输入、输出对象集合与规则集合的元素均为有限个; 2) 为每个节点添加其儿子节点时, 均以条件 $r, O_s \cap [\bigcup_{e_i \in l'} e_i, O_d] = \emptyset$ 避免了从根节点到新增节点的路径上出现对象间循环转化而使得路径无限长的情况; 3) 一旦新增节点被确定为 OVGS, 即时返回结果, 停止构造 CBT, 即通常情况下算法不需要构造完整的 CBT。

在最坏的情况下 S_CBTC&OVGPS 构造的一棵 CBT 包含 n 个节点; 每个节点包含的对象个数最多为 m ; R_{ws} 中的输出对象个数最多为 q ; 一个 r_q 最多包含 s 个输入对象。两个本体概念间的语义相似度计算的时间复杂度为 $O(1)$, 则有 S_CBTC&OVGPS 最坏情况下的时间复杂度为 $O(n \times m \times q + n \times m^2 \times q \times s)$, 空间复杂度为 $O(n)$ 。通常算法不需要为每个输出对象 o 构建完整的 CBT, 就能找到 OVGS, 即复杂度较最坏情况要少得多。

图 2 为 R_{ws} 示例, 其左、右两侧分别表示输入和输出对象集合, 内部为从输入对象 (集合) 到输出对象的转化规则。假

定有一个 $r_q = \{I', O'\}$, 其中 $I' = \{a, c, D, E, f\}$, $O' = \{A, FF\}$ 。我们以图 2 所示的 R_{ws} 为例, 分步说明 S_CBTC&OVGPS 的执行过程。

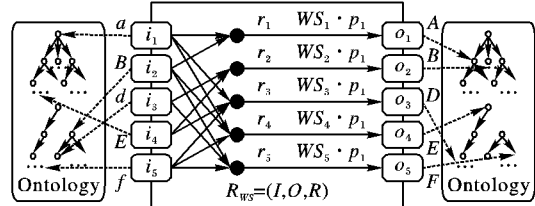


图2 语义 Web 服务规则库示例

1) 初始情况如图 3(a) 所示, 队列 tQueue 中仅有以输出对象 FF 为标识的根节点 r 。从 tQueue 中弹出 r , 得到 $tNode = r$, 即 $\chi(tNode) \cap O = \{FF\}$ 。此时 $o^x = FF$, FF 与 R_{ws} 的输出对象 o 逐一进行语义匹配, 算法选取语义相似度值大于 T_s 的输出对象 F 替换 FF, 即 $O^{tNode} = \{F\}$, $o' = F$ 。如图 3(b) 所示。

2) 由于规则 r_5 的目标对象集合 $r_5, O_d = \{F\}$ 且 $r_5, O_s \cap [\bigcup_{e_i \in l'} e_i, O_d] = \{B, d, f\} \cap \emptyset = \emptyset$, 即, r_5, O_d 与 o' 完全匹配且从 r 到后选节点 $\{B, d, f\}$ 的路径上不存在对象间的循环转化, 满足 CBT 的构造条件。为节点 r 添加儿子节点 n_1 以及从 n_1 指向 r 的有向边 $e_1 = (n_1, r, r_5)$, $\chi(n_1) = \{B, d, f\}$ 。由于 S_CBTC&OVGPS 每生成一个儿子节点就判断该节点是否为有效生成源, 因此需要判断 $\chi(n_1)$ 是否语义包含于 r_q 的输入对象集合 I' 中, 即判断 $\chi(n_1) \subseteq I'$ 是否成立。逐一计算 $\chi(n_1)$ 与 I' 所包含对象间的语义相似度值, 由于对象 B 无法在 I' 中找到最佳匹配, 因此, n_1 不是有效生成源, 继续构造 CBT。将节点 n_1 插入队列 tQueue 的尾部, 此时的 CBT 形如图 3(c)。

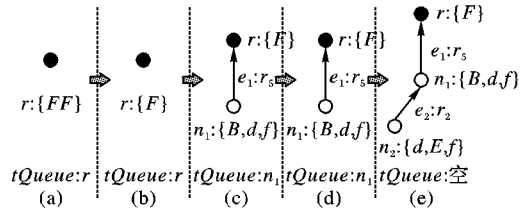


图3 S_CBTC&OVGPS 的执行过程

3) S_CBTC&OVGPS 弹出 tQueue 头部节点 n_1 , 得到 $tNode = n_1$, 即 $\chi(tNode) \cap O = \{B, d\}$ 。因 $simCC(B, B) = 1 \geq T_s$, $simCC(d, D) \geq T_s$, S_CBTC&OVGPS 选取输出对象 D 替换 d 。此时 $O^{tNode} = \{B, D\}$, $o' = \{B, D\}$ 。如图 3(d) 所示。

4) $O^{tNode} = \{B, D\}$, 分别考察 B 和 D 两个输出对象。首先, 当 $o' = B$ 时, 由于规则 r_2 的目标对象集合 $r_2, O_d = \{B\}$ 且 $r_2, O_s \cap [\bigcup_{e_i \in l'} e_i, O_d] = \{d, E\} \cap \{F\} = \emptyset$, 于是为节点 n_1 添加儿子节点 n_2 以及从 n_2 指向 n_1 的有向边 $e_2 = (n_2, n_1, r_2)$, $\chi(n_2) = \{d, E, f\}$, 如图 3(e) 所示。因 $\{E, f\} \in I'$ 且 $simCC(d, D) \geq T_s$, 故 $\chi(n_2) \subseteq I'$ 成立, n_2 是有效生成源。又因 CBT 是按树的层遍历方式逐层生成的, 则若 n_2 是有效生成源, n_2 也一定是 OVGS。S_CBTC&OVGPS 找到 OVGS, 返回 $OVGP(P_{n_2 \rightarrow r} = \langle e_1, e_2 \rangle)$ 及 $OVGS(g = (P_{n_2 \rightarrow r}, n_1, \{d, E, f\}))$, 算法终止。

3 仿真实验与评估

在仿真实验中选用了与文献[4]相似的实验环境。首先, 用基于 IBM XML Generator 开发的 Web 服务描述文档生成工具生成了 1000 个符合 OWL-S 文件规范的文档, 并保证其中 25% 的 Web 服务有 1 个输出、45% 的有 2~4 个输出、

25% 的有 5~9 个输出以及 5% 的有 10~15 个输出,对于输入也采用同样的比例。我们从 WordNet^[11]中挑选约 2 000 个本体概念的子本体,用于输入、输出对象的语义标注。其次,因为仅抽取了 Web 服务描述文档中的接口描述部分,为了构造用于测试的 R_{ws} ,需要针对每个 ws 的输入、输出对象随机生成产生式规则,即在生成一条新的 $r = (O_s, O_d, p)$ 时,从 ws_i 的输出对象集合中随机选取一个作为 r 的目标对象,从 ws_i 的输入对象集合中随机选取 1~5 个作为 r 的源对象集合,并以规则生成的序号及对应的 ws 序号作为该规则的标识。

表 1 服务请求在不同规模规则库中的处理结果

规则库	规则条数	可满足的用户请求数	不可满足的用户请求数
R_{ws1}	100	42	58
R_{ws2}	200	53	46
R_{ws3}	500	57	43
R_{ws4}	1 000	83	17
R_{ws5}	2 000	89	11
R_{ws6}	5 000	95	5

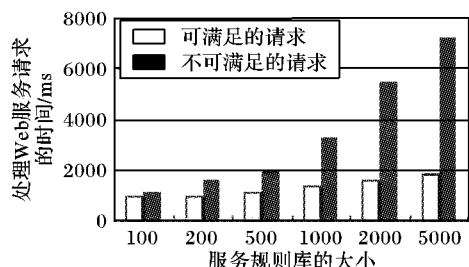


图 4 不同规模规则库处理服务请求的时间比较

采用上述服务规则的模拟生成方法,产生了 6 个不同规模的规则库 $R_{ws1} \sim R_{ws6}$,令它们分别包含 100、200、500、1 000、2 000 和 5 000 条不同的规则,在每一个 R_{ws} 中依次提交 100 次不同的 Web 服务请求 $r_q = \{I', O'\}_{\circ} r_q$ 采用与 ws 同样的方式随机生成,并与 ws 共享同一个子本体。表 1 表明在 R_{ws} 规模不等的情况下 CBT_ASWSC 方法处理 100 个 r_q 并返回组合服务的成功和失败次数。图 4 表明在 R_{ws} 规模不等的情况下 CBT_ASWSC 方法处理 100 个 r_q 并返回结果所耗费的时间。

由上述实验结果可知, CBT_ASWSC 方法与文献[4]的方法相比:1) 服务组合的成功率有所提高。从表 1 可知对不同规模的 R_{ws} ,每 100 个 r_q 被满足的个数均有所提高,不被满足的个数亦有所减少。2) 对可满足的 r_q 的处理时间有所减少,而对不可满足的 r_q 的处理时间有所增加。这是因为算法 S_CBTC&OVGPS 在 CBT 的构建过程中完成了 OVGP 的选取,对可满足的 r_q 不需要为其每个输出对象构建完整的 CBT,大大缩短了处理时间;而对不可满足的 r_q 则需要为其至少一个输出对象构建完整的 CBT,加上判定每一个新增节点是否为

OVGP 所耗费的时间,将导致处理时间的增加。

4 结语

本文提出了一种基于完备回溯树的语义 Web 服务自动组合方法(CBT_ASWSC)。该方法为 Web 服务引入语义,实现了对对象间的语义转化,能够处理对象间不存在完全匹配的情况,从而提高了 Web 服务组合的成功率。算法 S_CBTC&OVGPS 将搜索空间受限于完备回溯树中,但仅在最坏情况下需要生成和遍历完整的回溯树,进一步提高了服务组合的效率。

今后的工作包括:1)降低 S_CBTC&OVGPS 的时间复杂度,用更快速的匹配算法代替对象间逐一比较的匹配方法;2)对组合后的流程服务进行规约化简,以提高 CBT_ASWSC 方法的效率。

参考文献:

- [1] CARDOSO J, SHETH A. Semantice-workflow composition [J]. Journal of Intelligent Information System, 2003, 8(6): 191-225.
- [2] RAO J H, KUNGAS P, MATSKIN M. Logic-based web services composition: from service description to process model [C]// Proceedings of the International Conference on Web Services. Washington: IEEE Computer Society, 2004: 446-453.
- [3] 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[J]. 计算机研究与发展, 2007, 44(4): 643-650.
- [4] 邓水光, 吴健, 李莹, 等. 基于回溯树的 Web 服务自动组合[J]. 软件学报, 2007, 18(8): 1896-1910.
- [5] SILVA E, PIRES L F, Van SINDEREN M. An algorithm for automatic service composition [C]// 1st International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ICSOF 2007). Barcelona, Spain: INSTICC Press, 2007: 65-74.
- [6] 梁晟. 基于语义 Web 的服务自动组合技术的研究[D]. 北京: 中国科学院研究生院, 2004.
- [7] 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.
- [8] LI YU-HUA, BANDAR Z A, MCLEAN D. An approach for measuring semantic similarity between words using multiple information sources [J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(4): 871-882.
- [9] MARTIN D, BURSTEIN M, HOBBS J, et al. OWL-S: Semantic markup for Web services [EB/OL]. (2004-11-22) [2007-10-29]. <http://www.w3.org/Submission/OWL-s/>.
- [10] 张晨或. 基于语义度量的本体映射及语义查询的研究[D]. 合肥: 合肥工业大学, 2005.
- [11] MILLER G A. WordNet: A lexical database for English [J]. Communication of the ACM, 1995, 38(11): 39-41.

(上接第 1426 页)

- [9] HOFMANN T. Probabilistic latent semantic indexing [C]// Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 1999: 50-57.
- [10] BLEI D M, NG A Y, JORDAN M I. Latent dirichlet aladdress [J]. Journal of Machine Learning Research, 2003, 3: 993-1022.
- [11] MEI QIAO-ZHU, SHEN XUE-HUA, ZHAI CHENG-XIANG. Automatic labeling of multinomial topic models [C]// Proceedings of the Thirteen ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2007: 490-499.

- [12] WANG JIN-LONG, XU CONG-FU, LI GANG, et al. Understanding research field evolving and trend with dynamic Bayesian networks [C]// Proceedings of the Eleventh Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin: Springer-Verlag, 2007: 320-331.
- [13] HECKERMAN D. A tutorial on learning with Bayesian networks [M]// JORDAN M, ed. Learning in graphical models. Cambridge, MA: MIT Press, 1999: 301-354.
- [14] YU JING, SMITH V A, WANG P P, et al. Using Bayesian network inference algorithms to recover molecular genetic regulatory networks [C]// Proceedings of the Fourth International Conference on Systems Biology. Stockholm, Sweden: [s. n.], 2002.