

基于动态描述逻辑的语义 Web 服务推理

史忠植 常 亮

(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)

摘 要 语义 Web 服务的提出是为了解决 Web 服务资源在语义 Web 中的自动发现、组合和执行等问题,基本思路是将本体承载的静态知识与对 Web 服务动态功能的描述有机地结合起来. 动态描述逻辑是描述逻辑的一种动态扩展,支持面向语义 Web 的对动作的描述和推理. 文中利用动态描述逻辑 $DDL(SHOIN(\mathbf{D}))$ 的描述和推理功能,提出对语义 Web 服务进行建模和推理的一种有效途径. 从 OWL-S 中的 ProcessModel 出发,将语义 Web 服务建模为基于 $DDL(SHOIN(\mathbf{D}))$ 的动作理论,其中,对应于 ProcessModel 中的原子过程,可以对输入、输出、局部变量、前提条件、结果等多个方面进行建模;对应于 ProcessModel 中的复合过程,可以相应地对数据流以及顺序、选择、乱序、条件、迭代、循环等控制结构进行刻画. 以建模后得到的动作理论为基础,应用动态描述逻辑的推理机制,可以分别对语义 Web 服务的可实现性、可执行性、投影、规划等问题进行推理. 这些推理功能为语义 Web 服务的自动发现和自动组合提供了有效的支持.

关键词 语义 Web 服务; OWL-S; 动态描述逻辑; Web 服务推理; 服务发现和组合

中图法分类号 TP18

Reasoning About Semantic Web Services with an Approach Based on Dynamic Description Logics

SHI Zhong-Zhi CHANG Liang

(Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

Abstract The semantic Web services vision is to enable effective automation of various Web service related activities, such as Web service discovery, composition and execution. An obvious concern about the semantic Web service is to combine in some way the static descriptions of the information provided by ontologies with the dynamic descriptions of Web services' capabilities. Based on the dynamic description logic $DDL(SHOIN(\mathbf{D}))$, this paper presents an approach to model and reason about semantic Web services. With this approach, the OWL-S ProcessModel of Web services will be firstly translated into an action theory which is based on the $DDL(SHOIN(\mathbf{D}))$. This action theory allows for modeling atomic processes with their inputs, outputs, local variables, preconditions and results. It also offers considerable expressive power for modeling not only data flows of composite processes but also control flows such as the Sequence, Choice, Any-Order, If-Then-Else, Iterate, Repeat-While and Repeat-Until. Based on the action theory, the realizability, executability, projection and planning problems on semantic Web services can be reasoned about. These mechanisms provide effective supports for the discovery and composition of semantic Web services.

Keywords semantic Web service; OWL-S; dynamic description logic; reasoning about Web services; discovery and composition of services

收稿日期: 2008-07-10. 本课题得到国家自然科学基金(90604017, 60775035)、国家“八六三”高技术研究发展计划项目基金(2007AA01Z132)和国家“九七三”重点基础研究发展规划项目基金(2007CB311004)资助. 史忠植, 男, 1941 年生, 研究员, 博士生导师, 主要研究领域为人工智能、机器学习、多主体系统、语义 Web 等. E-mail: shizz@ics.ict.ac.cn. 常 亮, 男, 1980 年生, 博士, 主要研究方向为描述逻辑、语义 Web、智能主体.

1 引言

语义 Web 和 Web 服务是万维网发展的两个重要趋势,这两种技术的结合产生了一个新兴的研究课题——语义 Web 服务. 语义 Web 服务的目标是在 Web 服务的描述中加入足够的语义信息,使 Web 服务成为计算机可以理解的实体,从而支持 Web 服务的自动发现、自动组合和自动执行等^[1].

在针对语义 Web 服务的研究工作中,最有代表性的是 OWL-S^[2]. OWL-S 是一个用 OWL 语言描述的 Web 服务本体,从 ServiceProfile、ProcessModel 和 ServiceGrounding 3 个方面对 Web 服务进行了刻画. 其中,ServiceProfile 类似于服务的黄页,描述了服务的功能及相关属性;ProcessModel 对服务的过程模型进行刻画,描述了服务是如何工作的;ServiceGrounding 将过程模型与通信协议及消息格式等联系起来,描述了如何访问一个服务. 从最初版本的 DAML-S 开始,OWL-S 联盟陆续发布了 OWL-S 1.0 和 OWL-S 1.1;目前的最新版本是 OWL-S 1.2^[3].

OWL-S 本身不支持对 Web 服务动态功能的推理. 一方面,OWL-S 所基于的本体语言 OWL 建立在描述逻辑的基础上^[4]. 描述逻辑可以有效地表示和推理关于静态领域的知识,但对于动作或服务等具有动态特征的知识则无法处理. 原因在于描述逻辑的语义模型中不存在“状态”或“可能世界”这样的成分. 另一方面,虽然 OWL-S 在 OWL 的基础上引入了服务的前提条件和效果等属性,但这些属性仅仅作为描述逻辑中普通的角色名进行处理;OWL-S 没有从“执行服务后产生的效果”等动态内涵的角度提供相应的语义支持.

针对描述逻辑只能处理静态领域知识的局限,文献[5]将描述逻辑 ALC、动态逻辑、以及动作理论有机地结合起来,提出了一种动态描述逻辑 DDL. 应用 DDL,一方面可以基于描述逻辑对静态领域知识进行刻画,另一方面可以将上述知识作为背景,在其基础上对关于动作的知识进行描述和推理. 文献[6]在 DDL 的基础上采用可能模型途径来定义原子动作的语义,针对描述逻辑 ALCQO、ALCQO 和 ALCQIO 分别构建出 D-ALCO、D-ALCQO 以及 D-ALCQIO 等具有不同描述能力的动态描述逻辑,相应地为这些逻辑提供了适用于开世界假设的 Tableau 判定算法. 文献[7]研究了动态描述逻辑中关于动作的推理问题,将动作的可实现性、可执行

性、投影、规划等推理问题转换为公式的可满足性问题.

动态描述逻辑具有以下特点. 首先,从对动作进行刻画的角度看,描述能力强于基于命题语言的逻辑或系统(例如命题动态逻辑^[8]、命题 STRIPS 系统^[9]等). 其次,从对动作进行推理的角度看,相关推理问题都是可判定的,具有有效的推理机制和判定算法作为支撑. 与之形成对比的是情境演算^[10]和流演算^[11]等动作系统;这些系统具有很强的描述能力,但由于一阶谓词逻辑的不可判定性,使得相关推理受到很大限制. 此外,动态描述逻辑中关于动作的各种推理问题都可以转换为公式的可满足性问题,从而基于判定算法在信息不完全的情况下进行推理. 最后,基于描述逻辑的静态领域知识与动态领域中关于动作的知识被有机地结合了起来,使得动态描述逻辑适用于语义 Web 环境.

本文利用动态描述逻辑的上述特点,提出对语义 Web 服务进行建模和推理的一种有效途径. 首先,针对与本体语言 OWL DL 对应的描述逻辑 SHOIN(**D**),构建出相应的动态描述逻辑 DDL(SHOIN(**D**)). 接下来,从 OWL-S 中的 ProcessModel 出发,将语义 Web 服务建模为基于 DDL(SHOIN(**D**))的动作理论. 在此基础上,应用动态描述逻辑中的推理机制,可以对语义 Web 服务的可实现性、可执行性、投影、规划等问题进行推理. 这些推理功能为语义 Web 服务的自动发现和自动组合提供了有效的支持.

2 动态描述逻辑 DDL(SHOIN(**D**))

DDL(SHOIN(**D**)) 可以看作描述逻辑 SHOIN(**D**)、动态逻辑以及基于可能模型途径的动作理论的有机结合. 在介绍 DDL(SHOIN(**D**))之前需要先引入其中的有型域 **D**,用于处理数值、字符串或者时间等具有特定类型的对象.

定义 1. 有型域 (concrete domain) **D** 是一个二元组 $\mathbf{D} = (\Delta_{\mathbf{D}}, \text{pred}(\mathbf{D}))$, 其中:

(1) $\Delta_{\mathbf{D}}$ 为有型论域,其中的每个元素称为一个数据;

(2) $\text{pred}(\mathbf{D})$ 是由一元谓词组成的集合;其中的任一谓词 P 都对应于 $\Delta_{\mathbf{D}}$ 的某个子集 $P^D \subseteq \Delta_{\mathbf{D}}$;也将 $\text{pred}(\mathbf{D})$ 中的每个谓词 P 称为一个数据类型,将 P^D 称为 P 的值.

此外,为了具有可计算性,有型域 $\mathbf{D} = (\Delta_{\mathbf{D}}, \text{pred}(\mathbf{D}))$ 必须满足如下约束:存在某个可靠并且完

备的算法,使得对于 $pred(\mathbf{D})$ 中的任意 n 个谓词 P_1, P_2, \dots, P_n 来说,总可以判定 $P_1^D \cap P_2^D \cap \dots \cap P_n^D$ 是否为空集。

$DDL(SHOIN(\mathbf{D}))$ 的基本符号包括:

(1) 描述逻辑 $SHOIN(\mathbf{D})$ 中的基本符号,又分为以下几个部分:①由概念名组成的集合 N_C ;②由抽象角色名组成的集合 N_R 以及 N_R 的某个子集 N_{R+} ; N_{R+} 中的每个抽象角色名都称为传递角色名;③由个体名组成的集合 N_I ;④有型域 $\mathbf{D} = (\Delta_D, pred(\mathbf{D}))$;⑤由有型角色名组成的集合 N_{Rc} ;⑥概念构造符 $\{\}$ (枚举)、 \neg (否定)、 \sqcup (合并)、 \forall (值限定) 和 \leq (最大数量限定);⑦角色构造符 $^-$ (逆反);

(2) 由原子动作名组成的集合 N_A ;

(3) 公式构造符 \neg (否定)、 \vee (公式析取)、 $=$ (相等) 和 $\langle \rangle$ (动作存在性断言);

(4) 动作构造符 \cup (选择)、 $;$ (顺序)、 $*$ (迭代) 和 $?$ (测试);

(5) 其它符号,包括定义号“ \equiv ”、圆括号“()”、以及逗号“ $,$ ”。

从这些符号出发可构造出角色、概念、公式以及动作。

定义 2. $DDL(SHOIN(\mathbf{D}))$ 中的角色 (role) 由如下产生式生成:

$$R ::= r \mid r^- \mid T,$$

其中, $r \in N_R, T \in N_{Rc}$. 将 N_{Rc} 中的有型角色名形成的角色称为有型角色 (concrete role), 将其它角色称为抽象角色 (abstract role)。

为了表述方便,引入关于角色的函数 $Inv(\cdot)$,使得对于任一 $r \in N_R$ 都有 $Inv(r) = r^-$ 和 $Inv(r^-) = r$ 。

令 R, R' 为任意两个角色,则称 $R \sqsubseteq R'$ 为角色包含公理 (role inclusion axiom). 对于由角色包含公理组成的任一有限集合 \mathcal{R} , 将其称为 RBox。

令 \mathcal{R} 为任一 RBox, 则用 $\sqsubseteq_{\mathcal{R}}$ 表示二元关系“ \sqsubseteq ”在集合 $\mathcal{R} \cup \{Inv(R) \sqsubseteq Inv(S) \mid R \sqsubseteq S \in \mathcal{R}\}$ 上的自反传递闭包。

称某个角色 R 是相对于 RBox \mathcal{R} 的传递角色 (transitive role), 当且仅当 $\{R, Inv(R)\} \cap N_{R+} \neq \emptyset$ 或者存在某个相对于 \mathcal{R} 的传递角色 R' 使得 $R \sqsubseteq_{\mathcal{R}}^* R'$ 和 $R' \sqsubseteq_{\mathcal{R}}^* R$. 称某个角色 S 是相对于 RBox \mathcal{R} 的简单角色 (simple role), 当且仅当不存在相对于 \mathcal{R} 的传递角色 R 使得 $R \sqsubseteq_{\mathcal{R}}^* S$ 。

定义 3. 相对于有型域 $\mathbf{D} = (\Delta_D, pred(\mathbf{D}))$ 和某个 RBox \mathcal{R} , $DDL(SHOIN(\mathbf{D}))$ 中的概念 (concept) 由如下产生式生成:

$$C, C' ::= C_i \mid \{u\} \mid \neg C \mid C \sqcup C' \mid \forall R.C \mid \leq nS \mid \forall T.P,$$

其中 $C_i \in N_C, u \in N_I, R$ 为抽象角色, S 为相对于 \mathcal{R} 的简单角色, n 为自然数, $T \in N_{Rc}, P \in pred(\mathbf{D}), e \in \Delta_D$ 。

此外,可以引入形如 $C \sqcap C', \exists R.C, \geq nS, \exists T.P, \top$ 以及 \perp 的概念,分别作为 $\neg(\neg C \sqcup \neg C'), \neg(\forall R.\neg C), \neg(\leq (n-1)S), \neg(\forall T.\neg P), C \sqcup \neg C$ 以及 $C \sqcap \neg C$ 的缩写。

令 C_i 为概念名, D 为概念, 则称 $C_i \equiv D$ 为概念定义式 (concept definition). 对于由概念定义式组成的有限集合 \mathcal{T} , 如果每个概念名最多在 \mathcal{T} 中某个概念定义式的左边出现一次, 则称 \mathcal{T} 为 TBox。

相对于某个 TBox \mathcal{T} , 如果某个概念名 C 出现在 \mathcal{T} 中概念定义式的左边, 则称 C 为被定义的概念名 (defined concept name), 否则称其为简单概念名 (primitive concept name)。

定义 4. $DDL(SHOIN(\mathbf{D}))$ 中的公式 (formula) 由如下产生式生成:

$$\begin{aligned} \varphi, \psi &::= C(u) \mid R(u, v) \mid u = v \mid T(u, e) \mid \\ &\neg \varphi \mid \varphi \vee \psi \mid \langle \pi \rangle \varphi \end{aligned}$$

其中, $u, v \in N_I, e \in \Delta_D, C$ 为概念, R 为抽象角色, T 为有型角色, π 为动作。

此外,可以引入形如 $[\pi]\varphi, \varphi \wedge \psi, \varphi \rightarrow \psi, \text{true}$ 以及 false 的公式, 分别作为 $\neg \langle \pi \rangle \neg \varphi, \neg(\neg \varphi \vee \neg \psi), \neg \varphi \vee \psi, \varphi \vee \neg \varphi$ 以及 $\varphi \wedge \neg \varphi$ 的缩写。

令 \mathcal{T} 为 TBox, C_i 为相对于 \mathcal{T} 的简单概念名, $u, v \in N_I, e \in \Delta_D, R$ 为抽象角色, T 为有型角色, 则将形如 $R(u, v), \neg R(u, v), T(u, e), \neg T(u, e), C_i(u)$ 以及 $\neg C_i(u)$ 的公式都称为简单公式 (primitive formula)。

定义 5. $DDL(SHOIN(\mathbf{D}))$ 中的动作 (action) 由如下产生式生成:

$$\pi, \pi' ::= \alpha \mid \varphi? \mid \pi \cup \pi' \mid \pi; \pi' \mid \pi^*,$$

其中, $\alpha \in N_A, \varphi$ 为公式。

相对于某个 TBox \mathcal{T} , 将 $\alpha \equiv (P, E)$ 称为一个原子动作定义式 (atomic action definition), 其中:

(1) $\alpha \in N_A$, 表示所定义的原子动作;

(2) P 是由公式组成的有限集合, 表示动作执行前必须满足的前提条件;

(3) E 是由相对于 \mathcal{T} 的简单公式组成的有限集合, 表示执行该动作后将会产生的影响。

对于由原子动作定义式组成的有限集合 \mathcal{A} , 如果每个原子动作名最多在 \mathcal{A} 中某个原子动作定义式的左边出现一次, 则称 \mathcal{A} 为 ActBox。

相对于 RBox \mathcal{R} 、TBox \mathcal{T} 和 ActBox \mathcal{A} , 如果原子动作 α 出现在 \mathcal{A} 中某个原子动作定义式的

左边,则称 α 为被定义的原子动作 (defined atomic action).

动态描述逻辑 $DDL(SHOIN(\mathbf{D}))$ 在描述逻辑 $SHOIN(\mathbf{D})$ 的基础上引入了动态维,使得语义模型从整体上体现为由多个可能世界构成的可能世界空间.每个动作对应于可能世界之间的可达关系;在每个可能世界下分别对概念名、角色名以及个体名进行解释.

定义 6. $DDL(SHOIN(\mathbf{D}))$ 模型是一个三元组 $M=(\Sigma, \Delta_M, I)$. 其中:

(1) Σ 是形如 $\Sigma=(W, T_{a_0}, T_{a_1}, \dots)$ 的框架;在该框架中, W 是由可能世界组成的非空集合, N_A 中的每个原子动作名 α_i 被映射为 W 上的某个二元关系 $T_{\alpha_i} \subseteq W \times W$.

(2) Δ_M 是由个体组成的非空集合,作为该模型的论域.

(3) 函数 I 对 W 中的每个可能世界 w 赋予一个解释 $I(w)=(\Delta_M, \cdot^{I(w)})$, 其中的解释函数 $\cdot^{I(w)}$ 满足以下条件:

(i) 将 N_C 中的每个概念名 C_i 解释为 Δ_M 的某个子集 $C_i^{I(w)} \subseteq \Delta_M$;

(ii) 将 N_R 中的每个抽象角色名 r 解释为 Δ_M 上的某个二元关系 $r^{I(w)} \subseteq \Delta_M \times \Delta_M$; 并且, 如果 $r \in N_{R+}$, 则二元关系 $r^{I(w)}$ 具有传递性 (即对于 Δ_M 中的任意元素 x, y, z : 如果 $(x, y) \in r^{I(w)}$ 并且 $(y, z) \in r^{I(w)}$, 则有 $(x, z) \in r^{I(w)}$);

(iii) 将 N_{Rc} 中的每个有型角色名 T 解释为从 Δ_M 到 Δ_D 的某个二元关系 $T^{I(w)} \subseteq \Delta_M \times \Delta_D$;

(iv) 将 N_I 中的每个个体名 p 解释为 Δ_M 中的某个元素 $p^{I(w)} \in \Delta_M$, 并且满足: 对于 W 中任一可能世界 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, 因而也将 $p_i^{I(w)}$ 简记为 p_i^I .

定义 7. 令 $\mathbf{D}=(\Delta_D, pred(\mathbf{D}))$ 为有型域; $M=(\Sigma, \Delta_M, I)$ 为 $DDL(SHOIN(\mathbf{D}))$ 模型, 其中 $\Sigma=(W, T_{a_0}, T_{a_1}, \dots)$. 对 $DDL(SHOIN(\mathbf{D}))$ 中角色、概念、公式以及动作的语义归纳定义如下.

首先, 相对于 W 中的任一可能世界 w , 将任一抽象角色 R 解释为 Δ_M 上的二元关系 $R^{I(w)}$, 将任一有型角色 T 解释为从 Δ_M 到 Δ_D 的二元关系 $T^{I(w)}$, 将任一概念 C 解释为 Δ_M 的子集 $C^{I(w)}$; 归纳定义如下:

(1) $r^{I(w)} = r^{I(w)}$, $T^{I(w)} = T^{I(w)}$, $C_i^{I(w)} = C_i^{I(w)}$, 其中 $r \in N_R$, $T \in N_{Rc}$, $C_i \in N_C$;

(2) $(r^-)^{I(w)} = \{(y, x) \mid (x, y) \in r^{I(w)}\}$, 其中 $r \in N_R$;

(3) $\{u\}^{I(w)} = \{u^I\}$, 其中 $u \in N_I$;

(4) $(\neg C)^{I(w)} = \Delta_M \setminus C^{I(w)}$;

(5) $(C \sqcup D)^{I(w)} = C^{I(w)} \cup D^{I(w)}$;

(6) $(\leq nS)^{I(w)} = \{x \mid \#\{y \mid (x, y) \in S^{I(w)}\} \leq n\}$,

其中的 S 为简单角色;

(7) $(\forall R.C)^{I(w)} = \{x \mid \text{对于任一 } y \in \Delta_M: \text{如果 } (x, y) \in R^{I(w)}, \text{ 则有 } y \in C^{I(w)}\}$, 其中的 R 为抽象角色, C 为概念;

(8) $(\forall T.P)^{I(w)} = \{x \mid \text{对于任一 } y \in \Delta_D: \text{如果 } (x, y) \in T^{I(w)}, \text{ 则有 } y \in P^D\}$, 其中的 T 为有型角色, P 为有型域 \mathbf{D} 中的数据类型;

其次, 相对于 W 中的任一可能世界 w , 用 $(M, w) \models \varphi$ 表示公式 φ 在模型 M 中的可能世界 w 下成立; 根据 φ 的结构归纳定义如下:

(9) $(M, w) \models C(u)$ iff $u^I \in C^{I(w)}$;

(10) $(M, w) \models R(u, v)$ iff $(u^I, v^I) \in R^{I(w)}$, 其中的 R 为抽象角色;

(11) $(M, w) \models T(u, e)$ iff $(u^I, e) \in T^{I(w)}$, 其中的 T 为有型角色, $e \in \Delta_D$;

(12) $(M, w) \models u = v$ iff $u^I = v^I$;

(13) $(M, w) \models \neg \varphi$ iff $(M, w) \not\models \varphi$;

(14) $(M, w) \models \varphi \vee \psi$ iff $(M, w) \models \varphi$ 或者 $(M, w) \models \psi$;

(15) $(M, w) \models \langle \pi \rangle \varphi$ iff 存在某个可能世界 $w' \in W$ 使得 $(w, w') \in T_\pi$ 并且 $(M, w') \models \varphi$;

最后, 对于任一动作 π , 将其解释为 W 上的某个二元关系 T_π ; 根据 π 的结构归纳定义如下:

(16) $T_{\varphi?} = \{(w, w) \mid (M, w) \models \varphi\}$;

(17) $T_{\pi \cup \pi'} = T_\pi \cup T_{\pi'}$;

(18) $T_{\pi; \pi'} = \{(w, w') \mid \text{存在某个 } w'' \in W \text{ 使得 } (w, w'') \in T_\pi \text{ 和 } (w'', w') \in T_{\pi'}\}$;

(19) $T_{\pi^*} = T_\pi$ 的自反传递闭包.

在上述定义的基础上, 对 RBox、TBox 以及 ActBox 的语义定义如下.

定义 8. M 是某个 RBox \mathcal{R} 的模型, 记为 $M \models \mathcal{R}$, 当且仅当对于 \mathcal{R} 中的任一角色包含公理 $R \sqsubseteq R'$ 以及 W 中的任一可能世界 w 都有 $R^{I(w)} \subseteq R'^{I(w)}$.

定义 9. M 是某个 TBox \mathcal{T} 的模型, 记为 $M \models \mathcal{T}$, 当且仅当对于 \mathcal{T} 中的任一概念定义式 $C \equiv D$ 以及 W 中的任一可能世界 w 都有 $C^{I(w)} = D^{I(w)}$.

定义 10. 令 $M=(\Sigma, \Delta_M, I)$ 为 $DDL(SHOIN(\mathbf{D}))$ 模型, 其中 $\Sigma=(W, T_{a_0}, T_{a_1}, \dots)$; 则:

(1) M 是某个原子动作定义式 $\alpha \equiv (P, E)$ 的模型, 记为 $M \models \alpha \equiv (P, E)$, 当且仅当对于任一 $(w, w') \in T_a$ 都有: ① 对于任一 $\psi \in P$ 都有 $(M, w) \models \psi$; ② 对于任一 $\varphi \in E$ 都有 $(M, w) \models \neg \varphi$; ③ 对于任一

$\varphi \in E$ 都有 $(M, w') \models \varphi$; ④ 对于任一简单公式 $\varphi \notin E$: 如果 $(M, w) \models \varphi$, 则有 $(M, w') \models \varphi$; 如果 $(M, w) \not\models \varphi$, 则有 $(M, w') \not\models \varphi$.

(2) M 是某个 $\text{ActBox } \mathcal{A}_c$ 的模型, 记为 $M \models \mathcal{A}_c$, 当且仅当对于 \mathcal{A}_c 中的任一原子动作定义式 $\alpha \equiv (P, E)$ 都有 $M \models \alpha \equiv (P, E)$.

最后引入公式的可满足性.

定义 11. 令 $\mathbf{D} = (\Delta_D, \text{pred}(\mathbf{D}))$ 为有型域, \mathcal{R} 、 \mathcal{T} 和 \mathcal{A}_c 分别为 RBox 、 TBox 和 ActBox , φ 为公式, 并且 φ 中出现的各个原子动作都是相对于 \mathcal{R} 、 \mathcal{T} 和 \mathcal{A}_c 被定义的, 则称公式 φ 相对于 \mathcal{R} 、 \mathcal{T} 和 \mathcal{A}_c 是可满足的, 当且仅当存在某个模型 $M = (\Sigma, \Delta_M, I)$ 以及该模型中的某个可能世界 w 使得 $M \models \mathcal{T}$, $M \models \mathcal{R}$, $M \models \mathcal{A}_c$ 和 $(M, w) \models \varphi$.

公式的可满足性问题是 $\text{DDL}(\text{SHOIN}(\mathbf{D}))$ 中最基本的推理问题. 文献[12]证明了该推理问题是可判定的, 并且给出了适用于开世界假设的判定算法.

3 对语义 Web 服务的建模

本节首先以最新版本的 OWL-S 1.2 为参照^[2-3]考察 OWL-S 中刻画的语义 Web 服务. 接下来在 $\text{DDL}(\text{SHOIN}(\mathbf{D}))$ 的基础上引入变元, 构建出基于 $\text{DDL}(\text{SHOIN}(\mathbf{D}))$ 的动作理论. 最后从 OWL-S 中的 ProcessModel 出发, 应用上述动作理论对语义 Web 服务进行建模.

3.1 基于 OWL-S 的语义 Web 服务

OWL-S 从 ServiceProfile 、 ProcessModel 和 ServiceGrounding 3 个方面刻画 Web 服务. 其中, ServiceProfile 从服务的功能属性、服务层次以及服务的非功能属性(例如服务质量)3 个方面进行刻画, 描述了该 Web 服务能“作什么”, 为 Web 服务的查找和发现提供初步信息. ProcessModel 从过程模型的角度进行刻画, 描述了该 Web 服务“如何工作”, 为 Web 服务的建模、推理和组合提供了必要信息. ServiceGrounding 将过程模型与通信协议和消息格式等联系起来, 描述了“如何访问”该 Web 服务. 对语义 Web 服务进行建模和推理时, 我们只考察 OWL-S 中的 ProcessModel .

ProcessModel 包括 3 种类型的过程: 原子过程、复合过程和简单过程. 图 1 和图 2 分别给出了原子过程和复合过程的直观例子; 这两个例子来源于 OWL-S 联盟基于 OWL-S 1.2 刻画的图书购买服务^[3], 分别对应于本体 CongoProcess.owl 中的原子过程 ExpressCongoBuy 和复合过程 FullCongoBuy .

为了表述简洁, 我们借鉴文献[2]的方式, 在这两个例子中省略了与 RDF 和 XML 等语法相关的描述成分.

```
define atomic process ExpressCongoBuy
  (inputs: (ECB_BookISBN - ISBN
            ECB_SignInInfo - SignInData
            ECB_CreditCardNumber - xsd:decimal),
   locals: (ECB_AcctID - AcctID
            ECB_CreditCard - CreditCard),
   preconditions: (hasAcctID(ECB_SignInInfo, ECB_AcctID)
                  &. creditNumber(ECS_CreditCard,
                                   ECB_CreditCardNumber)
                  &. validity(ECB_CreditCard, Valid)),
   outputs: (ECB_Output - ECB_OutputType),
   results: ((ECB_Book - Book
              &. hasISBN(ECB_Book, ECB_BookISBN)
              &. InStockBook(ECB_Book)
              |-> type(ECB_Output,
                      OrderShippedAcknowledgment)
              &. shipment(ECB_Shipment)
              &. shippedTo(ECB_Shipment, ECB_AcctID)
              &. shippedBook(ECB_Shipment, ECB_Book))
            (→ ∃ hasBook. InStockBook(ECB_BookISBN)
             |-> type(ECB_Output, NotifyBookOutOfStock))
          )
  )
```

图 1 原子过程 ExpressCongoBuy ^①

原子过程由输入(inputs)、输出(outputs)、局部变量(local variables)、前提条件(preconditions)以及结果(results)5 个部分组成. 输入和输出体现了服务调用者(例如智能主体)与服务之间的信息交互; 每个输入或输出都由一个变量名及其类型声明组成. 例如, 图 1 中的 ECB_BookISBN 为变量名, 其类型为 ISBN. 前提条件刻画了调用服务之前必须满足的条件, 用逻辑公式表示. 例如, 图 1 中要求与信用卡号 $\text{ECB_CreditCardNumber}$ 相对应的信用卡 ECB_CreditCard 是有效的. 结果部分描述了服务被执行后将会产生的影响. 每个结果由一个条件式和一组效果(effect)及输出约束(output bindings)组成, 其中的条件式、效果以及输出约束都用逻辑公式表示; Web 服务在执行过程中将根据各个条件式产生相应的影响和输出. 例如, 如果与 ISBN 号 ECB_BookISBN 相对应的某本图书 ECB_Book 在库存中, 则图 1 中的 Web 服务在执行后将输出 $\text{OrderShippedAcknowledgment}$, 同时图书 ECB_Book 被发货到 ECB_AcctID ; 如果库存中不存在与 ECB_

① 与本体 CongoProcess.owl 中的描述相比, 这里对原子过程 ExpressCongoBuy 中的各个变量名进行了简写, 用字符串“ECB_”替换了字符串“ExpressCongoBuy”. 类似地, 图 2 对复合过程 FullCongoBuy 和 CongoBuy 中的变量名也进行了简写.

BookISBN 相对应的图书,则仅仅输出 NotifyBook-OutOfStock. 局部变量是除了输入和输出之外额外引入的一些变量,用于前提条件或者结果部分的相关刻画;每个局部变量由一个变量名及其类型声明

组成. 例如,图 1 中引入了 ECB_AcctID、ECB_CreditCard 和 ECB_Book 3 个局部变量名;这些变量名的类型分别为 AcctID、CreditCard 和 Book.

```
define composite process FullCongoBuy
  (inputs: (FCB_BookName - xsd: string
           FCB_CreateAcctInfo - AcctInfo
           ...
           ),
   outputs: (FCB_CreateAcctOutput - AcctID
            FCB_Output - FullCongoBuyOutputType),
   results: ((title(aBook, FCB_BookName) &. InStockBook(aBook)
             |-> FCB_CreateAcctOutput<= CongoBuyBook. CB_BookCreateAcctOutput
             &. type(FCB_Output, OrderShippedAcknowledgment))
            (title(aBook, FCB_BookName) &. ->InStockBook(aBook)
             |-> FCB_Output<= LocateBook. LocateBookOutput))
  )
  {LocateBook(LocateBookBookName<=FCB_BookName);
   if title(aBook, FCB_BookName) &. InStockBook(aBook)
   then CongoBuyBook(CB_BookISBN<= LocateBook. LocateBookOutput,
                     CB_BookCreateAcctInfo<= FCB_CreateAcctInfo,
                     ...
                     )
  }
```

图 2 复合过程 FullCongoBuy

复合过程在原子过程和其它复合过程的基础上通过控制流和数据流组织而得. OWL-S 引入的控制结构包括 Sequence、If-Then-Else、Repeat-While、Repeat-Until、Split、Split+Join、Choice、Any-Order 和 Iterate. 其中, Sequence、If-Then-Else、Repeat-While 和 Repeat-Until 对应于程序设计语言中的顺序、选择、循环等控制结构; Split 和 Split+Join 都表示并行结构,区别在于后者增加了路障同步,要求各个子过程在整个并行过程之前执行完毕; Choice 结构从多个子过程中任意选择一个执行; Any-Order 结构允许其中的各个子过程以任一次序执行,但在执行过程中不允许任意两个子过程之间存在并行; Iterate 结构允许其子过程迭代执行任意多次. 例如,在图 2 所示的复合过程 FullCongoBuy 中,首先将调用复合过程 LocateBook; 接下来,如果标题为 FCB_BookName 的某本图书 aBook 在库存中,则调用另外一个复合过程 CongoBuyBook.

除了应用上述控制结构之外,复合过程中还通过数据流将各个子过程以及复合过程之间的输出和输入绑定起来. 例如,复合过程 FullCongoBuy 在调用过程 LocateBook 时将其输入 FCB_BookName 与 LocateBook 的输入 LocateBookBookName 绑定起来;此外,在调用 CongoBuyBook 时将 LocateBook 的输出 LocateBookOutput 与 CongoBuyBook 的输入 CB_BookISBN 绑定起来,以及将自身的输入 FCB_CreateAcctInfo 与 CongoBuyBook 的输入

CB_BookCreateAcctInfo 绑定起来等.

通过控制结构和绑定关系将原子过程以及其它复合过程组织起来之后,形成了复合过程的过程体部分. 此外,复合过程中也需要声明自己的输入输出,并且可以进一步对前提条件和结果进行刻画. 其中,结果部分主要是将各个子过程的输出与复合过程自己的输出绑定起来;例如在图 2 中,如果标题为 FCB_BookName 的某本图书 aBook 在库存中,则过程 CongoBuyBook 的输出 CB_BookCreateAcctOutput 将绑定到复合过程 FullCongoBuy 的输出 FCB_CreateAcctOutput, 否则过程 LocateBook 的输出 LocateBookOutput 将绑定到 FullCongoBuy 的输出 FCB_Output.

简单过程是原子过程和复合过程的抽象形式,提供了原子过程或复合过程的一种抽象视图. 因此,本文对语义 Web 服务建模时不考虑简单过程.

对于任一语义 Web 服务,我们将其每一次的调度执行称为一个语义 Web 服务实例.

3.2 基于动态描述逻辑的动作理论

针对 ProcessModel 的上述特点,本节在 $DDL(SHOIN(\mathbf{D}))$ 的基础上引入变元,构建出基于 $DDL(SHOIN(\mathbf{D}))$ 的动作理论.

令 N_{IV} 是由个体变元组成的集合, N_V 是由数据变量组成的集合. 对于定义 4 来说,如果将其中的 $u, v \in N_I$ 更改为 $u, v \in N_I \cup N_{IV}$, 将 $e \in \Delta_D$ 更改为 $e \in \Delta_D \cup N_V$, 则称所得到的式子为伪公式 (pseudo-

formula);相应地,对应于 $DDL(SHOIN(\mathbf{D}))$ 中对简单公式的定义,如果将其中的约束 $u, v \in N_I$ 更改为 $u, v \in N_I \cup N_{IV}$, 将 $e \in \Delta_D$ 更改为 $e \in \Delta_D \cup N_V$, 则称所得到的伪公式为简单伪公式(primitive pseudo-formula)。

接下来引入带参数的原子动作定义式和带参数的复杂动作定义式。

定义 12. 带参数的原子动作定义式形如 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$, 其中:

- (1) α 为字符串,表示该定义式的名字;
- (2) (v_1, v_2, \dots, v_n) 是由出现在 P 和 E 中的所有个体变元组成的有限序列,称为该定义式的界面;
- (3) P 是由伪公式组成的有限集合,用于刻画执行动作之前必须满足的前提条件;
- (4) E 是由简单伪公式组成的有限集合,用于刻画执行动作后将会产生的影响。

每个带参数的原子动作定义式 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 都描述了关于一类原子动作的知识。对应于界面 (v_1, v_2, \dots, v_n) , 可以相应地代入任意 n 个个体名 p_1, p_2, \dots, p_n , 得到动态描述逻辑中的一个原子动作定义式 $\alpha(p_1, p_2, \dots, p_n) \equiv (P[p_1/v_1, p_2/v_2, \dots, p_n/v_n], E[p_1/v_1, p_2/v_2, \dots, p_n/v_n])$, 其中的 $\alpha(p_1, p_2, \dots, p_n)$ 可以看作动态描述逻辑中的原子动作名, $P[p_1/v_1, p_2/v_2, \dots, p_n/v_n]$ 表示将 v_1, v_2, \dots, v_n 在 P 中的出现分别用 p_1, p_2, \dots, p_n 进行替换后得到的公式集, $E[p_1/v_1, p_2/v_2, \dots, p_n/v_n]$ 表示将 v_1, v_2, \dots, v_n 在 E 中的出现分别用 p_1, p_2, \dots, p_n 进行替换后得到的由简单公式组成的集合。

对于由带参数的原子动作定义式组成的有限集合 \mathcal{AAct} , 如果 \mathcal{AAct} 中任意两个定义式的名字都不相同, 则称 \mathcal{AAct} 为 $\mathbf{AActBox}$ 。

令 \mathcal{AAct} 为 $\mathbf{AActBox}$, $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 为 \mathcal{AAct} 中的某个带参数的原子动作定义式, u_1, u_2, \dots, u_n 为 n 个个体名或个体变元, 则将 $\alpha(u_1, u_2, \dots, u_n)$ 称为以 \mathcal{AAct} 为参照的被定义的伪原子动作(defined pseudo atomic action);特别地, 如果 u_1, u_2, \dots, u_n 都为个体名, 则将 $\alpha(u_1, u_2, \dots, u_n)$ 称为以 \mathcal{AAct} 为参照的被定义的原子动作。

定义 13. 相对于某个 $\mathbf{AActBox} \mathcal{AAct}$, 带参数的复杂动作定义式形如 $A(v_1, v_2, \dots, v_n) \equiv \pi$, 其中:

- (1) A 为字符串,表示该定义式的名字;
- (2) (v_1, v_2, \dots, v_n) 是由出现在 π 中的所有个体变元组成的有限序列,称为该定义式的界面;
- (3) π 由如下产生式生成:

$$\pi, \pi' ::= \alpha(u_1, u_2, \dots, u_n) \mid \varphi? \mid \pi \cup \pi' \mid \pi; \pi' \mid \pi^*,$$

其中, $\alpha(u_1, u_2, \dots, u_n)$ 是以 \mathcal{AAct} 为参照的被定义的伪原子动作, φ 为伪公式。

对于由带参数的复杂动作定义式组成的有限集合 \mathcal{CAct} , 如果 \mathcal{CAct} 中任意两个定义式的名字都不相同, 则称 \mathcal{CAct} 为复杂动作 $\mathbf{CActBox}$ 。

给定某个 $\mathbf{CActBox} \mathcal{CAct}$, 其中的每个定义式 $A(v_1, v_2, \dots, v_n) \equiv \pi$ 都描述了关于一类复杂动作的知识。对应于界面 (v_1, v_2, \dots, v_n) , 可以相应地代入任意 n 个个体名或个体变元 u_1, u_2, \dots, u_n , 得到定义式 $A(u_1, u_2, \dots, u_n) \equiv \pi[u_1/v_1, u_2/v_2, \dots, u_n/v_n]$, 其中的 $\pi[u_1/v_1, u_2/v_2, \dots, u_n/v_n]$ 在 π 的基础上将 v_1, v_2, \dots, v_n 的出现分别用 u_1, u_2, \dots, u_n 进行了替换;特别地, 如果 u_1, u_2, \dots, u_n 都为个体名, 则 $\pi[u_1/v_1, u_2/v_2, \dots, u_n/v_n]$ 可以看作动态描述逻辑中的一个动作。由于 \mathcal{CAct} 中各个定义式的名字都不相同, 可以用 $A(u_1, u_2, \dots, u_n)$ 指代 $\pi[u_1/v_1, u_2/v_2, \dots, u_n/v_n]$ 。

最后引入基于动态描述逻辑的动作理论。

定义 14. 基于动态描述逻辑的动作理论 \mathcal{D} 为一个五元组 $\mathcal{D} = (\mathcal{R}, \mathcal{T}, \mathcal{AAct}, \mathcal{CAct}, \mathcal{A})$, 其中:

- (1) \mathcal{R} 为动态描述逻辑中的 \mathbf{RBox} , 是由角色包含公理组成的有限集合;
- (2) \mathcal{T} 为动态描述逻辑中的 \mathbf{TBox} , 是由概念定义式组成的有限集合;
- (3) \mathcal{AAct} 为 $\mathbf{AActBox}$, 是由带参数的原子动作定义式组成的有限集合;
- (4) \mathcal{CAct} 为 $\mathbf{CActBox}$, 是由带参数的复杂动作定义式组成的有限集合;并且, \mathcal{CAct} 和 \mathcal{AAct} 中不存在名字相同的定义式;
- (5) \mathcal{A} 是由动态描述逻辑中的公式组成的有限集合。

动作理论 $\mathcal{D} = (\mathcal{R}, \mathcal{T}, \mathcal{AAct}, \mathcal{CAct}, \mathcal{A})$ 刻画了 3 个方面的知识: (1) 由 \mathcal{R} 和 \mathcal{T} 表示的领域知识。通过角色构造符、概念构造符、概念定义符、角色包含符等符号, \mathcal{R} 和 \mathcal{T} 对静态领域的知识进行了结构化的描述;这些知识也为 \mathcal{AAct} 和 \mathcal{CAct} 中对动作的描述以及 \mathcal{A} 中对应用领域具体状态的描述提供了词汇。(2) 由 \mathcal{AAct} 和 \mathcal{CAct} 表示的关于动作的知识。其中, \mathcal{AAct} 中的定义式从执行动作之前必须满足的条件以及执行动作之后产生的影响两个方面刻画了关于原子动作的知识;在 \mathcal{AAct} 的基础上, \mathcal{CAct} 中通过动作构造符“?”、“;”、“ \cup ”以及“ $*$ ”刻画了关于复杂动作的知识。(3) 由 \mathcal{A} 刻画的关于应用领域某个具体状态的知识, 类似于情景演算中对初始情景的刻画^[10]。

3.3 应用动作理论对语义 Web 服务进行建模

下面从 ProcessModel 中的原子过程和复合过程出发,应用上述动作理论进行建模.

根据原子过程的组成,可以分别从前提条件、输入、输出、局部变量以及结果等方面进行相应的处理.建模的过程由以下 3 个步骤组成.

(1) 对于原子过程中的前提条件,可以在带参数的原子动作定义式 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 的 P 中进行刻画.

OWL-S 允许采用 SWRL、SWRL-FOL、KIF 等多种逻辑语言对前提条件、效果、输出约束等进行刻画;不同的语言具有不同的表达能力.本文仅考虑采用 SWRL 语言的情况. SWRL 语言将本体语言 OWL DL 与一元/二元 Datalog RuleML 子语言有机地结合了起来^[13]. 但 OWL-S 并没有完全使用 SWRL 的语言成分;OWL-S 中使用的 SWRL-Condition 仅仅是由 SWRL 中的 atom 组成的合取形式. SWRL 中的 atom 由如下产生式生成:

$$\begin{aligned} atom ::= & C(x) \mid D(z) \mid P(x, y) \mid Q(x, z) \mid \\ & sameAs(x, y) \mid differentFrom(x, y) \mid \\ & builtIn(r, z_1, z_2, \dots, z_n), \end{aligned}$$

其中, C 为 OWL DL 中的概念, D 为有型域中的数据类型, x, y 为个体名或个体变元, z 为有型域中的数据或变量, P 为关于个体名或个体变元的二元关系, Q 是个体名或个体变元与数据或数据变量之间的二元关系, $builtIn(r, z_1, z_2, \dots, z_n)$ 表示有型域中的数据或变量 z_1, z_2, \dots, z_n 之间满足关系 r .

对应于 atom 的上述产生式,可以将其中的 $C(x), P(x, y), Q(x, z), sameAs(x, y)$ 以及 $differentFrom(x, y)$ 分别转换为 $DDL(SHOIN(\mathbf{D}))$ 中形如 $C(x), P(x, y), Q(x, z), x=y$ 以及 $\neg(x=y)$ 的伪公式;同时,将形如 $D(z)$ 和 $builtIn(r, z_1, z_2, \dots, z_n)$ 的运算及判断交给与有型域 \mathbf{D} 相关的计算机机制去处理.在此基础上,对于 OWL-S 中由 atom 的合取形式构成的 SWRL-Condition,可以通过构造符“ \wedge ”构造出 $DDL(SHOIN(\mathbf{D}))$ 中相应的伪公式.因此,对应于原子过程中的各个前提条件,可以在集合 P 中加入 $DDL(SHOIN(\mathbf{D}))$ 中相应的伪公式.

(2) 对于原子过程中的输入、输出以及局部变量,可以在带参数的原子动作定义式 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 的界面 (v_1, v_2, \dots, v_n) 以及 P 和 E 中进行刻画.刻画过程可以分 4 个步骤进行.

(i) 对应于由变量名及其类型声明组成的各个输入、输出以及局部变量,将它们的变量名依次构成形如 $(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t)$

的序列,然后将该序列作为上述定义式的界面 (v_1, v_2, \dots, v_n) .

(ii) 对于原子过程中的各个局部变量,假设其变量名和类型分别为 x 和 C ,则相应地在集合 P 中加入公式 $C(x)$.通过这些公式,各个局部变量的类型在定义式中得到了体现.

(iii) 对于原子过程中的各个输入,假设其变量名和类型分别为 x 和 C ,则相应地在集合 P 中加入公式 $C(x)$ 和公式 $agentKnows(x)$,其中的 $agentKnows$ 是特别引入的概念名.加入公式 $agentKnows(x)$ 的原因在于:智能主体在调用 Web 服务时需要为原子过程中的各个变量代入具体的值;从智能主体的角度看,每个输入都对应于智能主体在调用 Web 服务之前必须具备的知识.

(iv) 对于原子过程中的每个输出,假设其变量名和类型分别为 x 和 C ,则相应地在集合 E 中加入公式 $C(x)$ 和 $agentKnows(x)$.与上面对输入的处理相对应,加入 $agentKnows(x)$ 的原因在于:对于原子过程中的各个输出来看,其对应于智能主体在调用 Web 服务之后将要获得的知识.

(3) 对原子过程中的结果进行处理.每个结果由一个条件式和一组效果及输出约束组成;对于每个结果来说,其中的条件式并不影响 Web 服务能否被执行,而是影响执行 Web 服务后将会产生的效果和输出.令原子过程中含有 n 个结果;令 $(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t)$ 是经过上面步骤(2)的处理后得到的界面, P 是经过上面步骤(1)和(2)的处理后得到的与前提条件、输入以及局部变量相对应的由伪公式组成的集合, E 是经过上面步骤(2)的处理后得到的与输出相对应的由简单伪公式组成的集合.下面分 3 个步骤进行处理.

(i) 对于原子过程中的每个结果,相应地构造一个带参数的原子动作定义式.具体来说,对于第 i 个结果(其中 $1 \leq i \leq n$),可以根据其中的条件式构造出 $DDL(SHOIN(\mathbf{D}))$ 中的某个伪公式 φ_i ,同时根据其中的效果和输出约束构造出一个或多个简单伪公式 $\psi_{i_1}, \psi_{i_2}, \dots, \psi_{i_k}$;在此基础上构造一个带参数的原子动作定义式 $A_i(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \equiv (P_i, E_i)$,其中,集合 $P_i = P \cup \{\varphi_i\}$,集合 $E_i = E \cup \{\psi_{i_1}, \psi_{i_2}, \dots, \psi_{i_k}\}$.

(ii) 构造一个带参数的原子动作定义式 $A_{n+1}(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \equiv (P_{n+1}, E_{n+1})$,其中,集合 $P_{n+1} = P \cup \{\neg(\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_n)\}$,集合 $E_{n+1} = E$.

(iii) 构造带参数的复杂动作定义式 $A(I_1,$

$I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \equiv A_1(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \cup \dots \cup A_n(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \cup A_{n+1}(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t)$, 其中的 A 与被建模的原子过程同名。

综上,对于 ProcessModel 中的每个原子过程, 可以将其建模为基于动态描述逻辑的某个带参数的复杂动作定义式以及若干个带参数的原子动作定义式。例如,对于原子过程 ExpressCongoBuy,建模后得到的结果如图 3 所示。

ExpressCongoBuy_{succ}(IF)≡(P∪P_{succ}, E∪E_{succ})
ExpressCongoBuy_{fail}(IF)≡(P∪P_{fail}, E∪E_{fail})
ExpressCongoBuy(IF)≡ExpressCongoBuy_{succ}(IF)∪ExpressCongoBuy_{fail}(IF)
其中:
(IF)=(ECB_BookISBN, ECB_SignInInfo, ECB_CreditCardNumber, ECB_Output, ECB_AcctID, ECB_CreditCard, ECB_Book)
P={ISBN(ECB_BookISBN), SignInData(ECB_SignInInfo), decimal(ECB_CreditCardNumber), agentKnows(ECB_BookISBN), agentKnows(ECB_SignInInfo), agentKnows(ECB_CreditCardNumber), AcctID(ECB_AcctID), CreditCard(ECB_CreditCard), hasAcctID(ECB_SignInInfo, ECB_AcctID), creditNumber(ECB_CreditCard, ECB_CreditCardNumber), validity(ECB_CreditCard, Valid)}
P_{succ}=P∪{Book(ECB_Book), hasISBN(ECB_Book, ECB_ISBN), InStockBook(ECB_Book)}
P_{fail}=P∪{¬∃ hasBook, InStockBook(ECB_BookISBN)}
E={ECB_OutType(ECB_Output), agentKnows(ECB_Output)}
E_{succ}=E∪{type(ECB_Output, OrderShippedAcknowledgment), Shipment(ECB_Shipment), shippedTo(ECB_Shipment, ECB_AcctID), shippedBook(ECB_Shipment, ECB_Book)}
F_{fail}=E∪{type(ECB_Output, NotifyBookOutOfStock)}

图 3 基于 DDL(SHOIN(D))对原子过程 ExpressCongoBuy 的建模

建立起原子过程的模型之后,可以进一步将每个复合过程建模为一个带参数的复杂动作定义式。下面从 3 个方面进行处理。

(1)对复合过程中控制结构的处理。由于动态描述逻辑 DDL(SHOIN(D))中尚未引入并行构造符,本文假设所考察的复合过程中不含有控制结构 Split 和 Split+Join。对于其它形式的控制结构,可以通过动态描述逻辑中的动作构造符进行建模,如表 1 所示。因此,对应于复合过程的过程体部分,可以初步构建出某个带参数的动作 π 。

| 表 1 OWL-S 中的控制结构在动作描述逻辑中的刻画 | | |
|-----------------------------|-------------------------------------|--|
| OWL-S 中的控制结构 | 例子 | 在动作描述逻辑中的刻画 |
| Sequence | Sequence(π, π') | $\pi; \pi'$ |
| Choice | Choice(π, π') | $\pi \cup \pi'$ |
| Any-Order | Any-Order(π, π') | $(\pi; \pi') \cup (\pi'; \pi)$ |
| If-Then-Else | if φ then π else π' | $(\varphi?; \pi) \cup ((\neg \varphi)?; \pi')$ |
| Iterate | Iterate(π) | π^* |
| Repeat-While | while φ do π | $(\varphi?; \pi)^*; (\neg \varphi)?$ |
| Repeat-Until | do π until φ | $\pi; ((\neg \varphi)?; \pi)^*; \varphi?$ |

(2)对复合过程中数据流的处理。扫描复合过程的过程体部分和结果部分,对于其中存在绑定关系的任意一对变量名 x 和 y ,如果 x 没有出现在当前进行建模的复合过程的输入或输出中,则将变量名 x 在 π 中的每一处出现都用 y 进行替换。令 π' 是经过上述替换后得到的带参数的动作,则 π' 中通过变量名相同的情况体现了各个过程的输入和输出之

间存在的绑定关系。

(3)对应于复合过程中的各个输入和输出,将它们的变量名依次构成形如 $(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n)$ 的序列;此外,令 L_1, L_2, \dots, L_t 是出现在 π' 中但没有出现在集合 $\{I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n\}$ 中的所有变量名,相应地构造序列 $(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t)$ 。最后,令 A 是所建模的复合过程的名字,返回带参数的复杂动作定义式 $A(I_1, I_2, \dots, I_m, O_1, O_2, \dots, O_n, L_1, L_2, \dots, L_t) \equiv \pi'$ 。

例如,对于复合过程 FullCongoBuy,建模后得到的结果如图 4 所示。

FullCongoBuy(FCB_BookName, FCB_CreateAcctInfo, ..., FCB_CreateAcctOutput, FCB_Output, aBook)
≡LocateBook(FCB_BookName, FCB_Output);
(title(aBook, FCB_BookName) ∧ InStockBook(aBook))?;
CongoBuyBook(FCB_Output, FCB_CreateAcctInfo, ..., FCB_CreateAcctOutput)

图 4 基于 DDL(SHOIN(D))对原子过程 FullCongoBuy 的建模

根据以上步骤可以依次建立起原子过程和复合过程的模型,得到由带参数的原子动作定义式组成的集合 $\mathcal{A}Act$ 以及由带参数的复杂动作定义式组成的集合 $\mathcal{CA}ct$ 。此外,对于原子过程或者复合过程中所参照的领域本体,当其中不含有一般概念包含公理时^[6],可以直接刻画为 DDL(SHOIN(D))中的

RBox、TBox 以及由公式组成的某个集合 \mathcal{A} .

4 对语义 Web 服务的推理

令 $\mathcal{D} = (\mathcal{R}, \mathcal{T}, \mathcal{A}Act, \mathcal{C}Act, \mathcal{A})$ 是经过上述建模后得到的动作理论; 本节在其基础上考察对语义 Web 服务的推理. 所考察的推理问题可以分为两大类: 针对语义 Web 服务的推理和针对语义 Web 服务实例的推理. 对应到上述动作理论中, 前者体现为对带参数的原子动作定义式以及带参数的复杂动作定义式进行推理, 后者体现为对动作的推理.

首先, 针对任一语义 Web 服务, 我们需要判断是否存在某些状态使得该语义 Web 服务能够被成功地调度执行; 如果永远不会被调度执行, 则这样的 Web 服务是没有意义的. 我们将该推理问题称为语义 Web 服务的可实现性问题. 当语义 Web 服务的 ProcessModel 为原子过程时, 对应到上述动作理论中, 该推理问题体现为带参数的原子动作定义式的一致性. 定义如下.

定义 15. 带参数的原子动作定义式 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 相对于 RBox \mathcal{R} 和 TBox \mathcal{T} 是一致的, 当且仅当存在 n 个未在 $\mathcal{R}, \mathcal{T}, P$ 和 E 中出现的个体名 p_1, p_2, \dots, p_n , 并且存在 $DDL(SHOIN(\mathcal{D}))$ 中的某个模型 $M = (\Sigma, \Delta_M, I)$ 以及该模型中的两个可能世界 w 和 w' , 使得 $M \models \mathcal{R}, M \models \mathcal{T}, M \models \alpha(p_1, p_2, \dots, p_n) \equiv (P[p_1/v_1, p_2/v_2, \dots, p_n/v_n], E[p_1/v_1, p_2/v_2, \dots, p_n/v_n])$ 以及 $(w, w') \in T_{\alpha(p_1, p_2, \dots, p_n)}$.

根据定义, 可以将该推理问题转换为公式的可满足性问题, 即有如下定理.

定理 1. 带参数的原子动作定义式 $\alpha(v_1, v_2, \dots, v_n) \equiv (P, E)$ 相对于 RBox \mathcal{R} 和 TBox \mathcal{T} 是一致的, 当且仅当公式 $\langle \alpha(p_1, p_2, \dots, p_n) \rangle \text{true}$ 相对于 \mathcal{R}, \mathcal{T} 以及 ActBox \mathcal{A} 是可满足的, 其中, p_1, p_2, \dots, p_n 是任意 n 个未在 $\mathcal{R}, \mathcal{T}, P, E$ 中出现的个体名, $\mathcal{A} = \{ \alpha(p_1, p_2, \dots, p_n) \equiv (P[p_1/v_1, p_2/v_2, \dots, p_n/v_n], E[p_1/v_1, p_2/v_2, \dots, p_n/v_n]) \}$.

当语义 Web 服务的 ProcessModel 为复合过程时, 语义 Web 服务的可实现性问题体现为带参数的复杂动作定义式的一致性问题. 此时, 由于复合过程中需要调用其它原子过程, 我们假设与这些原子过程相对应的带参数的原子动作定义式都是一致的. 在此基础上, 对带参数的复杂动作定义式的一致性问题定义如下.

定义 16. 带参数的复杂动作定义式 $A(v_1, v_2, \dots, v_n) \equiv \pi$ 相对于 RBox \mathcal{R} 和 TBox \mathcal{T} 是一致的,

当且仅当存在 n 个未在 \mathcal{R}, \mathcal{T} 和 π 中出现的个体名 p_1, p_2, \dots, p_n , 并且存在 $DDL(SHOIN(\mathcal{D}))$ 中的某个模型 $M = (\Sigma, \Delta_M, I)$ 以及该模型中的两个可能世界 w 和 w' , 使得 $M \models \mathcal{R}, M \models \mathcal{T}, M \models \mathcal{A}$ 以及 $(w, w') \in T_{\pi[p_1/v_1, p_2/v_2, \dots, p_n/v_n]}$, 其中的 \mathcal{A} 是由分别对应于 $\pi[p_1/v_1, p_2/v_2, \dots, p_n/v_n]$ 中各个原子动作的原子动作定义式组成的一个 ActBox.

根据定义, 也可以将该推理问题转换为公式的可满足性问题, 即有如下定理.

定理 2. 带参数的复杂动作定义式 $A(v_1, v_2, \dots, v_n) \equiv \pi$ 相对于 RBox \mathcal{R} 和 TBox \mathcal{T} 是一致的, 当且仅当公式 $\langle \pi[p_1/v_1, p_2/v_2, \dots, p_n/v_n] \rangle \text{true}$ 相对于 \mathcal{R}, \mathcal{T} 以及 ActBox \mathcal{A} 是可满足的, 其中, p_1, p_2, \dots, p_n 是任意 n 个未在 \mathcal{R}, \mathcal{T} 和 π 中出现的个体名, \mathcal{A} 是由分别对应于 $\pi[p_1/v_1, p_2/v_2, \dots, p_n/v_n]$ 中各个原子动作的原子动作定义式组成的 ActBox.

对于具有可实现性的语义 Web 服务, 服务请求者可以在提供相应的输入数据后对其调度执行. 此时, 针对任一语义 Web 服务实例, 我们主要关注两个方面的问题. 一方面, 我们希望知道在当前状态下该语义 Web 服务实例能否被成功地执行完成. 这类推理问题称为可执行性问题. 另一方面, 我们希望知道在当前状态下执行该语义 Web 服务实例后能否达到所希望的目标. 这类推理问题称为投影问题. 基于上述动作理论, 对可执行性问题和投影问题分别定义如下.

定义 17. 令 π 为动作, \mathcal{A} 是由分别对应于 π 中各个原子动作的原子动作定义式组成的 ActBox. 称动作 π 相对于 RBox \mathcal{R} 、TBox \mathcal{T} 以及 ActBox \mathcal{A} 来说在公式集 \mathcal{A} 所刻画的状态上是可执行的, 当且仅当对于任一模型 $M = (\Sigma, \Delta_M, I)$ (其中 $\Sigma = (w, T_{a_1}, T_{a_2}, \dots)$) 以及该模型中的任一可能世界 w : 如果 $M \models \mathcal{R}, M \models \mathcal{T}, M \models \mathcal{A}$ 并且对于任一 $\phi \in \mathcal{A}$ 都有 $(M, w) \models \phi$, 则必然存在某个模型 $M' = (\Sigma', \Delta, I')$ (其中 $\Sigma' = (w', T'_{a_1}, T'_{a_2}, \dots)$) 使得: ① $W \subseteq W'$; ② 对于任一原子动作名 α 都有 $T_\alpha \subseteq T'_\alpha$; ③ 对于任一可能世界 $w_i \in W$ 都有 $I'(w_i) = I(w_i)$; ④ $M' \models \mathcal{R}, M' \models \mathcal{T}, M' \models \mathcal{A}, (M', w) \models \mathcal{A}$; ⑤ 存在某个可能世界 $w' \in W'$ 使得 $(w, w') \in T'_\pi$.

定义 18. 令 π 为动作, \mathcal{A} 是由分别对应于 π 中各个原子动作的原子动作定义式组成的 ActBox. 称动作 π 在公式集 \mathcal{A} 刻画的状态上执行后相对于 RBox \mathcal{R} 、TBox \mathcal{T} 以及 ActBox \mathcal{A} 来说使得公式 φ 成立, 当且仅当对于任一模型 $M = (\Sigma, \Delta_M, I)$ 以及该模型中的任意两个可能世界 w, w' : 如果 $M \models \mathcal{R}$,

$M \models \mathcal{T}, M \models \mathcal{A}c$, 对于任一 $\phi \in \mathcal{A}$ 都有 $(M, w) \models \phi$ 并且 $(w, w') \in T_\pi$, 则必然有 $(M, w') \models \phi$.

根据上述定义, 类似于文献[7], 可以将可执行性问题和投影问题分别转换为公式的可满足性问题. 令 $\alpha_1, \alpha_2, \dots, \alpha_n$ 是出现在动作 π 中的所有原子动作, $\mathcal{A}c = \{\alpha_1 \equiv (P_1, E_1), \alpha_2 \equiv (P_2, E_2), \dots, \alpha_n \equiv (P_n, E_n)\}$ 是由分别对应于这些原子动作的原子动作定义式组成的 ActBox, 则有如下两个定理.

定理 3. 动作 π 相对于 RBox \mathcal{R} 、TBox \mathcal{T} 和 ActBox $\mathcal{A}c$ 来说在 \mathcal{A} 刻画的状态上是可执行的, 当且仅当如下公式相对于 \mathcal{R} 、 \mathcal{T} 和 $\mathcal{A}c$ 是不可满足的:

$$\neg((\llbracket ((P_1, E_1) \cup (P_2, E_2) \cup \dots \cup (P_n, E_n))^* \rrbracket \Pi) \rightarrow (\text{Conj}(\mathcal{A}) \rightarrow \langle \pi \rangle \text{true})),$$

其中, 对任一公式集 S 用 $\text{Conj}(S)$ 表示由 S 中的各个公式作为合取项构成的合取式, 用 Π 表示公式 $(\text{Conj}(P_1) \rightarrow \langle (P_1, E_1) \rangle \text{true}) \wedge (\text{Conj}(P_2) \rightarrow \langle (P_2, E_2) \rangle \text{true}) \wedge \dots \wedge (\text{Conj}(P_n) \rightarrow \langle (P_n, E_n) \rangle \text{true})$.

定理 4. 动作 π 在 \mathcal{A} 刻画的状态上执行后相对于 \mathcal{R} 、 \mathcal{T} 以及 $\mathcal{A}c$ 来说使得公式 φ 成立, 当且仅当公式 $\neg(\text{Conj}(\mathcal{A}) \rightarrow [\pi]\varphi)$ 相对于 \mathcal{R} 、 \mathcal{T} 和 $\mathcal{A}c$ 是不可满足的.

最后考察关于语义 Web 服务实例的规划问题. 给定用于刻画初始状态的公式集 \mathcal{A} 、目标公式 φ 以及由语义 Web 服务实例组成的有限集合 Σ , 规划问题是指从 Σ 中的元素出发构造某个序列, 使得在 \mathcal{A} 刻画的状态上依次执行该序列中的语义 Web 服务实例后能够实现目标 φ , 其中, 将满足上述条件的序列称为一个规划. 基于动作理论, 对规划定义如下.

定义 19. 令 $(\pi_1, \pi_2, \dots, \pi_n)$ 为动作序列, $\mathcal{A}c$ 是由分别对应于该序列中各个原子动作的原子动作定义式组成的 ActBox, φ 为公式. 称该动作序列是从公式集 \mathcal{A} 出发相对于 RBox \mathcal{R} 、TBox \mathcal{T} 以及 ActBox $\mathcal{A}c$ 达到目标 φ 的一个规划, 当且仅当动作“ $\pi_1; \pi_2; \dots; \pi_n$ ”相对于 \mathcal{R} 、 \mathcal{T} 和 $\mathcal{A}c$ 来说在 \mathcal{A} 刻画的状态上是可执行的并且在执行后使得公式 φ 成立.

根据上述定义, 可以应用定理 3 和定理 4 来判断某个序列是否为满足要求的规划.

此外, 还可以借助可满足性问题进行规划求解. 即有如下结论.

定理 5. 令 $\Sigma = \{\pi_1, \pi_2, \dots, \pi_m\}$ 是由动作组成的有限集合, φ 为公式; 令 $\alpha_1, \alpha_2, \dots, \alpha_n$ 是出现在 Σ 和 φ 中的所有原子动作, $\mathcal{A}c = \{\alpha_1 \equiv (P_1, E_1), \alpha_2 \equiv (P_2, E_2), \dots, \alpha_n \equiv (P_n, E_n)\}$ 是由分别对应于这些原子动作的原子动作定义式组成的 ActBox, 则应用 Σ 中的动作可以构造出一个从公式集 \mathcal{A} 出发相对于

RBox \mathcal{R} 、TBox \mathcal{T} 以及 ActBox $\mathcal{A}c$ 达到目标 φ 的规划, 当且仅当如下公式相对于 \mathcal{R} 、 \mathcal{T} 和 $\mathcal{A}c$ 是不可满足的:

$$\neg((\llbracket ((P_1, E_1) \cup (P_2, E_2) \cup \dots \cup (P_n, E_n))^* \rrbracket \Pi) \rightarrow (\text{Conj}(\mathcal{A}) \rightarrow \langle (\pi_1 \cup \pi_2 \cup \dots \cup \pi_m)^* \rangle \varphi)),$$

其中, 对于任一公式集 S , 用 $\text{Conj}(S)$ 表示由 S 中的各个公式作为合取项构成的合取式; 用 Π 表示公式 $(\text{Conj}(P_1) \rightarrow \langle (P_1, E_1) \rangle \text{true}) \wedge (\text{Conj}(P_2) \rightarrow \langle (P_2, E_2) \rangle \text{true}) \wedge \dots \wedge (\text{Conj}(P_n) \rightarrow \langle (P_n, E_n) \rangle \text{true})$.

如果上述定理中的公式被证明为不可满足的, 则基于证明过程可以构造出满足要求的规划.

上述各种推理机制为语义 Web 服务的自动发现和自动组合提供了有效的支持. (1) 通过对语义 Web 服务可实现性的推理, 可以在服务发现的过程中排除没有价值的语义 Web 服务. (2) 通过对可执行性和投影问题的推理, 可以进一步验证候选的语义 Web 服务是否能有效地满足使用者的需求. (3) 如果各个语义 Web 服务在单独使用的情况下都不能满足使用者的需求, 则可以根据使用者的需求首先确定出一定数量的语义 Web 服务实例, 然后通过规划求解得到关于 Web 服务的一个组合方案. (4) 假如已经通过某种途径得到关于 Web 服务的一个组合方案, 则可以验证该组合方案的正确性.

5 相关工作

Narayanan 等^[14]应用情景演算对 DAML-S 中原子过程的语义进行定义, 然后应用 Petri 网刻画 DAML-S 中的复合过程, 从而建立起语义 Web 服务的 Petri 网模型. 其中, 每个原子过程根据其结果部分含有的条件式的个数相应地建模为一个或多个转换关系; 原子过程的前提条件以及所产生的影响在这些转换关系中分别体现为前置集和输出集. 在此基础上, Narayanan 等对 Web 服务的组合问题以及 Web 服务组合方案的安全性等进行严格的定义, 然后将相关推理问题转换为 Petri 网的可达性问题. 这种方法的主要局限在于相关推理只是在命题层面上进行; 基于情景演算的关于原子过程的知识并没有在推理过程和组合过程中得到应用.

McIlraith 等^[15]更彻底地应用了情景演算的描述能力. 除了基于情景演算对 DAML-S 中的原子过程进行刻画之外, McIlraith 等进一步应用 Golog 语言对 DAML-S 中不含有控制结构 Split 和 Split + Join 的复合过程进行刻画; 在此基础上, 应用已有的支持 Golog 语言的规划算法和工具^[10], 可以有效地

解决 Web 服务的组合问题. 这种方法继承了情景演算的优点, 具有很强的描述能力和成熟的理论基础. 但是, 一阶谓词逻辑的不可判定性限制了这种方法对语义 Web 服务的推理能力; 此外, 这种方法没有将关于语义 Web 服务的知识与领域本体结合起来.

Wu 等^[16]应用规划工具 SHOP2 进行语义 Web 服务组合. 其中, 在假设原子过程中不会同时含有效果和输出的前提下, 将 DAML-S 中的原子过程建模为 SHOP2 中的操作; 在假设复合过程中不含有控制结构 Split 和 Split+Join 的前提下, 将复合过程建模为 SHOP2 中的方法; 然后在此基础上通过 SHOP2 实现语义 Web 服务组合. Klusch 等^[17]对 SHOP2 的规划过程进行改进, 针对语义 Web 服务组合问题研发了工具 OWLS-Xplan; 该工具首先将 OWL-S 中刻画的 Web 服务转换为基于 PDDL (Planning Domain Definition Language) 的描述形式, 然后通过规划工具 Xplan 实现 Web 服务组合. Wu 等以及 Klusch 等的方法都具有以下两个局限: 首先, 都采用了闭世界假设, 在规划过程中需要知道关于世界状态的完整知识; 其次, 都不能将语义 Web 服务作为一种知识进行推理, 从而不能为 Web 服务的发现和匹配提供支持.

Keller 等^[18]提出对 Web 服务进行功能描述并在此基础上进行推理. 他们首先将 Web 服务的功能刻画为三元组 $D = (\varphi^{\text{pre}}, \varphi^{\text{post}}, IF_D)$, 其中的 φ^{pre} 为执行 Web 服务之前需要满足的前提条件, φ^{post} 为执行 Web 服务后成立的后置条件, IF_D 是由出现在 φ^{pre} 和 φ^{post} 中的所有变元形成的界面. 在此基础上, 可以对功能描述的可实现性以及功能描述之间的包含关系等进行推理, 将它们转换为一阶谓词逻辑中的定理证明问题. 同样, 一阶谓词逻辑的不可判定性限制了这种方法对 Web 服务的推理能力; 此外, 这种方法也没有将 Web 服务的功能描述与领域本体结合起来.

Gu 等^[19]提出情景演算的一种改造形式并将其应用于对语义 Web 服务的推理. Gu 等一方面将情景演算中出现的所有公式都限制为 C^2 逻辑中的公式, 另一方面将描述逻辑中的 RBox 和 TBox 引入情景演算中; 利用改造后得到的动作理论, 可以对语义 Web 服务进行建模, 进而对语义 Web 服务的可执行性和投影问题进行推理. 这种方法保证了相关推理问题是可判定的, 并且关于语义 Web 服务的知识与领域本体被有机地结合了起来. 但是, 具体的推理过程仍然需要借助定理证明器来完成; 此外, 作为情景演算的一种特殊形式, 在信息不完全的情况下

进行规划时同样需要借助于定理证明器^[10].

6 结 论

动态描述逻辑 $DDL(SHOIN(\mathbf{D}))$ 将语义 Web 上由本体承载的大量知识与动态领域中关于动作的描述和推理有机地结合起来, 为语义 Web 服务的建模和推理以及在此基础上的服务发现和服务组合提供了一种有效途径. 这种途径还具有以下特点: 首先, 可以从 OWL-S 中的 ProcessModel 出发, 经过比较直接的转换后建立起语义 Web 服务的模型; 其次, 可以直观地对多种类型的推理问题进行刻画, 并且将这些推理问题转换为 $DDL(SHOIN(\mathbf{D}))$ 中公式的可满足性问题; 最后, 利用动态描述逻辑的基于开世界假设的判定算法, 可以在信息不完全的情况下实现上述各种推理.

针对语义 Web 服务的自动发现和自动组合问题, 基本途径是应用某种形式系统对语义 Web 服务进行建模, 在此基础上进行推理或规划. 形式系统的描述能力与推理能力(或推理性能)之间是一对矛盾关系; 除了寻求两者之间的折衷之外, 一个现实的途径是提供具有不同描述能力(相应地具有不同的推理能力)的多套工具, 让使用者按需使用. 从这个角度看, 本文的工作提供了一种有效的可供选择的工具和途径.

接下来的一项工作是在动态描述逻辑中引入关于动作的并行构造符, 从而可以处理复合过程中的控制结构 Split 和 Split+Join.

参 考 文 献

- [1] McIlraith S, Son T, Zeng H. Semantic Web services. IEEE Intelligent Systems, 2001, 16(2): 46-53
- [2] Martin D et al. Bringing semantics to Web services with OWL-S. World Wide Web, 2007, 10: 243-277
- [3] Martin D, Burstein M, McDermott D et al. OWL-S 1.2 Release. Available at: <http://www.daml.org/services/owl-s/1.2/>
- [4] Horrocks I, Patel-Schneider P F, Harmelen F V. From SHIQ and RDF to OWL: The making of a Web ontology language. Journal of Web Semantics, 2003, 1(1): 7-26
- [5] Shi Zhong-Zhi, Dong Ming-Kai, Jiang Yun-Cheng, Zhang Hai-Jun. A logical foundation for the semantic Web. Science in China (Series E), 2004, 34(10): 1123-1138(in Chinese) (史忠植, 董明楷, 蒋运承, 张海俊. 语义 Web 的逻辑基础. 中国科学(E辑), 2004, 34(10): 1123-1138)
- [6] Chang Liang, Shi Zhong-Zhi, Qiu Li-Rong, Lin Fin. A Tableau decision algorithm for dynamic description logic. Chinese Journal of Computers, 2008, 31(6): 896-909(in Chinese)

- (常亮, 史忠植, 邱莉榕, 林芬. 动态描述逻辑的 Tableau 判定算法. 计算机学报, 2008, 31(6): 896-909)
- [7] Chang Liang, Lin Fin, Shi Zhong-Zhi. A dynamic description logic for representation and reasoning about actions//Zhang Zi-Li, Siekmann J eds. Proceedings of the 2nd International Conference on Knowledge Science, Engineering and Management. Berlin: Springer-Verlag, 2007: 115-127
- [8] Giacomo G D, Lenzerini M. PDL-based framework for reasoning about actions//Gori M, Soda G eds. Proceedings of the 4th Congress of the Italian Association for Artificial Intelligence. Berlin: Springer-Verlag, 1995: 103-114
- [9] Bylander T. The computational complexity of propositional STRIPS planning. Artificial Intelligence, 1994, 69(1-2): 165-204
- [10] Reiter R. Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems. Cambridge, MA: MIT Press, 2001
- [11] Thielscher M. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. Artificial Intelligence, 1999, 111(1-2): 277-299
- [12] Chang Liang. Research on extended dynamic description Logic and its application[Ph. D. dissertation]. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 2008(in Chinese)
(常亮. 扩展的动态描述逻辑及其应用研究[博士学位论文]. 中国科学院计算技术研究所, 北京, 2008)
- [13] Horrocks I, Patel-Schneider P F, Boley H et al. SWRL: A semantic Web rule language combining OWL and ruleML. W3C Member Submission, 2004
- [14] Narayanan S, McIlraith S A. Simulation, verification and automated composition of Web services//Proceedings of the 11th International World Wide Web Conference. New York: ACM Press, 2002: 77-88
- [15] McIlraith S A, Son T C. Adapting Golog for composition of semantic Web services//Fensel D et al eds. Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning. 2002: 482-496
- [16] Wu Dan, Parsia B, Sirin E, Hendler J A, Nau D S. Automating DAML-S Web services composition using SHOP2//Fensel D, Sycara K P, Mylopoulos J eds. Proceedings of the 2nd International Semantic Web Conference. Berlin: Springer-Verlag, 2003: 195-210
- [17] Klusch M, Gerber A, Schmidt M. Semantic Web service composition planning with OWLS-XPlan//Proceedings of the AAAI Fall Symposium on Semantic Web and Agents. Arlington VA. USA: AAAI Press, 2005
- [18] Keller U, Lausen H, Stollberg M. On the semantics of functional descriptions of Web services//Sure Y, Domingue J eds. Proceedings of the 3rd European Semantic Web Conference. 2006: 605-619
- [19] Gu Yi-Lan, Soutchanski M. Decidable reasoning in a modified situation calculus//Veloso In M eds. Proceedings of the 20th International Joint Conference on Artificial Intelligence. 2007: 1891-1897



SHI Zhong-Zhi, born in 1941, professor, Ph. D. supervisor. His main research interests include artificial intelligence, machine learning, multi-agent system and semantic Web.

CHANG Liang, born in 1980, Ph. D.. His research interests include description logics, semantic Web, and intelligent agents.

Background

In order to represent and process the knowledge about semantic Web services, an obvious concern is to combine in some way the static description of the information provided by ontologies with the dynamic description of computations provided by Web services. In the authors' previous work, by embracing actions into description logics, a family of dynamic description logics named $DDL(X)$ was proposed, where X represents description logics ranging from the ALC to the $SHOIQ(\mathbf{D})$. The $DDL(X)$ offers considerable expressive power for representing knowledge about dynamic application domains; it also provides effective mechanisms and algorithms for related reasoning tasks.

Based on the dynamic description logic $DDL(SHOIQ(\mathbf{D}))$, this paper presents a framework for modeling and reasoning about semantic Web services. With this framework, the OWL-

S description of Web services will be firstly translated into an action theory which is based on the $DDL(SHOIQ(\mathbf{D}))$; then the realizability, executability, projection and planning problems on semantic Web services can be defined and reasoned about. These mechanisms provide effective support for the discovery and composition of semantic Web services.

This research is supported by the National Natural Science Foundation of China under grant No. 60775035, with the title "Logical Foundation for the Semantic Web Service". It was also partially supported by the National Natural Science Foundation of China under grant No. 90604017, the National High-Tech Research and Development Plan of China under grant No. 2007AA01Z132, and the National Basic Research Program of China under grant No. 2007CB311004.