

OWL-S 过程模型的一阶动态逻辑语义研究

李 明, 刘 冬

(兰州理工大学计算机与通信学院, 兰州 730050)

摘 要: 针对 Web 服务本体语言(OWL-S)过程模型语义不完善、难以对其进行有效形式化分析和验证的问题, 提出一种 OWL-S 过程模型的形式化方法。该方法对原子过程提供的输入、输出参数、前提条件、执行效果, 以及组合过程控制构造子的语义进行描述, 从而得到过程模型的一阶动态逻辑语义。实例结果验证了该方法的有效性。

关键词: 语义 Web; Web 服务; Web 服务本体语言; 过程模型; 一阶动态逻辑; 形式化

Research on First-order Dynamic Logic Semantic of OWL-S Process Model

LI Ming, LIU Dong

(School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China)

【Abstract】 The semantics of Ontology Web Language for Services(OWL-S) process model is not complete, which makes it difficult to carry out analysis and verification by machine. To solve this problem, this paper proposes a formalization method of OWL-S process model. It describes the atomic processes with their inputs, outputs, preconditions and effects and composite processes with their control construct, then the first-order dynamic logic semantics for process model is obtained. Example result validates the effectiveness of this method.

【Key words】 semantic Web; Web services; Ontology Web Language for Services(OWL-S); process model; first-order dynamic logic; formalization

DOI: 10.3969/j.issn.1000-3428.2012.12.013

1 概述

Web 服务本体语言(Ontology Web Language for Services, OWL-S)是语义 Web 中用来描述 Web 服务属性和功能的本体规范, 其目标是使 Web 服务成为计算机可理解的实体^[1]。过程模型作为 OWL-S 的核心, 描述了 Web 服务的执行流程, 一个服务也通常被称为一个过程。然而, 它的语义是不完善的, 控制构造子的语义无法被本体语言 OWL-DL 表达^[2]。为了能使 OWL-S 描述的模型是完全形式化的, 能为后期的形式化验证提供基础和支持, 需要选择一种形式化方法对 OWL-S 过程模型进行描述。

目前已有许多学者对过程模型的语义进行研究。文献[3]使用 Object-Z 语言对 OWL-S 的静态方面进行刻画。文献[4]将 OWL-DL 的语法和 F-logic 的语法进行映射, 给出 OWL-S 的语法和静态语义。文献[5]通过 rewrite logic 展示 OWL-S 的语法、静态和动态语义。文献[6]提出 OWL-S 过程模型的标签 Petri 网建模方法, 对 OWL-S 中的原子过程和控制构造子进行具体描述。

这些研究工作为 OWL-S 形式化描述和应用提供了良好的范例, 但也存在一些不足: 采用 Object-Z 语言和 F-logic 的方法未表示模型控制构造子的语义, 而采用 rewrite logic 和标签 Petri 的方法难以对过程模型的前提条件和效果进行表示。针对上述问题, 本文对 OWL-S 过程模型的一阶动态逻辑语义进行研究。通过分析过程模型和一阶动态逻辑的特点, 利用一阶动态逻辑的原子程序和复杂程序描述过程模型的原子过程和组合过程。

2 OWL-S 过程模型

在 OWL-S^[1]中, 描述服务的基本信息主要由 3 类本体:

Service Profile 描述服务能做什么; Service Model 描述服务如何工作; Service Grounding 描述如何访问一个服务。过程模型是 Service Model 的子类, 完整地展示了 Web 服务的运行方式。

过程模型定义了一个过程本体, 包含原子过程、简单过程和组合过程。

(1)原子过程是不可再分的过程, 可以被直接调用。它没有子过程, 并且能通过一步完成。每个原子过程都必须提供一个 grounding 信息, 用于描述如何去访问这个过程。

(2)简单过程是不可以被直接调用的过程, 但可以一步完成。根据不同的角度, 它既可以被看作原子过程, 又可以被当作组合过程。

(3)组合过程是由若干原子过程和组合过程构成的过程。每个过程由控制构造子来定义子过程的执行顺序, 由数据流反映子过程之间的数据依赖关系。目前, OWL-S 定义的控制构造子有 Sequence、Choice、Any-Order、If-Then-Else、Iterate、Repeat-While、Repeat-Until、Split、Split-Join。

3 一阶动态逻辑

动态逻辑是关于程序或动作推理的逻辑, 它可以视为一阶逻辑、模态逻辑和正则事件三者的结合, 文献[7]对其进行了详细的介绍。动态逻辑最显著的特点是拥有一个称为“程序”的语法结构, 通过它改变变量的值, 进而改变整个公式的值。

基金项目: 甘肃省自然科学基金资助项目(1014RJZA028)

作者简介: 李 明(1959—), 男, 教授, 主研方向: 智能信息处理, 软件工程; 刘 冬, 硕士研究生

收稿日期: 2011-08-30 **E-mail:** liudong0906@126.com

3.1 语法

一阶动态逻辑包含一个与一阶谓词逻辑相同的基本词汇表。基于该词汇表, 定义了程序和公式的集合。设 $\Sigma = \{f, g, \dots, p, r, \dots\}$ 是一阶词汇表, 其中, f, g 表示函数符号; p, r 表示关系符号。设 $V = \{x_0, x_1, \dots\}$ 为个体变量的集合。

定义 1 在一阶动态逻辑中, 原子程序是一个赋值语句 $x: t$ 。其中, $x \in V$, t 为 Σ 中的项。该赋值语句表示将项 t 的值赋给变量 x 。

定义 2 在一阶动态逻辑中, 原子公式是词汇表 Σ 上的公式。它的形式为: $r(t_1, t_2, \dots, t_n)$, 其中, r 为 Σ 上的 n 元关系符号; t_1, t_2, \dots, t_n 为 Σ 上的项。

定义 3 给定原子程序, 一阶动态逻辑的正则程序按照下列方式被定义:

- (1) 原子程序是程序;
- (2) 如果 ϕ 为公式, 则 $\phi?$ 为程序;
- (3) 如果 a_1, a_2 为程序, 则 $a_1; a_2, a_1 \cup a_2, a_1^*, a_1 \wedge a_2$ 均为程序;
- (4) 程序均由方式(1)~方式(3)产生。

定义 4 一阶动态逻辑中的公式同一阶谓词逻辑中公式类似, 只是增加了模态操作符。其定义如下:

- (1) 任何原子公式是公式;
- (2) \top 表示真, \perp 表示假是公式;
- (3) 如果 ϕ_1, ϕ_2 是公式, 则 $\neg\phi, \phi_1 \wedge \phi_2, \phi_1 \vee \phi_2, \phi_1 \rightarrow \phi_2, \phi_1 \leftrightarrow \phi_2$ 也是公式;
- (4) 如果 ϕ 是公式且 $x \in V$, 则 $\forall x\phi, \exists x\phi$ 是公式;
- (5) 如果 ϕ 是公式, a 是程序, 则 $[a]\phi, \langle a \rangle\phi$ 是公式;
- (6) 任何公式都是经过有限次使用方式(1)~方式(5)产生的。

3.2 直观含义

$\phi?$: 测试公式 ϕ , 若 ϕ 为真则程序执行, 否则不执行(测试);

$a_1; a_2$: 程序 a_1 先执行, 然后 a_2 再执行(组合);

$a_1 \cup a_2$: 不确定的选择 a_1 或 a_2 (不确定性的选择);

a_1^* : 程序循环执行(循环);

$a_1 \wedge a_2$: 程序 a_1 和 a_2 并行执行(并行);

$[a]\phi$: ϕ 在执行程序 a 后的所有状态中均保持成立;

$\langle a \rangle\phi$: ϕ 在执行程序 a 后所得的所有状态中, 至少有一个成立。

3.3 程序描述

文献[8]给出一阶动态逻辑程序的描述。通常一个程序拥有一个效果来表示对世界产生的影响和一个前提条件来保证程序的正确发生。效果可以用“有条件的后置条件公式”来表示。

条件效果公式: $\phi \rightarrow [a]\phi$ 。称 ϕ 是程序 a 关于 ϕ 的充分条件, 其中, ϕ, ϕ 为一阶动态逻辑的公式; a 为原子程序。

程序的前提条件可以由“守卫公式”来表示。

守卫公式: $\langle a \rangle \top \rightarrow \chi_j$ 。把公式 χ_j 称作程序 a 的警卫, 也可以称作程序 a 可能发生的必要条件。

静态约束: θ 。这些是程序在任何情况下都必须满足的非模态的断言公式。

4 OWL-S 过程模型的一阶动态逻辑语义

过程的执行可以导致世界状态的变化, 动态逻辑是对状

态和改变状态的程序进行推理的逻辑, 因此, 可用动态逻辑的程序来表示过程。

4.1 原子过程形式语义的描述

在 OWL-S 中, 原子过程与单个 Web 服务调用相对应。它主要由 4 个元素组成, 即输入(Input)、输出(Output)、前提条件(Precondition)和结果(Result), 简称“IOPR”。其中, 输入表示服务执行所需要的信息; 输出表示服务返回的信息; 前提条件表示服务执行必须满足的条件; 结果表示服务执行所造成的影响, 包括条件、效果和输出。

(1) 原子过程

OWL-S 的原子过程可用一阶动态逻辑的原子程序表示, 其中, 过程名为程序名。例如, 原子过程 Login 可用原子程序 Login 表示。

(2) 变量声明和数据绑定

OWL-S 原子过程中的变量声明可用一阶动态逻辑中程序的静态约束公式表示, 其中, 变量的类型名与变量名同公式的公式名和变量名保持一致。例如原子过程的变量 x 、声明的类型为 C , 则在一阶动态逻辑中被表示为 $C(x)$ 。

对于 OWL-S 中存在绑定关系的一对变量 x 和 y , 通过定义公式 $bind(x, values(y))$ 来表示, 其含义是变量 x 和变量 y 的值存在绑定关系。

(3) 前提条件和输入

本文将 OWL-S 原子过程的前提条件和输入看作一阶动态逻辑中程序执行的必要条件, 分别称为物理前提条件和知识前提条件。物理前提条件被表示为 $\langle a \rangle \top \rightarrow \chi_j$ 。其中, a 和 χ_j 分别为一阶动态逻辑中的程序和公式。如果有多个条件, 则被表达为: $\langle a \rangle \top \rightarrow \chi_1 \wedge \chi_2 \wedge \dots \wedge \chi_n$ 。

输入被看作知识前提条件, 其含义为在执行一个服务前, 主体必须知道输入参数的值。本文将知识条件被表示为 $\langle a \rangle \top \rightarrow know(x)$ 。其中, x 为原子过程的输入参数; $know(x)$ 为本文定义了公式, 含义为 x 的值是知道的。

(4) 条件效果和输出

OWL-S 原子过程的效果可以用一阶动态逻辑的效果公式 $\phi \rightarrow [a]\phi$ 表示, 称为物理效果。其中, ϕ 为当程序执行使 ϕ 为真的条件; a 为程序; ϕ 为原子过程产生的效果。

本文将过程模型的输出看作知识效果, 表示主体被告知信息。它可以用如下公式表示: $\phi \rightarrow [a]know(x)$ 。其中, ϕ 为程序执行的条件; a 为程序; x 表示输出参数。

(5) 结果

在 OWL-S 中, 结果为条件、输出和效果三者的绑定。服务根据不同的条件式产生不同的输出和效果。结果条件与服务的前提条件最大的不同在于结果条件并不会影响服务能否执行, 只会对服务产生的效果和输出造成影响。结果在一阶动态逻辑中可表示为:

$$\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \rightarrow [a] \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \wedge know(x_1) \wedge know(x_2) \wedge \dots \wedge know(x_n)$$

其中, $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n$ 为程序执行的前提条件和结果条件的合取; $know(x_1) \wedge know(x_2) \wedge \dots \wedge know(x_n)$ 为知识效果的合取; $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n$ 为物理效果的合取。

4.2 组合过程形式语义的描述

组合过程用于表示具有复杂业务逻辑的服务, 它通常由原子服务或其他组合服务通过控制构造子组合而成。同 OWL-S 组合过程类似, 一阶动态逻辑提供了复杂程序的构造

子来表示顺序、条件、不确定的选择等复杂程序。基于此, 本文将 OWL-S 组合过程和一阶动态逻辑的复杂程序进行映射, 以此来说明控制构造子的语义。

(1)Sequence

假设 C 是过程模型中由 Sequence 控制构造子构成的组合过程, C_1 和 C_2 为其子过程。则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1};\pi_{c_2}$, 其中, π_c 为定义 C 的复杂程序, π_{c_1} 、 π_{c_2} 为定义 C_1 与 C_2 程序。

(2)Choice

假设 C 是过程模型中由 Choice 控制构造子构成的组合过程, C_1 和 C_2 是其子过程, 则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1} \cup \pi_{c_2}$, 其中, π_c 为定义 C 的复杂程序; π_{c_1} 、 π_{c_2} 为定义 C_1 与 C_2 的程序。

(3)Any-order

假设 C 是过程模型中由 Any-order 控制构造子构成的组合过程, C_1 和 C_2 是其子过程。则 C 在 DL 中被映射为 $\pi_c = (\pi_{c_1} \cup \pi_{c_2});(\pi_{c_2} \cup \pi_{c_1})$, 其中, π_c 为定义 C 的复杂程序, π_{c_1} 、 π_{c_2} 为定义 C_1 与 C_2 的程序。

(4)If-then-else

假设 C 是过程模型中由 If-then-else 控制构造子构成的组合过程, C_1 、 C_2 为 C 子过程, ϕ 是子过程执行的条件。该构造子的含义为如果 ϕ 为真, 则 C_1 执行, 否则 C_2 执行。则 C 在 DL 中被映射为 $\pi_c = (\phi?;\pi_{c_1}) \cup ((\neg\phi)?;\pi_{c_2})$, 其中, π_c 为定义 C 的复杂程序; π_{c_1} 、 π_{c_2} 为定义 C_1 与 C_2 的程序; ϕ 为相应的公式。

(5)Iterate

假设 C 是过程模型中由 Iterate 控制算子构成的组合过程, C_1 是其子过程, 它的含义是不断地执行 C_1 。则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1}^*$, 其中, π_c 为定义了 C 的复杂程序; π_{c_1} 为定义了 C_1 的程序。

(6)Repeat-while

假设 C 是过程模型中由 Repeat-while 控制算子构成的组合过程, C_1 是其子过程, ϕ 为循环条件。该构造子的含义为当 ϕ 为真时, 不断执行 C_1 , 直到 ϕ 为假。则 C 在 DL 中被映射为 $(\phi?;\pi_{c_1})^*;\neg\phi?$, 其中, π_c 为定义了 C 的复杂程序; π_{c_1} 为定义了 C_1 的程序; ϕ 为 DL 中相应的公式。

(7)Repeat-until

假设 C 是过程模型中由 Repeat-until 控制算子构成的组合过程, 其中, C_1 为其子过程; ϕ 为循环条件。它的含义是首先执行子过程 C_1 , 然后判断 ϕ , 如果为假, 继续执行 C_1 , 直到 ϕ 为真为止。则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1};((\neg\phi)?;\pi_{c_1})^*;\phi?$ 。其中, π_c 为定义了 C 的复杂程序; π_{c_1} 为定义了 C_1 的程序; ϕ 为 DL 中相应的公式。

(8)Split

假设 C 是过程模型中由 Split 控制算子构成的组合过程, C_1 、 C_2 、 C_3 是其子过程, 它的含义为执行完 C_1 后, C_2 与 C_3 并发执行。则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1};(\pi_{c_2} \wedge \pi_{c_3})$, 其中, π_c 为定义了 C 的复杂程序; π_{c_1} 、 π_{c_2} 、 π_{c_3} 为定义了 C_1 、 C_2 、 C_3 的程序。

(9)Split+Join

假设 C 是过程模型中由 Split+Join 控制算子构成的组合过程, C_1 、 C_2 、 C_3 、 C_4 是其子过程, 它的含义执行完 C_1 后,

C_2 与 C_3 并发执行, 当 C_2 与 C_3 都完成后执行 C_4 。则 C 在 DL 中被映射为 $\pi_c = \pi_{c_1};(\pi_{c_2} \wedge \pi_{c_3});\pi_{c_4}$, 其中, π_c 为定义了 C 的复杂程序; π_{c_1} 、 π_{c_2} 、 π_{c_3} 、 π_{c_4} 为定义了 C_1 、 C_2 、 C_3 、 C_4 的程序。

5 实例验证

以 OWL-S 推荐的标准实例——BravoAirProcess.owl^[9]中的原子过程 *LogIn* 为例, 来说明如何应用一阶动态逻辑对 OWL-S 的过程模型进行描述。在子过程 *LogIn* 的表示中省略了与 RDF 和 XML 等语法相关的描述成分, 伪代码如下:

```
define atomic process LogIn
(
  Inputs: (LogIn_AcctName -AcctName
           LogIn_Password -Password),
  Output: (LogIn_Output -boolean),
  Results: ((hasPassword(LogIn_AcctName,
                          LogIn_Password)
             |->LogIn_Output<=true &LoggedIn(AcctName))
            &hasPassword(LogIn_AcctName,
                          Correct_Password)&different(Correct_Password, LogIn_Password)
             |->LogIn_Output<=false &NotLoggedIn(AcctName))
)
```

该原子过程描述的服务如下: 用户登录系统时, 系统要求用户输入账户名和登录密码。当用户输入的账户名和登录密码一致, 系统输出登录正确的信息, 产生用户的登录状态正确的效果; 当用户输入的登录密码与账户的正确密码不一致时, 系统输出登录错误信息, 产生用户登录状态无效的效果, 则登录过程 *LogIn* 用一阶动态逻辑表示为:

(1)静态约束公式

$$\text{AcctName}(\text{LogIn_AcctName}) \wedge \\ \text{Password}(\text{LogIn_Password}) \wedge \\ \text{boolean}(\text{LogIn_AcctName}) \wedge \\ \text{Password}(\text{Correct_Password})$$

(2)守卫公式

$$\langle \text{LogIn} \rangle \top \rightarrow \text{know}(\text{LogIn_AcctName}) \wedge \\ \text{knows}(\text{LogIn_Password})$$

(3)条件效果公式

1)登录成功

$$\text{know}(\text{LogIn_AcctName}) \wedge \\ \text{knows}(\text{LogIn_Password}) \wedge \\ \text{hasPassword}(\text{LogIn_AcctName}, \\ \text{LogIn_Password}) \rightarrow \\ [\text{LogIn}] \text{knows}(\text{LogIn_Output}) \wedge \\ \text{bind}(\text{LogIn_Output}, \text{true}) \wedge \\ \text{Logged}(\text{AcctName})$$

2)登录失败

$$\text{know}(\text{LogIn_AcctName}) \wedge \\ \text{knows}(\text{LogIn_Password}) \wedge \\ \text{knows}(\text{Correct_Password}) \wedge \\ \text{hasPassword}(\text{LogIn_AcctName}, \\ \text{Correct_Password}) \wedge \\ (\text{Correct_Password}, \text{LogIn_Password}) \rightarrow \\ [\text{LogIn}] \text{knows}(\text{LogIn_Output}) \wedge \\ \text{bind}(\text{LogIn_Output}, \text{false}) \wedge \\ \text{NotLogged}(\text{AcctName})$$

(下转第 51 页)