基于扩展CPN的OWL-S过程语义建模及分析方法研究

鲍爱华[1] 王晓璇[2,3] 文 艾[1] 丁 科[1] 刘 鹏[1]
(中国人民解放军理工大学全军网格技术研究中心 南京 210007)[1]
(中国水产科学研究院东海水产研究所渔业资源遥感信息技术重点开放实验室 上海 200090)[2]
(同济大学测量与国土信息工程系 上海 200092)[3]

摘 要 OWL-S过程语义的建模与分析是语义Web服务相关领域需要重点研究的问题。分析了目前OWL-S过程语义研究中存在的问题,提出了一种扩展的着色Petri网PM_net(过程模型网,Process Model net)来对OWL-S的过程语义进行转化与分析。结合OWL-S过程模型元素的特点,PM_net对基本着色Petri网的变迁和触发规则进行了扩展,使OWL-S的原子过程、组合过程和数据流等核心元素能够等价映射到PM_net。同时说明了如何基于PM_net对OWL-S的过程语义一致性进行分析,为OWL-S本体演化、语义Web服务组合和验证提供了合理的理论基础。

关键词 过程语义,OWL-S,过程模型网,着色Petri网,本体演化,语义Web
中图法分类号 TP393 **文献标识码** A

Modeling and Analysis for the Process Formal Semantics of OWL-S Based on Extended CPN

BAO Ai-hua[1] WANG Xiao-xuan[2,3] WEN Ai[1] DING Ke[1] LIU Peng[1]
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)[1]
(Key Laboratory of Fisheries Resources Remote Sensing and Information Technology Resources, East China Sea Fisheries Research Institute, Chinese Academy of Fishery Sciences, Shanghai 200090, China)[2]
(Department of Survying and Geoinformatics, Tongji University, Shanghai 200092, China)[3]

Abstract The process formal semantics of OWL-S is a key issue in the research on semantic Web service. This paper analyzed the current work and existing problem of research on OWL-S process formal semantics, and proposed an extended Colored Petri net, which is called Process Model net(PM_net), to model and analyze the process formal semantics of OWL-S. According to the characteristic of OWL-S process model, PM_net extended the transition and fire rule of basic CPN, so that, the atomic process, composite process and data flow of OWL-S process model can be mapped to PM_net equally. The method to check consistency of OWL-S process formal semantics based on PM_net was also introduced. The work of this paper provided reasonable theroy foundation for the evolution of OWL-S, semantic Web service composition and verification.

Keywords Process formal semantics, OWL-S, Process model net, Colored Petri net, Ontology evolution, Semantic Web

1 引言

随着Web服务的兴起和发展,如何将Web服务与语义Web技术相结合,通过Web服务的语义封装,为用户提供自动的Web服务(即语义Web服务)已成为一个重要的研究课题。语义Web服务主要的研究目标就是对Web服务进行语义标注,使之成为计算机可理解、Agent可处理和用户透明的实体[1]。

目前关于语义Web服务的研究中,大部分集中在语义Web服务的应用上,包括语义Web服务的描述、匹配、组合和监控等,DARPA组织所提出的OWL-S本体规范即是其中比较典型的研究成果。OWL-S的研究者从3个方面对Web服务进行语义标注[2]:ServiceProfile从整体上描述服务的功能和服务与用户的交互,它主要用于Web服务的发布和自动发现;ServiceGrouding描述如何对服务进行访问,它主要用于Web服务的自动调用;ServiceModel是OWL-S的核心,它通过一个过程本体对服务的执行流程进行描述,是语义Web服务自动组合、验证和监控的基础。

由于语义Web服务是面向机器的计算实体,因此严格的形式语义是机器对其进行自动处理的基础。而在目前的研究中,OWL-S仅从语法角度对Web服务的过程模型进行了描述,对过程模型语义的定义也停留在概念框架层次,这使得机器难以对Web服务过程模型进行自动分析(如死锁分析、一致性检查等),因此OWL-S过程模型的形式语义(特别是过程语义)成为语义Web服务研究中一个亟待解决的问题。

针对OWL-S过程语义问题,目前许多研究者对此进行

到稿日期:2010-05-24 返修日期:2010-09-11 本文受国家863项目(2007AA092202),国家高技术研究发展计划(863计划)(2009AA12Z214),国家自然科学基金(40801060)资助。
鲍爱华(1981—),男,博士,讲师,主要研究方向为分布式人工智能、语义Web。

了研究，并取得了一定的研究成果。Srini Narayanan 等人利用情景演算理论对 DAML-S(OWL-S 的前身)的操作语义进行了研究[3]，并以情景演算为中介，将情景演算转换为 Petri 网，以此研究语义 Web 服务的组合与推理。蒋运承等人在 Srini 的研究基础上，采用情景演算对 OWL-S 中组合服务的形式语义进行了表示[4]。李景霞等人将 OWL-S 的过程模型与 Petri 网进行了映射[5,6]，采用 Petri 网对 OWL-S 中的原子过程和控制构造子进行了具体描述，并采用 Petri 网的分析方法对 OWL-S 中过程的可达性进行了分析。

在对 OWL-S 的过程语义分析中，目前的研究也存在一定的不足。采用情景演算虽然能够表示 OWL-S 的形式语义，但是在映射时过程的 Input/Output 需要转化为 Precondition/Effect 进行表示，这使得过程模型的信息产生丢失；同时情景演算也缺乏直观表示，知识工程师难以直接将过程模型与情景演算进行对应，导致情景演算推理结果难以直接反馈到过程模型。采用基本 Petri 网对 OWL-S 的过程模型进行表示时，过程的 Precondition 和 Condition Results 难以直接表示，需要进行扩展。另外，在相关的研究中对 OWL-S 中数据流的分析也存在一定的不足。

基于上述原因，本文借鉴了 Petri 网严格的形式化语义和直观易懂的优点，对着色 Petri 网进行了扩展，提出了一种过程模型网(Process Model Net，PM_net)对 OWL-S 的过程语义进行建模，并基于 PM_net 对 OWL-S 的语义一致性问题进行分析。结合 OWL-S 过程模型元素的特点，PM_net 对着色 Petri 网的变迁和触发规则进行了扩展，使 OWL-S 的原子过程、组合过程和数据流等核心元素能够与 PM_net 进行等价映射，从而使知识工程师能够采用 Petri 网的分析技术来对 OWL-S 的过程语义进行分析，为后续 OWL-S 过程模型的演化[7,8]、验证和监控提供合理的理论基础。

2 过程模型网 PM_net

2.1 设计思想

在 OWL-S 的过程模型中，核心的元素包括过程、组合过程控制构造子和数据流等。OWL-S 中的过程代表了语义 Web 中可被访问的 Web 服务，主要由输入、输出、前提条件和效果(IOPE)表示。其中，Input/Output 隶属于信息空间范畴，表示 Web 服务在被调用时数据的流动；Precondition/Effect 隶属于状态空间范畴，表示 Web 服务在执行前后状态空间中状态变量的变化。另外，OWL-S 中的过程支持多种条件输出，因此也可以将 Effect 和 Output 进行捆绑，用来表示 Web 服务执行后的后果(Result)。但是在传统的 Petri 网中，库所中的 Token 仅代表资源的流动，它类似于信息空间中的数据，难以直接表示状态空间的变化。因此，必须对传统的 Petri 网进行适当的扩展，才能对 OWL-S 过程模型的各个元素进行表示，进而基于 Petri 网严格的形式语义来对 OWL-S 的形式语义进行建模。

针对上述问题，本文对着色 Petri 网进行了扩展，提出了一种过程模型网 PM_net 来对 OWL-S 过程模型中的核心元素进行建模。在 PM_net 中，颜色集表示 OWL-S 中与过程相关的各类参数数据类型，变迁与具体的过程相对应，库所和变迁之间 Token 的流动则代表信息空间中的输入和输出。除此之外，PM_net 还定义了一个状态变量集 V_s，用于对状态空间进行描述。与库所中的颜色不同，在 PM_net 的执行过程中，状态变量不随 Token 的流动而改变，只有在变迁触发后世界状态发生变化时状态变量才会进行修改。PM_net 中的变迁也与一般着色 Petri 网中的变迁不同。新变迁的标签函数被划分为两种类型 TF_{pre} 和 TF_{eft}，前者表示变迁在触发前状态空间所需要满足的条件，而后者则表示变迁在触发后状态空间如何进行修改，它们分别对应于 OWL-S 中过程的 Precondition 和 Effect。在 PM_net 中，变迁的触发规则也发生改变。变迁的触发不仅取决于前继库所中的资源数量，还需要变迁的前提条件函数为真；在变迁触发后，除了库所中的资源发生流动，状态空间中的状态变量也需要依照变迁的效果函数进行修改。

通过扩展，PM_net 不仅能够表示信息空间中的输入输出参数，也能表示状态空间中的前提条件与后继效果，使 OWL-S 过程模型中的核心元素能够与 PM_net 进行等价映射。

2.2 PM_net 的形式化定义

下面，基于基本着色 Petri 网结构[9]对过程模型网进行形式化定义。

定义 1(过程模型网) 过程模型网 $PM\_net=(K,V_s,P,T;F,TF,E)$，其中[1]：

1)K 是非空有限的颜色集，代表 OWL-S 中所涉及的数据类型；

2)V_s 是状态变量有限集，用以表示当前的世界状态，且 $\forall vs\in V_s; Type(vs)\in K$；

3)P 代表库所有限集，T 代表变迁有限集，$P\cup T\neq\emptyset\wedge P\cap T=\emptyset$；

4)存在一个源库所 i 和终点库所 o，满足 $i,o\in P, {}^{\cdot}i=\emptyset\wedge o^{\cdot}=\emptyset$；

5)$F\subseteq P\times T\cup T\times P$ 代表 PM_net 中的弧的集合；

6)$TF: T\rightarrow Exprs$ 是变迁表达式函数，$TF=TF_{pre}\cup TF_{eft}$，$\forall t\in T, TF=(t)=TF_{pre}(t)/TF_{eft}(t)$，$Type(Var(TF(t)))\subseteq V_s$，其中，$TF_{pre}: T\rightarrow$[布尔表达式]，称为变迁前提条件函数，$TF_{eft}: T\rightarrow$[状态变量赋值表达式]，称为变迁后继效果函数；

7)E 是弧表达式函数，$E: F\rightarrow K_{MS}$。

定义 1 对过程模型网的静态结构进行了描述，与一般着色 Petri 网相比，对变迁函数进行了扩展，同时增加了一个状态变量集，使过程模型网能够对过程的 IOPE 进行准确的描述。下面，本文对过程模型网的动态结构进行描述。

定义 2(过程模型网的标识) 令 $P_n=(K,V_s,P,T;F,TF,E)$ 为过程模型网，则称 $M_p: P\rightarrow K_{MS}$ 为 P_n 的流标识；$M_s: V_s\rightarrow\{\cup_{x\in V_s} val(x)\}$ 称为 P_n 的状态变量标识；$M=M_p+M_s$ 统称为 P_n 的标识。

由定义 2 可知，过程模型网的标识由两部分组成，在变迁被触发时，流标识在库所间流动，用以表示信息空间中数据的传输和处理；状态变量标识则表示了过程模型网所处的世界

[1] 本文的定义中，$Expr$ 表示表达式，$Type(vs)$ 表示状态变量的取值类型，$Val(vs)$ 表示状态变量的取值，$Var(Expr)$ 表示表达式中变量的集合，KMS 表示颜色集 K 的多重集。

状态，它对变迁的触发具有约束作用。下面，我们对 PM_net 系统及其触发规则进行明确的定义。

定义 3(PM_net 系统)　$\Sigma=(P_n, M_0)$被称为 PM_net 系统，如果 $P_n=(K, V_s, P, T; F, TF, E)$是过程模型网，同时 $M_0 \in M$ 是P_n 的初始标识。

定义 4(∑的触发规则)　令 $M=M_p+M_s$ 为∑的标识，$t \in T$ 为∑的变迁，那么：

1)变迁 t 在标识 $M=M_p+M_s$ 下有发生权，即 $M[t>$，当且仅当标识 M 满足以下条件：① $\forall p \in {}^{\cdot}t, M_p(p) \geqslant E(p,t)$；②在 M_s 下，$TF_{pre}(t)=\text{true}$。

2)若 $M[t>M'$，$M'=M_p'+M_s'$，那么 M'满足以下条件：

① $\forall p \in P, M_p'(p)=$

$$\begin{cases} M_p(p)-E(p,t), & s \in {}^{\cdot}t-t^{\cdot} \\ M_p(p)+E(t,p), & s \in t^{\cdot}-{}^{\cdot}t \\ M_p(p)-E(p,t)+E(t,p), & s \in t^{\cdot} \cap {}^{\cdot}t \\ M_p(p), & s \in \text{other} \end{cases};$$

② $\forall x \in V_s, M_s'(x)=TF_{eft}(t)(x)$，其中，$TF_{eft}(t)(x)$表示通过变迁 t 的迁移，继效果函数计算后状态变量 x 新的值。

通过上述定义，过程模型网的静态结构与动态特性得到了确定。由于过程模型网具有基本 Petri 网结构良好、形式语义明确的优点，因此我们可以采用过程模型网对 OWL-S 的过程模型进行建模，并通过过程模型网的动态特性来表示 OWL-S 中各项过程控制构造子的操作语义。

3　基于 PM_net 的 OWL-S 过程模型建模

3.1　原子过程

在 OWL-S 中，原子过程与单个 Web 服务调用相对应。原子过程由 4 个元素组成，即 AtomicProcess=〈Input, Output, Precondition, Result〉。其中，Input/Output 代表了该过程在执行时信息空间中数据的流动，Precondition 代表该原子过程在执行前所需要满足的条件，Result 则代表过程的执行对信息空间和状态空间所造成的影响。在 OWL-S 中，Result 可由三元组〈Condition, Output, Effect〉表示，即过程执行结果代表了过程输出条件、参数和效果的绑定。通过上述定义，OWL-S 中的原子过程能够接受多个输入，同时能够针对不同的世界状态产生多个不同的效果与输出。

在本文中，我们采用如下规则将原子过程与 PM_net 进行映射。

规则 1(原子过程与 PM_net 的映射规则)

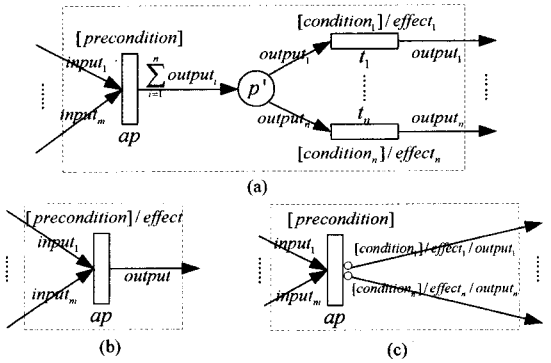
*S*1 将 OWL-S 中的原子过程 ap 映射为变迁 t，并将 t 命名为原子过程的名称；

*S*2 将原子过程 ap 的 m 个输入条件映射为变迁 t 的输入弧；

*S*3 对于 ap 的 n 个条件输出，建立一个新的库所 p'和 n 个变迁进行转化，其中 $\forall r_i=[cond_i]effect_i/output_i \in R, E(p', t_i)=output_i, TF_{pre}=cond_i, TF_{eft}=effect_i$；

*S*4 如果 ap 只有一个输出，那么将输出结果的 $effect$ 与变迁 ap 的效果函数进行合并。

原子过程 ap 与 PM_net 的转化如图 1 所示，图 1(a)表示一个多条件输出原子过程所对应的 PM_net 网结构，图 1(c)表示该网结构的简化表示，图 1(b)表示单输出原子过程所对应的 PM_net 网结构。



(a)具备多个条件输出的原子过程；(b)单个输出的原子过程
(c)多个条件输出原子过程的简化表示

图 1　原子过程与 PM_net 的转化

3.2　组合过程

在 OWL-S 中，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装而成。在 OWL-S 的过程模型中，组合过程是非常重要的元素，OWL-S 的过程语义也主要体现在过程控制构造子的操作语义上。但是，OWL-S 规范并没有对控制构造子如何运行进行精确的定义，因此，本文将采用 PM_net 对 OWL-S 中各类过程控制构造子进行描述，以此来说明控制构造子的操作语义。

在后续的说明中，本文将采用变迁(如图 1 中虚框所示)对原子过程、抽象过程和其他组合过程进行抽象表示，在描述中重点表示与控制构造子相对应的网结构，而忽略具体的输入输出参数。

(1)Sequence 控制结构

Sequence 用于连接一系列顺序执行的过程，在 PM_net 中，Sequence($proc_1$, $proc_2$)可由图 2(a)中的 PM_net 网结构表示。

(2)Split/Split+Join 控制结构

Split 用于表示 OWL-S 中一组过程的并发执行，当这些并发过程都开始执行时，Split 控制结构也就执行完毕。Split+Join在表示一组过程并发执行外，还对这些过程的完成进行同步，当所有的过程均执行完毕时，Split+Join 控制结构才执行完毕。Split($proc_1$, $proc_2$)和 Split+Join($proc_1$, $proc_2$)可由图 2(b)中的 PM_net 网结构表示。

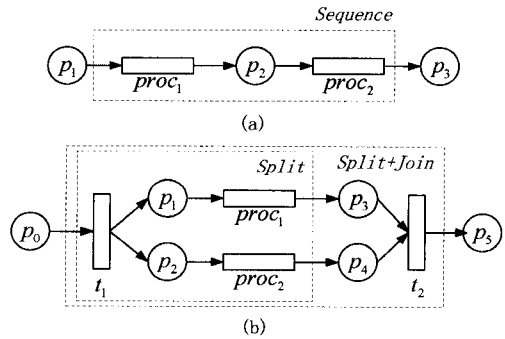


图 2　(a)Sequence 对应的 PM_net 网结构；(b)Split/Split+Join 对应的 PM_net 网结构

(3)Any-Order 控制结构

Any-Order 控制结构允许一组过程以无序但非并发的形式执行，具体而言，所包含的原子过程不能并发执行，组合过程之间也不能交叉执行。同时，Any-Order 结构要求所有的过程都必须执行，并且当所有过程都执行完毕后，Any-Order 结构才执行完毕。Any-Order($proc_1$, $proc_2$)可由图 3(a)中的 PM_net 网结构表示，它主要通过 p_2 这个控制库所来实现 $proc_1$ 和 $proc_2$ 的无序执行。

(4)Choice/If-Then-Else 控制结构

在 OWL-S 中，Choice 和 If-Then-Else 结构都用于表示在多个过程中进行条件选择。其中，Choice 表示在一组过程中随机选择一个过程执行；If-Then-Else 则表示根据世界状态的不同，在两个过程之间选择一个执行。Choice 和 If-Then-Else 可由图 3(b)所示的 PM_net 网结构表示，在 If-Then-Else 的情况下，$condition_1$ = ! $condition_2$；对于 m 个分支的 Choice 选择而言，$\forall 1 \leqslant i, j \leqslant m, [condition_i] \wedge [condition_j] = \text{false}, i \neq j$。

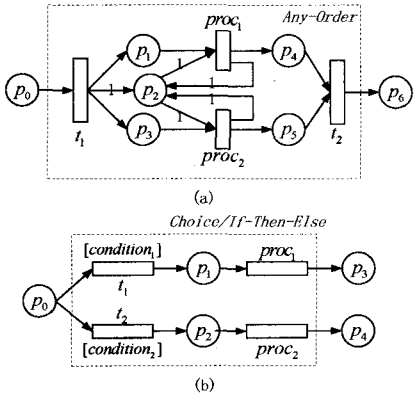


图 3 (a)Any-Order 对应的 PM_net 网结构；(b)Choice/If-Then-Else 对应的 PM_net 网结构

(5)循环控制结构

在 OWL-S 中，循环控制结构分为 Repeat-While 和 Repeat-Until 两种。其中，Repeat-While 首先对循环条件进行检查，在条件为真后循环执行过程，否则退出循环；Repeat-While 则在执行完循环过程后对循环条件进行检查，如果条件满足则继续执行，否则退出循环。Repeat-While 和 Repeat-Until 可分别由图 4(a)和图 4(b)所示的 PM_net 网结构进行表示。

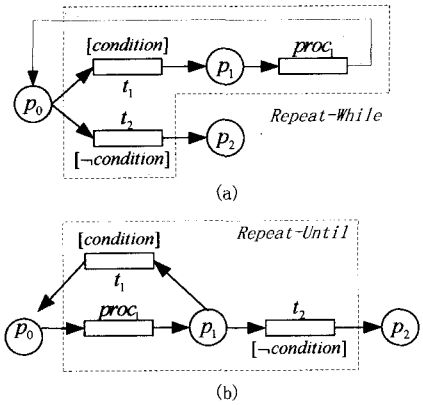


图 4 (a)Repeat-While 对应的网结构；(b)Repeat-Until 对应的网结构

3.3 数据流

在 OWL-S 的过程模型中，数据流是除过程和控制构造子之外的一个非常重要的元素，也是比较容易忽视的元素。数据流在过程模型中不仅表示了过程之间的数据依赖，也从一定程度上反映了过程之间的控制依赖，例如，在 Consumer-Pull 类型的数据流中，所涉及到的两个过程之间也间接存在着 Sequence 控制结构。因此，在对 OWL-S 的过程语义进行分析时，忽略数据流不仅会导致数据依赖的丢失，同时会造成对其过程语义的误解，造成过程信息的丢失。

鉴于此，本文对过程模型中的数据流同样采用 PM_net 网结构进行表示，从而对数据流的数据依赖特性和间接的控制依赖特性进行表示。OWL-S 中的数据流分为两种类型，Consumer-Pull 从数据消费者的角度描述过程输入参数的来源，而 Producer-Push 则从数据生产者的角度描述过程输出参数的流向。本文主要对 Consumer-Pull 类型的数据流进行分析。

图 5 表示了数据流 Consumer-Pull($desProc_1$, $desInput$, $srcProc$, $srcOutput$) 和 Consumer-Pull ($desProc_2$, $desInput$, $srcProc$, $srcOutput$)所对应的 PM_net 网结构。其中，源过程 $srcProc$ 的输出 $srcOutput$ 经过变迁 t_1 后复制为两份，分别作为过程 $desProc_1$ 和 $desProc_2$ 的输入参数。通过图 5 所示的 PM_net 网结构，过程间的数据流得到了形象的表示，同时过程之间的控制结构也可通过网结构得到体现。另外，数据流在转化为 PM_net 网结构后，数据流也可以与原子过程和组合过程采用统一的方法进行分析处理，简化了 OWL-S 过程语义分析的复杂度。

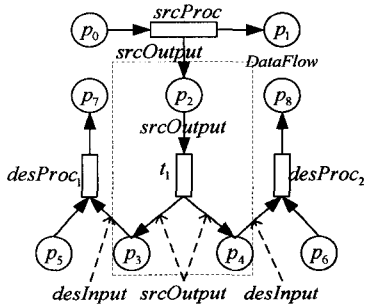


图 5 数据流对应的 PM_net 网结构

在完成过程模型与 PM_net 的转换后，下面本文将重点说明如何基于 PM_net 来分析过程模型的语义一致性。

4 基于 PM_net 的 OWL-S 过程语义一致性分析

作为面向机器的过程描述规范，OWL-S 的过程语义的一致性是机器正确理解并使用 OWL-S 本体的基本要素。例如，在领域背景发生变化导致 OWL-S 过程模型需要同步演化时，如果机器无法正确理解 OWL-S 的过程语义，那么在其过程模型发生变化后，机器就无法对修改后的过程模型进行自动处理，也就无法辅助知识工程师发现过程模型在修改过程中所发生的错误。因此，对于 OWL-S 过程模型的演化而言，OWL-S 过程语义的一致性是 OWL-S 本体自动或半自动演化的基础。

由于 OWL-S 本体规范仅对过程模型的语法进行了定义，对其过程语义并没有进行清晰的定义，因此如何分析 OWL-S 的过程语义一直是一个研究热点。本文采用过程模型网 PM_net 对 OWL-S 过程模型进行建模后，OWL-S 的过程语义就可以由对应的 PM_net 网及其动态运行特性所决定，因此，我们可以通过分析 PM_net 网的相关特性来对

OWL-S过程语义的一致性问题进行分析。基于上述思想，下面本文以PM_net为基础，对OWL-S过程语义一致性问题进行讨论。

定义5(过程数据依赖一致性) 在OWL-S中，过程 p 被认为是数据依赖一致的，当且仅当过程 p 所有的输入参数均得到满足，且与其来源保持一致。

由上述定义可知，数据依赖一致性对过程输入参数的约束包含两个方面：首先，对于过程的每个输入参数，都要有一个数据流来说明其来源或者由用户输入；其次，对于每个与过程输入相关的数据流而言，数据流所涉及到的输入和输出的概念类型要保持一致。在PM_net中，由于数据流已经与PM_net进行了转化，因此我们可以通过对PM_net库所中Token的分析来对过程数据依赖一致性进行检验。

结论1 令 P_n 是与OWL-S过程模型对应的PM_net网，变迁 t 由过程 p 转化而成，那么过程 p 是数据依赖一致的，当且仅当变迁 t 满足：$\forall p' \in {}^{\cdot}t$，若库所 p' 由过程 p 的输入数据流转化而成，那么 $\sum_{t' \in {}^{\cdot}p'} E(t',p') = E(p',t)$。

证明：充分性。在PM_net中，变迁 t 的前继库所包含两种：由过程控制构造子转化所得到的库所，仅表示控制作用；由数据流转化得到的库所，用来放置表示输入输出参数流动的Token。由于前者中所包含的Token并不表示数据的输入和输出，因此对后者进行分析即可。由于过程 p 是数据依赖一致的，因此它所涉及数据流的输入和输出是一致的，依据3.3节转化到PM_net中后，假设变迁 t 所对应的前继库所为 p'，那么必有 $E(t',p') = E(p',t)(t' \in {}^{\cdot}p')$。在对转化得到的PM_net进行合并化简后得到 $\sum_{t' \in {}^{\cdot}p'} E(t',p') = E(p',t)$。必要性。若过程 p 不是数据依赖一致的，那么存在一个输入数据流，该数据流的输入输出不一致；或者存在一个输入参数无法得到满足。依据3.3节转化到PM_net中后，变迁 t 必然存在一个前继库所 p'，$E(t',p') < E(p',t)(t' \in {}^{\cdot}p')$，这与前提条件矛盾。得证！

由结论1可知，通过对PM_net中变迁的前继库所进行分析，即可对对应过程数据依赖一致性分行分析。

定义6(过程前提一致性) 在OWL-S中，过程 p 被认为是前提一致的，当且仅当过程 p 的前提条件不是永假式。

过程前提一致性是对过程前提条件的约束，它要求过程的前提条件总有可能为真，从而避免过程的死锁。在PM_net中，由于过程的前提条件已经等价映射为变迁的前提条件函数，因此可以通过分析PM_net网来验证过程前提一致性。

结论2 令 P_n 是与OWL-S过程模型对应的PM_net网，M_0 是 P_n 的初始标识，变迁 t 由过程 p 转化而成，那么过程 p 是前提一致的，当且仅当变迁 t 满足：$\exists M = M_p + M_s \in R(M_0)$，$val(TF_{pre}(t))_{M_s} = \text{true}$。

结论2表明，如果存在一个由初始标识可达的世界状态，使得变迁 t 的前提条件函数为真，那么该变迁所对应的过程 p 是前提一致的，反之亦然。根据PM_net的定义和OWL-S过程模型与PM_net的转换规则，很容易给出结论2的证明，由于篇幅限制，本文不再赘述。

定义7(过程可达性) 在OWL-S中，过程 p 是可达的，当且仅当存在一条从初始过程出发并经过过程 p 的执行路径。

对于OWL-S过程模型而言，过程可达性是一个重要的性质。一个过程如果不是可达的，那么在任何环境下该过程都不会被执行，这也说明OWL-S的过程语义存在问题，需要对过程逻辑和约束条件进行检查。由于OWL-S规范对其过程语义缺乏清晰的定义，因此采用原有的推理方法通常难以自动分析过程的可达性。在PM_net中，由于OWL-S过程模型已经进行了等价映射，因此可以通过对PM_net变迁活性的分析来验证过程的可达性。基于该思想，本文提出如下结论。

结论3 令 P_n 是与OWL-S过程模型对应的PM_net网，变迁 t 由过程 p 转化而成，那么过程 p 是可达的，当且仅当 P_n 存在一个初始标识 M_0，使得变迁 t 是活的，即 $\exists M = M_p + M_s \in R(M_0), M[t>$。

据PM_net的定义和OWL-S过程模型与PM_net的转换规则，很容易给出结论3的证明，本文对此不再赘述。结论3表明，我们可以通过对PM_net中变迁活性的分析来验证对应过程的可达性，因此可以采用着色Petri网中的分析工具(如可达树、迁移矩阵等)来间接分析过程的可达性。

定义8(过程有效性) 在OWL-S中，过程 p 是有效的，当且仅当存在一条从过程 p 出发并到达过程模型终点的执行路径。

在OWL-S中，过程可达性要求一个过程总要有机会被执行，而过程有效性则要求过程模型所描述的一段过程总要能够结束。在采用PM_net对过程有效性进行分析之前，我们首先借鉴工作流网[10,11]相关研究中结束状态的思想对PM_net终止标识进行定义。

定义9(PM_net的终止标识) 令 P_n 是PM_net网，M_0 是 P_n 的初始标识，那么标识 $M_l = M_{lp} + M_{ls}$ 被称为 P_n 的终止标识，当且仅当 $M_l \in R(M_0)$，并且若 $p = 0$，$M_{lp}(p) \neq 0$，否则 $M_{lp}(p) = 0$。

在PM_net的终止标识下，除了终点库所外，其他库所中的标识均为空，这表明PM_net中所有能被触发的变迁都已经触发完成，并且没有新的变迁能够继续触发。因此，终止标识可以认为是PM_net的终止运行状态，与OWL-S过程模型相对应，它可表明过程模型中的过程已经执行完毕，进入结束状态。

结论4 令 P_n 是与OWL-S过程模型对应的PM_net网，M_0 是 P_n 的初始标识，M_l 是 P_n 的终止标识，变迁 t 由过程 p 转化而成，那么过程 p 是有效的当且仅当 $\forall M \in R(M_0)$，若 $M[t>$，那么 $M_l \in M$。

由第3节可知，OWL-S的过程模型可与PM_net进行等价转换，其中过程映射为变迁，过程的结束状态可映射为PM_net中的终止标识，因此结论4实际上是转换规则的延伸。由结论4可知，我们可以通过PM_net的可达性分析来验证OWL-S中过程的有效性。

结束语 本文是关于OWL-S本体自动演化研究中的一个基础性研究工作，主要借鉴了Petri网严格的形式化语义和直观易懂的特性，提出了一种过程模型网(PM_net)对OWL-S的过程语义进行表示与分析，使得在OWL-S本体演化过程中，机器能够自动发现OWL-S过程模型在修改后所出现的语义不一致，并提供候选的修补方案供知识工程师选择，从而修复过程模型的语义一致性。

本文目前的研究主要集中在过程模型网的定义、OWL-S

过程模型与 PM_net 的转化以及基于 PM_net 的 OWL-S 过程语义一致性分析方法上。在未来的工作中，我们将重点对 PM_net 的性质进行研究，例如如何将传统着色 Petri 网的分析方法(如活性、有界性等)应用到 PM_net、如何分析 PM_net 的可靠性等。同时，我们也将进一步研究如何基于 PM_net 实现 OWL-S 过程模型的语义一致性修复，从而实现 OWL-S 本体的自动演化。

参 考 文 献

[1] McIlraith S, Son T C. Semantic Web Services[J]. IEEE Intelligent Systems, 2001(Special Issue on the Semantic Web)

[2] Martin D, Barsfin M. OWL-S: Semantic Markup for Web Services(V1.2)[Z/OL]. http://www.ai.sri.com/daml/services/owl-s/1.2, 2006

[3] Narayanan S, McIlraith S. Analysis and Simulation of Web Services[J]. Computer Networks: The International Journal of Computer and Telecommunications Networking, 2003, 42(5): 675-693

[4] 蒋运承，史忠植. OWL-S 的形式语义[J]. 计算机科学，2005，32(7)：5-7，16

[5] 李景霞，肖政，侯紫峰. 基于标签 Petri 网的 OWL-S 建模与分析[J]. 计算机工程，2007，33(7)：8-10

[6] 李景霞，侯紫峰，基于颜色 Petri 网的 Web 服务组合建模及应用[J]. 计算机应用研究，2006(9)：149-151

[7] Stojanovic L. Methods and Tools for Ontology Evolution [D]. University of Karlsruhe, 2004

[8] Haase P, Stojanovic L. Consistent Evolution of OWL Ontologies [C]// *Proceedings of the Second European Semantic Web Conference*. Heraklion, Greece, 2005

[9] 吴哲辉. Petri 网导论[M]. 北京：机械工业出版社，2006

[10] van der Aalst W, van Hee K. Workflow Management: Models, Methods, and Systems[M]. USA: MIT Press, 2002

[11] van der Aalst W M P. Verification of Workflow Nets[C]// Proceedings of the 18th International Conference on Application and Theory of Petri Nets. Berlin: Springer-Verlag, 1997

(上接第 198 页)

设构件 R 具有端口 $P_1, P_2, \cdots, P_n$，构件 S 相应地具有端口 $P_1', P_2', \cdots, P_n'$，如果：$\forall i(1 \leqslant i \leqslant n): P_i \prec P_i'$，则构件 S 能够替换构件 R，并且用端口 $P_1', P_2', \cdots, P_n'$ 替换相应端口 $P_1, P_2, \cdots, P_n$。

(7) 构件双向替换分析

在构件进行组装、交易时，两个构件的型构、行为等同样可能不同，但仍然需要经常判定它们是否能够彼此相互替换进行使用。根据定义，能够运用进程之间的替换关系对此进行判定。于是存在如下检测法则：

设构件 R 具有端口 $P_1, P_2, \cdots, P_n$，构件 S 相应地具有端口 $P_1', P_2', \cdots, P_n'$，如果：$\forall i(1 \leqslant i \leqslant n): P_i \prec\succ P_i'$，则构件 S 与构件 R 能够相互替换，端口 $P_1', P_2', \cdots, P_n'$ 与端口 $P_1, P_2, \cdots, P_n$ 同时相互对应地进行替换。

(8) 端口连接兼容分析

在实现级别，通常并没有显式的连接件，连接件已经转化成了构件或者直接运用程序语言或中间件支持的连接件。这种情况之下，通常是构件之间的端口直接进行连接，相互协调进行交互。因此，必须判定两个端口能否协调地进行交互，而不是相互冲突，从而在组装阶段可发现可能存在的错误。根据定义，能够运用进程兼容关系进行判定。于是得到如下检测：

设两个构件端口的行为进程分别是 p, q，如果 $p \exists q$，则构件端口 p 与 q 能够直接连接，两者协调进行交互。

结束语 本文基于 π ADL 软件体系结构形式化描述方法，运用 π 演算的基本理论，结合软件体系结构领域的特定需求，定义了基本的概念和进程之间的新型关系，如兼容关系、替换关系等，形式化地给出和定义了 8 种体系结构的一致性分析方法，为运用 π 演算工具进行自动化分析打下了基础。运用该 8 种分析方法，能够检查体系结构规约中出现的某些错误，提高组装软件系统的质量。

π 演算只能描述系统的行为，相关的检测和分析仅仅限于系统行为方面一致性的检测和分析。进一步的研究工作包括对 π ADL 进行扩展以描述系统的更多属性并进行更加严格的检测，如时间属性的扩展和性能分析。

参 考 文 献

[1] 梅宏，申峻嵘. 软件体系结构研究进展[J]. 软件学报，2006，17(6)：1257-1275

[2] Allen R. A *formal approach to software architecture*[D]. Pittsburgh: Carnegie Mellon University, May 1997

[3] AT&T. Best Current Practices: Software Architecture Validation[R]. AT&T. 2004

[4] Kazman R, Bass L, Abowd G, et al. SAAM: A method for analyzing the properties of software architecture[C]// Proceedings of the 16th International Conference on Software Engineering. Sorrento. Italy, May 2001: 81-90

[5] Mario R, Barbacci S, Jeromy C, et al. Steps in an architecture tradeoff analysis method: quality attribute models and analysis [R]. CMU/SEI-97-TR-029. Carnegie Mellon University, 1997

[6] Debra J R, Alexander L W. Software testing at the architecture level[C]// Proceedings of the ISAW. 2005

[7] 云晓春，方滨兴. 基于构件设计的正确性验证[J]. 小型微型计算机系统，1999，20(5)：330-334

[8] Talor R N, Medvidovic N, Anderson K M, et al. A component- and message-based architectural style for GUI software [J]. IEEE Transactions on Software Engineering, 2001, 22(6): 390-406

[9] Shaw M, DeLine R, Klein D V, et al. Abstractions for software architecture and tools to support them[J]. IEEE Transactions on Software Engineering, 2000, 21(4): 314-335

[10] Magee J, Dulay N, Eisenbach S, et al. Specifying distributed software architectures[C]// Proceedings of the Fifth European Software Engineering Conference, ESEC'95. September 1995

[11] Luckham D C, Augustin L M, Kenney J J, et al. Specification and analysis of system architecture using Rapide[J]. IEEE Transactions on Software Engineering, 1995, 21(4): 336-355

[12] 任洪敏. 基于 π 演算的软件体系结构形式化研究[D]. 上海：复旦大学，2003

[13] Milner R. Communicating and Mobile Systems: the π-Calculus [M]. Cambridge: Cambridge University Press, 1999