文章编号:1001 - 9081(2007)07 - 1720 - 05

一种支持条件分支的语义 Web 服务组装方式及执行

江 岭,崔光佐
(北京大学 计算机系, 北京 100871)
(jiangling@ pku. edu. cn; cgz@ pku. edu. cn)

摘 要:Web 服务组装的目的是使多个 Web 服务协同工作以完成任务。由于简单的顺序控制结构的组装不能满足较为复杂的协同工作需求,介绍了一种支持条件分支控制结构的语义 Web 服务组装方式。用户利用该组装器,可在领域知识层进行可视化组装。该基于知识库的执行引擎支持带有条件分支控制结构的组装结果的执行。

关键词:语义;Web 服务组装;条件分支;执行引擎
中图分类号: TP182; TP311.5 **文献标志码**:A

Semantic Web service composition supporting If-Then-Else control structure

JIANG Ling, CUI Guang-zuo
(*Department of Computer Science and Technology, Peking University, Beijing* 100871, *China*)

Abstract: Web service composition enables the cooperation of multiple Web services to finish certain task. Because compositions with simple sequence control structure cannot satisfy the need of more complicated cooperation. A way of semantic Web service composition which supports the If-Then-Else control structure was proposed. Using the introduced composer, users can do visualizable composing at domain knowledge level. And the execution engine based on knowledge base supports the execution of composition result with If-Then-Else control structure.

Key words: semantic; Web service composition; If-Then-Else; execution engine

0 引言

当今的互联网,应用程序之间的通信越来越常见。Web Service 提供了异构应用之间的编程接口[1]。为了充分发挥 Web Service 的潜力,使多个 Web Service 可以协同完成某件任务,Web Service 的组装成为了一个研究课题。Web Service 组装为开发者提供了面向 Service 构建应用的能力[2]。

语义 Web 旨在为应用、企业和团体之间共享和重用数据提供一种公用的框架[3]。此公用框架一旦建成,便可大大节省异构应用及企业、团体之间的沟通成本。语义 Web Service 利用公用的概念对 Web Service 进行语义描述。利用语义,Web Service 可被准确地发现,领域人员能够基于领域知识进行 Service 组装,机器能够根据语义描述进行 Service 的自动组装,这些都使 Web Service 能够充分发挥其潜力。

在 Web Service 的组装中,由一些 Web Service(可能是原子 Service,也可能是复合 Service)组合起来,能够完成某个复合任务的 Service 叫作复合 Service,参与组装的 Web Service 称为组件 Service。复合 Service 的调用者与其组件 Service 交互的细节,需要在复合 Service 的描述中指定。复合 Service 的描述不仅要指定参与组装的组件 Service 之间的数据传递和时序关系,还应指定调用各组件 Service 的控制结构。OWL-S,一个基于 OWL(Ontology Web Language)语言的描述 Web Service 本体的规范,规定了复合 Service 的 8 种控制结构[4],分别是:Sequence, Split, Split + Join, Any-Order, Choice, If-Then-Else, Iterate, Repeat-While and Repeat-Until。现有的对 Web Service 组装的研究,一般局限于顺序结构(Sequence)这种最简单的控制结构。顺序控制结构将组件 Service 以一维线性方式组合起来完成复合任务;这种简单的组合方式可能将满足不了组件 Service 较复杂的协同工作的要求。

带有条件分支(If-Then-Else)结构的复合 Service 的组装过程中对条件的描述,及其执行过程中对条件的设置和判断,都需要语义的支持。在带有条件分支的 Web Service 的组装中,能很好地体现语义 Web Service 的优势,因此本文将条件分支结构作为支持复杂控制结构的 Web Service 组装的切入点。在真实的世界中,Web Service 运行的环境是不可预测的,即 Web Service 的调用者不能在 Web Service 执行之前事先预知 Web Service 的执行效果。仅支持顺序控制结构的复合 Web Service 只能在可预测,即每个组件 Service 的执行效果在 Service 组装阶段即可预知的环境中正确执行。支持条件分支结构的复合 Web Service 将能够在不可预测,即组件 Web Service 的执行效果不能在 Service 组装时预知的环境中执行;本文同时假定执行引擎对于 Web Service 的执行环境是完全可观察的。

本文的实验工作基于 OntoComposer(原名 Pico-Composer)平台[5]。OntoComposer 是一个混合式的语义 Web 服务组装平台,具备语义 Web Service 发现、自动组装、手动组装和执行功能。OntoComposer 基于智能规划算法,能根据用户指定的目标自动生成行动的线性序列,并将行动对应到相应的 Web Service。OntoComposer 原仅支持顺序结构的 Web Service 组装和执行。本文作者继承 OntoComposer 原作者的工作,对

收稿日期:2007 - 01 - 22;**修回日期**:2007 - 03 - 06。

作者简介:江岭(1981 -),男,四川资阳人,硕士研究生,主要研究方向:语义 Web 服务组装、智能规划; 崔光佐(1967 -),男,河北高阳人,教授,博士,主要研究方向:知识表示与推理、智能领域工程。

OntoComposer 加入对条件分支结构的 Web Service 手动组装和执行的支持,并与 OntoComposer 平台原有的手动组装有机地结合起来。

本文所介绍的基于 OntoComposer 平台的语义 Web Service 组装方式,主要具有如下特点:1)支持条件分支结构的复合 Web Service 的手动组装和执行; 2)在领域知识层进行组装,组装人员不需要 Web Service 技术方面的知识; 3)可视化组装,进行组装的领域人员不需要进行编程。

1　Web 服务的语义

Web Service 引入语义描述,成为语义 Web Service。语义的引入使 Web Service 可被自动发现、参与自动组装,并保证复合 Web Service 的正确执行。

1.1　共享概念和本体

语义 Web Service 是语义网(Semantic Web)的一个应用。在语义网中,信息被显示地赋予含义,这个含义可被机器自动处理[6]。要使信息具有机器可理解的语义,需要信息的传播者和接受者具有公有的概念。对于语义 Web Service 来说,服务的提供者、服务的聚合者和服务的使用者都需要维护相同的概念。

在语义网中,共享概念由本体来表达。本体是共享概念模型的、明确的、形式化规范说明[7]。在本体中,最基本的概念基于计算机能处理的数据类型来定义,更复杂的概念基于已建立的概念来定义。本体的广泛运用需要一个重要的先决条件,即一个描述本体并使得它们能够进行信息交换的语言标准。OWL(Web Ontology Language)即是这样一种语言。由于 OWL 语言本身定义了具有形式化语义的一些附加词汇,它表达 Web 上的内容,比基于 XML、RDF 和 RDF-S(RDF Schema)来表达更加便于机器处理[6]。对应于描述逻辑的 TBox 和 ABox,OWL 描述的本体由公理和事实组成[8]。前者描述概念及其关系的术语公理集,后者描述个体实例的断言公理集。本文利用 OWL 语言来表达本体。

1.2　Web 服务的语义描述

在本文的 Web Service 组装过程中,每一个参与组装的 Service 都被抽象为行动(Action)。行动由 Service 的输入(Inputs)、输出(Outputs)、前提(Preconditions)、效果(Effects)完全定义。Web Service 的输入是指外部调用 Web Service 的操作时需要提供的数据;输出则是 Web Service 操作结束后返回给外部调用者的数据;前提是指 Web Service 的操作被调用前要求为真的语义陈述;效果是 Web Service 的操作被调用后为真的语义陈述[9]。

在语义 Web Service 中,Web Service 的输入、输出、前提、效果都用语义加以描述。输入、输出参数的含义用语义模型(本体)中定义的词汇(概念)描述,表达前提和效果的概念和谓词也来自于语义模型(本体)。图 1 显示了 Web Service 的语义描述与本体所建立的语义模型之间的关系。

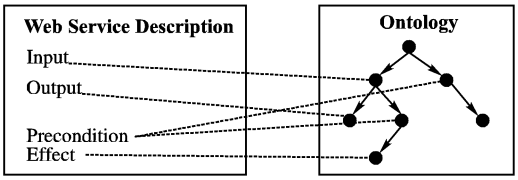

图 1　Web Service 的语义描述与本体的关系

1.3　Web 服务的条件效果

Web Service 的每一个效果(Effect)包括两个部分,分别为条件表达式 E1 和 E2。其执行时的结果为,当 Web Service 调用结束时,如果 Web Service 的上下文满足条件表达式 E1 所表达的语义,则在知识库中设置条件表达式 E2 为真,否则不作条件设置。E1 可为永真表达式;当 E1 不为永真时,这样的效果就是一个条件效果。

2　Web 服务组装

OntoComposer 组装的特色之一是最终用户在领域知识层进行组装,将 Web Service 抽象为由输入、输出、前提、效果(IOPE)描述的行动,从而摒弃了 Web Service 繁复的技术细节和内部模型。在领域知识层,条件分支控制节点的条件用 RDF 三元组表示; 在实现层,条件用可嵌入 OWL-S 的 SWRL[10] 语言表达。领域知识层与实现层的关系如图 2 所示。

利用对 Web Service 的抽象行动进行组装有下面两个优点:1) 组装生成的复合 Web Service 被抽象为由 IOPE 描述的行动之后,能与原子 Service 一样参与组装;这样,就使大型的 Web Service 组装成为可能,大型的复合 Service 可由其他规模较小的复合 Service 组装而成。2) 领域人员在进行组装时,所面对的是抽象的行动,因此只需要具有相关的领域知识,不需要具有 Web Service 技术相关的知识。

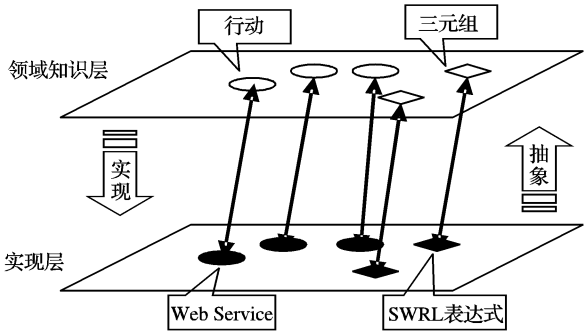

图 2　领域知识层与实现层的关系

2.1　RDF 三元组和 SWRL 语言

资源描述框架(Resource Description Framework,RDF)[11] 的数据模型基于三元组来描述资源的属性和关系。RDF 三元组形如〈p, s, o〉,其中 p 是一个谓词,s 是一个主体,o 是一个客体。主体或客体可为空。每个形如〈p, s, o〉的三元组表达了一个断言,即说明了主体 s 的一个属性 p 具有值 o,或主体 s 与客体 o 具有 p 所表达的关系。本文组装的条件表达式用 RDF 三元组的集合表示,其含义是集合中 RDF 三元组表达的断言的总和,即它们的逻辑合取。

当条件表达式嵌入 OWL-S 描述时,三元组用 SWRL 来进行描述。SWRL 是一种可嵌入 OWL-S 的逻辑表达语言[10]。在本文的 Web Service 的 OWL-S 描述中,其前提、效果和分支条件都用 SWRL 表达。

2.2　Web 服务组装的交互过程

用户手动组装的交互过程如下:

1) 用户选择当前组装所需要的领域本体并导入。

2) 用户可新建组装图,也可以打开之前保存的组装图继续进行组装或修改。

3) 用户在组装图上插入行动节点,代表一个组件

Service;或插入条件分支控制节点,在组装图上引入分支。插入条件分支节点时,需指定分支的条件。分支条件表达式是RDF 三元组的集合。其中三元组的谓词是在领域本体中定义的属性,主体和客体是领域本体中定义的概念或实体。用户通过添加三元组进行条件编辑。由于组装器读入并解析了领域本体,在添加三元组时,用户可通过界面选取三元组的谓词、主体、客体,也可手动输入。

3) 指定参数绑定。用户可将某行动的特定输出参数与另一行动的特定输入参数绑定,也可以将某些行动的特定输出指定为组装得到的复合 Service 的输出。

4) 用户可保存当前组装图。

5) 将已完成的组装图转换为 OWL-S 描述文件保存。OWL-S 描述的复合 Service 中记录了组装图所指示的时序、分支控制结构和参数绑定的信息。

2.3 可视化组装器的实现

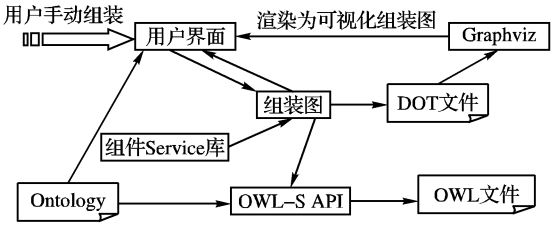

图 3 可视化组装器模块图

图 3 中的组装图记录了当前的组装结果(未完成的或已完成的)。组装结果是一个由起始节点、终止节点、行动节点、条件分支控制节点以及参数绑定节点组成的有向图,指定了复合 Service 调用组件 Service 的流程;行动节点对应了组件 Service 库中的 Web Service。用户通过用户界面操作组装图;用户设置条件分支控制节点的条件时需要来自本体中的概念和实体。组装图可转化为 DOT 文件描述;DOT 文件可通过 Graphviz[12] 渲染为可视化的图形。组装图通过调用马里兰大学 MIND 实验室 MINDSWAP 组开发的 OWL-S API[13],转化为 OWL 语言描述的 Web Service;在 OWL-S 描述中,组装图的行动节点转换为对应的 Web Service,条件分支控制节点的条件用 SWRL 表示。

3 复合 Web 服务执行

对于顺序结构的复合 Web Service 的执行,执行引擎只须根据 Web Service 的 OWL-S 描述,顺序地调用各组件 Service,并在各组件 Service 之间传递数据。而对于支持条件分支结构的复合 Web Service,执行引擎需要维护知识库,并利用本体中的概念来判断当前环境的状态,以决定组件 Service 的调用顺序。执行引擎是一个智能体(Agent),它的知识库维护着当前环境状态的描述。知识库采用封闭世界假设,即知识库中未提到的条件被假定为假。执行引擎根据 OWL-S 描述调用 Web Service,当执行时遇到条件分支控制构造,执行引擎判断知识库的状态描述是否使条件分支的条件为真,根据判断结果选择下一步的执行动作。执行引擎的知识库与 Web Service 描述和 Web Service 的组装享有相同的本体。图 4 显示了执行引擎、环境和本体之间的关系。

本文假设执行引擎与之交互的环境是完全可观察的,也没有别的智能体对环境做出改变。执行引擎的知识库维护着 RDF 三元组的集合。知识库最初存储着环境的基本知识和初始状态,即那些由领域本体的事实部分所表达的关系和属性。执行引擎对环境状态的感知来自所调用的 Web Service 的输出和效果。被调用的 Web Service 由 SWRL 表达效果的 $E1$ 和 $E2$ 部分分别被解析为 RDF 三元组集合 $T1$ 和 $T2$;执行引擎的推理器判断当前知识库中存储的三元组集合 T 是否蕴含 $T1$,若是,则改变知识库,使新的知识库的三元组集合为 $T \cup T2$。执行引擎遇到 OWL-S 的条件分支控制构造时,它解析得到分支条件的三元组集合 C,如果当前知识库的三元组集合蕴含 C,则判断分支条件为真,否则为假。根据对分支条件的判断,执行引擎调用相应的组件 Web Service,作为对环境的反应。

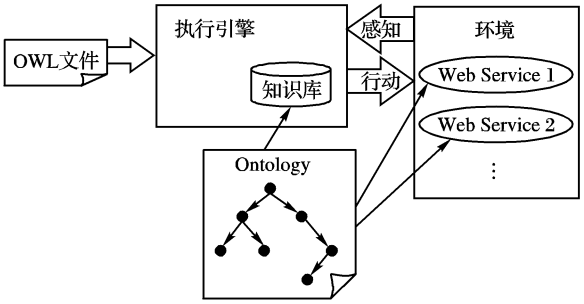

图 4 执行引擎、环境和本体的关系

OntoComposer 的执行引擎修改自 OWL-S API[13] 的执行引擎。OWL-S API 最新的 nightly build 版本支持 If-Then-Else 结构的复合 Web Service 描述,但因其不能在知识库中对 Web Service 的执行效果进行设置,故不能够很好地支持 If-Then-Else 结构的复合 Web Service 执行。本文对 OWL-S API 做了修改,使其能根据 Web Service 的执行效果对知识库进行设置,从而能很好地支持本文的工作,同时也与 OWL-S API 原有的功能有机结合起来。

4 示例和实验

4.1 示例

本文引入一个简单的示例。我们假设有两个由两家不同的公司提供的 Web 服务:在线购书公司提供的订书 Web 服务 OrderBookService,以及由快递公司提供的预订快递服务的 Web 服务 OrderExpressDeliveryService。作为客户,现在我们有这样的需求:由于希望订购一本书给朋友,所以同时需要在线购书公司提供的订书服务和快递公司提供的快递服务。但订书不是每次都能成功,即调用 OrderBookService 可能有两种效果:书库中有这本书,订书成功;或书库缺货,订书失败。因此,我们需要在订书成功后,调用 OrderExpressDeliveryService 预订快递服务;如果订书失败,则不预订快递服务。

我们将把这两个由两个不同公司提供的 Web Service 组装为一个复合的 Web Service,叫作 OrderBook&Express。这样,我们只需要调用 OrderBook&Express 这个复合的 Web Service,就可以满足上述的需求了。

对于这个示例,需要一个由 OrderBookService、OrderExpressDeliveryService 服务的提供者和服务的调用者共享的领域本体。在本例用到的本体的公理部分,包括服务的用户 User、订购的书 Book 等概念及其属性的定义,还包括用户与书的关系的描述,如某用户订购某书成功的概念 OrderBookSuccess。

如果调用 OrderBookService 订书成功,返回订书成功的信息和订单详情,否则返回订书失败的提示信息,因此

OrderBookService 的条件效果为,如果服务返回的信息不为订书失败的提示信息,则当前用户订书成功。

在 OWL-S 的过程模型对复合 Web Service 的描述中,复合过程由原子过程或复合过程通过控制构造(Control Construct)连接组成。用于构造复合过程的控制构造包括顺序(Sequence)、条件分支(If-Then-Else)等控制构造;顺序、条件分支等控制构造也由控制构造构成。有一种特殊的控制构造叫作 Perform,Perform 是对其他过程的调用。简言之,我们可以将任何复合过程视为一棵树,其非叶子节点是控制构造;不同的控制构造将其子节点用不同的方式组合起来;Perform 是位于叶节点的控制构造。复合 Web Service OrderBook&Express 的过程模型如图 5 所示,Sequence 节点指向其子节点的边上标记的序号表示 Sequence 结构执行时调用子结构的顺序,序号较小的边指向的结构先调用。If-Then-Else 节点也有两条边,其中一条边指向当条件分支的条件满足时调用的子结构,用"Then"标记;另一条边指向当条件不满足时调用的子结构,用"Else"标记。

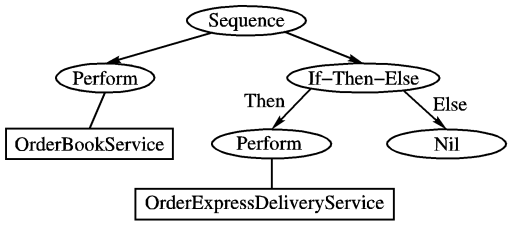

图 5　OrderBook&Express 过程模型

执行引擎执行复合 Web Service OrderBook&Express 时,首先调用其组件 Service OrderBookService(设输入为 User 类型的 usr 和 Book 类型的 bk),根据得到的输出判断是否订书成功,如果成功则在知识库中设置 OrderBookSuccess (usr, bk)为真。设条件分支控制构造的条件设置为 User 类型的 usr1 订购 Book 类型的 bk1 成功,执行引擎根据知识库判断 OrderBookSuccess (usr1, bk1)是否为真。如果为真,则调用 OrderExpressDeliveryService。

4.2　实验

我们利用 Resin-2. 1. 16[14] 作为 Web 服务器,Axis1. 2. 1 作为组件 Web 服务的发布工具。发布的组件 Web 服务利用 OWL-S API 的 WSDL2OWLS 工具转化为 OWL-S 描述。

用户指定当前的领域本体如图 6 所示。

OntoComposer 将已有的组件 Service 抽象为行动(Action),用户以行动为单位进行可视化组装。图 8 显示了可视化组装图。图 7 为条件编辑对话框,可利用当前本体中的概念对条件节点进行编辑。条件以谓词、主体、客体三元组表示。

OntoComposer 过程模型组装完成后须指定数据流。图 8 的界面右侧白板内显示了完成数据流指定的组装图。图中矩形框为行动节点,菱形框为条件分支节点,田字矩形框为数据流指定节点。与图中 P 节点绑定的输出将作为复合 Web Service 的输出。从图中可见 OrdeBookService 和 OrderExpressDeliveryService 的实例的输出都与复合 Web Service 的输出绑定。

Web Service 组装图完成后,可转化为 OWL-S 描述保存,保存后的复合 Web Service 可作为组件 Service 参与新的组装。在服务端的组件 Web 服务可用的情况下,OntoComposer 的执行模块可调入复合 Service 的 OWL-S 描述并执行。图 8 显示了 OrderBook&Express 的执行情况。界面左侧的输入框为复合 Web Service 的输入。填好输入项后执行,在界面左下方显示了执行的输出。

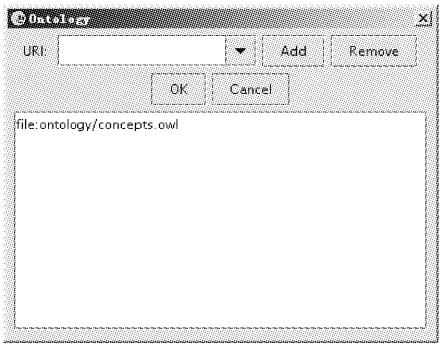

图 6　OntoComposer 领域本体选择和导入对话框

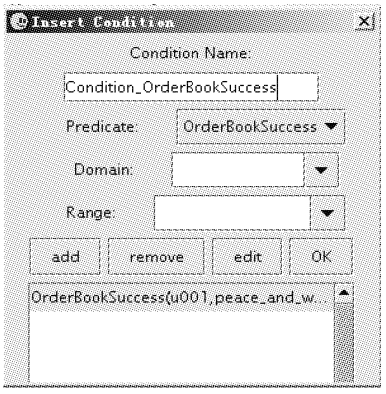

图 7　条件编辑对话框

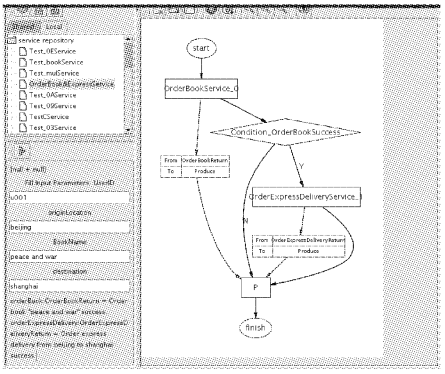

图 8　OrderBook&Express 的组装图、数据流指定,执行复合 Web Service

用 OWL-S API 提供的过程监视器对执行过程进行监视。如果对订阅书名输入框填入库中没有的书,如"from earth to moon",控制台显示只调用了 OrderBookService;当填入库中正常供应的书,如"peace and war",控制台显示调用 OrderBookService 后,由于订阅成功,继续调用 OrderExpressDeliveryService。对复合 Web Service 执行的实验成功。

5　结语

本文介绍了一种支持条件分支控制结构的语义 Web Service 组装方式。这主要包括两个方面:组装和执行。要使 Web Service 的组装和执行支持条件分支控制结构,重点在于对条件的描述、判断和对效果的设置。而对条件的描述和判断,关键在于语义的运用。领域本体所提供的共享概念使语义的表达和判断成为可能。本文所介绍的 Web Service 执行

引擎由于带有知识库,能够将效果的设置反映到知识库中去。

OntoComposer 平台将领域本体的解析、语义 Web Service 的可视化组装和 Web Service 的 OWL-S 描述的执行有机地结合起来。在领域知识层进行可视化组装是本文所介绍的组装方式的另一个特点。组装人员并不需要关于 Web Service 技术的知识,也不需要具有编程能力。而目前较流行的另一组装工具,OWL-S Editor[15],由于在 Web Service 的实现层进行组装,因此需要有 Web Service 和 OWL-S 的知识,并且组装过程较为复杂。如果单独用 OWL-S API 进行组装,则需要进行编程。

致谢 感谢我的导师崔光佐教授的悉心指导和刘杨师兄为 OntoCompser 所做的出色工作。

参考文献:

[1] W3C. Web Service Activity [EB/OL]. [2006 - 11 - 01]. http://www.w3.org/2002/ws/#drafts.

[2] MILANOVIC N, MALEK M. Current solutions for web service composition [J]. IEEE Internet Computing, 2004, 8(6):51 - 59.

[3] W3C. Semantic web activity [EB/OL]. [2006 - 12 - 31]. http://www.w3.org/2001/sw/.

[4] MARTIN D, BURSTEIN M, HOBBS J, *et al*. OWL-S: semantic markup for web services [EB/OL]. [2004 - 11 - 30]. http://www.w3.org/Submission/OWL-S/.

[5] 刘杨. 基于本体的混合式语义 Web 服务组装机制及原型实现[D]. 北京:北京大学, 2006.

[6] MCGUINNESS D L, VAN HARMELEN F. OWL web ontology language overview[EB/OL]. [2004 - 02 - 28]. http://www.w3.org/TR/owl-features/.

[7] 宋炜,张铭. 语义网简明教程[M]. 北京: 高等教育出版社, 2004.

[8] 梅婧,刘升平,林作铨. 语义 Web 语言的逻辑分析[C]// 第二届全国智能信息网络学术会议论文集. 谭铁牛. 北京:科学出版社,2004.

[9] AKKIRAJU R, FARRELL J, MILLER J, *et al*. Web Service Semantics - WSDL-S[EB/OL]. [2005 - 11 - 01]. http://www.w3.org/Submission/WSDL-S/.

[10] HORROCKS I, PATEL-SCHNEIDER P F, BOLEY H, *et al*. SWRL: A semantic web rule language combining OWL and RuleML[EB/OL]. [2004 - 03]. http://www.w3.org/Submission/SWRL/.

[11] KLYNE G, CARROLL J J. Resource description framework (RDF): Concepts and abstract syntax[EB/OL]. [2004 - 02 - 01]. http://www.w3.org/TR/rdf-concepts/#section-data-model.

[12] Graphviz-graphvisualizationsoftware[EB/OL]. [2006 - 12 - 01]. http://graphviz.org/.

[13] Maryland Information and Network Dynamics Lab. Semantic web agents project. OWL-S API[EB/OL]. [2006 - 12 - 01]. http://www.mindswap.org/2004/owl-s/api/.

[14] RESIN C[EB/OL]. [2006 - 12 - 01]. http://www.caucho.com/.

[15] The OWL-S Editor [EB/OL]. [2006 - 12 - 01]. http://owlseditor.semwebcentral.org/.

[16] World Wide Web Consortium [EB/OL]. [2006 - 12 - 01]. http://www.w3.org/.

[17] TRAVERSO P, PISTORE M. Automated composition of semantic web services into executable processes [C]// Proceedings of ISWC'04. Hiroshima, Japan: [s.n.], 2004:380 - 394.

[18] SIRIN E, HENDLER J, PARSIA B. Semi-automatic composition of Web services using semantic descriptions[C]// Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS 2003. 2002:17 - 24.

[19] RAO JH, SU XM. A survey of automated web service composition methods [C]// Proceedings of Semantic Web Services and Web Process Composition workshop. CA, USA: Springer, 2004.

[20] (美) RUSSELL S, NORVIG P. 人工智能——一种现代方法[M]. 2 版. 姜哲,张敏,杨露,等译. 北京:人民邮电出版社, 2004.

[21] SRIVASTAVA B, KOEHLER J. Web service composition - current solutions and open problems[C]// ICAPS 2003 Workshop on Planning for Web Services. Trento, Italy: [s.n.], 2003.

[22] COSTA LD, PIRES P, MATTOSO M. Automated composition of web services with contingency plans[C]// IEEE International Conference on Web Services. USA: IEEE Computer Society, 2004.

[23] Description Logic[EB/OL]. [2006 - 12 - 01]. http://dl.kr.org/.

[24] Axis[EB/OL]. [2006 - 12 - 01]. http://ws.apache.org/axis/.

(上接第 1716 页)

有了上面的定理,判断 DTD 是否具有一致性的算法就非常直接简单了:首先对 DTD D 进行遍历,判定 DTD 是否存在“|”操作符,若存在,则对 DTD D 进行分解,使得分解后的 DTD 不受“|”的约束。接着对分解的 DTD 创建 DTD 图,然后对每一个DTD图从根节点开始沿着强约束边做深度优先遍历,每个第一次访问的节点做“visited”的标记,当有节点访问两次时,就表明出项了闭合回路,那么 D 就是不一致的,否则,D 就具有一致性。

3 结语

鉴于在 DTD 的定义中存在不合理的结构的可能,本文提出了 DTD 一致性的概念,对能够导致 DTD 不一致的各种因素进行了分析,给出了 DTD 一致性的判定条件,这些研究能够在快速判断 DTD 结构设计是否合理等方面提供了有效的帮助。由于 XML Schema 作为一种新的结构模式,具有 DTD 不可比拟的优越性,所以今后的工作是 XML Schema 一致性和 XML 键及 XML 结构模式相互作用的完整性约束。

参考文献:

[1] FAN W, LIBKIN L. On XML integrity constraints in the presence of DTDs[C]// Proceedings of ACM Symposium on Principles of Database Systems. New York: ACM Press, 2001: 114 - 125.

[2] ARENAS M, FAN W, LIBKIN L. On verifying consistency of XML specification[C]// Proceedings of 21st ACM Symposium on Principles of Database Systems. New York: ACM Press, 2002: 259 - 270.

[3] ARENAS M, FAN W, LIBKIN L. What's hard about XML schema constraints?[C]// Proceedings of 13th International Conference on Database and Expert Systems Applications. Berlin: Springer-Verlag, 2002: 269 - 278.

[4] BUNEMAN P, DEVIDSON S, FAN W, *et al*. Reasoning about keys for XML[C]// Database Programming Languages, the 8th International Workshop. Frascati: Springer - Verlag, 2001:133 - 148.