Web 服务之间数据关联的建模与应用

顾志峰 李涓子 胡建强 许 斌 王克宏
(清华大学计算机科学与技术系 北京 100084)

摘 要 数据关联是 Web 服务的输入输出数据之间存在的对应关系,它反映了服务与服务在业务逻辑上的相关性,这种相关性信息对于服务组装、服务发现等任务具有重要的意义.已有的数据关联建模方法根据服务接口所使用的数据模型或本体模型进行机器推理间接地表达数据关联,这种表达方式在表达能力上存在局限性.针对这个问题,文中借鉴超级链接的思想,提出了显式数据关联的概念,通过静态声明的方法将数据关联直接表达出来.文中定义了显式数据关联的模型,给出了该模型的 XML 实现.在此基础上,文中讨论了显式数据关联的应用,并给出了利用显式数据关联优化 WS-Challenge 2007 组装算法的一个应用案例,实验结果表明,该优化方法在处理具有复杂继承关系的数据集时能够有效提高算法的效率,最终该算法在比赛中夺得组装性能冠军.

关键词 数据关联;Web 服务;WSDL;服务描述;WS-Challenge
中图法分类号 TP331

Modeling and Application of Data-Links Among Web Services

GU Zhi-Feng LI Juan-Zi HU Jian-Qiang XU Bin WANG Ke-Hong
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract Data correlations are relationships among I/O documents of Web services. They reflect the correlations among services on business logics, and are important to service tasks such as service composition, service discovery, and etc. Existing modeling methods of data correlation express data correlations indirectly by reasoning on data models or ontology models used by service interface definitions, and they have limitations on expressiveness. In order to solve this problem, referring to the idea of hyperlinks, this paper proposes the concept of Explicit Data-Link, which expresses data correlations as static explicit declarations. It first gives the abstract model of explicit data-link and its XML implementation. Then, the application of explicit data-link is discussed in detail. A study case that makes use of explicit data-link to optimize the composition algorithm for WS-Challenge 2007 is given. The experiment results show that this optimization method is effective when dealing with data sets that have complex inheritance structures, and the algorithm wins the championship of performance in the competition.

Keywords data correlation; Web service; WSDL; service description; WS-Challenge

1 引 言

从软件复用的角度来讲,Web 服务定义了一种以 XML,HTTP 等开放标准为基础的软件复用框架,每个服务都是一个可复用的访问接口,将数据或业务逻辑等资源提供给调用者.虽然存在一类 Web 服务,其提供的访问接口十分独立,与其它服务没有

收稿日期:2008-04-13;最终修改稿收到日期:2008-07-02.本课题得到国家"九七三"重点基础研究发展规划项目基金(2007CB310803,2003CB317007)、国家"八六三"高技术研究发展计划项目基金(2007AA010306)、国家自然科学基金(90604025)与中国博士后科学基金(20070410061)资助.顾志峰,男,1979 年生,博士研究生,研究方向为服务的发现与组装技术.E-mail: gu_zhifeng@mails.tsinghua.edu.cn.李涓子,女,1964 年生,博士,副教授,研究方向为语义 Web 与 Web 服务、文本挖掘与知识发现.胡建强,男,1971 年生,博士后,主要研究方向为分布对象技术和软件工程.许 斌,男,1973 年生,博士,副教授,主要研究方向为 Web 服务和语义 Web.王克宏,男,1941 年生,教授,博士生导师,研究领域为网络计算与知识处理.

业务逻辑上的联系,比如提供天气查询的 Web 服务,但更普遍的情况是多个 Web 服务隶属于同一个应用系统,其中每个服务是对该系统中可复用资源的抽象,例如某个物流管理系统中定义的多个 Web 服务.在后一种情况下,服务与服务之间在业务逻辑上往往存在比较紧密的相关性.这种相关性基本上可以分为两类:(1) 时序约束.即服务的调用顺序必须满足一定的时序关系,这种约束实质上就是业务流程的体现;(2) 数据关联.即某个服务的输出数据与另一个服务的输入数据之间存在的关联,数据关联的关键在于输入数据与输出数据的语义必须匹配.服务之间的这两类相关性对于服务组装、服务发现等任务具有重要的意义,而 Web 服务的描述语言 WSDL 无法描述这两类相关性信息,因此,已有的文献提出了各种方法对这两类相关性信息进行建模.

对于时序约束,主要利用有限状态机(FSM)和 Petri-Net 等过程建模方法进行描述,比如文献[1]和[2]使用 FSM 描述服务的时序行为;文献[3]使用 Petri-Net 对服务内部和外部特征进行描述.

对于数据关联,主要通过两种方法描述:(1) 类型系统,即通过类型匹配构造数据关联.这种方法的缺陷十分明显,即它可能引入错误的数据关联(类型相同的数据不一定具有相同的语义,比如产品 ID 和产品名称完全可能同是字符串类型),也无法在异构数据之间构造数据关联;(2) 本体标注,即通过对数据模型进行语义本体标注来明确数据的语义,同时解决异构数据之间的映射问题,比较有代表性的是 METEOR-S 框架[4-6]中提出的 WSDL-S 以及几个知名的语义 Web 服务框架:OWL-S[7-8]、WSML[9-10]、SWSF①等.这种方法原则上可以解决上述的两个问题,但是该方法的核心假设,即一个全局共享的本体库,在实际中往往很难实现.

同时,上述这两种数据关联建模方法还存在一个共同的问题,即数据关联在这两种方法中都隐含于服务接口所使用的数据模型或本体模型中,需要通过机器推理的方法获取.这样虽然可以保持服务描述的独立性(因为机器推理过程只依赖于数据模型或本体模型,与服务无关),却限制了其表达能力,这主要表现在几个方面:(1) 可能引入大量无意义的数据关联;(2) 无法定制数据关联的语义;(3) 无法表达服务实例之间的数据关联;(4) 无法有效支持需要消息转换的数据关联.从另一个角度来讲,这种隐含的数据关联反应的是数据模型或者本体模型内部的语义关系,并不能很好地刻画服务之间的数据关联,因此在为服务之间数据关联建模的时候有必要将服务引入数据关联模型中.

本文研究服务之间的数据关联.针对上述问题,本文提出显式数据关联的概念,将服务引入数据关联模型中,并通过静态声明的方法将数据关联直接表达出来.

显式数据关联借鉴了 Web 页面中的超级链接的思想.我们将 Web 服务与 Web 页面进行对比:Web 服务抛弃了 Web 页面中的表示层信息,同时使用 XML 规范了数据格式,使它成为一种适合于应用程序与应用程序之间通信的交互方式.毫无疑问,超级链接是 Web 页面中最重要的元素之一,它反映了页面与页面之间在业务逻辑上的相关性,对于 Web 服务而言,也需要类似的概念来描述服务与服务在业务逻辑上的相关性,但是已有的 Web 服务协议栈中并没有相应的规范用于描述服务之间的"超级链接".

例如,在一个 Web 应用中,页面 A 返回一个产品的列表,其中每个产品定义了一个指向页面 B 的超链,用于显示产品的详细信息,产品的 ID 作为该超链的参数.在这个例子中,页面 A 返回结果中的超链反映了页面 A 和页面 B 之间的数据关联.如果我们将页面 A 和 B 分别映射到服务 A 和 B,那么这种数据关联就会丢失,因为已有的服务描述框架都缺乏有效的方法来定义这种数据关联.虽然我们可以在服务 A 的返回数据中添加类似超链的属性或标签来定义这个数据关联,但是这样做会破坏数据层与业务逻辑层的独立性,同时也增加了数据的冗余度.比较合理的做法是在服务描述中定义如上所述的数据关联.本文正是基于这样一个想法提出了显式数据关联的概念.

本文定义的显式数据关联模型允许可扩展的语义,即一个显式数据关联可以附加任意的数据来描述该数据关联所具有的特定语义,这意味着显式数据关联不仅可以用来表达多个服务在业务逻辑上的相关性,也可以作为一种通用的数据关联模型用于描述特定类型的数据关联.这极大地扩展了显式数据关联的应用空间,本文分别从服务组装、服务发现、经验知识和启发式信息 4 个方面讨论了显式数据关联的应用.

本文第 2 节给出服务及服务实例的抽象模型,在此基础上定义服务之间的显式数据关联模型;第 3 节给出显式数据关联的 XML 表示,并讨论如何与 WSDL 集成;第 4 节讨论显式数据关联的几个应用

① http://www.daml.org/services/swsf/

场景;第 5 节给出显式数据关联的一个应用实例,即如何利用显式数据关联优化 WS-Challenge 的自动组装算法;第 6 节简要讨论显式数据关联的生成与维护问题;第 7 节是本文的总结.

2　相关概念及定义

2.1　服务与服务实例

我们将服务定义成一个抽象的功能黑盒. 每个服务以一个结构化的文档作为输入,返回一个结构化的文档作为输出,形式化表示如下.

定义 1.　服务. 服务是一个二元组 $S=(I,O)$,其中 I 和 O 分别是输入输出文档的类型.

文档的类型是一个十分基本的概念,它在特定的名字空间中唯一地标识了一个概念,同时它也规定了数据的格式和语义. 对 Web 服务来讲,输入输出是 XML 文档,因此文档类型主要有 3 种定义方式,即 XML Schema、DTD 和 RELAX NG.

相比 WSDL 给出的服务模型,本文给出的服务模型略有简化. WSDL 为了兼顾面向文档与面向 RPC 的接口定义方式,在服务模型中加入了一些与传统对象模型类似的元素,使其看上去更像一个对象接口,而本文给出的定义是一个完全面向文档的服务模型.

一个服务可能有多个服务实例与之对应. 服务实例是服从某个服务接口定义的一个可访问的端点(endpoint),定义如下.

定义 2.　服务实例. 服务实例是一个二元组 $E=(S,L)$,其中 S 是一个服务,L 是访问入口.

我们认为服务与服务实例之间的关系如图 1 所示. 每个服务可能有多个实现,每个实现也可能有多个实例被部署,而实例之间可能共享数据源或者使用独立的数据源.

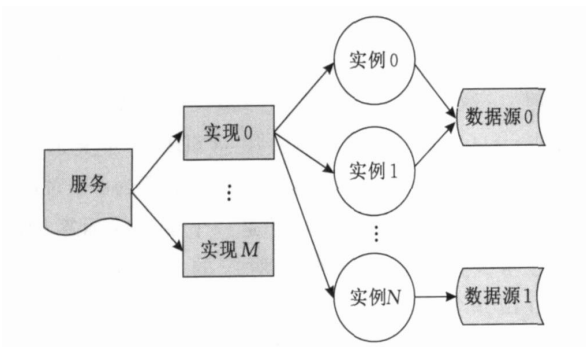

图 1　服务与服务实例

2.2　显式数据关联

显式数据关联描述两个服务之间的数据关联关系,形式化定义如下.

定义 3.　显式数据关联. 显式数据关联是一个四元组 $A=(S,T,M,X)$,其中 S 和 T 是两个服务或两个服务实例,S 称为源服务,T 称为目标服务,M 是一组映射,X 是一个权值.

该定义中 S 和 T 的意义十分明确,即 S 的输出文档与 T 的输入文档之间具有数据关联.

M 是一组映射,用来表达具体的数据映射关系,其成员具有如下形式:

$$p=f(d) \text{ 或 } p^{*}=f(d),$$

其中:d 是 S 的输出文档;p 是 T 输入文档中的某个属性,而 f 是映射规则. 我们允许 f 的返回值与 d 之间存在一对一或多对一两种数量关系,分别使用 = 和 $^{*}=$ 进行区分. 直观地讲,M 中的映射规则指定了如何将 S 输出文档中的数据转换成 T 输入文档所需要的数据元素. 以第 1 节中的例子为例,M 中应定义如下映射规则:

$$ProductID^{*}=ProductIDOf(Products).$$

在 Web 服务中,M 中的映射规则一般可以通过 XPath 或者 XSLT 表达.

显式数据关联的四元组中,X 是一个扩展元素. 在模型中,我们将其定义为一个权值,以此来表达其描述的数据关联的优先级,这仅仅是一个扩展实例,在具体实现时,X 可以是任意的数据,用于描述扩展的语义,在 3.2 节中会有进一步介绍.

3　显式数据关联的实现

第 2 节中我们给出了服务、服务实例以及显式数据关联的形式化定义,本节将介绍如何实现第 2 节中给出的这些定义,并与已有的 Web 服务规范相结合.

3.1　服务与服务实例的 WSDL 描述

服务与服务实例可以直接使用 WSDL 进行描述. 我们使用如下的 WSDL 模板来描述本文中服务的概念,即给定服务 $S=(I,O)$,可以用 WSDL 表示为:

```
〈definitions〉
 〈message name = "S_input"〉
  〈part name = "input" element = "I"/〉
 〈/message〉
 〈message name = "S_output"〉
  〈part name = "output" element = "O"/〉
 〈/message〉
 〈portType name = "A_portType"〉
  〈operation name = "S"〉
```

```
    〈input message = "S_Input"〉
    〈output message = "S_Output"〉
   〈/operation〉
  〈/portType〉
 〈/definitions〉
```

该模板使用一个 operation 标签表示服务 S,S 的输入输出文档分别被定义成该 operation 的输入输出 message. 于此对应,服务 S 的实例 $E=(S,L)$ 可以通过如下的 WSDL 模板描述:

```
〈definitions〉
 〈binding name = "A_binding",
          type = "A_portType"〉
  〈soap:binding style = "document" …/〉
  〈operation name = "S"〉
   〈input〉
    〈soap:body use = "literal" …/〉
   〈/input〉
   〈output〉
    〈soap:body use = "literal" …/〉
   〈/output〉
  〈/operation〉
 〈/binding〉
 〈service name = "A_service"〉
  〈port name = "E"
        binding = "A_binding"〉
   〈soap:address location = "L"/〉
  〈/port〉
 〈/service〉
〈/definitions〉
```

该模板首先给出 S 的一个 binding,该 binding 使用 document-literal 编码方式,即直接传输 XML 文档,不做编码转换,这是一种面向文档的编码方式. 基于这个 binding,将服务实例 E 描述为一个 port.

WSDL 的服务模型中,每个 portType 中允许包含多个 operation,这类似于对象模型中对象与方法的关系. 而本文将服务的概念直接对应到 operation,这本质上与 WSDL 的模型并不冲突,因为与 HTTP,SMTP 等底层的传输协议类似,Web 服务本身是无状态的或者说状态是在高层的应用逻辑中维护,而服务框架本身并不提供状态支持,因此 WSDL 定义的 portType 只是表示其中包含的 operation 在业务逻辑上是相关的,比如某个 operation 返回的某个 ID 可以作为另外一个 operation 的参数,但是 WSDL 本身并没有提供这种描述机制,这正是显式数据关联要解决的问题.

在 WSDL 的服务模型中,异常(fault)也是很重要的一个组成部分,由于异常处理不是本文的研究内容,因此,本文对此不做展开.

3.2 显式数据关联的 XML 描述

我们定义了一个 XML Schema 用于描述显式数据关联,本节通过一个直观的模板来说明这个 XML Schema. 给定显式数据关联 $A=(S,T,M,X)$,其 XML 表示如下:

```
〈datalink type = "service"〉
 〈src〉〈!-- S --〉
  〈portType〉ns:srcPtName〈/portType〉
  〈operation〉srcOpName〈/operation〉
 〈/src〉
 〈des〉〈!-- T --〉
  〈portType〉ns:desPtName〈/portType〉
  〈operation〉desOpName〈/operation〉
 〈/des〉
 〈mappings〉〈!-- M --〉
  〈mapping〉〈!-- p = f(d) --〉
   〈despart〉desPartName〈/despart〉
   〈property〉XPath_Expr〈/property〉
   〈srcpart〉srcPartName〈/srcpart〉
   〈transform type = "xpath"
               multiplicity = "single"〉
    XPath_Expr
   〈/transform〉
  〈/mapping〉
  〈mapping〉
   …
  〈/mapping〉
 〈/mappings〉
 〈!-- X --〉
 〈weight〉X〈/weight〉
 〈desc〉A demo assoc〈/desc〉
 …
〈/datalink〉
```

该 XML 模板与显式数据关联定义中元素的对应关系在代码片段中已经简单标明. 首先描述的是 S 和 T,S 对应 WDSL 中的一个 operation,因此,在 src 标签下,我们通过 portType 和 operation 两个标签唯一地标识一个服务. portType 通过一个 QName(Qualified Name)引用 WSDL 中定义的某个 portType,而 operation 则是该 portType 下对应的某个 operation. Operation 是一个可选的标签,因为在本文定义的服务模型对应的 WSDL 模板中 operation 具有唯一性(见 3.1 节),因此 operation 可以省略. 定义 operation 标签的目的是为了与已有的 WSDL 文档结合,使得显式数据关联可以描述任意的两个合法的 WSDL 文档之间的数据关联,而不仅限于本文给出的 WSDL 模板. T 的描述和 S 完全类

似,不做展开.

需要注意的是,datalink 标签包含一个 type 属性. 在定义的 Schema 中,type 属性具有两个可能的值:service 和 instance. 如果 type 的值为 service,该显式数据关联描述的是两个服务之间的数据关联;如果 type 的值为 instance,则该显式数据关联描述的是两个服务实例之间的数据关联,在这种情况下,src 和 des 标签应变成如下的形式:

```
〈datalink type = "instance"〉
 〈src〉〈!-- E1 = (S,L1)--〉
  〈service〉ns:srcSvcName〈/service〉
  〈port〉srcPtName〈/service〉
  〈operation〉srcOpName〈/operation〉
 〈/src〉
 〈des〉〈!-- E2 = (T,L2)--〉
  〈service〉ns:desSvcName〈/service〉
  〈port〉desPtName〈/service〉
  〈operation〉desOpName〈/operation〉
 〈/des〉
 …
〈/datalink〉
```

对比前面的 XML 片段,当 type =“instance”的时候,应当使用 service,port 和 operation 3 个标签来唯一地标识一个服务实例,这与 3.1 节中给出的 WSDL 模板结构完全对应.

引入这两种不同类型的显式数据关联是为了提供更强的表达能力. 根据图 1 所示服务与服务实例的关系,由于实现细节的差异或者后台数据源的不一致,服务之间的显式数据关联在应用到某些服务实例时可能产生不正确的结果,这种情况下,需要提供一种机制来描述特定的服务实例之间的数据关联,以此来修复对应的服务之间显式数据关联所存在的问题或将其标注为无效的关联,而这正是服务实例之间显式数据关联的必要性. 另一方面,服务实例之间的数据关联往往会有一些附加的信息需要描述,比如针对特定实例的约束条件、QoS 属性等. 这些扩展的描述信息只作用于两个特定的服务实例,很显然,这种情况下服务实例之间的显式数据关联也是必须的.

如果从面向对象的角度来理解这两种不同类型的显式数据关联,那么两个服务之间显式数据关联可以被认为是一个基类,而这两个服务的实例之间的显式数据关联则是这个基类的子类. 子类可以重载(修改)基类描述信息,也可以添加子类所特有的描述信息.

显式数据关联四元组中 M 是一组映射,每个映射通过 mappings 标签下的 mapping 标签描述. 每个映射由两部分组成,第一部分是服务 T 的输入文档中的某个元素,通过标签 despart 和 property 描述. Despart 引用服务 T 输入消息中的某个 part,与 src,des 标签下的 operation 标签类似,despart 是一个可选的标签,如果 part 具有唯一性,则可以省略. Property 的值是一个 XPath 表达式,指向 T 输入文档中的某个标签或属性.

第二部分是映射规则,以 S 的输出文档作为其输入,通过标签 srcpart 和 transform 描述. Srcpart 与 despart 完全对应. Transform 的属性 type 控制映射规则的表达方式. 在前面给出的模板中,type 的值是 xpath,这表示 transform 下包含了一个 XPath 表达式,指向 S 输出文档中某个标签或属性. 映射规则可以通过多种方式表达,比如 XPath,XSLT 甚至嵌入式脚本语言等. 在本文定义的 Schema 中,type 有两个枚举值:xpath 和 xslt. Transform 的属性 multiplicity 描述映射规则返回的数据与被映射的 property 之间是一对一还是多对一的关系. 它有两个枚举值:single 和 multiple,分别对应显式数据关联模型中的 = 和 *= . 以前面的服务 A 和 B 为例,ProductID 和 Products 之间的映射可以定义为:

```
〈mapping〉
 〈property〉/ProductID〈/property〉
 〈transform type = "xpath"
   multiplicity = "multiple"〉
  /Products/Product/ProductID
 〈/transform〉
〈/mapping〉
```

需要指出的是,我们并不要求根据 M 中映射规则构造的输入文档对调用 T 而言是完备的,即显式数据关联并不一定是一个可用的“超级链接”,可能需要其它的数据才能构造一个完备的输入文档,以实现对 T 的调用. 在这一点上,显式数据关联与超级链接是有区别的.

四元组中 X 被定义为一个权值,与前面给出的模板中的 weight 标签对应. 需要说明的是,在具体实现中,我们将 X 定义为一个扩展点,权值只是扩展信息的一种. 具体来讲,在 XML Schema 中,我们将 X 定义成一个〈any〉元素,因此,我们可以在一个显式数据关联中添加任意的元素来描述与该数据关联相关的信息,比如前面模板中一个简单的文本描述信息.

扩展的描述信息在实际应用中往往具有重要的意义,但是也需要相应的能够理解这些扩展信息的程序对其进行处理. 本文定义的权值扩展在自动服务组装算法中可以作为一种启发式信息来提高算法

的效率,这在第 4 节中会有进一步的介绍.

3.3 显式数据关联与 WSDL

原理上 3.2 节给出的显式数据关联 XML 描述可以独立于 WSDL 定义和发布,但是,最终应用显式数据关联的时候必须结合相应的 WSDL 文件,因此,将显式数据关联嵌入源服务 S 或者目标服务 T 的 WSDL 文档,作为服务描述的一部分共同发布是一个更加合理的方法.

W3C 给出的 WSDL 规范允许某些标签下嵌入任意的 XML 片段,比如 binding,service 等标签,但是 WS-I Basic Profile 1.1 中定义了更加灵活的规则,即 WSDL 中的任意一个标签都可以包含任意的扩展属性和扩展标签. 因此,原则上显式数据关联可以嵌入 WSDL 的任意标签之下,但一般情况下总是将两个服务之间的显式数据关联嵌入相应的 operation 标签之下,而将两个服务实例之间的显式数据关联嵌入相应的 port 标签之下. 除此之外,另一种嵌入方式是将显式数据关联记录在单独的文件中,然后在 WSDL 中通过扩展标签导入,原理上跟直接嵌入完全一致,但这样可以使得 WSDL 文件保持简洁.

4 显式数据关联的应用

显式数据关联所具备的描述能力使其拥有广阔的应用空间,本节分别从服务组装、服务发现、经验知识和启发式信息 4 个方面讨论显式数据关联的应用.

4.1 服务组装

显式数据关联反映了服务与服务在业务逻辑上的相关性,因此其主要的应用在于服务组装. 显式数据关联是面向服务组装人员的超级链接,服务组装人员可以根据显式数据关联构造服务的输入文档,虽然这个输入文档并不一定完备,但是这可以在很大程度上减轻服务组装的开销.

可以从两个方面来理解显式数据关联给服务组装带来的好处. 首先,一个 Web 服务的语义往往通过以自然语言表达的技术文档描述,涉及复杂的领域知识,因此,对于非领域专家的服务组装人员而言,给定一系列可用的服务,往往很难快速理解这些服务在业务逻辑上的关联,更无从谈起有效利用这些服务来构造应用程序. 显式数据关联有助于改善这一状况,我们可以将显式数据关联可视化成易于理解的有向图,从而帮助服务组装人员快速理解服务之间的业务逻辑关系,降低学习成本. 其次,对于跨公司、跨组织的服务组装,可能存在多个类型系统,这些类型系统描述了相似的概念却无法直接替代,在这种情况下,消息转换就不可避免. 虽然目前已经存在很多商业的工具可以辅助开发人员自动或者半自动地构造消息转换,但是设计并实现一个消息转换依旧不是一项简单的任务,尤其是当消息类型比较复杂的时候,往往需要花费大量的时间和人力才能设计出一个真正能满足实际需求的消息转换. 因此,消息转换的复用十分重要. 显式数据关联中的映射规则正好满足了这样一个需求,它提供了一种很直接的消息转换的发布与复用机制. 从服务组装人员的角度而言,可以避免自己开发或者查找相应的消息转换模块,这将极大减轻服务组装的难度,并提高服务组装的可靠性.

4.2 服务发现

显式数据关联中包含了源服务 S 和目标服务 T,因此,如果显式数据关联被嵌入到服务描述中,那么得到 S 就可以发现 T,反之,得到 T 亦可以发现 S. 那么,给定一个服务的集合,通过这种传递式的发现过程,我们很容易得到一个更大的服务集合,其中包含了一系列相关的服务,我们认为这是一种十分有效的分布式服务发现机制.

这种发现机制的一大特点是不需要任何在线可访问服务发现引擎,包括像 UDDI 这样集中式的发现引擎,或者基于 P2P 网络的分布式发现引擎[11-12]. 由于服务之间的关联信息自包含于服务描述中,因此整个发现过程完全可以在本地通过类似 Crawler 的方式进行,当然前提是 WSDL 文档必须包含有效的显式数据关联. 这种发现机制的另一个特点是,其输出结果对服务组装具有很强的指导意义,因为通过这种方式发现的服务在业务逻辑上具有很强相关性,对应用逻辑的构造可以起到很好的辅助与提示作用.

4.3 经验知识

在实际应用中,IT 环境往往十分复杂,IT 工程师主要依靠各自的经验解决各种问题. 如果能够将经验知识通过一种形式化的方式记录下来,并加以复用,这对 IT 系统的运营具有重要的意义. 显式数据关联的可扩展性使得它可以成为任何与服务之间数据传递相关的经验知识的载体. 这里我们从服务之间互操作性的角度出发,给出一个利用显式数据关联记录经验知识的应用场景.

根据 3.2 节的介绍,由于实现、配置、部署上的差异,服务实例之间可能存在各种互操作问题. 显式数据关联允许描述两个服务实例之间的数据关联,并允许将任意的扩展语义附加给该数据关联,因此,

我们可以使用显式数据关联刻画服务实例之间的互操作性. 以本文给出的权值扩展为例,我们规定:如果权值小于 0,那么这个数据关联存在互操作问题,应该避免使用. 回到前面的例子,假设服务 A 中包含的 ProductID 不能完全被服务 B 所识别,那么我们就可以声明一条权值为 - 1 的显式数据关联,并将其存入知识库中.

随后,这类经验知识可以被应用到服务组装的绑定过程中. 抽象设计与具体实现严格分离一直是软件工程的一个重要准则,SOA 也不例外,BPEL 在抽象层面描述业务流程,而其中调用的服务在部署时或者运行时才得以绑定. 我们可以设计分析工具提取 BPEL 中的数据流,并与知识库中的显式数据关联进行匹配,如果发现有权值为负的显式数据关联被匹配上,则尝试替换成其它的服务实例或给出警告信息,以避免运行时出现互操作问题. 当然,这其中有很多技术细节需要深入研究,本文不做展开.

4.4　启发式信息

服务自动组装方法的研究工作中有一类算法根据服务的输入输出来构造服务组装,比如 SWORD[13]、Liang 提出的基于与或图的自动组装算法[14]以及 WS-Challenge①定义的服务组装问题. 这类算法的求解过程中,一个很重要的步骤是根据某个给定服务的输入文档查找相关的服务,使得这些服务的输出文档能够构造出给定服务的输入文档. 显然,显式数据关联所描述的服务输入输出文档之间的关联关系恰好可以作为一种启发式信息应用到此类算法中.

以 Liang 的与或图搜索算法[14]为例,在这个算法中,服务及其输入输出文档被抽象成一个与或图,服务节点是与节点,文档节点是或节点. 算法对与或图进行搜索,根据给定的输入输出条件构造与或图的解(即一个组装方案),每得到一个解,就将结果输出给组装人员进行评估,如果不能满足需求则尝试构造一个新的解. 该算法存在这样一个问题,当服务数量比较大的时候,算法的解空间也会随之增大,这样一来,人工评估的次数可能会急剧上升从而导致算法失效. 这种情况下,如果服务组装人员具有一些先验知识,比如服务 A 与服务 B 的之间的连接具有高优先级,那么当算法搜索到 A 节点的时候,可以优先考虑 B 节点,从而指导算法更快地输出满足需求的解. 很明显,用显式数据关联来表达这种启发式信息是一个有效的方法,我们可以将权值作为优先级来控制搜索算法的寻径策略,从而达到优化该算法的目的.

5　WS-Challenge 应用案例

本节以 WS-Challenge 2007 的参赛程序为例,讨论如何利用显式数据关联优化自动组装算法.

5.1　WS-Challenge 简介

WS-Challenge 是 CEC/EEE 年会组织的一个 Web 服务自动组装竞赛,其目标是在给定的服务集合中以尽可能短的时间找出所有满足需求的组合方案. 每个组合方案都是一个服务链表,如图 2 所示. 该链表满足:(1) $S(k)$ 的输入文档包含于 $S(k-1)$ 的输出文档与算法的输入文档的并集,$1 < k < n$;(2) $S(1)$ 的输入文档包含于算法的输入文档;(3) $S(n)$ 的输出文档包含算法要求的输出文档.

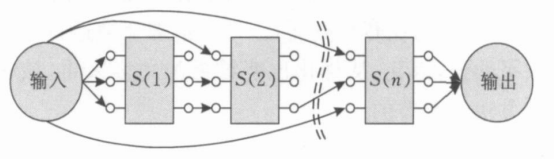

图 2　服务链表

WS-Challenge 2005 只考虑语法级的关联,从 2006 年的比赛开始引入文档之间的继承关系. 继承关系是应用最广泛的一种语义信息,继承关系的引入表明 WS-Challenge 正逐步向语义组装过渡,在最新的 WS-Challenge 2008 数据集中已经出现了 OWL 描述的本体. 基于语法关联的组装只需要简单的匹配文档类型,因此速度较快,但是无法处理复杂或者异构的数据模型. 本文介绍的组装算法适用于 WS-Challenge 2006 和 2007 中语义组装(Semantic Composition)的数据集.

5.2　优化方案

求解 WS-Challenge 定义的组装问题的关键在于,给定某个文档,如何高效地获取所有能够输出该文档的服务. 我们将 WS-Challenge 2006 的参赛程序作为基准算法,在基准算法中,所有的服务被保存在一个以文档为索引的倒排表中,算法在构造组合方案的时候通过查找该倒排表快速地获取能够输出给定文档的所有服务. 对于图 2 所示服务链表中的某个服务而言,其前驱服务可以通过对其输入文档的倒排表查找结果求交获取,如图 3 所示.

本文利用显式数据关联对基准算法进行改进. 基准算法查询倒排表的过程本质上是构造数据关联的过程,在改进的算法中,我们将这个过程转移到一

① http://www.ws-challenge.org/

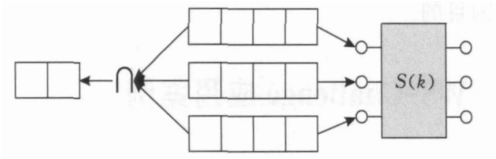

图 3 构造服务 $S(k)$ 的前驱服务列表

个预处理算法中,预先完成数据关联的构造,并按照显式数据关联的格式保存至服务描述中. 通过对数据集进行预处理,使得改进的组装算法可以省去查找倒排表的时间开销,从而也就减少整个算法的时间开销. 当然预处理程序本身也需要一定的时间开销,因此,如果从一次运行的角度来衡量,改进后的算法在整体时间消耗上并没有优势,然而,预处理结果可以被重复利用,对同一个数据集而言,一旦预处理完成之后,所有的请求都可以得到优化,因此,改进算法的优势在多次请求下才会逐步显现. 就 WS-Challenge 的比赛规则而言,每个请求会被运行 5 次求平均值,因此,从优化比赛结果的角度来讲,该优化方法是有效的.

5.3 实验结果

本文利用往届 WS-Challenge 提供的 4 个数据集对改进后的自动组装算法进行评估. 实验平台的硬件环境是 Celeron 1 GHz,512MB RAM,软件环境是 Debian Sarge,时间值通过 gettimeofday 这个 API 获取.

4 个测试数据集的统计信息如表 1 所示. 前两个数据集来自 WS-Challenge 2005,后两个数据集来自 WS-Challenge 2006. 2005 年的数据集不考虑 XML Schema 中的继承关系,因此继承关系的数量都是零.

表 1 中也给出了每个数据集预处理的时间开销.

实验的结果如图 4 所示. 可以看到,对于 2005 年的两个数据集,改进算法与基准算法没有明显差别;而对于 2006 年的两个数据集,改进算法明显优于基准算法,这个结果是符合预期的. 根据前面介绍,改进算法与基准算法的主要差别在于省去了查找倒排表的开销,这一步在预处理过程中完成. 由于 2005 年的数据集不考虑继承关系,因此基准算法在查找某个服务的前继服务的时候只需要针对每个输入文档查找一次倒排表,而倒排表经过哈希优化,其查询复杂度几乎是常数,因此查询开销在总时间开销中所占比例很小,改进后的算法不能表现出明显的优势. 而 2006 年的数据集需要考虑继承关系,使得基准算法查询倒排表的次数急剧上升,尤其在最

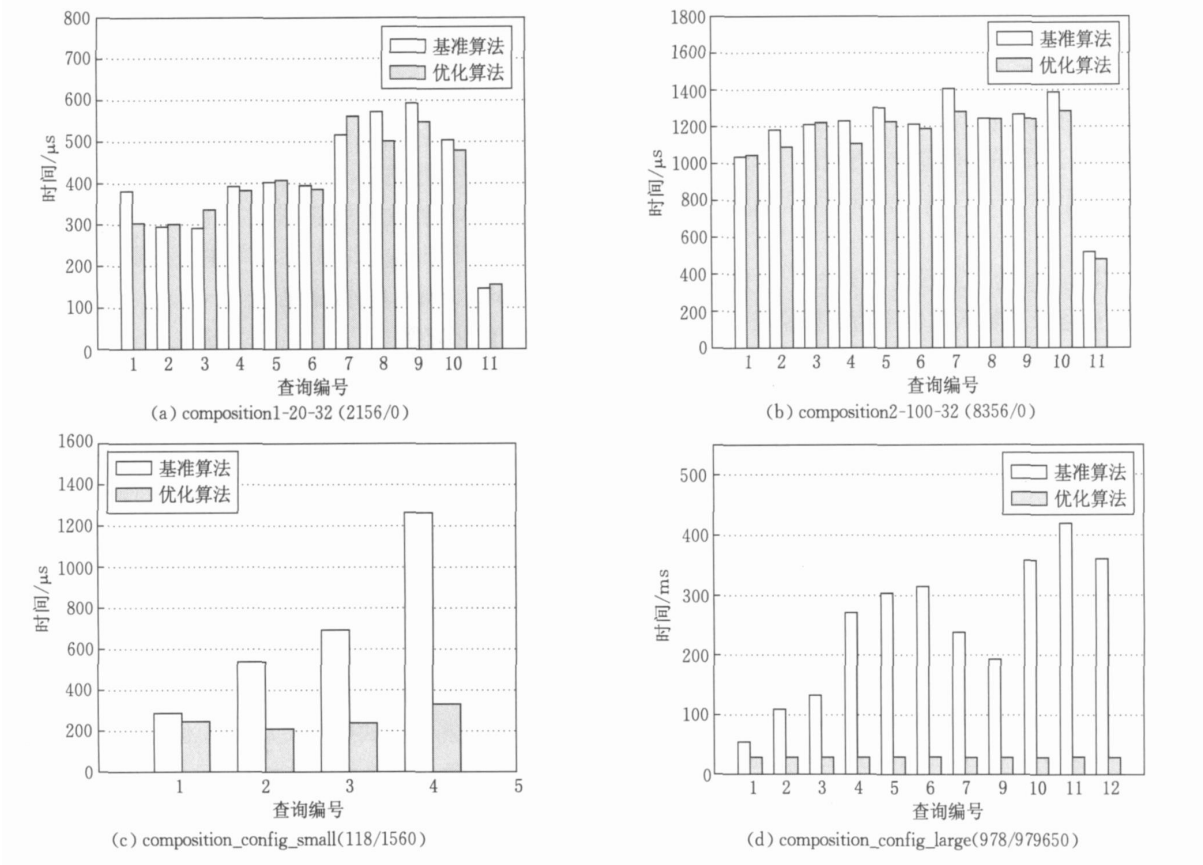

图 4 结果比较

后一个数据集中,有接近 100 万个继承关系,某些类型的继承深度有上百层,使得查询倒排表的开销在总时间开销中所占的比重大幅提高,因此,相比之下改进算法的执行效率明显提升.

表 1　测试数据集

数据集	服务数量	继承数量	预处理时间/μs
composition1-20-32	2156	0	25694
composition2-100-32	8356	0	92341
composition_config_small	118	1560	3918
composition_config_large	978	979650	298000

对比表 1 中数据集的预处理时间开销,可以发现对于 2006 年的数据集而言,预处理后组装算法所节省的时间(即每个 Query 的时间差之和,再乘以运行次数)完全可以抵消预处理的时间开销,因此该优化方法是有效的. 最终该算法在 WS-Challenge 2007 中取得组装性能的冠军[15].

6　显式数据关联的生成与维护

显式数据关联模型定义了数据关联的一种描述方式,但是对于如何生成显式数据关联本文并没有给出具体的方法. 另一方面,由于显式数据关联是一种静态声明,必须保证它与相应服务描述的一致性,因此,显式数据关联的维护也是一个重要的问题. 这一小节从自动和手工两个角度简要讨论显式数据关联的生成与维护问题.

首先,显式数据关联可以通过构造算法自动生成并加以维护. 比如,第 5 节给出的 WS-Challenge 应用案例中显式数据关联通过预处理程序自动生成,如果数据集发生改变,我们可以重新运行预处理算法更新数据关联. 对于大数据量以及实时性要求较高的应用而言,每次服务描述更新都运行预处理程序是不可行的,这种情况下可以对预处理程序进行扩展,提供添加、删除、更新服务描述的接口,通过这些接口快速更新相应的显式数据关联. 自动的显式数据关联生成与维护方法能够快速处理大量数据,基本没有维护开销,但是,这种方法只是将原本隐含在数据模型或本体模型中的数据关联提取出来,并不能充分利用显式数据关联的表达能力,对实际应用的意义不大.

其次,显式数据关联可以在相关工具的辅助下通过人工构造和维护. 我们认为显式数据关联可以由 3 种角色来生成:(1) 服务提供商. 服务提供商可以将自己提供的服务以及商业伙伴提供的服务之间的业务逻辑关系通过显式数据关联的方式提供给客户;(2) 独立的第三方. 如何在互联网海量的服务中寻找最佳的组合,这是一类重要的经验知识,任意一个组织都可以以第三方的身份将这类知识通过显式数据关联的方式提供给用户;(3) 开发人员. 开发人员也可以定义自己的显式数据关联,将成功的或失败的服务组合通过显式数据关联记录下来.

人工维护显式数据关联难度相对较大. 一旦某个服务描述发生变化,所有相关的显式数据关联都应该得到更新,因此,必须提供相应的工具帮助维护人员快速定位需要更新的显式数据关联,并对其进行更新. 从整个互联网的角度来讲,显式数据关联的维护工作具有分布性,比如服务提供商只需要维护自己发布的显式数据关联,因此不存在海量数据集需要集中维护的问题.

最后需要指出的是,虽然显式数据关联的描述依赖于服务,但服务的描述并不依赖显式数据关联,因此,在服务设计的阶段可以完全不考虑显式数据关联,即不需要预知所有可能与其交互的服务. 与某服务相关的显式数据关联集合可以在该服务设计实现完成后逐步生成与完善.

7　结　论

服务之间的数据关联反映了服务与服务在业务逻辑上的相关性,对服务组装、服务发现等任务具有重要的意义. 本文借鉴超级链接的思想,提出了显式数据关联的概念,解决了隐式表达数据关联的方法所存在的问题.

本文给出了显式数据关联的形式化模型以及具体的 XML 实现,并讨论了显式数据关联与 WSDL 的集成. 在此基础上,本文讨论了显式数据关联在服务组装、服务发现等方面的应用,并给出了利用显式数据关联优化 WS-Challenge 组装算法的一个应用案例. 实验结果表明,该优化方法在处理具有复杂继承关系的数据集时能够有效提高算法的效率.

参　考　文　献

[1] Berardi D, Calvanese D et al. Automatic composition of e-services that export their behavior// Proceedings of the International Conference on Service-Oriented Computing. Trento, Italy, 2003: 43-58

[2] Bultan T, Fu X et al. Conversation specification: A new approach to design and analysis of e-service composition// Proceedings of the International Conference on World Wide Web. Budapest, Hungray, 2003: 403-410

[3] Zhang J, Chang C-K et al. WS-Net: A Petri-net based specification model for Web services// Proceedings of the IEEE International Conference on Web Services. San Diego, California, USA, 2004: 420-427

[4] Sivashanmugam K, Verma K, Sheth A, Miller J. Adding semantics to Web services standards// Proceedings of the 1st International Conference on Web Services. Las Vegas, Nevada, 2003: 395-401

[5] Patil A, Oundhakar S, Sheth A, Verma K. METEOR-S Web service annotation framework// Proceedings of the International World Wide Web Conference. New York, USA, 2004: 553-562

[6] Verma K, Sivashanmugam K, Sheth A et al. METEOR-S WSDI: A scalable infrastructure of registries for semantic publication and discovery of Web services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, 2005, 6(1): 17-39

[7] McIlraith S, Martin D. Bringing semantics to Web services. IEEE Intelligent Systems, 2003, 18(1): 90-93

[8] Paolucci M, Srinivasan N, Sycara K. Toward a semantic choreography of Web services: From WSDL to DAML-S// Proceedings of the 1st International Conference on Web Services. Las Vegas, Nevada, USA, 2003: 22-26

[9] Bruijn J D, Fensel D et al. Using the Web service modelling ontology to enable semantic eBusiness. Communications of the ACM, Special Issue on Semantic eBusiness, 2005, 48(12): 43-47

[10] Haller A, Cimpian E, Mocan A et al. WSMX—A semantic service-oriented architecture// Proceedings of the IEEE International Conference on Web Services. Orlando, Florida, USA, 2005: 312-328

[11] Schmidt C, Parashar M. A peer-to-peer approach to Web service discovery. World Wide Web, 2004, 7(2): 211-229

[12] Papazoglou M P, Kramer B J, Yang J. Leveraging Web-services and peer-to-peer networks// Proceedings of the International Conference on Advanced Information Systems Engineering. Klagenfurt, Austria, 2003: 485-501

[13] Ponnekanti S R, Fox A. SWORD: A developer toolkit for Web service composition// Proceedings of the 11th International World Wide Web Conference. Honolulu, Hawaii, USA, 2002

[14] Liang Q-H A, Su S Y-W. AND/OR graph and search algorithm for discovering composite web services. International Journal of Web Services Research, 2005, 2(4): 48-67

[15] Gu Z-F, Xu B, Li J-Z. Inheritance-aware document-driven service composition// Proceedings of the 9th IEEE Conference on E-Commerce Technology (CEC'07) and the 4th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE'07). Tokyo, Japan, 2007: 513-516

GU Zhi-Feng, born in 1979, Ph. D. candidate. His research interests include service discovery and service composition.

LI Juan-Zi, born in 1964, Ph. D., associate professor. Her research interests include semantic Web, Web service, text mining and knowledge discovery.

HU Jian-Qiang, born in 1971, Ph. D.. His research interests include distributed object-oriented technologies and software engineering.

XU Bin, born in 1973, Ph. D., associate professor. His research interests include Web services and Semantic Web.

WANG Ke-Hong, born in 1941, professor, Ph. D. supervisor. His research interests include network computing and knowledge engineering.

Background

Service-oriented computing (SOC) has been widely accepted as the next generation programming paradigm. It defines promising technologies that enable future computing models over the Internet. It is obvious that the information encoded in service model is the basis of high-level technologies, and many works have tried to define extended service models to support advanced service technology. This paper introduces another extended model, so-called explicit data-link, for the modeling of data correlations among Web services. Unlike existing works, which usually model data correlations in an implicit way, and create them dynamically by reasoning on the data models or ontology models used by service definitions, the authors' method defines a way to express data correlations with static explicit declarations, which makes it more expressive than existing methods. This work is supported by the National Basic Research Program of China (973 Program) under grant Nos. 2007CB310803, 2003CB317007, the National High Technology Research and Development Program (863 Program) under grant No. 2007AA010306, National Natural Science Foundation of China (NSFC) under grant No. 190604025, and China Postdoctoral Science Foundation under grant No. 20070410061. These foundations cover a lot of research topics, while all of them adopt SOC as the infrastructure, which is the research topic of this work. This work is also inspired by the WS-Challenge competition. The authors have taken part in WS-Challenge 2006 and 2007. In WS-Challenge 2006, they won the championship of syntactic composition, and won the number 4 of semantic composition. WS-Challenge 2007 only held semantics composition, and they won the championship in this time. Now the authors are preparing for the coming WS-Challenge 2008.