Active XML 文档重写机制的研究

高永兵，胡文江，靳 荣
（内蒙古科技大学 信息工程学院，内蒙古 包头 014010）

摘 要：Active XML 定义为一种动态的 XML，主要思想是在 XML 中嵌入 Web Services 调用。当 Active XML 文档用作表示和交换时，通过调用相应的 Web Services，用返回结果取代 Active XML 文档中的 Web Services 调用或者作为子节点插入，这个过程称为重写。针对应用要求不同的场景，总结了两类重写方式：无模重写，即在重写时不受 Schema 约束；有模重写，重写时要符合预定的 Schema。在对无模方式做了简单讨论后，对有模方式的算法做了详细讨论，为拓展 Active XML 技术的应用领域提供了技术支持。

关键词：Active XML；Web Services；重写
中图分类号：TP393 **文献标识码**：A **文章编号**：1673－629X(2008)03－0085－04

Research of Rewriting of Active XML Documents

GAO Yong-bing, HU Wen-jiang, JIN Rong
(School of Information Engineering, Inner Mongolia University of Science & Technology, Baotou 014010, China)

Abstract: Active XML document is a type of dynamic XML that some of the data are defined by means of embedded calls to Web services. When such documents are represented and exchanged among applications, one can materialize Web services calls and replace these calls by return results or inserted as subnode, and the process is called rewriting. Concludes two methods that named no schema and schema to address this problem and describes respective rewriting algorithm in detail that is foundational to farther utilize.

Key words: Active XML; Web services; rewriting

1 Active XML 介绍

XML 作为一种自描述的半结构化数据模型，已经成为一种各种应用之间数据表示和交换的标准。XML 分两个层次：结构完整性与规范性。结构完整性要求符合 XML 文档语法，规范性要求采用 DTD 或 Schema 实现对文档节点的约束，使之符合模式定义要求，因此，数据表示和交换也基于这两个层次展开。Web Services[1]是由组件技术发展起来的新标准，它的意图是允许一些活动对象为潜在的请求提供松散耦合的分布式服务。Web Services 是建立在 XML、SOAP（简单对象访问协议）、WSDL（Web Services 描述语言）、UDDI（统一描述、发现和集成）基础之上的分布式应用架构。在 Web Services 架构的各模块间以及模块内部，消息以 XML 格式传递，其原因在于，以 XML 格式表示的消息易于阅读和理解，并且 XML 文档具有跨平台性和松散耦合的结构特点。

Active XML[2]定义为一种分布式信息管理语言，主要包含两部分：Active XML 文档和 Active XML 服务组件。Active XML 服务组件提供对 Active XML 文档的支持；Active XML 文档是对 XML 的扩展，其主要思想是在 XML 文档中嵌入 Web Services 调用，实现对 Internet 上的 Web Services 资源的动态访问，但语法上仍是有效的 XML 文档。通过触发嵌入的 Web Services 调用，相关 Web Services 被激活（采用 SOAP 协议），并返回结果，文档动态地从数据源（Web Services）获取相关数据。一个 Active XML 文档包含两种信息，在文档中直接表示出来的数据称为“显式数据”，以 Web Services 调用表示的数据称为“隐含数据”。在需要（典型为查询和更新）时将这种调用“具体化”，即通过调用相应的 Web Services，用返回结果取代 Web Services 调用或者作为子节点插入，这个过程称为“重写”。通过对

收稿日期：2007－06－19
基金项目：内蒙古自然科学基金资助项目（200608010808）；内蒙古科技大学青年科研基金资助项目（KY200654）
作者简介：高永兵（1974－），男，内蒙古包头人，副教授，硕士生导师，CCF 会员，研究方向为数据库与 XML；胡文江，硕士生导师，教授，研究方向为计算机网络控制；靳 荣，硕士生导师，教授，CCF 高级会员，研究方向为计算机网络技术。

重写的控制，Active XML 文档实现能够比原 XML 表达更加丰富准确实时的内容。

为了能清楚准确地描述 Active XML 文档，给出规范化模型和一个实例。Active XML 文档可以看作一颗标签树，标签由两类节点组成：数据节点和函数节点，函数节点对应相应的 Web Services 调用。引入大写字母 L 表示标签集，F 表示函数名集，D 表示数据值集。

标签属于 $L \cup D$ 的节点称数据节点，而标签属于 F 的节点称函数节点，一个函数节点的子树可作为函数参数，当函数被调用时子树作为参数传递，用返回值重写该函数节点。一个 Active XML 文档的例子如图 1 所示，其中函数节点用黑体表示，为了使描述更为清楚，文档忽略了函数调用的具体格式和命名空间等细节。该文档表示 yahoo 体育新闻报道，其中新闻标题和日期为显式信息；获取 yahoo 的体育新闻以函数调用的方式给出，具体化时，将用 date 的体育新闻列表重写函数节点。

```
<? xml version="1.0">
<news>
  <title> Sports report </title>
  <date>2004-11-16</date>
  <sports>
    <call server="GetSports@yahoo.com">
      sports
    </call>
  </sports>
</news>
```

图 1 Active XML 文档例子

2 问题描述

作为一种新兴的 Web 应用模式，Web Services 在 Web 的任何地方都可调用，这就为实现一种灵活的数据交换范型提供了可能，Active XML 正是利用了 Web Services 这个特性，发送端不一定要对 Active XML 文档进行具体化，可以选择发送包含 Web Services 调用的 Active XML 文档给接收端，由接收端根据需要适时地完成具体化。这种灵活的调用方式允许实现分布式运算，即考虑端点负载、访问控制、功能和性能等因素，将 Web Services 调用方法由“有能力”或者“需要”的端点来实现调用的具体化，而不同端点不同时间不同方式的调用能够得到动态的显式数据。一个简单的模型如图 2 所示，需要强调的是，图 2 中的 Server 也可以驻留在 Sender 或 Receiver 端。

但在利用 Active XML 文档的灵活交换特点的同时，会遇到如下两个问题：

第一，如何来实现 Active XML 文档中的 Web Services 调用？这里包括端点、时间、调用方式等具体参数的选择，端点参数指 Web Services 调用选择在发送端还是接收端进行，时间参数主要有实时和定时方式，而调用方式主要有推模式或拉模式。

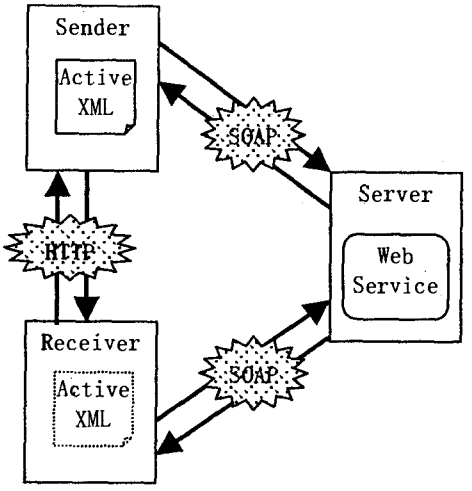

图 2 调用模型

第二，由于 Web Services 调用的结果具有不确定性，可能返回一个包含新的 Web Services 调用的列表，这将会导致递归调用，如何控制递归调用的次数（深度），调用返回列表的数量（广度要求），比如 Top n 等，使其在重写时满足模式（Schema 或 DTD）的要求，例如在图 1 示例的 GetSports 调用中，由于可能返回错误或者一个当日最新的体育新闻列表和一个新的指向更多体育新闻的调用，从而可能形成无限的搜索空间。

对于这两个问题，根据 Active XML 文档中的语法和规范性要求将其分成无模方式和有模方式两类来讨论其重写控制和过程。

3 无模方式

无模方式建立在随机通信的基础之上，事前双方没有预定的模式，但在发送端和接收端交互的过程中，会有上下文存在，类似 Web 上的浏览器与 Web 服务器交互，使用 HTTP 将浏览器类型、接受文档类型、保存的 Cookie 等内容放在请求消息头中；Web 服务端将状态消息、日期和 Web 服务器类型等放在相应消息中。在发送端发送 Active XML 文档之前需有接收端的 Active XML“能力”的信息，根据其能力来选择是否将 Active XML 具体化。当对方不能处理 Active XML 文档时，在本地实现具体化后交换，否则，可以在规定策略（如发送端调用还是接收端调用）的基础上来完成具体化，实现 Active XML 文档向普通 XML 的转化。这种方式可用在对客户端要求较低的系统中，如 RSS[3]

等。

无模方式的目标是在选定端实现对函数节点的重写,使其具体化为普通的 XML 文档。重写算法需要处理 Web 调用深度和广度的问题,过程如下:

设函数调用广度最大为 n,深度不超过 k(n,k 取自然数),自顶向下遍历 Active XML 文档树,对于每一个函数节点 f,如果调用的返回列表大于 n,则截取前 n 项,重写函数,否则判断是否有新函数调用,如果深度 k 大于 0,则继续调用,$k = k - 1$,将新返回列表与前返回列表的和与 n 比较,重复这个过程,直至列表大于 n 或者没有新的调用为止。所有函数都进行重写后,则 Active XML 文档可具体化为普通的 XML 文档。

4　有模方式

在无模方式中,一个较为明显的问题是控制调用具体化的粒度过大,只能选择在哪一端实现,这在一些特殊的应用中不能满足要求,如数据集成[4,5]、工作流管理等。如何来导向 Active XML 文档中具体每一个调用的具体化呢? 在 XML 文档中,模式(DTD 和 Schema)被用来详细说明每一个节点的规范,为了维持 Active XML 文档的 XML 特性,目前的研究普遍采用扩展 XML Schema 模式作为 Active XML 文档的模式。接发双方可以根据预定的 Active XML 文档模式对 Active XML 文档效验,使其符合发送端和接收端的要求。

Active XML 文档模式 s 可用 (L,F,τ) 形式表示,其中 τ 表示相应的映射关系,每一个标签 $l \in L$ 映射到一个 $L \cup F$ 上的正则表达式或原子数据类型,每一个函数 $f \in L$ 也映射一个 $L \cup F$ 上的正则表达式,描述了 f 的名字以及输入和输出类型,记为 $\tau_{in}(f)$ 和 $\tau_{out}(f)$。

正则表达式描述每一个元素的名字及其数据结构,数据节点的正则表达式在 DTD 和 XML Schema 定义。在这里引入了函数节点的正则表达式的描述,与数据节点描述不同的是,它同时规定了函数的输入类型和输出类型表达,图 3 表示一个 Schema 简化的例子,其中 data 表示原子数据类型,| 表示或关系,* 为通配符,表示 0 个或者多个。

data:

　　τ(news) = title. date. (GetSports|Sports *)

　　τ (title) = data

　　τ (date) = data

　　τ (sports) = title. (Getdate|date)

functions:

　　τ_{in}(Getdate) = title

　　τ_{out}(Getdate) = date

　　τ_{in}(GetSports) = data

　　τ_{out}(GetSports) = (sports * |performance)

图 3　Schema 例子

Active XML 文档 t 是模式 s = (L,F,τ) 的一个实例需满足如下条件:每一个有标签 $l \in L$ 的数据节点 $n \in t$,则有 n 的孩子标签形成一个属于 lang($\tau(l)$) 的词;每一个有标签 $f \in F$ 的函数节点 $m \in t$,则 f 的一个输入参数列实例(相应的输出参数列实例) 能形成属于 lang($\tau_{in}(f)$) (相应的 lang($\tau_{out}(f)$)) 的词。

lang(R) 表示正则表达式 R 所能描述的词汇,如 lang($\tau_{in}(f)$) 表示 f 的所有输入类型所形成的词汇集。发送端根据双方提前协定的 Schema 将其 Active XML 文档转化为符合 Schema 规定的格式文档后再进行发送,但这里可能有如下问题:

第一,文档 t 是否是模式 s 的实例,例如图 1 所示的 Active XML 文档就是图 3 所示 Schema 的一个实例,但将图 3 的 Schema 中 Data 的 τ(news) = title. date. (GetSports | Sports *) 修改为 τ'(news) = title. date. Sports * ,则图 1 所示的 Active XML 文档就不一定是 τ'的一个实例,如果调用 GetSports 执行并且返回新闻列表,则可成为 τ'的一个实例;否则由于不进行 GetSports 函数调用,则其不能成为 τ'的实例。

第二,能否找到一个算法来测试一个 Active XML 文档确定会成为一个 Schema 的实例,如果不能确定,是否可以使一个 Active XML 文档能够成为一个 Schema 的实例。接下来将对这种情况的重写进行讨论。

设 t 为 Active XML 文档树,s 为预定交换模式,目标是重写 t 使其成为 s 的模式,这个重写过程涉及到相关函数调用和调用产生的新函数的嵌套调用,每次调用都在改变 t,最终使其成为 s 的一个实例,满足交换模式。为了使这个过程更为高效,执行如下算法:

第一步:对 t 中函数的参数验证,使 t 中的每一个函数参数都能成为合适的输入类型,符合 UDDI 和 ACL 的要求。因为每个函数的参数有可能是一个函数调用,存在嵌套的关系,所以该过程应该从最内层函数开始处理,如果最内层参数不符合,则失败;否则自内向外,内层函数标识或其 $\tau_{out}(f_{in})$ 应为外层函数的输入参数 $\tau_{in}(f_{out})$,如不符合,则失败。这个过程正常完成后,则所有函数的参数应是正确的类型。

第二步:自顶向下处理每一个节点 n,使其符合模式的要求。设 w 为 n 子树标签形成的词,要使 w 成为正则表达式 $R = \tau$(label(n)) 的词,需调用 w 中的函数以及由此函数产生的新的函数,每次调用的返回结果由

$\tau_{out}(f_i)$ 决定。由于函数标识和目标语法都用正则表达式给出，采用它们的有限状态机比较二者的关系。如果所有的节点都符合模式要求，那么 t 必为 s 的实例。

接下来讨论如何测试一个 Active XML 文档确定会成为一个 Schema 的实例，如果不能确定，是否可以使一个 Active XML 文档能够成为一个 Schema 的实例。

在函数调用有限状态机中，一般都会有分支节点的出现，以如图 4 所示的 τ(news) = title. date. (GetSports|Sports*)的有限状态机为例，P3 为分支节点，无论是否经过 P4，进行函数调用，都能到达目标状态 P5。这种情况称为确定的重写，我们的算法按照优先函数调用分支的过程进行，最后必然能到达终态。

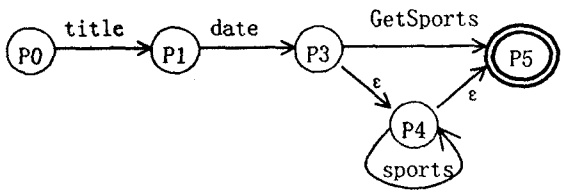

图 4 τ(news)的有限状态机

在图 5 所示的 τ'(news) = title. date. Sports* 的有限状态机中，必须经过 P4，才能到达目标状态 P5。我们的算法使用试探的方式进行，将每一函数节点(分支节点)压栈，如果不符合模式要求，则回溯，寻找下一分支，否则继续，直至到达终态，如果所有分支都失败，则过程失败，返回错误。调用广度和深度的要求都按照无模的方式执行。

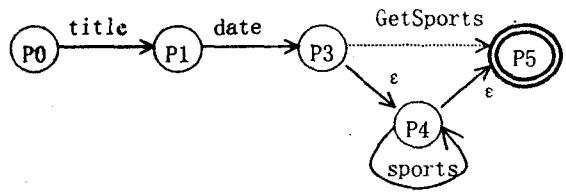

图 5 τ'(news)的有限状态机

5 结束语

针对 Active XML 文档的具体化提出无模方式和有模方式两种方案，与 XML 的结构完整性与规范性要求对应，无模方式适应于具体化控制粒度较大的情况，只需在既定端完成全部函数调用。笔者等设计了 Windows 平台下基于 MSXML 的 Active 控件的实现方式，装有该插件的端点具有 Active XML 能力，能够根据交互信息上下文实现 Web 服务调用具体化。有模方式的设计借鉴了 Tova Milo 等人提出扩展 Schema 的方式[6]，他们使用基于笛卡儿有限状态自动机的一套算法来寻找符合交换 Schema 的函数重写，这在一个 Active XML 文档拥有多个函数调用时精确而复杂，但是有函数组合产生巨大搜索空间的问题，尤其是函数存在多深度调用时体现得更为明显；采用压栈回溯每个函数分别处理的方式虽然使用了试探法，在 Active XML 文档中函数较少并且交换 Schema 较为简单时具有明显优势，函数较多时且深度又大也能避免象笛卡儿有限状态自动机那样出现巨大搜索空间。在 Active XML 重写优化方面还有待研究，这也是进一步的工作方向。

目前，Active XML 已经开展的应用有：数据集成[4,5]，分布式工作空间管理[7]等。笔者总结了无模式要求和有模式要求情况并且实现了其重写算法，这为拓展 Active XML 技术的应用领域如工作流管理[8]、Web 服务组合[9]提供了基础的技术支持。

参考文献：

[1] The World Wide Web Consortium. Web Services Homepage [EB/OL]. 2002. http://www.w3.org/2002/ws.

[2] Benjelloun O. Active XML homepage [EB/OL]. 2003. http://activexml.net/.

[3] RSS-DEV Working Group. RSS homepage[EB/OL]. 2000. http://web.resource.org/rss.

[4] Abiteboul S, Benjelloun O, Manolescu I, et al. Active XML: Peer-to-Peer Data and Web Services Integration[C]//Proceeding of 28th International Conference on Very Large Data Bases. Hong Kong, China: Morgan Kaufmann Publishers Inc., 2002:1087-1090.

[5] Abiteboul S, Benjelloun O, Milo T. Web Services and Data Integration[C]//Proceeding of The 3rd International Conference on Web Information Systems Engineering. Grand Hyatt, Singapore: IEEE CS Press, 2002:3-6.

[6] Milo T, Abiteboul S, Amann B, et al. Exchanging Intensional XML Data[C]//Proceeding of the 2003 ACM International Conference on Management of Data. San Diego, USA: ACM Press, 2003:289-300.

[7] Abiteboul S, Baumgarten J, Bonifati A, et al. Managing Distributed Workspaces with Active XML[C]//In Proceeding of 29th International Conference on Very Large Data Bases. Berlin, Germany: Morgan Kaufmann Publishers Inc., 2003: 1061-1064.

[8] 靳 荣，史海军，高永兵，等. AXML 在基于 Web Services 的工作流过程建模中的应用[J]. 微计算机信息，2006，22(6):253-255.

[9] 靳 荣，赵军富，高永兵，等. 基于 Active XML 的动态 Web 服务组合实现[J]. 计算机应用，2007，33(18):125-127.