对 XPath模式定位能力的扩充

王 强 武港山
(南京大学计算机科学与技术系 南京 210093)
(南京大学计算机软件新技术国家重点实验室 南京 210093)
(qangw@ graphics. nju. edu. cn)

摘 要 基于一般结构模式的定位手段对一个完备的面向 XML文档内容的定位机制来说是重要的,而 XPath在这方面还有所欠缺.首先通过实例分析阐明了 XPath提供基于一般结构模式的定位手段的必要性,随后通过一些扩充定义来改善它的表示能力.扩充从概念、语法描述和执行机制 3个方面展开,概念上,引入了结点序列模式和结点正则表达式的定义;语法描述上,为 XPath扩充了数据类型 *NRegex Type*和结点序列模式匹配函数 *match*;执行机制上,定义了一个用于分析结点序列模式匹配过程的形式化自动机.

关键词 结点序列模式,正则表达式,模式匹配函数,自动机,XPath

中图法分类号 TP391. 4

EXTENSION OF PATTERN LOCATING CAPABILITY IN XPATH

WANG Qiang and WU Gang-Shan
(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)
(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract XPath is deficient in locating method based on general structures, which is important for a complete XML document oriented query mechanism. Examples first are illustrated to elucidate the need to make the extensions and then some extensions are made for XPath. The extensions are developed from three perspectives. From the perspective of concept, definitions of node sequence pattern and node regular expression are introduced. From the perspective of syntax description, a data type called "*NRegex Type*" and a function called "*match*" are extended into XPath. From the perspective of execution mechanism, a formal automaton is defined, which is used to analyze the recognition process for node sequence patterns.

Key words node sequence pattern, regular expression, pattern match function, automaton, XPath

1 引 言

XML[1]为网络文档处理带来了丰富的语义和良好的结构,许多前台的信息表现和后台的数据存储开始采用 XML的描述方式,面向 XML文档的信息查询也迅速成为业界研究的一个方向.

在面向 XML文档的查询机制中,对 XML文档内部对象的定位是一个重要的部分. W3C推出的 XML path language(XPath)[2]正是这样一种定位语言,它建立在 XML文档树的抽象层次之上,为各种结点对象位置的描述提供语法和相关语义.目前,

原稿收到日期: 2000-01-17;修改稿收到日期: 2000-06-02
本课题得到国家自然科学基金资助 (60073030)

XPath被作为基础成分广泛地应用在 W3C制定的其它语言中,如 XPointer, XSLT等.

XPath的定位机制与传统的文档或数据库管理系统中的定位机制是有区别的,区别的实质在于XPath具有更加严格的面向文档层次结构的特性,基于这样的特性,XPath提供了许多面向文档实例的模式定位手段,但对于一些复杂的面向文档一般结构信息的模式定位需求,XPath则缺乏足够的表达能力,本文针对这个问题展开讨论,并提出了相应的解决方法.

本文的讨论是基于有效(valid)XML文档展开的,非有效的 XML文档由于其文档结构是随意的,不具有代表性.

2　XML文档查询实例分析

2.1　实例 1

下面是一个用于记录文献目录的 XML文档类型定义(DTD)及实例.

(1) 文档类型定义 catalog.dtd

```
〈! ELEMENT Catalog (Paper | Book)* 〉
〈! ELEMENT Book (Title, Author+ , ISBN)〉
〈! ELEMENT Paper (Title, Author+ )〉
〈! ELEMENT Title (# PCDATA)〉
〈! ELEMENT Author (# PCDATA)〉
〈! ELEMENT ISBN (# PCDATA)〉
```

(2) *catalog.dtd*的 XML文档实例

```
〈Catalog〉
  〈Book〉
    〈Title〉 Theory of Computation 〈/Title〉
    〈Author〉 Derick Wood 〈/Author〉
    〈ISBN〉 0-06-047208-1〈/ISBN〉
  〈/Book〉
  〈Paper〉
    〈Title〉XML Path Language〈/Title〉
    〈Author〉 James Clark 〈/Author〉
    〈Author〉 Steve DeRose 〈/Author〉
  〈/Paper〉
  〈Paper〉
    〈Title〉 Semistructured Data 〈/Title〉
    〈Author〉 Peter Buneman 〈/Author〉
  〈/Paper〉
  …
〈/Catalog〉
```

若需要定位上述文档中“Semistructured Data”这篇文献前面的所有文献,采用 XPath可以将定位需求表示为“*child*::* [*following-sibling*:: *Paper* [*child*:: *Title*= “Semistructured Data”]]”,或将这个需求转换成“定位整个文档的前两篇文献”,并用XPath表示为“*child*::* [*position*()= 1 or *position*()= 2]”.

上面使用到的表示文档结点对象位置的函数 *position*(),表示结点对象次序关系的标识符 *following-sibling*以及表示结点对象层次关系的标识符 *child*都是 XPath的基于模式定位表示方法中的基本要素.

2.2　实例 2

*catalog.dtd*的文档结构是简明的,下面的这个文档类型定义片段相比之下有些复杂,它可以说明XPath表达能力的欠缺,

```
…
〈! ELEMENT queer (((C,D)* | E)* , A* ,
                   ((D,C)* | E)* , A* ) 〉
…
```

如果把 *queer*内容模型(content model)中前面的 A称为 A_1,后面的称为 A_2,希望给出一个结点定位模式,定位符合上述文档类型定义的一批文档中的所有 A_2要素实例.

一类文档中的对象定位问题应该从文档一般结构的角度来考虑,就本例而言,*queer*内容模型中的一般结构模式 $(D,C)^*$ 是区分 A_1和 A_2要素实例的关键模式,但这种一般结构层次上的模式却无法用XPath的现有手段表示出来,原因在于需求驱动影响下的 XPath很大程度上是面向文档实例而不是文档一般结构的.这样,对于需要借助一般结构模式来表达的定位需求,XPath是无法胜任的.

对于一般结构层次上模式信息的表达,DeRose教授在面向 XML文档的查询语言—— XQuery[3]中提出了解决方案,即“Node Sequence Pattern”概念和相应的表示方法“Node Regular Expression”,本文将它们引入 XPath中并加以改造,使 XPath以及建筑在 XPath基础之上的各种查询系统拥有更加完备的模式定位能力.

3　对 XPath的扩充定义

3.1　结点序列模式

结点序列模式是上下文中结点对象有序排列关系的一般表示模式.如“A,B”是表达要素结点 A和 B在上下文中前后次序关系的一种结点序列模式.

3.2 结点正则表达式 *NRegex*

结点正则表达式(*NRegex*)是用于描述结点序列模式的表达式,下文给出其形式化的定义.该定义在语法上限制了 XQuery 中字符正则表达式(character regular expression)[3]的表达能力,增加了模糊结点模式(fuzzy node pattern)的表示方法,在分割符号上沿用 XPath 的表示方法.关于 XQuery 中 *NRegex* 的定义请参考文献[3].

NRegex 主要部分的 BNF 表示如下:

```
⟨NRegex⟩::= ⟨NSequence⟩| ⟨NChoice⟩
⟨NSequence⟩::= ⟨NPiece⟩| ⟨NPiece⟩',' ⟨NSequence⟩
⟨NChoice⟩::= ⟨NPiece⟩| ⟨NPiece⟩'|' ⟨NChoice⟩
⟨NPiece⟩::= ⟨ElementType⟩⟨Suffix⟩?
              //一般要素类型模式
            |'"' ⟨Charegex⟩'"' ⟨Suffix⟩?
              //要素类型的字符正则表达模式
            |'…' ⟨Loop⟩?
              //模糊结点模式
            |'[' ⟨Predicate⟩']' ⟨Loop⟩?
              //单一谓词模式
            |'(' ⟨NRegex⟩')' ⟨Loop⟩?
              //成组模式
⟨ElementType⟩::= ⟨Name⟩
                  //⟨Name⟩参考 XML规范 1.0[1]
⟨Suffix⟩::= ⟨Loop⟩|'[' ⟨Predicate⟩']'
⟨Loop⟩::= '?'|'*'|'+'|'{' ⟨Counts⟩'}'
⟨Counts⟩::= ⟨Count⟩| ⟨Count⟩'|' ⟨Counts⟩
⟨Count⟩::= unsigned| unsigned'-' unsigned
           //unsigned为无符号整数
⟨Predicate⟩::= ⟨PrediateExpr⟩
           //⟨PrediateExpr⟩参考 XPath规范 1.0[2]
⟨Charegex⟩::= ⟨Charpiece⟩⟨Loop⟩?
⟨Charpiece⟩::= ⟨Char⟩|'[' ⟨Charpiece⟩('∪'
               ⟨Charpiece⟩)*']'|
               '[' ⟨Charpiece⟩⟨Charpiece⟩*']'|'[' ⟨Char⟩'-'⟨Char⟩']'|
               '^' ⟨Charpiece⟩|'[' ⟨Charpiece⟩'-' ⟨Charpiece⟩']'
               //⟨Char⟩参考 XML规范 1.0[1]
```

必要的说明:

(1) 以上的 BNF 定义的描述并不完全,定义中⟨*ElementType*⟩、⟨*Char*⟩以及⟨*PredicateExpr*⟩将分别遵照 XML 规范 1.0[1]以及 XPath 规范 1.0[2]中相关的定义.

(2) ⟨*Charegex*⟩的定义源于 XQuery 的 Character Regular Expression[3],在 XQuery 中,Character Regular Expression 的表示功能十分强大,可以用来表示任意的复杂结点序列模式.本文根据实际需要将⟨*Charegex*⟩的功能限定为表示一定格式的要素类型,这样⟨*Charegex*⟩将与⟨*ElementType*⟩的地位对等.⟨*Charegex*⟩在使用时两端加上了引号分割符,主要是为了避免'*'等字符表达的二义性.

(3) "…"称为模糊结点模式,是本文提出来的一种新的表示手段,它被用来表示任意类型、任意数量的结点序列.附属的⟨*Loop*⟩用于限制结点的出现次数.例如,在实例 2中 A_2 要素实例前出现的结点模式可以用"…, $(D,C)^*$, …"来表示.

(4) ⟨*Counts*⟩的概念在 XML 的内容模型(content model)中是不存在的,它的引入来源于 XQuery 中的⟨*Pick*⟩[3].本文中⟨*Counts*⟩被用来限定模式出现的具体次数,地位与⟨*Loop*⟩中定义的'*','+','?'对等.例如{1| 2}表示相同模式出现1次或2次,{3- 5}表示相同模式出现的3次、4次或5次.

(5) ⟨*Predicate*⟩可以单独使用,也可以作为要素类型的后缀使用,本文推荐仅在结点类型是非要素类型的情况下使用单一谓词模式.

3.3 数据类型和函数的扩充

定义 1. 数据类型 *NRegex Type*,它是结点正则表达式的类型抽象.

结点序列模式匹配函数 *match*:

boolean *match* (*NRegex Type regex*).

match 是一个布尔函数,作为谓词表达式的一部分对 XPath 执行过程中的上下文结点表(context node list)作出判断; *match* 的参数是一个 *NRegex Type* 对象,是上下文结点表中结点序列的期望模式.

实例 2可以用 *match* 函数表达成: "…… \ *A*[*preceding-sibling*::* [*match* (…, (*D*, *C*)*, …)]]";执行时某个 *A* 结点之前的结点序列被整体代入 *match* 函数的形参 *regex* 进行匹配计算,*match* 返回结果为真时保留当前 *A* 结点,结果为假时则放弃它.

将上下文结点表作为结点模式匹配的单位是本文的一个想法,主要基于以下几点考虑:

(1) 上下文结点表是已有的 XPath 执行过程中的一个重要的变量,使用它来存储待匹配的结点序列可以省去额外变量的使用.

(2) 上下文结点表包含要素、属性、名字空间、注释等各种类型的结点,可以满足用户的各种需求.

(3) 上下文结点表可以包含不同层次的结点内

容,基于它的模式匹配将比 XQuery中单一层次上的模式匹配更加灵活.

4 结点序列模式匹配自动机

为了进一步说明模式匹配的执行过程,下文给出一个形式化自动机[4,5]的定义,该定义建立在非确定下推自动机基础上,其中合法要素类型、结点实例、合法谓词等概念是对某个具体文档及其遵循的一般结构(DTD)而言的.

预定义:

(1) $QNodetype$= 合法要素类型∪ {*Root*, *Text*, *Attribute*, *PI*, *Comment*, *NSNode*}, $QNodetype$ 与 XPath数据模型(data model)[2]中的结点类型定义是完全一致的.

(2) NT= $QNodetype$∪ {# *None*, # *Any*}.

(3) *Predicate*定义为文档相关的合法谓词表达式的集合.

(4) E定义为文档中所有结点实例的集合.

(5) 函数 *TypeofNode*: E→ $QNodetype$, ∀*node* ∈ E, *TypeofNode*(*node*)返回结点的类型.

(6) 函数 *ContextCompute*: E× *Predicate*→ boolean, ∀*node*∈ E, ∀*pre*∈ *Predicate*,假定 *node*作为上下文结点时 *pre*的计算结果是 *result*,则 *ContextCompute*(*node*, *pre*)= *result*.

(7) 函数 *Satisfy*: E× NT× *Predicate*→ boolean, ∀*node*∈ E, ∀*nodetype*∈ NT, ∀*pre*∈ *Predicate*,

$$Satisfy(node,nodetype,pre)=\begin{cases} ContextCompute(node,pre) & nodetype=\#\ Any \\ pre & nodetype=\#\ None \\ (TypeofNode(node)=nodetype)\ \text{and} & \\ (ContextCompute(node,pre)) & nodetype\text{为其它类型}. \end{cases}$$

定义 **2**.

结点序列模式匹配自动机 M为一个六元组:

$$M=\{Q,E,\Gamma,W,s,F\},$$

其中: Q为有限状态集; E同预定义; Γ为一个非负整数栈(不考虑栈底符号⊥),栈内符号 *symbol*∈ {⊥}∪ {*n*: *n*≥ 0 and *n*∈ *I*},⊥规定为栈底符号; W为转换关系,$W\subseteq Q\times NT\times P\times\Gamma\times Q\times\Gamma^*$; $s\in Q$为开始状态; $F\subseteq Q$为结束状态集;

M中的一个格局(configuration)定义为⊥($\Gamma-\{\perp\}$)* $Q E^*$ 中的一个元素;

给定一个格局 $gcpax$,其中 $g\in\Gamma^*$, $c\in\Gamma$, $p\in Q$, $a\in E\cup\{\lambda\}$, $x\in E^*$,当存在元组(p, *nodetype*, *pre*, c, q, c')∈ W且 *Satisfy*(a, *nodetype*, *pre*)结果为真时,格局 $gcpax$ 可以转换成 $gc'qx$,记为 $gcpax\vdash gc'qx$;格局转换的传递闭包记为$\vdash^+$,自反传递闭包记为$\vdash^*$.

一个结点序列 x 能被 M 接受,当且仅当存在格局转换序列$\perp sx\vdash^*\perp f, f\in F$;

M接受的语言 $L(M)=\{x: x\in E^*$ 并且 x 被 M 接受$\}$.

必要的说明:

(1) 为了语言定义的方便,输入符号表 E包含了所有的结点实例,而不仅仅是结点实例的类型.

(2) Γ栈用于存储结点序列模式的计数值,鉴于模式的嵌套性,栈结构的采用是自然的.

(3) ∀*tuple*∈ W,如果 *tuple* 的 NT 分量为 # *None*,则该元组 *tuple*称为一个λ转换,在做λ转换时不需要当前输入结点的参与,输入读头也不用向后移动;λ转换主要用于结点序列模式的计数.

(4) ∀*tuple*∈ W,如果 *tuple*的 NT分量为# *Any*,则该元组 *tuple*称为一个 *Any*转换,在做 *Any*转换时不需要匹配结点的类型,其它动作与一般转换相同; *Any*转换用于模糊结点模式的匹配.

(5) 本文定义的自动机不是一个单一的识别机器,而是一个组合机器,因为识别过程中对谓词表达式的计算以及对字符正则表达式的识别需要专门的机器来完成.

(6) 几种模式实例的状态转换图分析:

① 选择成组模式: 实例 1"$B|C|D$",如图 1所示:

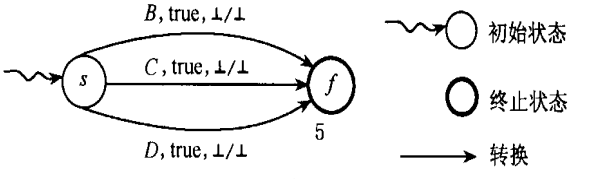

图 1

② 与〈*Loop*〉有关的顺序成组模式: 实例 2 "(B{3}, C) {2}",如图 2所示:

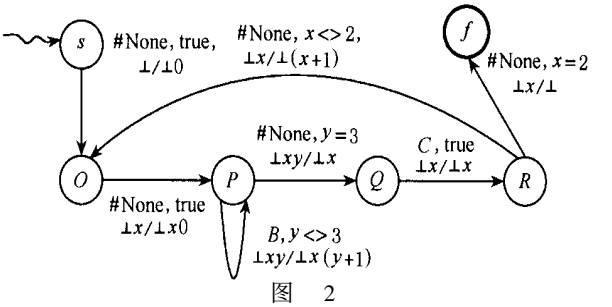

图 2

③ 要素类型的字符正则表达模式、模糊结点模式、带谓词的一般要素类型模式: 实例 3 “ $IE\{3\}$” , … , B [@ *singleAttr* = “ single”] ,如图 3所示:

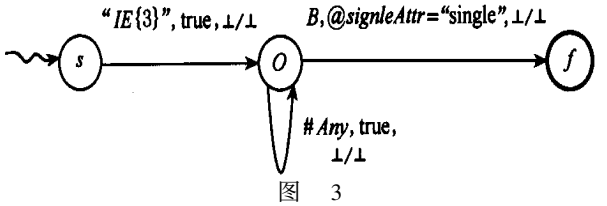

图 3

其中字符正则表达式“ $IE\{3\}$”和谓词表达式 @ *singleAttr* = “ single” 将分别由适当的自动机来识别和计算 ,篇幅所限 ,不再展开描述.

(7) 关于执行模型 ,以上的形式化表示对于理论分析、优化分析等是重要的 ,但最终的开发实现需要绑定到某种算法、某种程序设计语言和某种执行平台之上 ,就目前业界的状况来说 ,在和 XPath的实现不冲突的前提条件下基于 W3C的 DOM API进行面向对象的开发是一种较好的选择.

5 结束语

本文分析了 XPath在一般结构层次上的结点模式表达能力的不足 ,继而引入结点序列模式的概念 ,并在这个概念基础上扩充定义了 XPath.本文的扩充一方面借鉴了 XQuery中的许多思想和表示方法 ,另一方面根据实际需求提出了新的表示方法 ,使扩充后的 XPath具有更加完善的定位能力 .

一般来说 ,在基于模式的查询处理中 ,模式的选择和表示是一个问题的两个不同方面 ,其中模式表示问题是本文讨论的重点 ,至于模式选择问题 ,有一点是值得注意的 ,即模式的选择受到文档一般结构与文档实例之间的多对多关系的影响 ,例如 ,在实例 2中既可以选择 $(D,C)^*$ 作为区分 A_1 和 A_2 要素实例的关键模式 ,也可以选择 $(D,C)^+$ 模式 ,具体的选择结果将视需求而定.篇幅所限 ,模式选择问题不再展开讨论.

有关 XPath模式定位能力扩充的进一步工作包括结点正则表达式表达能力的评测、结点正则表达式的优化分析以及在实现模型上对扩充的数据类型和函数的开发等.

参 考 文 献

1 Tim Bray, Jean Paoli, C M Sperberg-McQueen. Extensible Markup Language(XML), version 1.0, 1998. http://www.w3.org/TR/1998/REC-xml-19980210

2 James Clark, Steve DeRose. XML Path Language(XPath), version 1.0, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116

3 DeRose, Steven J. XQuery: A unified syntax for linking and querying general XML documents. In: Proc of QL'98- The Query Languages Workshop. Boston: World Wide Web Consortium, 1998

4 Derick Wood. Theory of Computation. New York: Harper & Row Publishers Inc, 1987

5 Hartmut Liefke. Horizontal query optimization on ordered semistructured data. In: WebDB'99. 1999. http://citeseer.nj.nec.com/246796.html

王 强 男 ,1976年生 ,硕士研究生 ,主要研究领域为多媒体文档技术、置标系统.

武港山 男 ,1968年生 ,副教授 ,主要研究领域为多媒体文档技术、置标系统.