维普资讯 http://www.cqvip.com

XQuery 语言的形式化语义

杨红丽[1,2] 韩俊刚[2] 郝克刚[1]

(西北大学计算机科学系 西安 710069)[1] (西安邮电学院计算机系 西安 710061)[2]

摘 要 XQuery 语言用于查询 XML 文档。目前,该语言规范还是 W3C 的工作草稿。语言的形式化语义有助于语言的标准化,本文通过重用 XML 家族语言通用语义构件的方法,形式化建模 XQuery 语言的语义,语义的描述采用 Object-Z 规范语言。这种面向对象的语义描述不仅具有简洁性、可扩展性和可组合性,而且有助于规范之间的一致性和协调性。

关键词 XML, XPath, XML Schema, Object-Z, XQuery

The Formal Semantics of XQuery Language

YANG Hong-Li[1,2] HAN Jun-Gang[2] HAO Ke-Gang[1]

(Deparmtment of Computer, Northwest University, Xi'an 710069)

(Department of Computer, Xi'an Institute of Post & Communications, Xi'an 710061)[2]

Abstract XQuery language is used to search XML document. The specification of the language is W3C working draft now. The formal semantics will help to the standization of the language. This paper models the semantics of XQuery language by reusing the common semantic construct of XML family languages. The presentation of semantics uses formal specification language Object-Z notation. This object-oriented description is not only concise, extensible and composable, but also consistency between specifications.

Keywords XML, XPath, XML schema, Object-Z, XQuery

1 XQuery 简介

XML[14,21]作为应用程序之间交换数据的标准格式,应用程序数据可以存储为 XML 文档格式,也可以存储在关系数据库中,这样就需要设计的 XML 文档查询语言能查询各种数据源。XQuery[4,7,17]正是为此目的而设计的查询语言,其工作组开始于 1999 年 10 月。目前 XQuery 规范仍然处于不断变化中,工作组已经发表了其工作草稿,最新也是最重要的工作草稿是文[18]。XQuery 是 Xpath[3]的扩展,任何表达式如果在 Xpath 和 XQuery 中语法有效且成功执行,则其计算结果相同。由于两种语言如此紧密联系,两个规范的编辑者也一起工作,以确保两种语言的一致性。

XQuery 的类型系统基于 XML Schema[16],包括全局/局部元素和属性声明、复杂和简单类型定义、命名和匿名类型、通过限制和扩展获取新的类型,以及列表和联合类型等。XQuery 的操作对象是 XML 文档的抽象逻辑结构[15]。XQuery 的表达式类型有:路径表达式、元素构造器、函数调用、算术和逻辑表达式、条件表达式、量词表达式、序列类型上的表达式等。这些表达式由关键字、符号和操作数构成。表达式的操作数一般也是表达式。

XQuery 有两个重要的设计特点:

· XQuery 是一个功能语言,语言的每个构件是一个表达式,表达式可以任意组合,表达式的结果可以用于任何其它表达式的输入,只需满足前一个表达式的结果类型匹配后一个表达式的输入类型。

· XQuery 是一个类型语言,类型可以从多个 XML Schema 文档输入。XQuery 语言对这些类型进行操作。此外,XQuery 也支持静态类型分析,系统可以推理一个表达式的输出类型,静态类型容许早期错误识别。

2 面向对象的语义描述

2.1 重用的 XML 家族通用语义构件

XML 语言属于 XML 家族语言,因此可重用 XML 家族通用语义构件,包括:XML Schema[10,16,20]类型定义;XML 文档数据模型[15];Xpath[3]表达式。这里简单介绍本文用到的 XML 家族通用语义构件,详细的描述参见文[2]。

XML Schema 中共有 19 种原子类型,如:字符串类型 String、布尔类型 Bool、名称类型 QName、整数类型 Integer 等。原子类型定义为 Z[12,13]自由类型:

AtomicType == String | QName | Bool | Integer |…

XML 文档数据模型中最重要的数据类型是结点。一个结点可以是下面七种结点类型之一:文档结点(document)、元素结点(element)、属性结点(attribute)、文本结点(text)、注释结点(comment)、处理指令结点(processing instruction)、名字空间结点(namespace)。结点类型 nodeType 定义为 Z 自由类型。

$$nodeType ::= docTypet \mid elemType \mid attrType \mid textType \mid commType \mid piType \mid nmsType$$

结点类 Node 包括属性:类型 type、名称 name、结点值 valve、双亲 parent、孩子结点序列 *children*、属性结点序列 *attribute*、名字空间结点序列 *namespace*。此外还包括辅助属性:文档结点 *doc*、子孙结点序列 *descend*、祖先结点序列 *ancestor*、结点标识 *id*。

Node

type: nodeType
name: QName
value: String
parent: Node
children, attribute, namespace: seq Node
△
id: ID
doc:Node
descend, ancestor: seq Node

Xpath 中最重要的类型是表达式,一个表达式可以是基

维普资讯 http://www.cqvip.com

本表达式、路径表达式、序列表达式、算术表达式、比较表达式、逻辑表达式、条件表达式、量词表达式、序列类型表达式或者 *for* 表达式。所有表达式的公有属性包括：类型 *type*、值 *val*，以及上下文结点 *context*，建模为类 baseExp（在文[2]中定义）。Xpath 表达式类 Exp 定义为联合类(class union)：

$$Exp \triangleq primExp \cup pathExp \cup seqExp \cup arithExp \cup compExp \cup logicExp \cup condExp \cup quanExp \cup seqtypeExp \cup forExp$$

XQuery 定义为 Xpath 的超集。除了 Xpath 中描述的九种表达式外(不包括 forExp 表达式)，XQuery 还包括：FLWR 表达式和构造器(constructors)表达式。

XQuery 的表达式类(xqExp)定义为联合类。图 1 表示 XQuery 表达式类的组成关系。

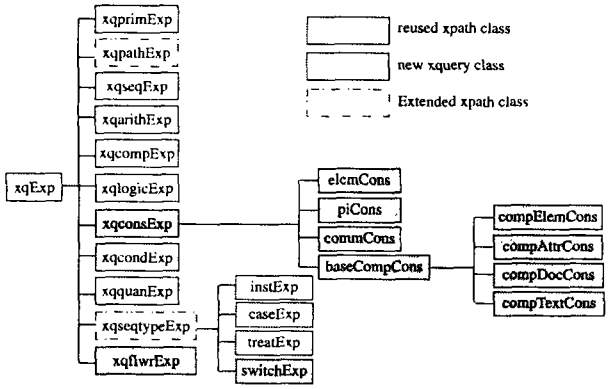

图 1　XQuery 表达式类

$$xqExp \triangleq xqprimExp \cup xqpathExp \cup xqseqExp \cup xqarithExp \cup xqcompExp \cup xqlogicExp \cup xqcondExp \cup sqquanExp \cup xqseqtypeExp \cup xqflworExp \cup xqconsExp$$

基本表达式 *xqprimExp* 与 Xpath 中类 *primExp* 等价。

$$xqprimEx\ p == primExp$$

XQuery 中的路径表达式与 Xpath 中的稍微不同。XQuery 支持的搜索轴有：*child*、*desc*、*attr*、*self*、*dslesOrself* 以及 *parent*。XQuery 的轴类型定义为：

$$xqAxis ::= child \mid desc \mid attr \mid self \mid dslesOrself \mid parent$$

XQuery 中的步表达式类 xqStep 的定义通过替换相应的 Xpath 步表达式类 *Step* 中的轴类型 *Axis* 为新的轴类型 xqAxis。

xqStep

Step[*xqAxis*/*Axis*]

XQuery 的路径表达式类 *xqpathExpr* 的定义通过替换相应的 XPath 路径表达式类 *pathExp* 中类 *Step* 为类 xqStep。

xqpath

pathExp[*xqStep*/*Steq*]

XQuery 中的序列表达式、算术表达式、比较表达式、逻辑表达式、条件表达式和量词表达式都与相应的 XPath 表达式类等价。

$$xqseqExp == seqExp$$
$$xqarithExp == arithExp$$
$$xqcompExp == compExp$$
$$xqlogicExp == logicExp$$
$$xqcondExp == condeExp$$
$$xqquanExp == quanExp$$

2.2　typeswitch 表达式

XQuery 中序列类型表达式同 XPath 中一样，支持 *instanceOf*、*cast* 和 *treat* 表达式。另外，XQuery 还支持序列类型上的 *typeswitch* 表达式。XQuery 的序列类型表达式类 *xqseqtypeExp* 定义为联合类。

$$xqseqtypeExp \triangleq instExp \cup castExp \cup treatExp \cup switchExp$$

typeswitch 表达式包含属性：操作数表达式 *test*(其类型将被测试)、一个或多个 *case* 语句(每个包含一个类型和一个表达式)。*case* 语句定义为类 *Case* 的对象。

Case

type : *scalType*
exp : *Exp*

type_of (*exp*) = *seqType*

类 switchExp 定义如下：

switchExp

baseExp

test: *Exp*
case: *seqCase*
default : *Exp*

$\#case \geq 1$
$(\exists i : 1..\#case \cdot case(i).type = test.type) \Rightarrow val = case(i).exp.val$
$(\exists i : 1..\#case \cdot case(i).type = test.type) \Rightarrow val = default.val$

typeswitch 表达式的计算：依次测试 test 表达式的动态类型是否匹配 case 语句中的类型，第一个类型匹配的语句称为 *effective case*，该语句的表达式将被执行，其执行结果作为整个 *typeswitch* 表达式的结果；如果没有任何 *case* 语句匹配 *test* 表达式的类型，则整个 *typeswitch* 表达式的结果为缺省表达式(*default*)的计算结果。

2.3　FLWR 表达式

XQuery 提供 FLWR 表达式用于支持循环和中间结果的变量绑定。名称 FLWR 来源于关键字：*For*、*Let*、*Where*、*Return*。FLWR 表达式中 *for* 和 *let* 语句生成一个变量索引序列；*where* 语句用于过滤变量绑定元组；*return* 语句返回 FLWR 表达式结果。

for 语句可能包含多个变量，每个变量与一个表达式相连，表达式返回一个序列值，变量可以依次取序列中的每个值，这样形成一个变量绑定元组列表，在多个变量的情况下，是多个变量取值序列的笛卡尔积。一般，变量绑定元组列表保留变量取值序列的顺序。类 *For* 包含变量名序列 *nms*、表达式序列 *exps*，以及辅助属性：变量索引序列 *vefs*。

let 语句可能包含一个或多个变量，每个变量关连一个表达式。不同于 *for* 语句，一个 *let* 语句绑定每个变量与其关连的表达式的值。

For

nms : seq *String*
exps : seq *Exp*
△
refs : *Varef*

$\#nms = \#exps = \#refs$
$\forall i : 1..\#nms \cdot refs(i).nm = nms(i) \wedge refs(i).val \in ran\ exps(i).val$

维普资讯 http://www.cqvip.com

Let

nms : seq *String*
exps : seq *Exp*
△
refs : *Varef*

 # nms = # exps = # refs
∀ *i* : 1.. # nms • *refs*(*i*). *nm* = *nms*(*i*) ∧ *refs*(*i*).*val* = exps(i).val

下面的 *for* 语句和 *let* 语句生成一个变量绑定元组序列，包括三个绑定元组：

```
for $i in (1 to 3)
let j:=(1 to $i)
$i = 1, $j = 1
$i = 2, $j = (1,2)
$i = 3, $j = (1,2,3)
```

类 *xqflwrExp* 需要定义下面的类型和函数：

(1)类型 Binds 定义为变量绑定元组序列：

Binds == seq seq *Varef*

(2)函数 *binding* 用于构造变量绑定序列：

| *binding* : *For* × *Let* → *Binds*

(3)函数 *evalSingle* 用于计算包含单个变量绑定元组的 FLWR 表达式的值。

(4)函数 *evalflwr* 用于计算一个 FLWR 表达式的值。

evalSingle: seq Varef × *Exp* × *Exp* → *seqVal*

∀ *s: seq Varef; wh:Exp; re:Exp; val:seqVal* •
where.val ⇒ *val = re.val*

evalflwr: Binds × *Exp* × *Exp* → *seqVal*

∀ *b: Binds; wh:Exp; re:Exp; val:seqVal* •
where.val ⇒ *val = re.val*
 # b = 1 ⇒ *val = evalSingle*(*b*(1),*wh*,*re*)
 # b = 1 ⇒ *val* = *evalSingle*(*b*(1),*wh*,*re*)⌢ *evalFLWR*(tail *b*,*wh*,*re*)

类 *xqflwrExp* 包含属性：*for* 表达式、*let* 表达式、*where* 表达式，以及辅助属性：变量索引序列 *refs* 和变量作用范围 *vscope*。

xqflwrExp

baseExp

for : *For*
let : *Let*
wh : Exp
re : *Exp*
△
type = *seqType*
refs : *seq Varef*
vscope : *QName* ⤖ *VarLoc*

#{v:(Varef ∩ scontain) • v.nm} = #(Varef ∩ scontain) [pred1]
{vr:(Varef ∩ scontain) • *v.nm* α *vr.vl*} ⊆ *vscope* [pred2]
vscope = {vr:ran *for.refs* • *v.nm* α *vr.vl*} ∪ {vr:ran *let.refs* • *v.nm* α *vr.vl*} [pred3]
 # refs = #*for.refs* + #*let.refs*
val = *evalflwr*(*binding*(*for*,*let*),*wh*,*re*)

不变性 *pred*1 说明：FLWR 表达式中没有两个不同的变量有相同的名称。不变性 *pred*2 说明：变量的发生绑定在其作用范围内。不变性 *pred*3 说明：FLWR 表达式的作用范围是所包含的 *for* 语句及 *let* 语句中变量的作用范围。

2.4 构造器

XQuery 提供构造器(constructors)表达式用于在查询中创建 XML 结构。XQuery 提供了 XML 文档数据模型[15]中六种结点类型的构造器(不包括名字空间结点类型)。

一个构造器可以是元素构造器、注释构造器、处理指令构造器或者计算构造器(computed constructor)，用于创建一个命名的元素或属性结点，或者创建一个文档结点或文本结点。

构造器类 *xqconsExp* 定义为联合类：

xqconsExp::=*elemCons* | *piCons* | *commCons* | *compElemCons* | *compAttrCons* | *compDocCons* | *compTextCons*

所有构造器的共有属性建模为类 baseCons，该类继承基本表达式类 baseExp。

baseCons

baseExp

type = *seqType*
 # val = *1*
type_of(*val* (*1*)) = *nodeType*

元素构造器创建一个 XML 元素，其结果是一个新的元素结点。类 *elemCons* 包含属性：新元素名称字符串 *nm*、属性结点序列 *attrs*、内容结点序列 cont。

elemCons

baseCons

nm : *QName*
attrs : *seq Node*
cont : *seq Node*

∀ *a* : *Node* • a ∈ ran *attrs* ⇒ a.*type* = *attrType*
∀ *a* : *Node* • a ∈ ran *attrs* ⇒ a.*type* ∈ {*elemType*,*textType*, *commType*,piType}
val(1).*name* = *nm*
val(1).*type* = *elemType*
val(1).*attribute* = *attrs*
val(1).*children* = *cont*

处理指令构造器包含名称 nm 和内容 cont 属性，类 piCons 定义如下：

piCons

baseCons

nm : *QName*
cont : *String*

val(1).*type* = *piType*
val(1).*name* = *nm*
val(1).*value* = *cont*

注释构造器包含字符串内容 *cont*，类 *commCons* 定义如下：

commCons

baseCons

cont : *String*

val(1).*type* = *commType*
val(1).*value* = *cont*

可计算的构造器(computed constructor)包含一个关键字用于标识所创建的结点类型，如：*element*、*attribute*、*document*、*text*，结点类型 *nodeType*1定义为：

*nodeType*1 == *element* | *attribute* | *document* | *text*

所有可计算构造器的通用属性描述为类 *baseCompCons*，该类继承基本构造器类 *baseCons*。

维普资讯 http://www.cqvip.com

baseCompCons

baseCons

ndType : nodeType1
cont : Exp

可计算的元素构造器包含属性：名称表达式 *nm*，计算结果为新创建元素的名称；内容表达式 *cont*，计算结果为结点序列，用于构造新创建元素的孩子结点。类 *compElemCons* 定义为：

compElemCons

baseCompCons

nm : *Exp*

ndType = *element*
type_of (*nm.val*) = *QName*
cont.type = seq *Node*
val(1).*type* = *elemType*
val(1) = *create*1(*nm.val*,*cont.val*)

其中，函数 *create*1根据结点名称和内容序列值创建一个新的元素结点。

create1 : QName × seq *Node* → *Node*

∀ *nm* : QName; *node* : *seq Node*; •
n.type = *elemType*
n.name = *nm* ∧
∀ *x* : *Node* | *x* ∈ ran *nodes* •
x.type = {*textType*,*elemType*,*commType*,*piType*} ⇒
x ∈ ran *n.children*

可计算的属性构造器包含属性：名称表达式 *nm*，用于计算新创建属性结点的名称；内容表达式 *cont*，用于计算新创建属性结点的值。类 *compAttrCons* 定义如下：

compAttrCons

baseCompCons

nm : *Exp*

ndType = *attribute*
cont.type = seq *String*
type_of (*nm.val*) = *QName*
val(1).*type* =*attrType*
val(1).*name* = nm.val
val(1).*value* = *catStr*(*cont.val*)

其中，函数 *catStr* 用于连接多个字符串，返回单个字符串。

catStr : sep *String* → String

∀ *s* : sep *String* •
s = 〈 〉 ⇒ *catStr*(*s*) = *empty*
s ≠ 〈 〉 ⇒ *catStr*(*s*) = *s*(1)⌢*catStr*(tail *s*)

compDocCons

baseCompCons

ndType = *document*
val(1).*type* = *docType*
type_of(*cont*) = seq *Node*
val(1).*children* = *cont.val*

可计算的文档构造器类 *compDocCons* 继承类 *baseCompCons*，因此包含该类所有属性，其内容表达式 *cont* 计算的结果结点序列将作为新创建文档结点的孩子。

可计算的文本结点类 *compTextCons* 继承类 *baseCompCons*，包含的内容表达式 *cont* 计算的结果字符串将作为新创建的文本结点的值。

compTextCons

baseCompCons

ndType = *text*
val(1).*type* = *textType*
type_of(*cont*) = seq *String*
val.value = *catStr*(*cont.val*)

3 相关研究及比较

3.1 相关研究

XQuery 的语义已经在文[5,8,11]中描述，这些方法提出的 XQuery 核为相对较小但却富有表达力的 XQuery 子语言，通过类型推理规则描述其静态语义，通过值推理规则描述其动态语义。这种语义还给出从核心子语言的表达式到 XQuery 数据的简单操作的映射，另外还定义了 XQuery 语言到其核心子语言的映射，目的是使所有语义构件都可以直接用于 XQuery 语言的高效实现。

本文用于描述 XQuery 语义的方法与上面所述的方法完全不同，采用面向对象方法描述 XQuery 语言的指称语义，重用了 XML 家族语言通用语义构件。这种方法的好处是：每种语言构件的抽象语法、静态和动态语义都包括在一个类中，不仅有助于语义的可读性，而且如果语言加强，可以很容易地对现有的语义进行修改。通过重用 XML 家族通用语义构件，也有助于说明各规范之间的相互依赖性，保持规范间的一致性。

3.2 XQuery 与 XSLT 的比较

XQuery 和 XSLT[6,22]都是基于抽象数据模型的高级描述性语言。两者的应用领域不同：XSLT 主要用于 XML 文档转换，XQuery 主要用于 XML 文档查询。从语义的角度比较两种语言，可以发现两者在功能上正在收敛：

·都对 XML 文档的抽象树结构进行操作；

·都基于 XPath 表达式语言：XSLT 采用 XPath 作为子语言，XQuery 是 XPath 的超集；

·都采用 XML Schema 数据类型系统；

·都可以创建新的结点(node)，不同的是：XSLT 用构造指令实现，XQuery 用构造表达式实现。

XSLT 中有一些语言特点(features)没有包括在 XQuery 中，即模板规则，这是因为它们专门用于转换，而 XQuery 中几乎所有功能都包括在 XSLT 中。目前已经有一些研究，主要实现两种语言间的相互转换，因为 XSLT 语言已经有许多成熟的开发工具，可以很方便地首先开发 XSLT 应用，然后转换为 XQuery 应用。

小结 XML 文档查询语言 XQuery[18]仍然处于演化阶段，其语言规范目前仅为工作组发布的工作草稿(15/11/2002)。本文通过重用 XML 家族通用语义构件的方法建模每个 XQuery 语言构造为 Object-Z[9,19]类。这种面向对象的语义描述方法有利于语义的重用，随着 XQuery 语言的不断发展，可以很容易地修改语言的语义。

(下转第27页)

维普资讯 http://www.cqvip.com

26 Filar J, Vrieze K. Competitive Markov decision process. Springer-Verlag, 1997. Theorem 4.6.4

27 Hu J, Wellman M P. Online learning about other agents in a dynamic multi-agent system. In: Second Intl. Conf. on Autonomous Agents, Minneapolis, 1998. 239～246

28 Bowling M, Velsoso M. Convergence of gradient Dynamics with a variable learning rate. In: Proc. of the Eighteenth Intl. Conf. on Machine Learning, Williams College, June 2001. 27～34

29 Bowling M, Velsoso M. Multiagent learning using a variable learning rate. Artificial Intelligence, 2002

30 Astom K J. Optimal control of Markov decision processes with incomplete state estimation. J. Math. Anal. Appl., 10: 174～205

31 Sondik E J. The optimal control of partially observable Markov decision processes: [Ph. D. thesis]. Stanford University, 1971

32 Singh S P, Jaakkola T, Jordan M I. Model-free reinforcement learning for non-Markovian decision problems. In: Proc. of the 11th Intl. Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1994. 288～292

33 Kaelbling L P, Littman M L, Cassandra A R. Planing and acting in partially observable stochastic domains. Artificial Intelligence, 1998, 101: 99～134

34 Smallwood R D, Sondik E J. The optimal control of partially observable Markov decision processes over a finite horizon. Operations Research, 1973, 21: 1071～1088

35 Sondik E J. The optimal control of partially observable Markov decision processes over the infinite horizon: Discounted cost. Operations Research, 1978, 26: 282～304

36 Madani O, Hank S, Condon A. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov Decision Processes. In: Proc. of the 16th National Conf. on Artificial Intelligence (AAAI-99), 1999

37 Hauskrecht M. Planing and control in stochastic domains with imperfect information: [Ph. D. thesis]. Massachusetts Institute of Technology, Cambridge, 2000

38 Littman M L, Cassandra A R, Kaelbling L P. Learning policies for partially observable environments: scaling up. In: Proc. of the 12th Intl. Conf. on Machine Learning, 1995. 362～370

39 Lovejoy W S. A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes. Annals of Operations Research, 1991, 28: 47～66

40 Lovejoy W S. Suboptimal Policies with Bounds for Partially Observed Markov Decision Processes. Annals of Operations Research, 1993, 41: 583～599

41 Simmons R, Koenig S. Probabilistic Robot Navigation in Partially Observable Environments. In: Proc. of Fourteenth Intl. Joint Conf. on Artificial Inteligence, Montreal, Canada, 1995. 1080～1087

42 Thrun S, Fox D, Burgard W. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. Machine Learning, 31: 29～53

43 Arai S, Sycara K. Effective learning approach for planning and scheduling in multi-agent domain. In: Proc. of the 6th Intl. Conf. on Simulation of Adaptive Behavior, 2000. 507～516

44 荒井幸代. 多 Agent 强化学习: 面向实用化的课题、理论和诸技术的结合. 人工智能学会志, 16卷4号, vol7, 2001

45 Matatric J. Reinforcement learning in the multi-robot domain. Autonomous Robots, 1997, 4: 77～83

46 宫崎和光, 荒井幸代, 小林重信. Profit-sharing 在多 Agent 系统强化学习报酬分配中的理论研究. 人工智能学会志, 14卷6号, 1999. 1157～1164

47 Singh S, Jaakkola T, Littman M L, Szepesv'ari C. Convergence results for single-step on-policy reinforcement learning algorithms. Machine Learning, 2000

48 Stone P, Veloso M. MultiAgent systems: A survey from a machine learning perspective. Autonomous Robots, 2000, 8(3)

49 Humphrys M. Action selection methods using reinforcement learning. In: Proc. of the Eighteenth Intl. Conf. on Simulation of Adaptive Behavior, 1996. 135～144

50 Boutilier C. Planning, learning and coordination in multiagent decision processes. In: Proc. of the Sixth Conf. on the Theoretical Aspects of Rationality and Knowledge, 1996. 195～210

(上接第18页)

致谢 感谢联合国大学软件技术研究所对该研究的资助。

参 考 文 献

1 See http://www.w3.org.

2 杨红丽, 郝克刚, 韩俊刚. 基于 Object-Z 的 XPath 形式化语义. 计算机科学, 2003, 30(2)

3 Boag S, et al. XML Path Language (XPath) 2.0. 2002. http://www.w3.org/TR/xpath20/

4 Per Bothner. What is XQuery?

5 Simeon J, Choi B, Fernandez M. The XQuery Formal Semantics: A Foundation for Implementation and Optimization, May 2002

6 Clark J. XSL Transformations (XSLT) Version 2.0, 2002. http://www.w3.org/TR/xslt20/

7 Chamberlin D. XQuery: An XML query language

8 Fernández M, et al. XQuery 1.0 and XPath2.0 Formal Semantics. Nov. 2002. http://www.w3.org/TR/2002/WD_query_semantics_20021115/

9 Duke R, Rose G. Formal Object Oriented Specification Using Object_Z. Macmillan. 2000

10 David C. Fallside. XML Schema part 0: primer, 2001

11 Fankhauser P. XQuery Formal Semantics State and Challenge, 2001

12 Davies J, Wookcock J. Using Z: Specification, Refinement and Proof. Prentice Hall, 1996. http://www.comlab.ox.ac.uk/igdp/usingz

13 Spivey J M. The Z Notation: A Reference Manual. Prentice Hall, 1992

14 Diamond J, et al. Professional XML 2nd Edition. Wrox Press Ltd., 2001

15 Marsh J, et al. XQuery 1.0 and XPath 2.0 Data Model, 15 November 2002. http://www.w3.org/TR/2002/WD-query-datamodel-200211115/

16 Malhotra A, Biron P V. XML Schema Part 2: DAtatypes, 2001. htt://www.w3.org/TR/2001/REC-REC-xmlschema-2-200105-02/

17 Peter. XQuery Formal Semantics State and Challenge. http://ibm.com/developerWorks

18 Fernandez M F, et al. Scott Bodg, Donchamberlin. XQuery 1.0: An XML Query language. 15 November 2002. http://www.w3.org/TR/2002/WD-query-datamodel-200211115/

19 Smith G. The Objice-Z Specification Language. Kluwer Academic Publishers, 1999

20 Thompson H S, Beech D, Maloney M, Mendelsohn N. XML Schema Part 1: Structures, 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010102

21 Sperberg-McQueen C M, Bray T, Paoli J. Extensible Markup Language(XML) 1.0 (Second Edition). 2000. http://www.w3.org/TR/REC-xml/

22 Yang H L, Dong J S, Hao K G, Han J G. Formalizing Semantics of xslt using object-z. In: Proc. of APWeb2003: Asia Pacific Web Conference. Springer-Verlag, April 2003