Et tu, XML?

Philip Wadler, Avaya Labs

wadler@avaya.com
Acknowledgements

This talk is joint work with:

Mary Fernandez (AT&T)
Jerome Simeon (Lucent)
The W3C XML Query Working Group

Disclaimer: This talk. is a personal view.
Other members of XML Query may disagree.
Prologue

Friends, romans, computer scientists, lend me your ears! I come to bury the relational database, not to praise it. The evil that standards committees do lives after them; the good is oft interred with their core dumps; so let it be with relations. The noble XML hath told you that relations were ambitious. Here under leave of XML and the W3C — for XML is an honourable standard; so are they all, all honourable standards — come I to speak at relations funeral. Relations were my friend, faithful and just to me. But XML says relations were ambitious, and XML is an honourable standard. Relations hath brought many captives home to Rome whose ransom did the coffers of Oracle, IBM, and Microsoft fill. Did this in relations seem ambitious? Yet XML says relations were ambitious, and XML is an honourable standard.
<SPEECH>
  <SPEAKER>ANTONY</SPEAKER>
  <LINE>Friends, Romans, countrymen, lend me your ears;</LINE>
  <LINE>I come to bury Caesar, not to praise him.</LINE>
  <LINE>The evil that men do lives after them;</LINE>
  <LINE>The good is oft interred with their bones;</LINE>
  <LINE>So let it be with Caesar. The noble Brutus</LINE>
  <LINE>Hath told you Caesar was ambitious:</LINE>
  <LINE>Here, under leave of Brutus and the rest--</LINE>
  <LINE>For Brutus is an honourable man;</LINE>
  <LINE>So are they all, all honourable men--</LINE>
  <LINE>Come I to speak in Caesar’s funeral.</LINE>
  <LINE>He was my friend, faithful and just to me:</LINE>
  <LINE>But Brutus says he was ambitious;</LINE>
  <LINE>And Brutus is an honourable man.</LINE>
  ...
</SPEECH>
<LINE>Peace, ho! let us hear him.</LINE>
</SPEECH>
- <SPEECH>
  <SPEAKER>ANTONY</SPEAKER>
  <LINE>Friends, Romans, countrymen, lend me your ears;</LINE>
  <LINE>I come to bury Caesar, not to praise him.</LINE>
  <LINE>The evil that men do lives after them;</LINE>
  <LINE>The good is oft interred with their bones;</LINE>
  <LINE>So let it be with Caesar. The noble Brutus</LINE>
  <LINE>Hath told you Caesar was ambitious;</LINE>
  <LINE>If it were so, it was a grievous fault;</LINE>
  <LINE>And grievously hath Caesar answer'd it.</LINE>
  <LINE>Here, under leave of Brutus and the rest--</LINE>
  <LINE>For Brutus is an honourable man;</LINE>
  <LINE>So are they all, all honourable men--</LINE>
  <LINE>Come I to speak in Caesar's funeral.</LINE>
  <LINE>He was my friend, faithful and just to me;</LINE>
  <LINE>But Brutus says he was ambitious;</LINE>
  <LINE>And Brutus is an honourable man.</LINE>
  <LINE>He hath brought many captives home to Rome</LINE>
</SPEECH> Whose names are untasted of thy lips? <LINE>
(SPEECH
  (SPEAKER "ANTONY")
  (LINE "Friends, Romans, countrymen, lend me your ears;")
  (LINE "I come to bury Caesar, not to praise him.")
  (LINE "The evil that men do lives after them;")
  (LINE "The good is oft interred with their bones;")
  (LINE "So let it be with Caesar. The noble Brutus")
  (LINE "Hath told you Caesar was ambitious:")
  (LINE "Here, under leave of Brutus and the rest--")
  (LINE "For Brutus is an honourable man;")
  (LINE "So are they all, all honourable men--")
  (LINE "Come I to speak in Caesar’s funeral.")
  (LINE "He was my friend, faithful and just to me:")
  (LINE "But Brutus says he was ambitious;")
  (LINE "And Brutus is an honourable man.")
  ...
)

Shakespeare in Lisp
Part I

Some XML applications
The Four Webs

- Computers
- Voice
- Wireless
- Television
The Four Webs

- Computers — xHTML
- Voice — Voice XML
- Wireless — WAP/WML
- Television — bHTML
The Business of VoiceXML

The July/August issue of Speech Technology magazine offers an insightful look at VoiceXML as told through interviews conducted with various VoiceXML Forum members. The article, entitled "The Business Side of VoiceXML: Faster Time to Market is Only Part of the Story", covers topics including:
ebXML enables enterprises of any size, in any location to meet and conduct business through the exchange of XML-based messages.

**ebXML NEWS**
- [01 August 2001] OpenTravel Alliance Endorses ebXML
- [30 July 2001] UN/CEFACT Forms e-Business Transition Ad hoc Working Group
- [21 June 2001] OASIS Forms ebXML Technical Committees
- [22 May 2001] UN/CEFACT and OASIS Launch ebXML SE Working Group

**INDUSTRY SUPPORT**
- Open Applications Group to incorporate ebXML into 162 mature Business Object Documents
- Korea Institute for Electronic Commerce (KIEC) Opens Prototype ebXML Registry & Repository
- Covisint Supports ebXML Technology Findings
- More ebXML Adoption News

**ebXML-DEV MAIL LIST**
Join this open forum to exchange ideas on implementing ebXML.

- Subscribe
- View Archives

Book excerpt: ebXML: The Smart Choice for Business
Discover businesses worldwide that offer the exact products and services that you need. Register the products and services of your own business for others to discover. Or both. Technology and business champions are leading the development and deployment of an open, Internet-based Universal Description, Discovery, and Integration (UDDI) specification. UDDI is the building block that will enable

The following PDFs are available for download:

- Version 2.0 Programmer's API Specification (464 KB)
- Version 2.0 Data Structure Specification (243 KB)
- Version 2.0 Replication Specification (229 KB)
- Version 2.0 Operator's Specification (223 KB)
- Version 1.0 Programmer's API Specification (330 KB)
- Version 1.0 Data Structure Specification (193 KB)
- Executive White Paper (30 KB)
XML Schema Recommendation

XML Schema Part 1: Structures

W3C Recommendation 2 May 2001

This version:
http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
(in XML (with its own DTD, XSL stylesheet) and HTML), with separate provision of the schema and DTD for schemas described herein.

Latest version:
http://www.w3.org/TR/xmlschema-1/

Previous version:
http://www.w3.org/TR/2001/TR-xmlschema-1-20010330/

Editors:
Henry S. Thompson (University of Edinburgh) <ht@cogsci.ed.ac.uk>
David Beech (Oracle Corporation) <David.Beech@oracle.com>
Murray Maloney (for Commerce One) <murray@muzmo.com>
Noah Mendelsohn (Lotus Development Corporation) <Noah_Mendelsohn@lotus.com>

Copyright ©2001 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.
Part II

An introduction to XQuery
Influences

Languages that influence XQuery include:

- SQL
- OQL, O₂
- Nested Relational Algebra, Kleisli
- Xduce
- XML-QL, YatI, Lorel
- XPath, XQL, XSLT
- XML Schema, TRex, Relax
- Quilt
Part III

Data model
Some XML data

<BOOKS>
  <BOOK YEAR="1999 2003">
    <AUTHOR>Abiteboul</AUTHOR>
    <AUTHOR>Buneman</AUTHOR>
    <AUTHOR>Suciu</AUTHOR>
    <TITLE>Data on the Web</TITLE>
    <REVIEW>A truly <EM>fine</EM> book.</REVIEW>
  </BOOK>
  <BOOK YEAR="2002">
    <AUTHOR>Buneman</AUTHOR>
    <TITLE>XML in Scotland</TITLE>
    <REVIEW><EM>Truly the <EM>best</EM> ever!</EM></REVIEW>
  </BOOK>
</BOOKS>
Data model

XML

```xml
<BOOK YEAR="1999 2003">
  <AUTHOR>Abiteboul</AUTHOR>
  <AUTHOR>Buneman</AUTHOR>
  <AUTHOR>Suciu</AUTHOR>
  <TITLE>Data on the Web</TITLE>
  <REVIEW>A truly <EM>fine</EM> book.</REVIEW>
</BOOK>
```

XQuery

```xquery
element BOOK {
  attribute YEAR { 1999, 2003 },
  element AUTHOR { "Abiteboul" },
  element AUTHOR { "Buneman" },
  element AUTHOR { "Suciu" },
  element TITLE { "Data on the Web" },
  element REVIEW { "A truly", element EM { "fine" }, "book." }
}
```
Part IV

Types
DTD (Document Type Definition)

<!ELEMENT BOOKS (BOOK*)>
<!ELEMENT BOOK (AUTHOR+, TITLE, REVIEW?)>
<!ATTLIST BOOK YEAR CDATA #OPTIONAL>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ENTITY % INLINE "(#PCDATA | EM)*">
<!ELEMENT REVIEW %INLINE;>
<!ELEMENT EM %INLINE;>
Schema

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="BOOKS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="BOOK"
          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
<xsd:element name="BOOK">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="AUTHOR" type="xsd:string"
                        minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element name="TITLE" type="xsd:string"/>
            <xsd:element name="REVIEW" type="INLINE"
                        minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:attribute name="YEAR" type="NONEMPTY-INTEGER-LIST"
                            use="optional"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:complexType name="INLINE" mixed="true">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="EM" type="INLINE"/>
  </xsd:choice>
</xsd:complexType>
<xsd:simpleType name="INTEGER-LIST">
  <xsd:list itemType="xsd:integer"/>
</xsd:simpleType>
<xsd:simpleType name="NONEMPTY-INTEGER-LIST">
  <xsd:restriction base="INTEGER-LIST">
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
XQuery types

define element BOOKS { BOOK* }
define element BOOK { YEAR?, AUTHOR+, TITLE, REVIEW }
define attribute YEAR { integer+ }
define element AUTHOR { string }
define element TITLE { string }
define group INLINE { ( string | EM )* } 
define element REVIEW { INLINE }
define element EM { INLINE }
XQuery types

There is vigorous debate over whether XQuery should use any type notation other than Schema!
Every XQuery type is a group

```xquery
define attribute YEAR { integer+ }
define element AUTHOR { string } =

define group YEAR {
    attribute YEAR { integer+ }
}
define group AUTHOR {
    element AUTHOR { string }
}
```
Nesting XQuery types

define element BOOKS { BOOK* }

define element BOOK {
    attribute YEAR { integer+ } ?,
    element AUTHOR { string } +,
    element TITLE { string },
    element REVIEW { INLINE } ?
}

define group INLINE {
    ( string | element EM { INLINE } )* 
}
Data integration — DTD

<!ELEMENT AMAZON-CATALOGUE (AMAZON-BOOK*)>
<!ELEMENT AMAZON-BOOK (TITLE,AUTHOR+,PRICE,ISBN)>  
<!ELEMENT FATBRAIN-CATALOGUE (FATBRAIN-BOOK*)>
<!ELEMENT FATBRAIN-BOOK (AUTHOR+,TITLE,ISBN,PRICE)>
Data integration — Schema

```xml
<xsd:element name="AMAZON-CATALOGUE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="BOOK"
        minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="TITLE"/>
            <xsd:element ref="AUTHOR"
              minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element ref="PRICE"/>
            <xsd:element ref="ISBN"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```
<xsd:element name="FATBRAIN-CATALOGUE">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="BOOK" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element ref="AUTHOR" minOccurs="1" maxOccurs="unbounded"/>
                        <xsd:element ref="TITLE"/>
                        <xsd:element ref="ISBN"/>
                        <xsd:element ref="PRICE"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
Data integration — XQuery

```xml
define group AMAZON-BOOK {
    element BOOK { TITLE,AUTHOR+,PRICE,ISBN }
}
define group FATBRAIN-BOOK {
    element BOOK { AUTHOR+,TITLE,ISBN,PRICE }
}
define element CATALOGUE { AMAZON-BOOK*,FATBRAIN-BOOK* }
```

Like "DTD with Specialization".
Violates Schema “consistent element restriction”.
Tree grammars and tree automata

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>non-deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>top-down</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom-up</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tree grammars and tree automata

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>non-deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>top-down</td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>bottom-up</td>
<td>Class 2</td>
<td>Class 2</td>
</tr>
</tbody>
</table>
Tree grammars and tree automata

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>non-deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>top-down</td>
<td>Class 1</td>
<td>Class 2</td>
</tr>
<tr>
<td>bottom-up</td>
<td>Class 2</td>
<td>Class 2</td>
</tr>
</tbody>
</table>

Tree grammar **Class 0**: DTD

Tree automata **Class 1**: Schema

Tree automata **Class 2**: XQuery, XDuce, Relax

Class 0 $< \text{Class 1} < \text{Class 2}$

Class 0 and Class 2 have good closure properties.
Class 1 does not.
**Wildcard types**

Wildcard types model unstructured data

```plaintext
define group AnySimpleType {
    string | integer | ...
}
define group AnyAttribute {
    attribute * { AnySimpleType* }
}
define group AnyElement {
    element * { AnyItem* }
}
define group AnyItem {
    AnySimpleType | AnyAttribute | AnyElement
}
define group AnyType {
    AnyItem*}
```
Part V

XQuery expressions
Projection and construction

Return title and authors of all books

for $book in /BOOKS/BOOK return

  <BOOK>{ $book/TITLE, $book/AUTHOR } </BOOK>

⇒

  <BOOK>
    <TITLE>Data on the Web</TITLE>
    <AUTHOR>Abiteboul</AUTHOR>
    <AUTHOR>Buneman</AUTHOR>
    <AUTHOR>Suciu</AUTHOR>
  </BOOK>

  <BOOK>
    <TITLE>XML in Scotland</TITLE>
    <AUTHOR>Buneman</AUTHOR>
  </BOOK>
Projection and construction

Return title and authors of all books

```xml
for $book in /BOOKS/BOOK return
    <BOOK>{ $book/TITLE, $book/AUTHOR }</BOOK>
∈

element BOOK {
    element TITLE { string },
    element AUTHOR { string }+
}*
```
Selection and existential

Return year and title of all books published before 2000

for $book in /BOOKS/BOOK
where $book/@YEAR < 2000
return
  <BOOK>{ $book/@YEAR, $book/TITLE }</BOOK>
⇒
  <BOOK YEAR="1999 2003">
    <TITLE>Data on the Web</TITLE>
  </BOOK>
∈

element BOOK {
  attribute YEAR { integer+ },
  element TITLE { string }
}
An equivalent formulation

Return year and title of all books published before 2000

for $book in /BOOKS/BOOK
where $book/@YEAR < 2000
return
  <BOOK>{ $book/@YEAR, $book/TITLE }</BOOK>

==

for $book in /BOOKS/BOOK return
  if $book/@YEAR < 2000 then
    element BOOK { $book/@YEAR, $book/TITLE }
  else
    ()

$book/@YEAR < 2000

==

some $year in $book/@YEAR satisfying $year < 2000
Selection and projection with XPath

Return title of all books written before 2000

/BOOKS/BOOK[@YEAR < 2000]/TITLE
⇒
<TITLE>Data on the Web</TITLE>
∈
element TITLE { string } *
An equivalent formulation

Return title of all books written before 2000

/BOOKS/BOOK[@YEAR < 2000]/TITLE

==

for $root in / return
   for $books in $root/BOOKS return
      for $book in $books/BOOK return
         where $book/@YEAR < 2000 return
            $book/TITLE
Getting the number wrong

Return book with title ”Data on the Web”

/BOOK/BOOK[TITLE = "Data on the Web"] ∈ BOOK*

How do we exploit keys and relative keys?
Getting the number right

Return book with title "Data on the Web"

treat as BOOK? ( /BOOK/BOOK[TITLE = "Data on the Web"] ) ∈ BOOK?
Nesting

Return titles for each author

let $books := /BOOKS/BOOK
for $author IN distinct($books/AUTHOR) return
  <AUTHOR NAME={ $author }>{
    $books/BOOK[AUTHOR = $author]/TITLE
  }</AUTHOR>

⇒

  <AUTHOR NAME="Abiteboul">
    <TITLE>Data on the Web</TITLE>
  </AUTHOR>

  <AUTHOR NAME="Buneman">
    <TITLE>Data on the Web</TITLE>
    <TITLE>XML in Scotland</TITLE>
  </AUTHOR>

  <AUTHOR NAME="Suciu">
    <TITLE>Data on the Web</TITLE>
  </AUTHOR>
Nesting

Return titles for each author

let $books := /BOOKS/BOOK
for $author IN distinct($books/AUTHOR) return
  <AUTHOR NAME={ $author }>{
    $books/BOOK[AUTHOR = $author]/TITLE
  }</AUTHOR>
∈

element AUTHOR {
  attribute NAME { string },
  element TITLE { string }*
}
Nesting, getting the number right

Return titles for each author

define element TITLE { string }
let $books := /BOOKS/BOOK
for $author IN distinct($books/AUTHOR) return
  <AUTHOR NAME={ $author }>{
    treat at TITLE+ (
      $books/BOOK[AUTHOR = $author]/TITLE
    )
  }<\AUTHOR>
∈

element AUTHOR {
  attribute NAME { string },
  element TITLE { string }+
}
Join

Titles of all books that cost more at Amazon than at Fatbrain

    $fatbrain := document("http://www.fatbrain.com/books.xml")
for $amazon_book IN $amazon/BOOKS/BOOK,
    $fatbrain_book IN $fatbrain/BOOKS/BOOK
    and $amazon_book/PRICE > $fatbrain_book/PRICE
return $amazon_book/TITLE
Unordered

Titles of all books that cost more at Amazon than at Fatbrain, in any convenient order

unordered(
      $fatbrain := document("http://www.fatbrain.com/books.xml")
  for $amazon_book IN $amazon/BOOKS/BOOK,
      $fatbrain_book IN $fatbrain/BOOKS/BOOK
    and $amazon_book/PRICE > $fatbrain_book/PRICE
  return $amazon_book/TITLE
)
An error

Return title and ISBN of each book

for $book in /BOOKS/BOOK return

∈

element ANSWER { TITLE }*
Finding an error by assertion

Return title and ISBN of each book

define element ANSWER {
    element TITLE { string },
    element ISBN { string }
}

for $book in /BOOKS/BOOK return
    assert as ANSWER (  
    )

Assertions might be added automatically when there is a global element declaration and no conflicting local declarations.
Finding an error by omission

Return title and ISBN of each book

define element BOOKS { BOOK* }
define element BOOK { AUTHOR+, TITLE }
define element AUTHOR { string }
define element TITLE { string }
for $book in /BOOKS/BOOK return

Note $book/ISBN ∈ ()

Idea: Report an error when $e ∈ () and $e ≠ ()
Wildcards, computed names

Turn all attributes into elements, and vice versa

define function swizzle (AnyElement $x) returns AnyElement {
    element {name($x)} {
        for $a in $x/@* return element {name($a)} {$a/data()},
        for $e in $x/*/ return attribute {name($e)} {$e/data()}
    }
}

swizzle(<TEST A="a" B="b">
    <C>c</C>
    <D>d</D>
</TEST>)

⇒

<TEST C="c" D="D">
    <A>a</A>
    <B>b</B>
</TEST>
Typing can lose information

Return all Amazon and Fatbrain books by Buneman

define element CATALOGUE { AMAZON-BOOK*,FATBRAIN-BOOK* }
for $book in /CATALOGUE/BOOK
where $book/AUTHOR = "Buneman" return
  $book
∈
( AMAZON-BOOK | FATBRAIN-BOOK )*
Part VI

Syntax
Templates

Convert book listings to HTML format

```html
<HTML><H1>My favorite books</H1>
  <UL>
    for $book in /BOOKS/BOOK return
    <LI>
      <EM>{$book/TITLE/data() }</EM>,
      { $book/@YEAR/data()[position()=last()] }.
    </LI>
  </UL>
</HTML>

⇒

<HTML><H1>My favorite books</H1>
  <UL>
    <LI><EM>Data on the Web</EM>, 2003.</LI>
    <LI><EM>XML in Scotland</EM>, 2002.</LI>
  </UL>
</HTML>
```
XQueryX

A query in XQuery:

```xml
FOR $b in document("bib.xml")//book
  WHERE $b/publisher = "Morgan Kaufmann" AND $b/year = "1998"
  RETURN
  $b/title
```

The equivalent in XQueryX:

```xml
$q:query xmlns:q="http://www.w3.org/2001/06/xqueryx">
  $q:flwr
    $q:forAssignment variable="$b">
      $q:step axis="SLASHSLASH">
        $q:function name="document">
          $q:constant datatype="CHARSTRING">bib.xml</q:constant>
        </q:function>
        $q:identifier>book</q:identifier>
      </q:step>
  </q:forAssignment>
</q:query>
```
XQueryX, continued

```xml
<q:where>
  <q:function name="AND">
    <q:function name="EQUALS">
      <q:step axis="CHILD">
        <q:variable>$b</q:variable>
        <q:identifier>publisher</q:identifier>
      </q:step>
      <q:constant datatype="CHARSTRING">Morgan Kaufmann</q:constant>
    </q:function>
    <q:function name="EQUALS">
      <q:step axis="CHILD">
        <q:variable>$b</q:variable>
        <q:identifier>year</q:identifier>
      </q:step>
      <q:constant datatype="CHARSTRING">1998</q:constant>
    </q:function>
  </q:function>
</q:where>
```
XQueryX, continued²

  <q:return>
   <q:step axis="CHILD">
     <q:variable>$b</q:variable>
     <q:identifier>title</q:identifier>
   </q:step>
  </q:return>
</q:flwr>
</q:query>
Part VII

Typing rules
Types

unit type $u ::=$ string string

| integer integer
| attribute $a \{ t \}$ attribute
| attribute $* \{ t \}$ any attribute
| element $a \{ t \}$ element
| element $* \{ t \}$ any element

type $t ::=$ unit type

| () empty sequence
| $t, t$ sequence
| $t \mid t$ choice
| $t?$ optional
| $t+$ one or more
| $t*$ zero or more
| $x$ type reference
Documents

string \( s ::= "" , "a" , "b" , ... , "aa" , ... \)

integer \( i ::= ... , -1 , 0 , 1 , ... \)

document \( d ::= s \) \hspace{1cm} \text{string} \\
\quad \mid i \hspace{1cm} \text{integer} \\
\quad \mid \text{attribute } a \{ d \} \hspace{1cm} \text{attribute} \\
\quad \mid \text{element } a \{ d \} \hspace{1cm} \text{element} \\
\quad \mid () \hspace{1cm} \text{empty sequence} \\
\quad \mid d , d \hspace{1cm} \text{sequence}
Type of a document — \( d \in t \)

\[
\begin{align*}
    s & \in \text{string} \\
    i & \in \text{integer} \\
    d & \in t \\
    \text{element} \ a \ \{ \ d \ \} & \in \text{element} \ a \ \{ \ t \ \\
    d & \in t \\
    \text{element} \ a \ \{ \ d \ \} & \in \text{element} \ * \ \{ \ t \ \\
    d & \in t \\
    \text{attribute} \ a \ \{ \ d \ \} & \in \text{element} \ a \ \{ \ t \ \\
    d & \in t \\
    \text{attribute} \ a \ \{ \ d \ \} & \in \text{element} \ * \ \{ \ t \ \\
    d & \in t \\
    \text{define group} \ x \ \{ \ t \ \\
    d & \in x
\end{align*}
\]
Type of a document, continued

\[
\begin{align*}
() & \in () & \text{(empty)} \\
\frac{d_1 \in t_1 \quad d_2 \in t_2}{d_1, d_2 \in t_1, t_2} & \quad \text{(sequence)} \\
\frac{d_1 \in t_1}{d_1 \in t_1 \mid t_2} & \quad \text{(choice 1)} \\
\frac{d_2 \in t_2}{d_2 \in t_1 \mid t_2} & \quad \text{(choice 2)} \\
\frac{d \in t+?}{d \in t*} & \quad \text{(star)} \\
\frac{d \in t, t*}{d \in t*} & \quad \text{(plus)} \\
\frac{d \in () \mid t}{d \in t?} & \quad \text{(option)}
\end{align*}
\]
Subtyping and type equivalence

Definition. Write \( t_1 \subseteq t_2 \) iff for all \( d \), if \( d \in t_1 \) then \( d \in t_2 \).

Definition. Write \( t_1 = t_2 \) iff \( t_1 \subseteq t_2 \) and \( t_2 \subseteq t_1 \).

Examples

\[
\begin{align*}
  t & \subseteq t? \subseteq t^* \\
  t & \subseteq t+ \subseteq t^* \\
  t_1 & \subseteq t_1 \mid t_2 \\
  t, () & = t = () , t \\
  t_1 , (t_2 \mid t_3) & = (t_1 , t_2) \mid (t_1 , t_3) \\
  \text{element } a \{ t_1 \mid t_2 \} & = \text{element } a \{ t_1 \} \mid \text{element } a \{ t_2 \}
\end{align*}
\]

Can decide whether \( t_1 \subseteq t_2 \) using tree automata.
Type of an expression — $E \vdash e \in t$

**Environment** $E ::= \$v_1 \in t_1, \ldots, \$v_n \in t_n$

- **(variable)**
  \[
  \frac{E \text{ contains } \$v \in t}{E \vdash \$v \in t}
  \]

- **(let)**
  \[
  \frac{E \vdash e_1 \in t_1 \quad E, \$v \in t_1 \vdash e_2 \in t_2}{E \vdash \text{let } \$v := e_1 \text{ return } e_2 \in t_2}
  \]

- **(empty)**
  \[
  \frac{}{E \vdash () \in ()}
  \]

- **(sequence)**
  \[
  \frac{E \vdash e_1 \in t_1 \quad E \vdash e_2 \in t_2}{E \vdash e_1, e_2 \in t_1, t_2}
  \]

- **(treat as)**
  \[
  \frac{E \vdash e \in t_1 \quad t_1 \cap t_2 \neq \emptyset}{E \vdash \text{treat as } t_2 (e) \in t_2}
  \]

- **(assert as)**
  \[
  \frac{E \vdash e \in t_1 \quad t_1 \subset t_2}{E \vdash \text{assert as } t_2 (e) \in t_2}
  \]
Quantifiers

quantifier $q ::= ()$ exactly zero $t \cdot () = ()$
  | $-$ exactly one $t \cdot - = t$
  | $?$ zero or one $t \cdot ? = t?$
  | $+$ one or more $t \cdot + = t+$
  | $*$ zero or more $t \cdot * = t*$

\[
\begin{array}{c|cccc}
  , & (()) & - & ? & + & * \\
\hline
  () & (()) & - & ? & + & * \\
  - & - & + & + & + & + \\
  ? & ? & + & * & + & * \\
  + & + & + & + & + & + \\
  * & * & + & * & + & * \\
\end{array}
\quad
\begin{array}{c|cccc}
  | & (()) & - & ? & + & * \\
\hline
  () & (()) & - & ? & + & * \\
  - & - & + & + & + & + \\
  ? & ? & + & * & + & * \\
  + & + & + & + & + & + \\
  * & * & + & * & + & * \\
\end{array}
\quad
\begin{array}{c|cccc}
  . & (()) & - & ? & + & * \\
\hline
  () & (()) & - & ? & + & * \\
  - & - & + & + & + & + \\
  ? & ? & + & * & + & * \\
  + & + & + & + & + & + \\
  * & * & + & * & + & * \\
\end{array}
\]
Typing for loops

Return all Amazon and Fatbrain books by Buneman

define element CATALOGUE { AMAZON-BOOK*, FATBRAIN-BOOK* }
for $book in /CATALOGUE/BOOK
where $book/AUTHOR = "Buneman" return

$book ∈ ( AMAZON-BOOK | FATBRAIN-BOOK )*

$E \vdash e_1 \in t_1

$E, \$x \in P(t_1) \vdash e_2 \in t_2

$E \vdash \text{for } \$x \text{ in } e_1 \text{ return } e_2 \in t_2 \cdot Q(t_1)$

$P(AMAZON-BOOK*, FATBRAIN-BOOK*) = AMAZON-BOOK | FATBRAIN-BOOK$

$Q(AMAZON-BOOK*, FATBRAIN-BOOK*) = *$
Prime types

unit type \( u \) ::= string \quad \text{string}
\quad | \quad \text{integer} \quad \text{integer}
\quad | \quad \text{attribute} \ a \ \{ \ t \ \} \quad \text{attribute}
\quad | \quad \text{attribute} \ * \ \{ \ t \ \} \quad \text{any attribute}
\quad | \quad \text{element} \ a \ \{ \ t \ \} \quad \text{element}
\quad | \quad \text{element} \ * \ \{ \ t \ \} \quad \text{any element}

prime type \( p \) ::= u \quad \text{unit type}
\quad | \quad p \mid p \quad \text{choice}
Factoring

Factoring theorem.  For every type $t$, prime type $p$, and quantifier $q$, we have $t \subseteq p \cdot q$ iff $P(t) \subseteq p$ and $Q(t) \leq q$.

Corollary.  For every type $t$, we have $t \subseteq P(t) \cdot Q(t)$. 

\[
P'(u) = \{u\}, \quad Q'(u) = -
\]
\[
P'() = \{\}, \quad Q() = ()
\]
\[
P'(t_1, t_2) = P'(t_1) \cup P'(t_2), \quad Q(t_1, t_2) = Q(t_1), Q(t_2)
\]
\[
P'(t_1 \mid t_2) = P'(t_1) \cup P'(t_2), \quad Q(t_1 \mid t_2) = Q(t_1) \mid Q(t_2)
\]
\[
P'(t?) = P'(t), \quad Q(t?) = Q(t) \cdot ?
\]
\[
P'(t+) = P'(t), \quad Q(t+) = Q(t) \cdot +
\]
\[
P'(t\ast) = P'(t), \quad Q(t\ast) = Q(t) \cdot *
\]
\[
P(t) = () \quad \text{if } P'(t) = \{\}
\]
\[
= u_1 \mid \cdots \mid u_n \quad \text{if } P'(t) = \{u_1, \ldots, u_n\}
\]
Uses of factoring

\[ E \vdash e_1 \in t_1 \]
\[ E, \; \$x \in P(t_1) \vdash e_2 \in t_2 \]
\[ E \vdash \text{for } \$x \text{ in } e_1 \text{ return } e_2 \in t_2 \cdot Q(t_1) \]

(for)

\[ E \vdash e \in t \]
\[ E \vdash \text{unordered}(e) \in P(t) \cdot Q(t) \]

(unordered)

\[ E \vdash e \in t \]
\[ E \vdash \text{distinct}(e) \in P(t) \cdot Q(t) \]

(distinct)

\[ E \vdash e_1 \in \text{integer} \cdot q_1 \quad q_1 \leq ? \]
\[ E \vdash e_2 \in \text{integer} \cdot q_2 \quad q_2 \leq ? \]
\[ E \vdash e_1 + e_2 \in \text{integer} \cdot q_1 \cdot q_2 \]

(arithmetic)
Part VIII

Conclusions
XML Schema: Formal Description

W3C Working Draft, 20 March 2001

This version:
http://www.w3.org/TR/2001/WD-xmlschema-formal-20010320/

Latest version:
http://www.w3.org/TR/xmlschema-formal/

Editors:
Allen Brown (Microsoft) alienbr@microsoft.com
Matthew Fuchs (Commerce One) matthew.fuchs@commerceone.com
Jonathan Robie (Software AG) jonathan.robie@SoftwareAG-USA.com
Philip Wadler (Avaya) wadler@avaya.com

Copyright ©2001 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.
6 Validation

6.1 Content validation

We write $d \in_r g$ if forest $d$ matches group $g$.

**EMPTY:**

\[ \varepsilon \ in e \]

**SEQUENCE:**

\[ \frac{d_1 \ in g_1 \ \ \ d_2 \ in g_2}{d_1 , d_2 \ in g_1 , g_2} \]

**CHOICE 1:**

\[ \frac{d \ in g_1}{d \ in g_1 \ | \ g_2} \]

**CHOICE 2:**

\[ \frac{d \ in g_2}{d \ in g_1 \ | \ g_2} \]
RELAX NG Specification

Committee Specification 11 August 2001

This version:
   Committee Specification 11 August 2001

Editors:
   James Clark <jje@jclark.com>, MURATA Makoto <mura034@actglobal.net>

Copyright © The Organization for the Advancement of Structured Information Standard [OASIS] 2001. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.
§ 6.2. Patterns

The axioms and inference rules for patterns use the following notation:

- \( p \) ranges over patterns (elements matching the pattern production)
- \( cx \vdash \alpha; m \sim p \)
  - asserts that with respect to context \( cx \), the attributes \( \alpha \) and the sequence of elements and strings \( m \) matches the pattern \( p \)

6.2.1. choice pattern

The semantics of the choice pattern are as follows:

(choice 1)

\[
\frac{cx \vdash \alpha; m \sim p_1}{cx \vdash \alpha; m \sim <\text{choice}> p_1 p_2</\text{choice}>}
\]

(choice 2)

\[
\frac{cx \vdash \alpha; m \sim p_2}{cx \vdash \alpha; m \sim <\text{choice}> p_1 p_2</\text{choice}>}
\]
Choose a sample query:

Submit Query

Query in english:

Q1: List books published by Addison-Wesley after 1991, including their year and title.

Query text:

```xml
<bib>
    { FOR $b IN $bib/book
        WHERE $b/publisher/data(.) = "Addison-Wesley"
            AND $b/year/data(.) > 1991
        RETURN
            <book year="($b/year/data(.))">($b/title)
                </book>
    }
</bib>
```
Some research topics

How to trade-off accuracy for simplicity in types?

How to exploit keys and relative keys?

How to integrate keys and relative keys into types?

How to handle graphs?

How to handle fixpoints as in Datalog?

How to integrate with OO view of data?
Links

My XML page

http://www.research.avayalabs.com/~wadler/xml/

W3C XML Query page

http://www.w3.org/XML/Query.html

XML Query demonstrations

Galax - AT&T, Lucent, and Avaya
   http://www-db.research.bell-labs.com/galax/

Quip - Software AG
   http://www.softwareag.com/developer/quip/

XQuery demo - Microsoft
   http://131.107.228.20/xquerydemo/
Conclusions

There is nothing to XML
There is everything to XML

Industry resists formal methods
Industry welcomes formal methods

The best thing you can do is work on standards
The worst thing you can do is work on standards

You need XML
XML needs you
Beware the ides of March!