

The Next 700 Markup Languages

Philip Wadler, Bell Labs

The Next 700 Programming Languages

“... today ... 1,700 special programming languages are used to ‘communicate’ in 700 application areas.” — *Computer Software*, July

A family of unimplemented computing languages is described that is intended to span differences of application area by a unified framework. This framework dictates the rules about the uses of user-coined names, and the conventions about characterizing functional relationships. Within this framework, the design of a language splits into two independent parts. One is the choice of written appearances of programs (or, more generally, their physical representation). The other is the choice of abstract entities (such as numbers, character strings, lists of them, functional relations among them) that can be referred to in the language.

– Peter J. Landin, The Next 700 Programming Languages, *CACM*, March 1966.

Part I

From HTML to XML

Some XML

```
<html>
  <head>
    <title>Philip Wadler's bibliography</title>
  </head>
  <body>
    <h1>Philip Wadler's bibliography</h1>
    <ul>
      <li><a href="xslt.pdf">A formal semantics of patterns in XSLT</a><br/>
        P. Wadler, <em>Markup Technologies 99</em>.</li>
      <li><a href="query.pdf">A data model and algebra for XML Query</a><br/>
        M. Fernandez, J. Simeon, and P. Wadler, <em>draft, Feb 00</em>.</li>
    </ul>
  </body>
</html>
```

Some more XML

```
<bibliography heading="Philip Wadler's bibliography">
  <paper>
    <author>P. Wadler</author>
    <title>A formal semantics of patterns in XSLT</title>
    <conference>Markup Technologies 99</conference>
    <file>xslt.pdf</file>
  </paper>
  <paper>
    <author>M. Fernandez</author>
    <author>J. Simeon</author>
    <author>P. Wadler</author>
    <title>A data model and algebra for XML Query</title>
    <conference>draft</conference>
    <file>query.pdf</file>
  </paper>
</bibliography>
```

It's just a tree!

```
html
|
|--head
|  |
|  '--title
|     |
|     '--"Philip Wadler's bibliography"
'--body
    |--h1
    |  |
    |  '--"Philip Wadler's bibliography"
    '--ul
        |
        |--li ...
        |
        '--li ...
```

Some LISP

```
(html
  (head
    (title "Philip Wadler's bibliography"))
  (body
    (h1 "Philip Wadler's bibliography")
    (ul
      (li
        (a (@href "xslt.pdf) "A formal semantics of patterns in XSLT") (br)
        "P. Wadler, " (em "Markup Technologies 99") ".")
      (li
        (a (@href "query.pdf") "A data model and algebra for XML Query") (br)
        "M. Fernandez, J. Simeon, and P. Wadler, " (em "draft") ".")))))
```

Some more LISP

```
(bibliography
```

```
  (@heading "Philip Wadler's bibliography")
```

```
  (paper
```

```
    (author "P. Wadler")
```

```
    (title "A formal semantics of patterns in XSLT")
```

```
    (conference "Markup Technologies 99")
```

```
    (file "xslt.pdf"))
```

```
  (paper
```

```
    (author "M. Fernandez")
```

```
    (author "J. Simeon")
```

```
    (author "P. Wadler")
```

```
    (title "A data model and algebra for XML Query")
```

```
    (conference "draft")
```

```
    (file "query.pdf")))
```

DTD: Document Type Description

```
<!ELEMENT bibliography (paper*)>
```

```
<!ATTLIST bibliography heading CDATA #REQUIRED>
```

```
<!ELEMENT paper (author+,title,conference,file)>
```

```
<!ELEMENT author #PCDATA>
```

```
<!ELEMENT title #PCDATA>
```

```
<!ELEMENT conference #PCDATA>
```

```
<!ELEMENT file #PCDATA>
```

XML Schema: An alternative to DTDs

```
<x:element name="bibliography">
  <x:type>
    <x:attribute name="heading" type="x:string"/>
    <x:element name="paper" minOccurs="0" maxOccurs="*">
      <x:type>
        <x:element name="author" type="x:string" maxOccurs="*" />
        <x:element name="title" type="x:string" />
        <x:element name="conference" type="x:string" />
        <x:element name="file" type="x:string" />
      </x:type>
    </x:element>
  </x:type>
</x:element>
```

DTDs and Schemas are like types

```
class Bibliography {  
    String heading;  
    Paper[] papers;  
}
```

```
class Paper {  
    String[] author;  
    String title;  
    String conference;  
    String file;  
}
```

Part II

XML Applications and Standards

The Four Webs

- Computers — xHTML
- Voice — Voice ML
- Wireless — WAP/WML
- Television — bHTML

Additional applications

- [configuration](#) — Universal Plug-n-Play (UPnP)
- [directories](#) — Directory Services Markup Language (DSML)
- [palmtops](#) — SyncML Consortium
- [e-commerce](#) — BizTalk, eCo Framework, Rosetta Net
- [mathematics](#) — MathML
- [standards](#) — W3C Standard DTD
- [and finally](#) —

Additional applications

- [configuration](#) — Universal Plug-n-Play (UPnP)
- [directories](#) — Directory Services Markup Language (DSML)
- [palmtops](#) — SyncML Consortium
- [e-commerce](#) — BizTalk, eCo Framework, Rosetta Net
- [mathematics](#) — MathML
- [standards](#) — W3C Standard DTD
- [and finally](#) — Theological ML

WML: Wireless Markup Language

```
<wml>
  <card>
    <p>
      <do type="accept">
        <go href="#card2"/>
      </do>
      Hello world!
      This is the first card...
    </p>
  </card>
  <card id="card2">
    <p>
      This is the second card. Goodbye.
    </p>
  </card>
</wml>
```

RETS : Real Estate Transaction Standard

<PROPERTY>

<MLNUM>99040102</MLNUM>

<STATUS>ACT</STATUS>

<AREA>101</AREA>

<ADDR>123 Main</ADDR>

<LISTPR>105000</LISTPR>

</PROPERTY>

<PROPERTY>

<MLNUM>99030204</MLNUM>

<STATUS>ACT</STATUS>

<AREA>101</AREA>

<ADDR>100 WASHINGTON SQ.</ADDR>

<LISTPR>550000</LISTPR>

</PROPERTY>

UPnP: Universal Plug-and-Play

```
<W:ROOT xmlns:W="urn:www.microsoft.com:Windows"
  xmlns:LSCam="urn:www.w3c.org:Schema:LiveStillCamera">
  <W:DeviceType>LiveStillCamera</W:DeviceType>
  <W:DeviceDesc>
    <W:Name>Sentry</W:Name>
    <W:Manufacturer>
      <W:Name>Axis Communications</W:Name>
      <W:URL>http://www.axis.com</W:URL>
    </W:Manufacturer>
    <W:ModelName>NetEye</W:ModelName>
    <W:SerialNumber>00408C242A27</W:SerialNumber>
  <LSCam:ImageResolution>
    <LSCam:Width>704</LSCam:Width>
    <LSCam:Height>576</LSCam:Height>
  </LSCam:ImageResolution>
</W:ROOT>
```

Some XML W3C standards

- Namespaces
- DOM — Document Object Model
- XSL — Stylesheet Language (XSL-T, XSL-FO)
- XLL — Linking Language (XLink, XPointer)
- XPath — common subset of XSL and XLL
- XSchema
- XQuery

Part III

XSLT

Convert a bibliography to HTML

```
<xslt:template match="bibliography">
  <html>
    <head>
      <title><xslt:value-of select="@heading"/></title>
    </head>
    <body>
      <h1><xslt:value-of select="@heading"/></h1>
      <ul><xslt:apply-templates select="paper"/></ul>
    </body>
  </html>
</xslt:template>
```

Convert a paper to HTML

```
<xslt:template match="paper">
  <li>
    <a href="file"><xslt:value-of select="title"/></a>
    <br/>
    <xslt:apply-templates select="author"/>
    <xslt:text>, </xslt:text>
    <em><xslt:value-of select="conference"/></em>
    <xslt:text>.</xslt:text>
  </li>
</xslt:template>
```

Convert and author list to HTML

```
<xslt:template match="author[position()=1]">
  <xslt:apply-templates/>
</xslt:template>

<xslt:template match="author[position()=last()]">
  <xslt:text>, and <xslt:text><xslt:apply-templates/>
</xslt:template>

<xslt:template match="author">
  <xslt:text>, <xslt:text><xslt:apply-templates/>
</xslt:template>
```

Part IV

Xduce

Haruo Hosoya, Benjamin C. Pierce,
Peter Buneman — University of Pennsylvania

The type of a bibliography

```
type Biblio      = bibliography[Heading,Paper*]
type Heading    = @heading[String]
type Paper      = paper[Author+,Title,Conference,Href]
type Author     = author[String]
type Title      = title[String]
type Conference = conference[String]
type File       = file[String]
```

The type of HTML

```
type Html      = html[Head?, Body]
type Head     = head[Title]
type Title    = title[String]
type Body     = body[Mix]
type Mix     = ( h1[Mix]
                | a[@href[String], Mix]
                | p[Mix]
                | em[Mix]
                | ul[li[Mix]*]
                | String )*
```

Convert bibliography to HTML

```
fun do_biblio : Biblio -> Html =  
  bibliography[@heading[h], p] ->  
    html[head[title[h]],  
         body[h1[h], ul[do_paper*(p)]]]
```

```
fun do_paper : Paper -> li[Mix] =  
  paper[x : Author*, title[t], conference[c], file[f]] ->  
    li[a[href[f],t], br[], do_authors(x), ", ", em[c], "."]
```

```
fun do_authors : Author* -> Mix =  
  author[a] ->  
    a  
| author[a], author[b] ->  
  a, "and, ", b  
| author[a], x ->  
  a, ", ", do_authors(x)
```

Can we do better with the types?

- Polymorphic types
- Higher-order types

Part V

A query algebra for XML

Mary Fernandez, Dan Suciu — AT&T

Jerome Simeon, Philip Wadler — Lucent

Nested Relational Algebra

```
[ (value t, value a)
| x <- follow "bibliography" bib0,
  b <- follow "paper" x,
  t <- follow "title" b,
  a <- follow "author" b ]
```

==>

```
[("Formal semantics of XSLT", "P. Wadler"),
 ("An algebra for XML Query", "M. Fernandez"),
 ("An algebra for XML Query", "J. Simeon"),
 ("An algebra for XML Query", "P. Wadler")]
```

Nested Relational Algebra — Tuples and Lists

$$\frac{}{\Gamma, x : s \vdash x : s} \quad \frac{\Gamma, x : s \vdash e : t}{\Gamma \vdash \lambda x. e : s \rightarrow t} \quad \frac{\Gamma \vdash e_1 : s \rightarrow t \quad \Gamma \vdash e_2 : s}{\Gamma \vdash e_1(e_2) : t}$$

$$\frac{\Gamma \vdash e_1 : t_1 \quad \dots \quad \Gamma \vdash e_k : t_k}{\Gamma \vdash (e_1, \dots, e_k) : t_1 \times \dots \times t_k} \quad \frac{\Gamma \vdash e : t_1 \times \dots \times t_k}{\Gamma \vdash \pi_{i,k}(e) : t_i} \quad (\text{for } 1 \leq i \leq k; k \geq 2)$$

$$\frac{}{\Gamma \vdash [] : [s]} \quad \frac{\Gamma \vdash e : s}{\Gamma \vdash [e] : [s]} \quad \frac{\Gamma \vdash e_1 : [s] \quad \Gamma \vdash e_2 : [s]}{\Gamma \vdash e_1 \uplus e_2 : [s]} \quad \frac{\Gamma, x : s \vdash e_1 : [t] \quad \Gamma \vdash e_2 : [s]}{\Gamma \vdash \bigcup[e_1 \mid x \in e_2] : [t]}$$

$$\frac{}{\Gamma \vdash \text{true} : \text{Bool}} \quad \frac{}{\Gamma \vdash \text{false} : \text{Bool}}$$

$$\frac{\Gamma \vdash e_1 : \text{Bool} \quad \Gamma \vdash e_2 : t \quad \Gamma \vdash e_3 : t}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t} \quad \frac{\Gamma \vdash e_1 : t \quad \Gamma \vdash e_2 : t}{\Gamma \vdash e_1 \text{ op } e_2 : \text{Bool}}$$

where op is one of =, <, >, ≤, ≥, ≠

$$\frac{}{\Gamma \vdash n : \text{Nat}} \quad \frac{\Gamma \vdash e_1 : \text{Nat} \quad \Gamma \vdash e_2 : \text{Nat}}{\Gamma \vdash e_1 \text{ op } e_2 : \text{Nat}} \quad \frac{\Gamma, x : s \vdash e_1 : \text{Nat} \quad \Gamma \vdash e_2 : [s]}{\sum[e_1 \mid x \in e_2] : \text{Nat}}$$

where n is a natural number and op is one of +, −, *, /, %

$$\frac{}{\Gamma \vdash \perp^t : t} \quad \frac{\Gamma \vdash e : [s]}{\Gamma \vdash \text{the}(e) : s} \quad \frac{\Gamma \vdash e : [s]}{\Gamma \vdash \text{index}(e) : [\text{Nat} \times s]}$$

Nested Relational Algebra — Parsing and Recursion

$$\frac{\Gamma \vdash e : \text{String}}{\Gamma \vdash \text{text}(e) : \text{Node}} \quad \frac{\Gamma \vdash e : [\text{Node}]}{\Gamma \vdash \text{elem}_i(e) : \text{Node}}$$

$$\frac{}{\Gamma \vdash \text{textItem} : \text{Reg String}} \quad \frac{\Gamma \vdash e : \text{Reg } t}{\Gamma \vdash \text{step}_i(e) : \text{Reg } t} \quad \frac{\Gamma \vdash e_1 : \text{Reg } t \quad \Gamma \vdash e_2 : [\text{Node}]}{\Gamma \vdash \text{match}(e_1, e_2) : [t]}$$

$$\frac{}{\Gamma \vdash ([]) : \text{Reg } s} \quad \frac{\Gamma \vdash e : s}{\Gamma \vdash ([e]) : \text{Reg } s}$$

$$\frac{\Gamma \vdash e_1 : \text{Reg } s \quad \Gamma \vdash e_2 : \text{Reg } s}{\Gamma \vdash e_1 \oplus e_2 : \text{Reg } s} \quad \frac{\Gamma, x : s \vdash e_1 : \text{Reg } t \quad \Gamma \vdash e_2 : \text{Reg } s}{\Gamma \vdash \bigoplus([e_1 \mid x \in e_2]) : \text{Reg } t}$$

$$\frac{\Gamma \vdash e : \text{Reg } t}{\Gamma \vdash \text{rep}(e) : \text{Reg } [t]}$$

$$\frac{\Gamma, x : [t] \vdash e_1 : t \quad \Gamma, y : \text{String} \vdash e_2 : t \quad \Gamma \vdash e_3 : \text{Node}}{\Gamma \text{fold}(\lambda x. e_1)(\lambda y. e_2)(e_3) : t}$$

Can we do better with the types?

- Types in our current formulation:
 - (t, u) — tuple
 - $[t]$ — list
 - *String* — string
 - *Node* — XML tree
- Types in Xduce:
 - t, u — sequence
 - t^* — repetition
 - $t \mid u$ — alternation
 - *String* — text
 - $a[t]$ — element

Part VI

Conclusion

Conclusion

- XML is just a tree.
- Use DTD or Schema or Xduce to define types.
Use XSLT or Xduce to define transformations.
- Types can help with transformations.
Can we do better with the types?
- XML is not the end of our problems —

Conclusion

- XML is just a tree.
- Use DTD or Schema or Xduce to define types.
Use XSLT or Xduce to define transformations.
- Types can help with transformations.
Can we do better with the types?
- XML is not the end of our problems —
it is the beginning of our problems.

Conclusion

- XML is just a tree.
- Use DTD or Schema or Xduce to define types.
Use XSLT or Xduce to define transformations.
- Types can help with transformations.
Can we do better with the types?
- XML is not the end of our problems —
it is the beginning of our problems.
- On to the next 700 Markup Languages!