XPath
References

• http://www.w3.org/TR/xpath/
Values

• XQuery and XPath manipulate ordered sequences of *items*; an *item* is either a node or an atomic value

• Nodes as those of XDM
XPath

- `<bookstore>`
  `<book lang=en>`
    `<title>Harry Potter</title>`
    `<author>J K. Rowling</author>`
    `<year>2005</year>`
    `<price>29.99</price>`
  `</book>`
`</bookstore>`
The tree

```
$doc /child::element(bookstore)/child::element(book)
$doc /bookstore/book
$doc /descendant::element(author)/child::text()
$doc //author/text()
$doc //book[./author/text() = "K. Rowling"]/title/text()
```
Abbreviations

• Omitted axis: `child::`
  – `book = child::book`
• `@` = `attribute::`
  – `book/@lang = child::book/attribute::lang`
• `// = /descendant-or-self::*`/
  – `$doc//book = $doc/d-o-s::*/child::book`
• `.. = parent::node()`
  – `$doc//book/@lang/../author/text()`
  – `$doc//book[./@lang]/author/text()`
Other axes

- self
- parent
- child: only elements, no attribute
- attribute: the attributes
- namespace
- ancestor, ancestor-or-self
- descendant, descendant-or-self
- following: follow but are not descendants
- following-sibling
- preceding: precede but are not ancestors
- preceding-sibling
Tree patterns

- Trees with simple “son” edges and double “descendant” edges
- Nodes can be labeled with a kind ("element" "text" "attribute"), a name, a value
- A starting node and a destination node
Tree patterns

• They express XPath queries:
  – book[//last/text() = 'Buneman']/title
  – book//last/text()//ancestor::element(book)/title

• They express in a canonical way a set of equivalent XPath queries
Boolean and numerical predicates

• Comparison:
  – descendant::book[@lang = "en"]

• Quantification:
  – .//book[@lang]
  – .//bool[@lang = ("en", "ita")]

• Evaluating path / step: if [[path]] is L=N1,...,Nn, we evaluate step n-times, once for each context (L,Ni)

• Numerical predicates:
    ./bookstore/book/author[position()=1]
  – (./bookstore/book/author)[1]
Quantification: the effective boolean value of a sequence

- Empty: false
- The first element is a node: true
- First is not a node, and length >1: fail
- First is not a node, and length = 1 (singleton):
  - Is a string or untyped: true if not empty
  - Is numeric: true if not zero
  - Is a boolean: true if is true
  - Otherwise: fail
The order of the result

• Document order: the root, the first child and all of its descendants, the second child...

• XPath: every expression that returns nodes, returns them with no duplicates and in document order

  – //a[1]/text() -> “first”, “second”
  – //a[2]/text() -> “first”
  – //a//text()
    -> reorder(“first”, “second”, “first”)
    = (“first”, “second”)

  a
  a
  second
  first
Atomization

- $\text{john/weight} + 1$?
- <$\text{weight}>5</$\text{weight}> + 1$?
  - $6$, if $\text{weight}$ is declared numeric
  - Error otherwise

- Every function that expects an atomic value (+, = ...) triggers atomization

- Atomization:
  - Identity over atomic values
  - The typed value for nodes

- $\text{$doc$///book[./author = "K. Rowling"]/title/text()}$
String value e typed value

• String value of a document/element: all the strings of the text leaves, read in document order

• Typed value:
  – Untyped element or typed element with mixed content: the string value
  – Simple content-type element: the string value converted according the content-type
  – Element with element-only content: error
XDM types

- xs:anyType
  - xs:anySimpleType
    - xs:anyAtomicType
      - xs:untypedAtomic and specific atomic types such as xs:integer, xs:string, and xs:dayTimeDuration
  - xs:untyped and specific complex types
    - Specific list and union types such as xs:IDREFS
Let us assume an `IdRefAuth` attribute of type `IDREF` in `books`, and an attribute `Id` of type `ID` in `authors`.

fn:id(E): finds, in the current document, the nodes whose ID appears in the string sequence E:
- `$doc//book[year=2010]/@IdRefAuth`:
  - The idrefs of the authors of 2010 books
- `$doc//fn:id($doc//book[year=2010]/@IdRefAuth)`:
  - The element nodes of those authors
  - Same as:
    - `$doc//fn:root()/*[@Id= (...) ]`
- `$doc//fn:id($doc//book[year=2010]/@IdRefAuth)/name`:
  - The names of those authors