XPath

Path Expressions
Conditions

1

Paths in XML Documents

- XPath is a language for describing paths in XML documents.
- Really think of the semi-structured data graph and *its* paths.

2

Example DTD

```
<!DOCTYPE BARS [
  <!ELEMENT BARS (BAR*, BEER*)>
  <!ELEMENT BAR (PRICE+)>
      <!ATTLIST BAR name ID #REQUIRED>
  <!ELEMENT PRICE (#PCDATA)>
      <!ATTLIST PRICE theBeer IDREF #REQUIRED>
  <!ELEMENT BEER EMPTY>
      <!ATTLIST BEER name ID #REQUIRED>
      <!ATTLIST BEER soldBy IDREFS #IMPLIED>
]>
```

3

Example Document

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = Miller">3.00</PRICE>
  </BAR> …
  <BEER name = "Bud" soldBy = "JoesBar
      SuesBar … "/> …
</BARS>
```

4

Path Descriptors

- Simple path descriptors are sequences of tags separated by slashes (/).
- If the descriptor begins with /, then the path starts at the root and has those tags, in order.
- If the descriptor begins with //, then the path can start anywhere.

5

Value of a Path Descriptor

- Each path descriptor, applied to a document, has a value that is a sequence of elements.
- An *element* is an atomic value or a node.
- A *node* is matching tags and everything in between.
  - I.e., a node of the semistructured graph.

6

Example: /BARS/BAR/PRICE

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = "Miller">3.00</PRICE>
  </BAR> …
  <BEER name = "Bud" soldBy = "JoesBar
      SuesBar …"/> …
</BARS>
```

/BARS/BAR/PRICE describes the set with these two PRICE elements as well as the PRICE elements for any other bars.

7

Example: //PRICE

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = "Miller">3.00</PRICE>
  </BAR> …

  <BEER name = "Bud" soldBy = "JoesBar
      SuesBar …"/>…
</BARS>
```

//PRICE describes the same PRICE elements, but only because the DTD forces every PRICE to appear within a BARS and a BAR.

8

Wild-Card *

- A star (*) in place of a tag represents any one tag.
- Example: /*/*/PRICE represents all price objects at the third level of nesting.

9

Example: /BARS/*

<BARS>
<PRICE theBeer = "Bud">2.50</PRICE>
<PRICE theBeer = "Miller">3.00</PRICE>
…
</BARS>

/BARS/* captures all BAR and BEER elements, such as these.

Attributes: /BARS/*/@name

<BARS>
<PRICE theBeer = "Bud">2.50</PRICE>
<PRICE theBeer = "Miller">3.00</PRICE>
</BARS> …
SuesBar …"/> …
</BARS>

/BARS/*/@name selects all name attributes of immediate subelements of the BARS element.

Attributes of a tag may appear in paths as if they were nested within that tag.

Selection Conditions

- A condition inside […] may follow a tag.
- If so, then only paths that have that tag and also satisfy the condition are included in the result of a path expression.

Example: Selection Condition

- /BARS/BAR[PRICE < 2.75]/PRICE

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = "Miller">3.00</PRICE>
  </BAR> ...
```

The condition that the PRICE be < $2.75 makes this price but not the Miller price satisfy the path descriptor.

13

Example: Attribute in Selection

- /BARS/BAR/PRICE[@theBeer = "Miller"]

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = "Miller">3.00</PRICE>
  </BAR> ...
```

Now, this PRICE element is selected, along with any other prices for Miller.

14

Axes

- In general, path expressions allow us to start at the root and execute steps to find a sequence of nodes at each step.
- At each step, we may follow any one of several *axes*.
- The default axis is child:: --- go to all the children of the current set of nodes.

15

Example: Axes

- /BARS/BEER is really shorthand for /BARS/child::BEER
- @ is really shorthand for the attribute:: axis.
  - Thus, /BARS/BEER[@name = "Bud"] is shorthand for /BARS/BEER[attribute::name = "Bud"]

16

More Axes

- Some other useful axes are:
  1. parent:: = parent(s) of the current node(s).
  2. descendant-or-self:: = the current node(s) and all descendants.
     - Note: // is really shorthand for this axis.
  3. ancestor::, ancestor-or-self, etc.

17

XQuery

CS317/387

18

XQuery

- XQuery extends XPath to a query language that has power similar to SQL.
- XQuery is an expression language.
  - Like relational algebra --- any XQuery expression can be an argument of any other XQuery expression.
- Unlike RA, with the relation as the sole datatype, XQuery has a subtle type system.

19

The XQuery Type System

1. *Atomic values* : strings, integers, etc.
   - Also, certain constructed values like true(), date("2004-09-30").
2. *Nodes*.
   - Seven kinds.
   - We'll only worry about four, on next slide.

20

Some Node Types

1. *Element Nodes* are like nodes of semistructured data.
   - Described by !ELEMENT declarations in DTD's.
2. *Attribute Nodes* are attributes, described by !ATTLIST declarations in DTD's.
3. *Text Nodes* = #PCDATA.
4. *Document Nodes* represent files.

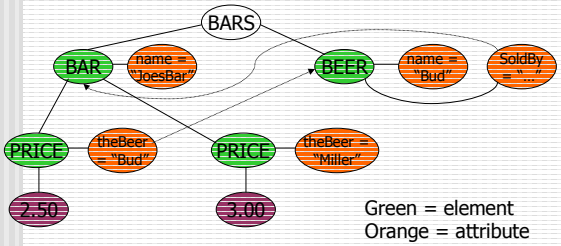
21

Example Document

```
<BARS>
  <BAR name = "JoesBar">
      <PRICE theBeer = "Bud">2.50</PRICE>
      <PRICE theBeer = "Miller">3.00</PRICE>
  </BAR> …
  <BEER name = "Bud" soldBy = "JoesBar SuesBar … "/> …
</BARS>
```

22

Example Nodes



Green = element
Orange = attribute
Purple = text

23

Document Nodes

- Form: document("<file name>").
- Establishes a document to which a query applies.
- Example: document("/users/umesh/bars.xml")

24

XQuery Values

- *Item* = node or atomic value.
- *Value* = ordered sequence of zero or more items.
- Examples:
  1. () = empty sequence.
  2. ("Hello", "World")
  3. ("Hello", <PRICE>2.50</PRICE>, 10)

25

FLWR Expressions

1. One or more for and/or let clauses.
2. Then an optional where clause.
3. A return clause.

26

Semantics of FLWR Expressions

- Each for creates a loop.
  - let produces only a local definition.
- At each iteration of the nested loops, if any, evaluate the where clause.
- If the where clause returns TRUE, invoke the return clause, and append its value to the output.

27

FOR Clauses

`for <variable> in <expression>, . . .`

- Variables begin with $.
- A for-variable takes on each item in the sequence denoted by the expression, in turn.
- Whatever follows this for is executed once for each value of the variable.

28

Example: FOR

for $beer in document("bars.xml")/BARS/BEER/@name

Return

<BEERNAME> {$beer} </BEERNAME>

"Expand the enclosed string by replacing variables and path exps. by their values."

- $beer ranges over the name attributes of all beers in our example document.
- Result is a list of tagged names, like <BEERNAME>Bud</BEERNAME> <BEERNAME>Miller</BEERNAME> . . .

29

LET Clauses

`let <variable> := <expression>, . . .`

- Value of the variable becomes the *sequence* of items defined by the expression.
- Note let does not cause iteration; for does.

30

Example: LET

let $d := document("bars.xml")
let $beers := $d/BARS/BEER/@name
return
<BEERNAMES> {$beers} </BEERNAMES>

- Returns one element with all the names of the beers, like:

<BEERNAMES>Bud Miller …</BEERNAMES>

31

Following IDREF's

- XQuery (but not XPath) allows us to use paths that follow attributes that are IDREF's.
- If *x* denotes a sequence of one or more IDREF's, then ***x =>y*** denotes all the elements with tag *y* whose ID's are one of these IDREF's.

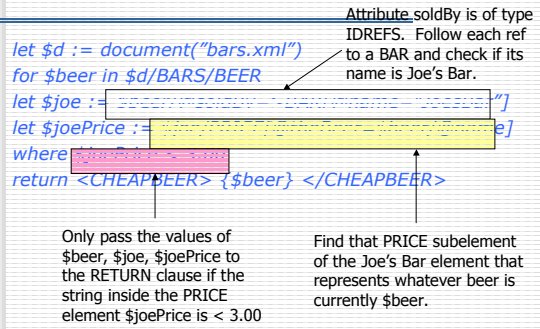
32

Example

- Find all the beer elements where the beer is sold by Joe's Bar for less than 3.00.
- Strategy:
  1. $beer will for-loop over all beer elements.
  2. For each $beer, let $joe be either the Joe's-Bar element, if Joe sells the beer, or the empty sequence if not.
  3. Test whether $joe sells the beer for < 3.00.

33

Example: The Query



34

Order-By Clauses

- FLWR is really FLWOR: an order-by clause can precede the return.
- Form: order by <expression>
  - With optional ascending or descending.
- The expression is evaluated for each output element.
- Determines placement in output sequence.

35

Example: Order-By

- List all prices for Bud, lowest first.

```
let $d := document("bars.xml")
for $p in $d/BARS/BAR/PRICE[@theBeer="Bud"]
order by $p
return { $p }
```

36

Predicates

- Normally, conditions imply existential quantification.
- Example: /BARS/BAR[@name] means "all the bars that have a name."
- Example: /BARS/BAR[@name="JoesBar"]/PRICE = /BARS/BAR[@name="SuesBar"]/PRICE means "Joe and Sue have at least one price in common."

37

Other Operators

- Use Fortran comparison operators to compare atomic values only.
  - eq, ne, gt, ge, lt, le.
- Arithmetic operators: +, - , *, div, idiv, mod.
  - Apply to any expressions that yield arithmetic or date/time values.

38

Effective Boolean Values

- The *effective boolean value* (EBV) of an expression is:
  1. The actual value if the expression is of type boolean.
  2. FALSE if the expression evaluates to 0, "" [the empty string], or () [the empty sequence].
  3. TRUE otherwise.

39

EBV Examples

1. @name="JoesBar" has EBV TRUE or FALSE, depending on whether the name attribute is "JoesBar".
2. /BARS/BAR[@name="GoldenRail"] has EBV TRUE if some bar is named the Golden Rail, and FALSE if there is no such bar.

40

Boolean Operators

- E_1 and E_2, E_1 or E_2, not(E), if (E_1) then E_2 else E_3 apply to any expressions.
- Take EBV's of the expressions first.
- Example: not(3 eq 5 or 0) has value TRUE.
- Also: true() and false() are functions that return values TRUE and FALSE.

41

Quantifier Expressions

some $x in E_1 satisfies E_2

1. Evaluate the sequence E_1.
2. Let $x (any variable) be each item in the sequence, and evaluate E_2.
3. Return TRUE if E_2 has EBV TRUE for at least one $x.

- Analogously:

every $x in E_1 satisfies E_2

42

Document Order

- Comparison by document order: << and >>.
- Example: $d/BARS/BEER[@name="Bud"] << $d/BARS/BEER[@name="Miller"] is true iff the Bud element appears before the Miller element in the document $d.

43

Set Operators

- union, intersect, except operate on sequences of nodes.
  - Meanings analogous to SQL.
  - Result eliminates duplicates.
  - Result appears in document order.

44