Markup Languages and Semistructured Data - SS'02

http://www.pms.informatik.uni-muenchen.de/lehre/markupsemistrukt/02ss/

XPath 1.0 Tutorial

28th of May, 2002

Dan Olteanu

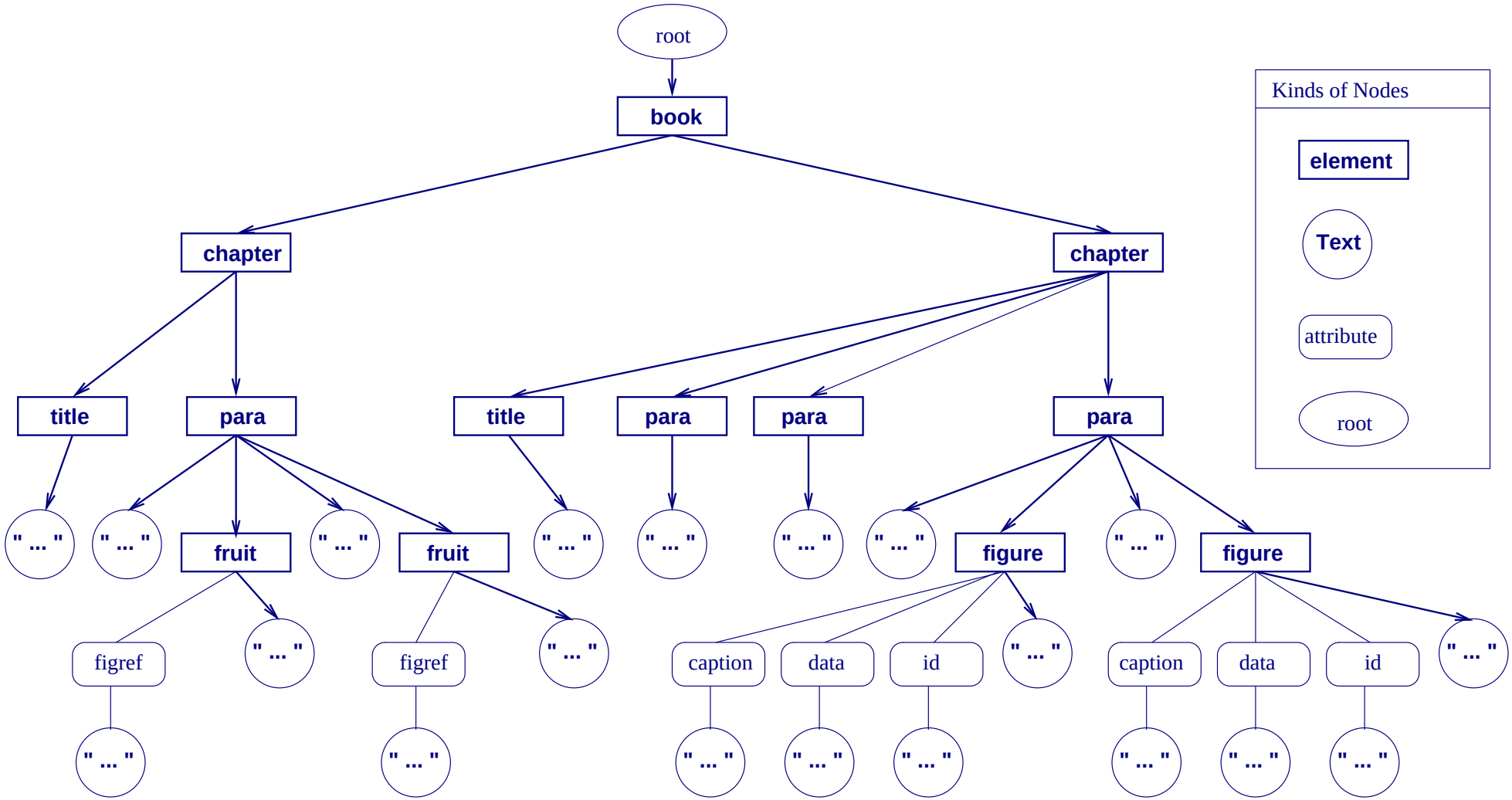
XPath 1.0 - W3C Recommendation

- language for **addressing** and **matching** parts of an XML document.
- designed to be used standalone, but also by **XPointer**, **XSLT**, **XQuery**.
- uses a compact, **non-XML** format to facilitate use within URIs and attribute values.
- provides basic facilities for string, number and boolean **manipulation**.
- supports namespaces.

XPath 1.0 Data Model

- can be derived from **XML InfoSet**.
- XML document is viewed as a **tree**, containing different kinds of nodes.
- kinds of nodes: root, element, text, attribute, namespace, processing instruction, comment nodes.
- imposes a document **order** defined on all nodes except attribute and namespace nodes (order of occurence of element start-tags).
- the root node is the **first** node.
- the namespace nodes are defined to occur **before** the attribute nodes.
- root and element nodes have an **ordered list** of children.
- an element node is the **parent** of the associated **set** of attribute/namespace nodes, the attributes/namespaces **are not children** of the associated element node.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<book>
 <chapter>
  <title>Various Fruits</title>
  <para>
   The next chapters introduce different kinds of fruits, like
   <fruit figref="fr_virg">strawberries</fruit> or <fruit figref="apple">apples</fruit>.
  </para>
 </chapter>
 <chapter>
  <title>Strawberries</title>
  <para>
   stre[a]w berige; stre[a]w straw + berie berry;
   perhaps from the resemblance of the runners of the plant to straws.
  </para>
  <para>
   A fragrant edible berry, of a delicious taste and commonly of a red colour.
  </para>
  <para>
   The common American strawberry is
   <figure caption="Fragaria virginiana" data="fr_virg.jpg" id="fr_virg">Fragaria virginiana</figure>,
   the European is
   <figure caption="Fragaria vesca" data="fr_vesca.jpg" id="fr_vesca">Fragaria vesca</figure>.
  </para>
 </chapter>
</book>
```



Tree Representation for the XML fragment example, cf. XPath 1.0 Data Model

XPath Expressions: Syntactic Building Blocks

- **primary** expressions: strings, numbers, booleans, location paths, predicates, function calls, variable reference.
- **complex** expressions: unions, filters, relational expressions.
- basic expression **types**: string, number, boolean, node-set.
- expression **evaluation** done in a context, consisting of:
  - the context **node**.
  - the context **position** and **size**.
  - a set of variable bindings.
  - a function library.
  - a set of namespace declarations in expression scope.
- examples
  - 'Markup-Sprachen und semi-strukturierte Daten'
  - ($x + $y) $\cdot$ 2 $>$ 10.7
  - //lecture[@name = 'Markup' and contains(author,'Bry')]

Location Paths: The MOST Important Constructs

A path is constructed from steps, which have:

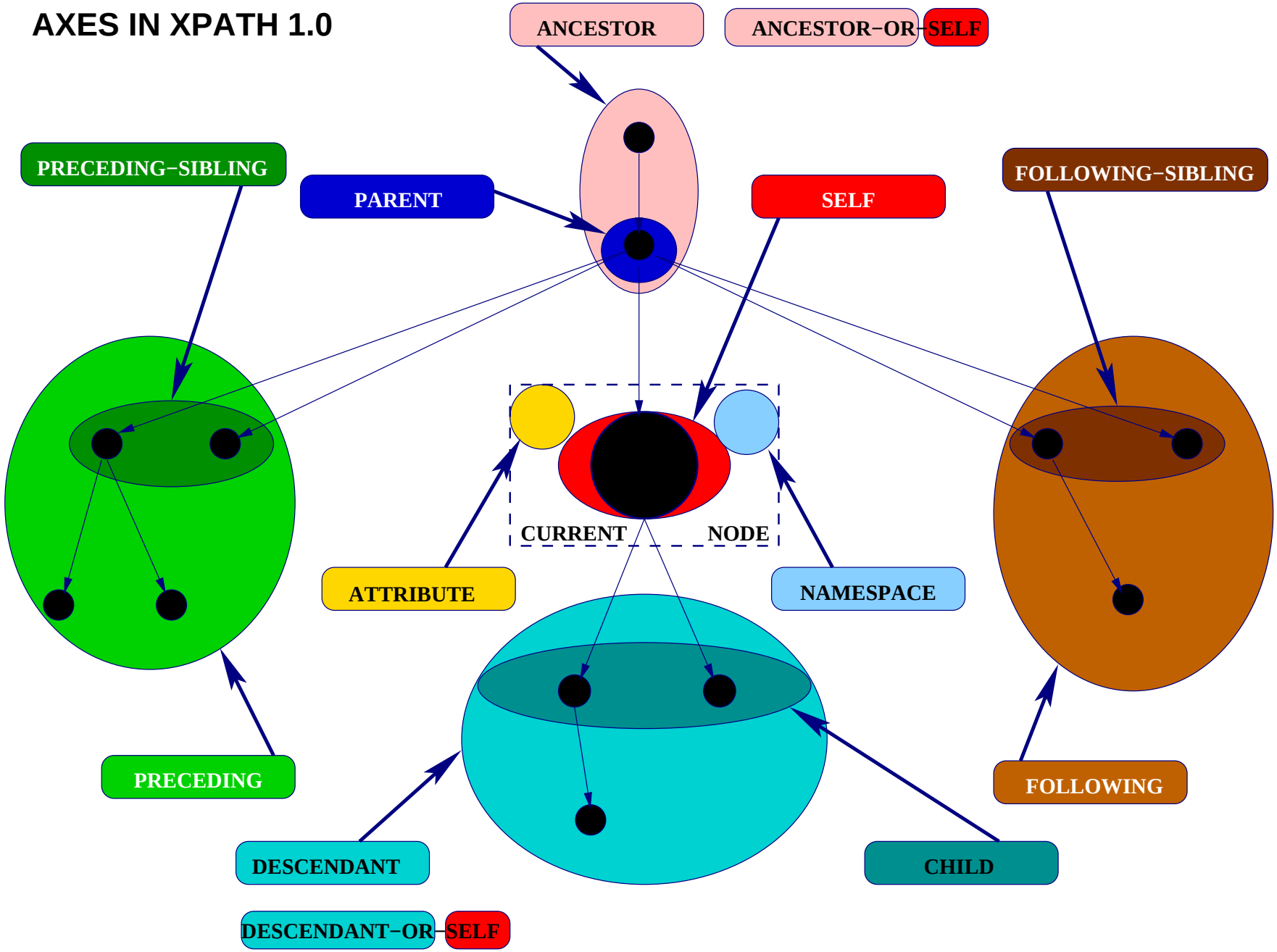
- an **axis**, which specifies the tree relationship between the nodes.
- a **node** test, which specifies the name of the selected nodes.
- zero or more **predicates**, which refine the set of nodes selected by the location step.

Examples:

- **child**::para
- **child**::figure[**attribute**::id="fr_vesca"]
- **child**::*[position()=last()]

Abbreviated syntax:

- **@name** for `attribute::name`
- **para[1]** for `child::para[position()=1]`
- **.//para** for `self::node()/descendant-or-self::node()/child::para`
- **.** for `self::node()`
- **../para** for `parent::node()/child::para`


AXES IN XPATH 1.0
ANCESTOR
ANCESTOR-OR-SELF
PRECEDING-SIBLING
PARENT
SELF
FOLLOWING-SIBLING
CURRENT NODE
ATTRIBUTE
NAMESPACE
PRECEDING
FOLLOWING
DESCENDANT
CHILD
DESCENDANT-OR-SELF

Example Break (1) - Testing Location Paths

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- downloadable from 5Sight:
  `http://www.fivesight.com/downloads/xpathtester.asp`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.

Core Function Library: Node-Set Functions

- *number* **last()**
  returns the context size from the expression evaluation context.
- *number* **position()**
  returns the context position from the expression evaluation context.
- *number* **count(node-set)**
  returns the number of nodes in the argument node-set.
- *node-set* **id(object)**
  selects elements by their unique ID, as declared in DTD.
- *string* **name(node-set?)**, *string* **local-name(node-set?)**,
  string **namespace-uri(node-set?)**
  returns the expanded/local name/namespace URI of the node in the argument node-set that is first in document order.

Example Break (2) - Testing Node-Set Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- downloadable from 5Sight:
  `http://www.fivesight.com/downloads/xpathtester.asp`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.

Core Function Library: String Functions

- *string* **string**(*object*?)
- *string* **concat**(*string*, *string*, *string**)
- *boolean* **start-with**(*string*, *string*)
- *boolean* **contains**(*string*, *string*)
- *string* **substring-before**(*string*, *string*)
- *string* **substring-after**(*string*, *string*)
- *string* **substring**(*string*, *number*, *number*?)
- *number* **string-length**(*string*?)
- *string* **normalize-space**(*string*?)
- *string* **translate**(*string*, *string*, *string*)

Example Break (3) - Testing String Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- downloadable from 5Sight:
  `http://www.fivesight.com/downloads/xpathtester.asp`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.

Core Function Library: Boolean & Number Functions

Boolean Functions

- *boolean* **boolean**(*object*)
- *boolean* **not**(*boolean*)
- *boolean* **true**()
- *boolean* **false**()

Number Functions

- *number* **number**(*object*?)
- *number* **sum**(*node-set*?)
- *number* **floor**(*number*)
- *number* **ceiling**(*number*)
- *number* **round**(*number*)

Example Break (4) - Testing Boolean & Number Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- downloadable from 5Sight:
  `http://www.fivesight.com/downloads/xpathtester.asp`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.

More Examples (1)

- Select the *figure* elements without attributes:
  //figure[not(@*)]

- Select the *chapter*s having three *para*graphs:
  //chapter[count(.//para) = 3]

- Select the first *para*graph of each *chapter*:
  //chapter//para[1]

- Select the first *para*graph of all *chapter*s:
  (//chapter//para)[1]

- Select the *figure*s with an attribute *caption* 'Fragaria virginiana' from the second *chapter*:
  //chapter[2]//figure[@caption = 'Fragaria virginiana']

- Select the *figure*s in *chapter*s 2 through 5:
  //chapter[position() >= 2 and position() <= 5]//figure

- Select *caption*s of *figure*s that are referenced by *figref* attributes of *fruit* elements in the first *chapter*:
  id(//chapter[1]//fruit/@figref)[self::figure]/caption

More Examples (2)

- Select *chapter*s in which the word 'Strawberry' is mentioned in at least one *para*graph:
  //chapter[.//para[contains(.,'Strawberry')]]

- Select *chapter*s in which the word 'Strawberry' is mentioned in every *para*graph:
  //chapter[count(.//para) = count(.//para[contains(.,'Strawberry')]) and .//para]
  OR
  //chapter[not(.//para[not(contains(.,'Strawberry'))]) and .//para]

- List the *name*s of the second-level *manager*s of all *employee*s whose *rating* is 'Good':
  id(id(/emp[rating = "Good"]/@mgr)[self::emp]/@mgr)[self::emp]/name

- List all distinct *employee*s from a *company*:
  (//company//employee)[not(.=preceding::employee)]

- Prepare a critical sequence report consisting of all elements that occur between the first and second *incision*:
  (//incision[2]/preceding::*) [count(. | (//incision[1]/following::*)) = count (//incision[1]/following::*)]

What's coming with XPath 2.0 ?

- support for **XML Schema** primitive datatypes.
- explicit **For Any** and **For All** quantifiers (`some` and `every`).
- **FR** construct from XQuery **FLoWeR** expression (`for-return`).
- extended set of **aggregation** functions (e.g. `min`, `max`, `avg` ...).
- **conditional** expressions (`if-then-else`).
- node-set **intersection** and **difference** functions (`intersect`, `except`).
- string matching using **regular expressions**.

Useful links

- W3C XPath Recommendation:
  http://www.w3.org/TR/xpath.

- XPath Tutorial from Zvon:
  http://www.zvon.org/xxl/XPathTutorial/General/examples.html.

- XPath Tester from 5Sight:
  http://www.fivesight.com/downloads/xpathtester.asp.

- XPath Tester from PhPXML:
  http://www.phpxml.org/scripts/testsuite/.

- XPath Tutorial from Resin:
  http://www.caucho.com/products/resin/ref/xpath.xtp.

- XPath Implementation from Xalan:

  http://xml.apache.org/xalan-j/index.html.