Manipulating XML Trees

CS 431 – March 5, 2007

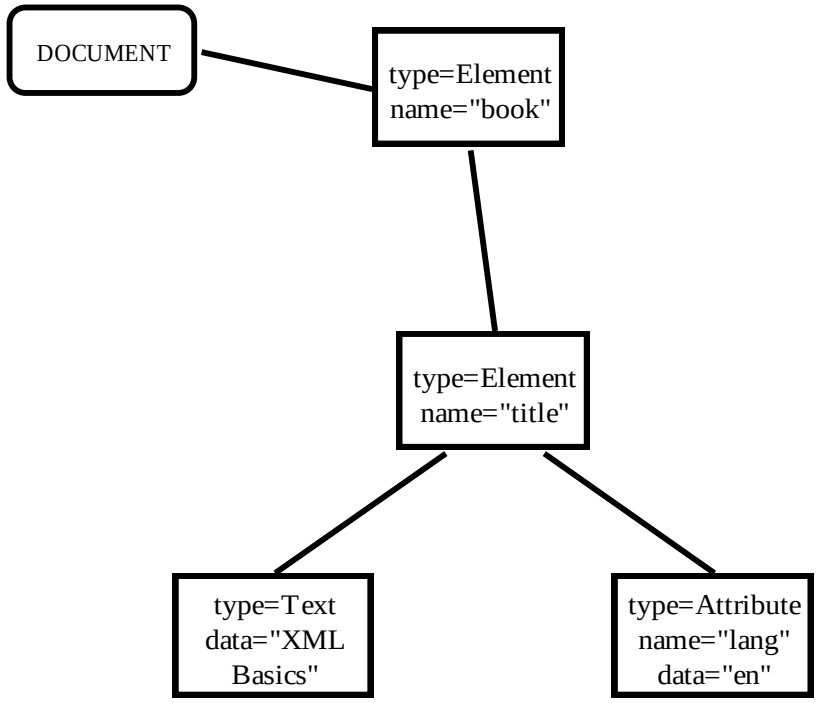
Carl Lagoze – Cornell University

XPath

- Language for addressing parts of an XML document
  - XSLT
  - Xpointer
  - XQuery
- Tree model based onDOM
- W3C Recommendation
  - 1999 – 1.0
    - http://www.w3.org/TR/xpath
  - 2007 – 2.0 (Backwards compatible)
    - http://www.w3.org/TR/xpath20/

Remember to think in terms of DOM trees

```
<?xml version="1.0"
    encoding="UTF-8"?>
<book>
    <title lang='"en"'>"XML
    Basics"</title>
</book>
```



Xpath Concepts

- Context Node (starting point)
  - current node in XML document that is basis of path evaluation
  - Default to root (remember that root is "Document")
- Location Steps (directions)
  - Sequence of node specifications
  - Evaluation of each node specification creates a new context
    - always within previous context
  - Think of file paths
    - /nodeSpec/nodeSpec/nodeSpec
- Node Specification
  - Increasingly detailed specification of the sequence of nodes to which the location step should lead

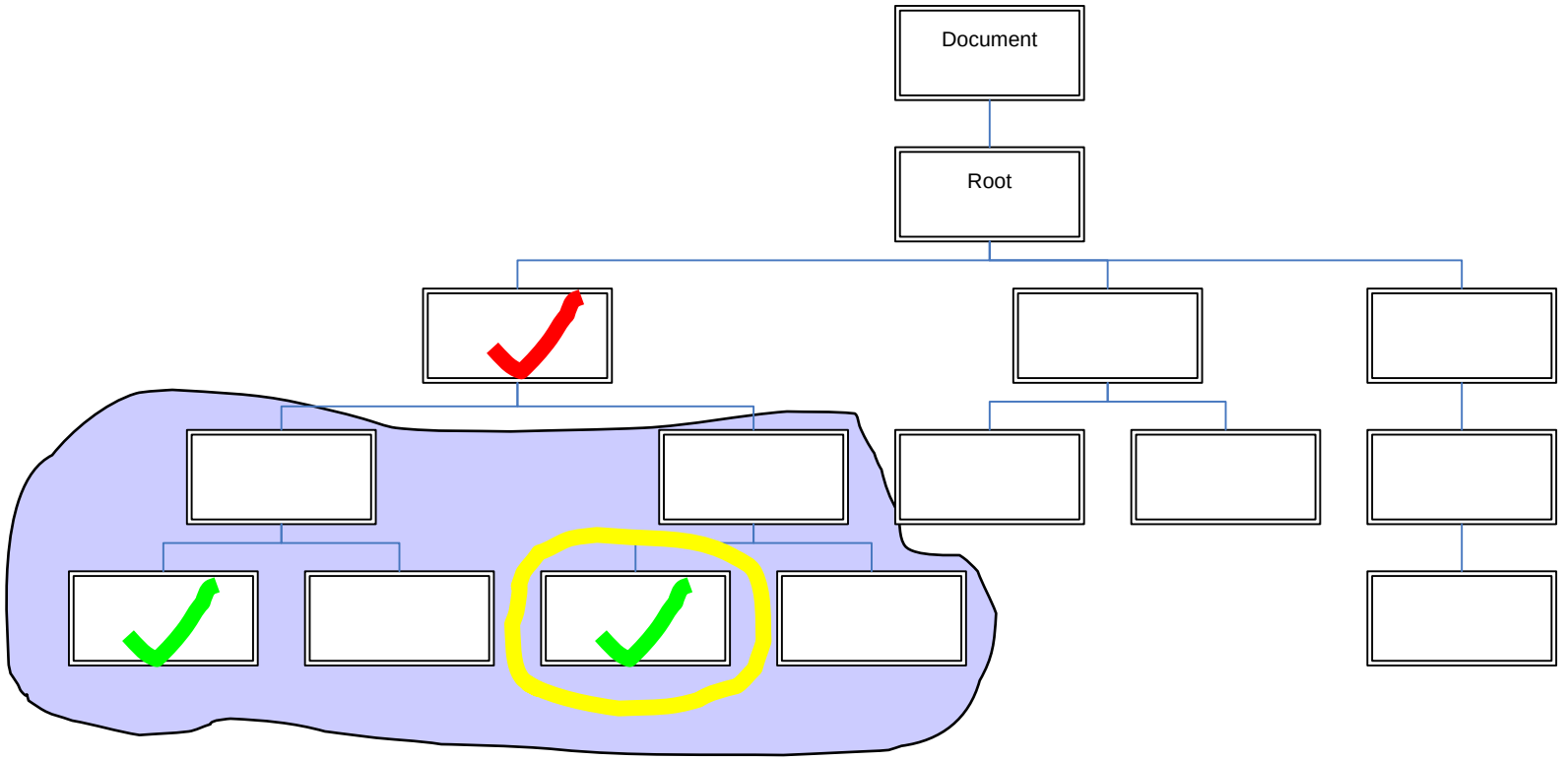
Location Step Context Syntex

- /nodeSpec/nodeSpec/…. – absolute from document root
- nodeSpec/nodeSpec …. – relative from context
- //nodeSpec/nodeSpec – anywhere in document tree

Each nodeSpec is: axis::node-test[predicate]

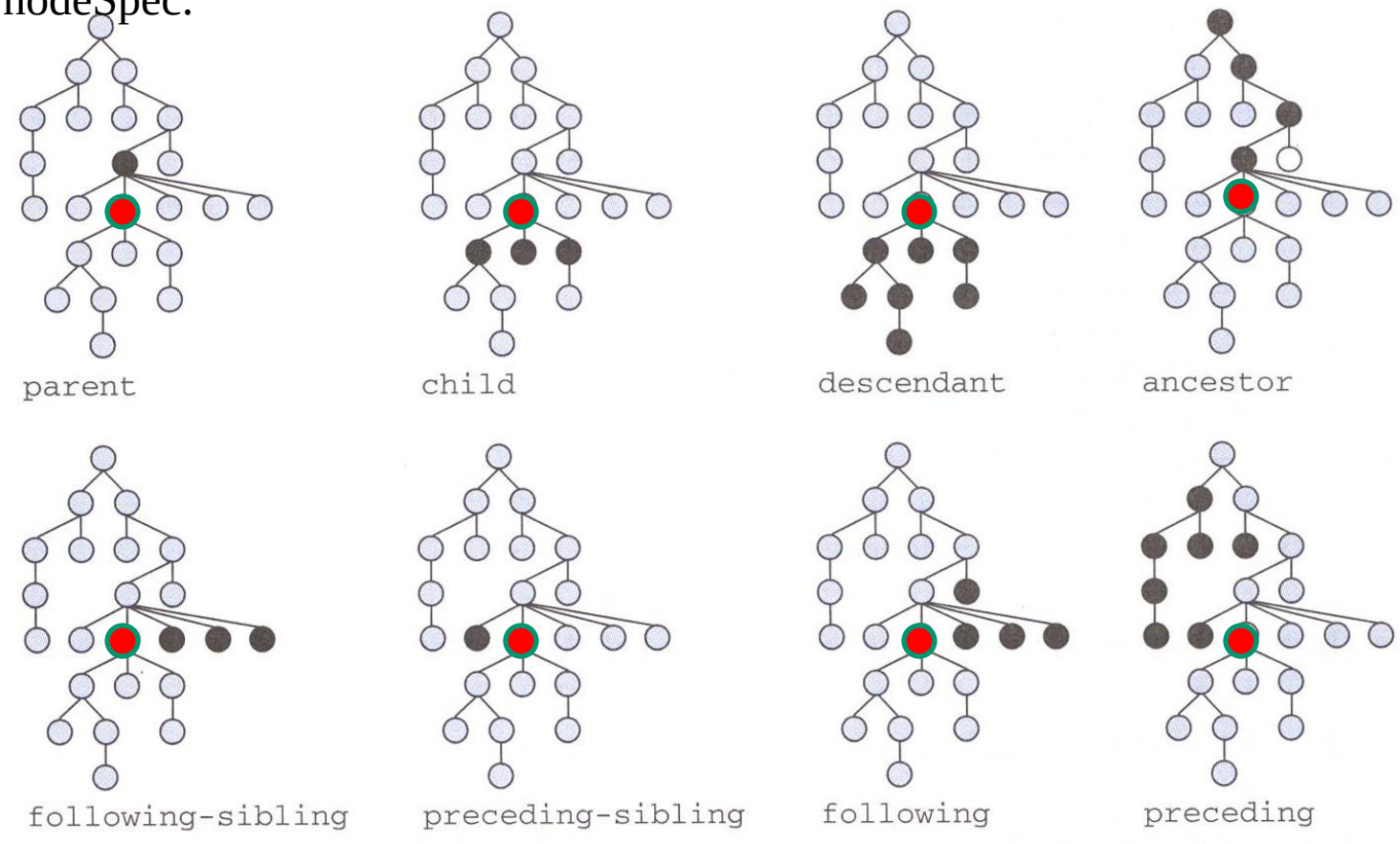
- Axis – sub-tree(s) selection from context node
- Node Test – select specific elements or node type(s)
- Predicates – predicate for filtering after axis and node tests

Context, Axis, Node Test, Predicate



axis::node-test[predicate]

First approximation of the sequence of nodes to obtain from the location nodeSpec.



From Moller and Schwartzback XML & Web Techologies

axis::node-test[predicate]

- Filter the kind of nodes within the specified axis
  - text() – only character data
  - comment() – only comments
  - node() – all nodes of axis type (attribute or element)
  - [name] – all nodes of given name
    - e.g. "Book"
      - make sure to pay attention to namespaces!!!!
  - Wildcard: *
- *Remember in DOM that everything is a node*

axis::node-test[predicate]

- Boolean and comparative operators
- Types
  - Numbers
  - Strings
  - node-sets (the set of nodes selected)
- Functions
  - Examples
    - *boolean* starts-with(*string, string*)
    - *number* count(*node-set*)
    - *number* position()

xpath examples

- /child::source/child::*AAA*
  - or /source/*AAA* since child is default axis
- /child::source/child::*[position()=2]
  - or /source/*[2]
- /child::source/child::*AAA*[position()=2]/attribute::id
  - or /source/*AAA*[2]/@id
- /child::source/child::*AAA*/@*
  - or /source/*AAA*/@*
- /child::source/child::*AAA*[contains(. ,'a1')]
  - /source/*AAA*[contains(. ,'a1')]
- /descendant::BBB/child::*CCC*

http://www.cs.cornell.edu/courses/CS431/2007sp/examples/xpath/base.xml

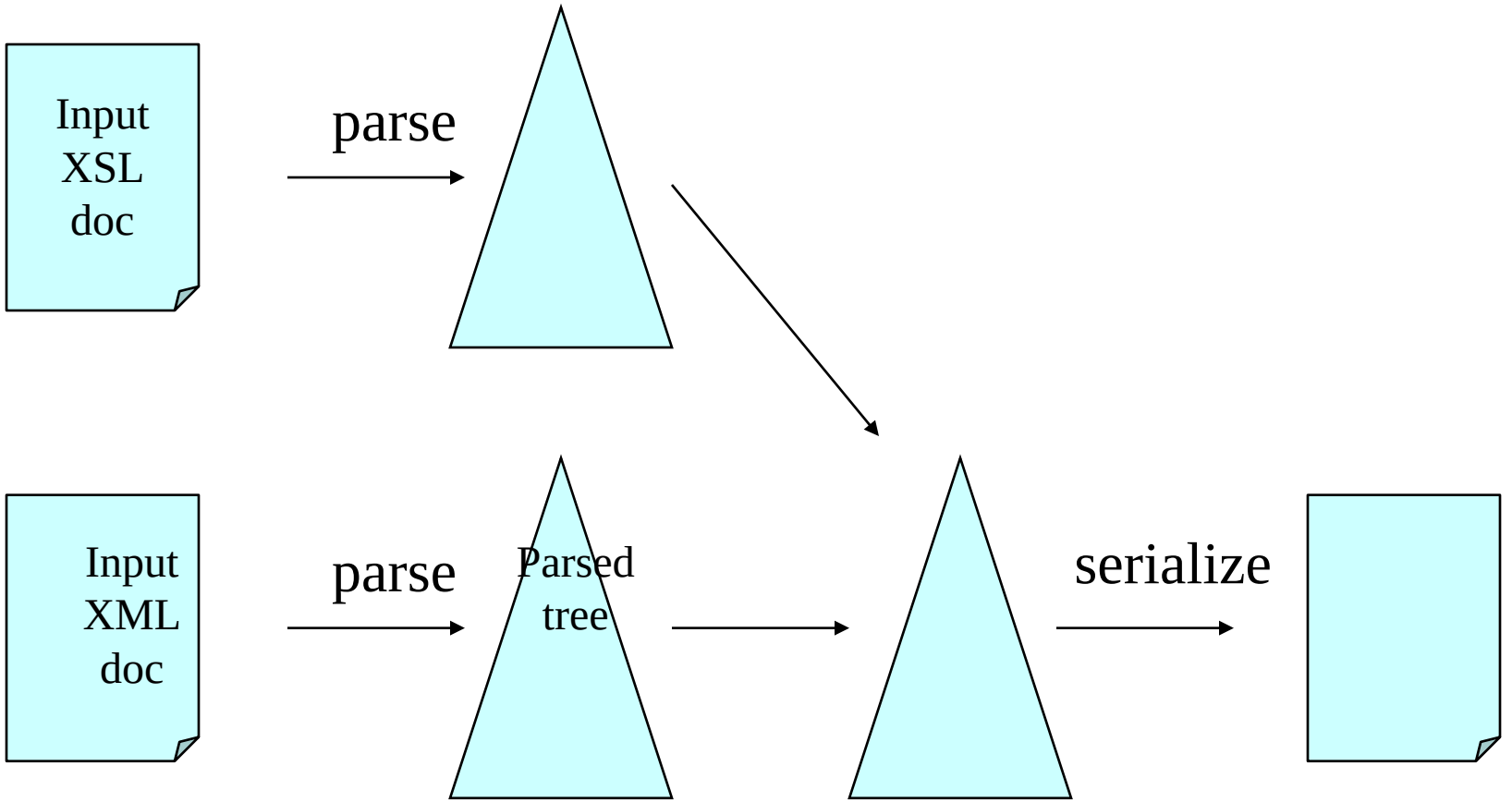
XML Transformations (XSLT)

- Origins: separate rendering from data
  - Roots in CSS
- W3C Recommendation
  - http://www.w3.org/TR/xslt
- Generalized notion of transformation for:
  - Multiple renderings
  - Structural transformation between different languages
  - Dynamic documents
- XSLT – rule-based (declarative) language for transformations

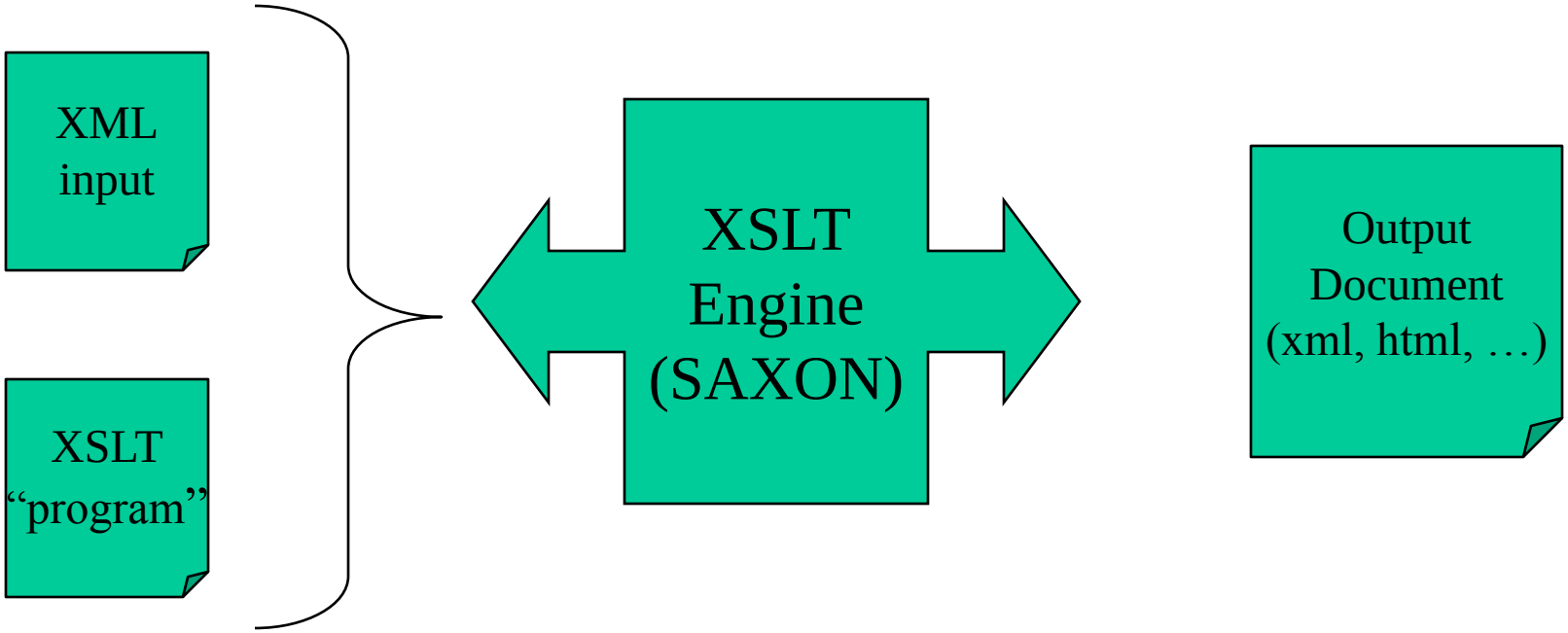
XSLT Capabilities

- Produce any type of document
  - xHTML, XML, PDF…
- Generate constant text
- Filter out content
- Change tree ordering
- Duplicate nodes
- Sort nodes
- Any computational task (XSLT is "turing complete")
  - extra credit if you write an OS in XSLT

XSLT Processing Model



XSLT "engine"


XML
input
XSLT
"program"
XSLT
Engine
(SAXON)
Output
Document
(xml, html, …)

Stylesheet Document or Program

- XML document rooted in <stylesheet> element
- XSL tags are in namespace http://www.w3.org/1999/XSL/Transform
- Body is set of templates or rules
  - match attribute specifies xpath of elements in source tree
  - Body of template specifies contribution of source elements to result tree

Stylesheet Document or Program

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:fo="http://www.w3.org/1999/XSL/Format">
        <xsl:template match="para">
                <p>
                        <xsl:apply-templates/>
                </p>
        </xsl:template>
        <xsl:template match="emphasis">
                <b>
                        <xsl:apply-templates/>
                </b>
        </xsl:template>
</xsl:stylesheet>
```

XSL Execution Model

- Templates represent a set of rules
- Rule matching is done within current tree context
- Rules are not executed in order
- Default behavior is depth-first walk of tree, outputting element values

- http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/base.xml
- http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/null.xsl

Template Form

```
<xsl:template match="para">
    <p>
        <xsl:apply-templates/>
    </p>
</xsl:template>
```

- Sequential execution within template
- Elements from xsl namespace are transform instructions
- Match attribute value is xpath expression setting rule for execution of body
- Non-xsl namespace elements are literals.
- <xsl:apply-templates>
  - set context to next tree step
  - re-evaluate rules

Result Tree Creation

- Literals – any element not in xsl namespace
- <xsl:text> - send content directly to output (retain whitespaces)
- <xsl:value-of> - expression processing
- <xsl:copy> and <xsl:copyof> - Copy current node or selected nodes into result tree
- <xsl:element> - instantiate an element
- <xsl:attribute> - instantiate an attribute

A simple example

- XML base file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/simple.xml
- XSLT file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/simple.xsl

Modifying rule set and context

- Mode setting
  - <xsl:apply-templates mode="this">
  - <xsl:template match="foo" mode="this">
  - <xsl:template match="foo" mode="that">
- Context setting
  - <xsl:apply-templates select="//bar">
  - Modifies default depth-first behavior
- Conflict resolution rules

- http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/elements.xsl
- http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/elements2.xsl

XSLT Procedural Programming

- Sequential programming style
- Basics
  - for-each – loop through a set of elements
  - call-template – like a standard procedure call

For-each programming example

- XML base file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/foreach.xml
- XSLT file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/foreach.xsl

Call-template programming example

- XML base file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/call.xml
- XSLT file
  - http://www.cs.cornell.edu/lagoze/courses/CS431/2007sp/Examples/Lecture10/call.xsl

Various other programming constructs

- Conditionals
- Variables (declaration and use)
- Some type conversion
- Sorting

Result Tree Creation

- Literals – any element not in xsl namespace is inserted into result tree

```
<xsl:template match="para">
  <p>
    This is the sentence
    <xsl:apply-templates/>
  </p>
</xsl:template>
```

Result Tree Creation

- <xsl:text> - send content directly to output (retain whitespaces)

```
<xsl:text>, and </xsl:text>
```

Result Tree Creation

- <xsl:value-of> - extract element values (anywhere in the tree)

```
<tr>
 <td><xsl:value-of select="catalog/cd/title"/></td>
 <td><xsl:value-of select="catalog/cd/artist"/></td>
</tr>
```

Result Tree Creation

- <xsl:copyof> - Copy selected nodes into result tree

```
<xsl:copy-of select="table"/>
```

Result Tree Creation

- <xsl:element> - instantiate an element
- <xsl:attribute> - instantiate an attribute

```
<xsl:element name="newEl">
   <xsl:attribute name="newAttr">
      <xsl:value-of select="/top/AAA[1]"/>
   </xsl:attribute>
</xsl:element>
```

Default Rules (Must replace to change them)

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

- Applies to root node and element nodes
- Recurses depth first

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

- Applies to text and attribute nodes
- Copies value to output tree

A simple example

- XML base file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/simple.xml
- XSLT file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/simple.xsl

Modifying rule set and context

- Context setting
  - <xsl:apply-templates select="//bar">
  - Modifies default depth-first behavior
- There are conflict resolution rules

- http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/elements.xsl
- http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/elements2.xsl

Modifying rule set and context

- Mode setting
  - <xsl:apply-templates mode="this">
  - <xsl:template match="foo" mode="this">
  - <xsl:template match="foo" mode="that">

  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/modes.xsl

Namespaces in XSLT

- The XSL document MUST know about the namespaces of elements that it references (via XPATH expressions) in the instance document
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/baseNS.xml
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/elementsNS.xsl
- Watch out for the default namespace!!
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/baseNoNS.xml
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/elementsNoNS.xsl

XSLT Procedural Programming

- Sequential programming style
- Basics
  - for-each – loop through a set of elements
  - call-template – like a standard procedure call

For-each programming example

- XML base file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/foreach.xml
- XSLT file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/foreach.xsl

Call-template programming example

- XML base file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/call.xml
- XSLT file
  - http://www.cs.cornell.edu/courses/CS431/2006sp/examples/xslt/call.xsl

Associating an XML document with a transform

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="http://www.cs.cornell.edu/Courses/cs502/2002SP/Demos/xslt/simple.xsl"?>
<para>
  this is
  <emphasis>
    big
  </emphasis>
  text
</para>
```