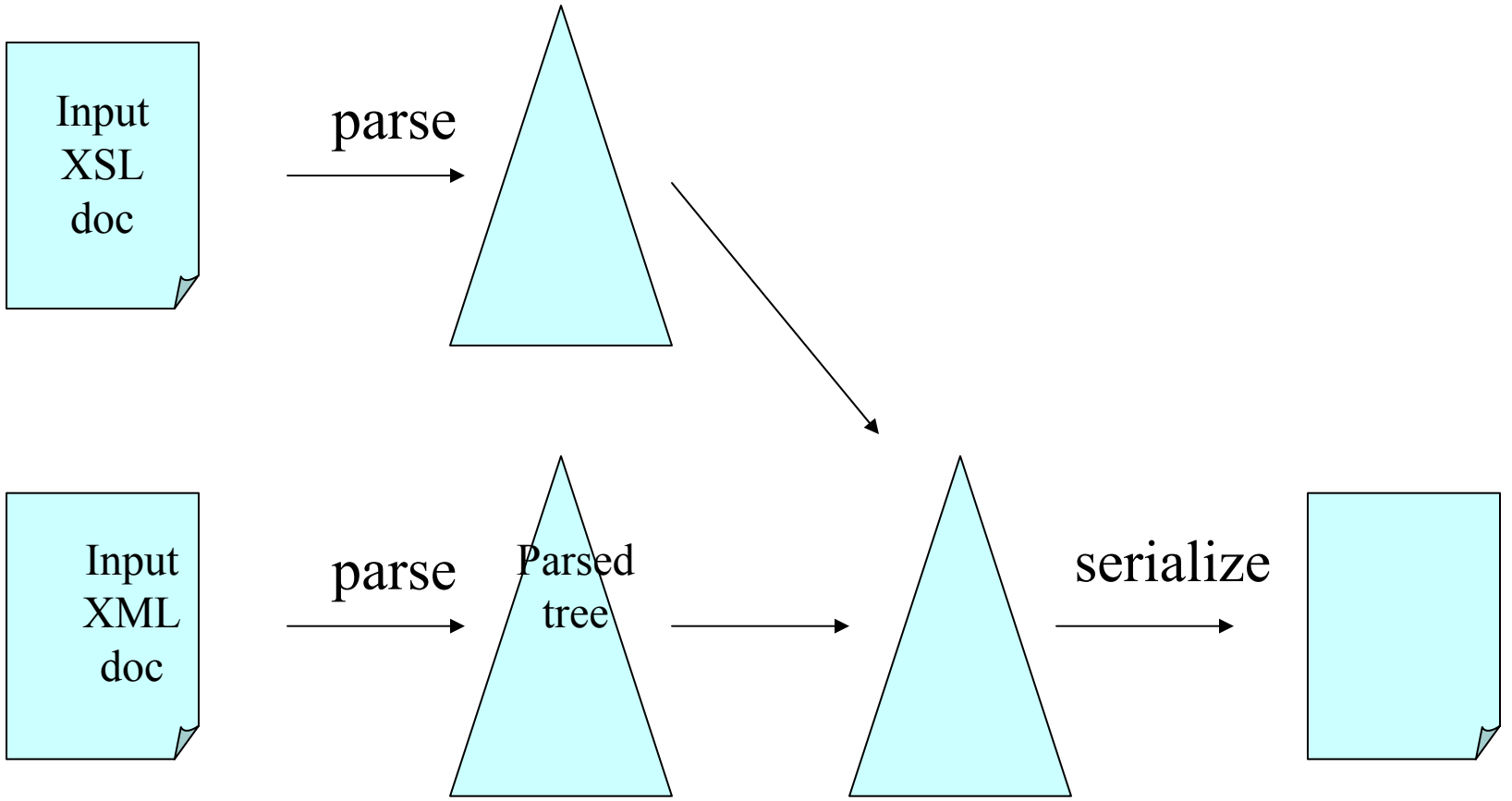
XML Wrap-up

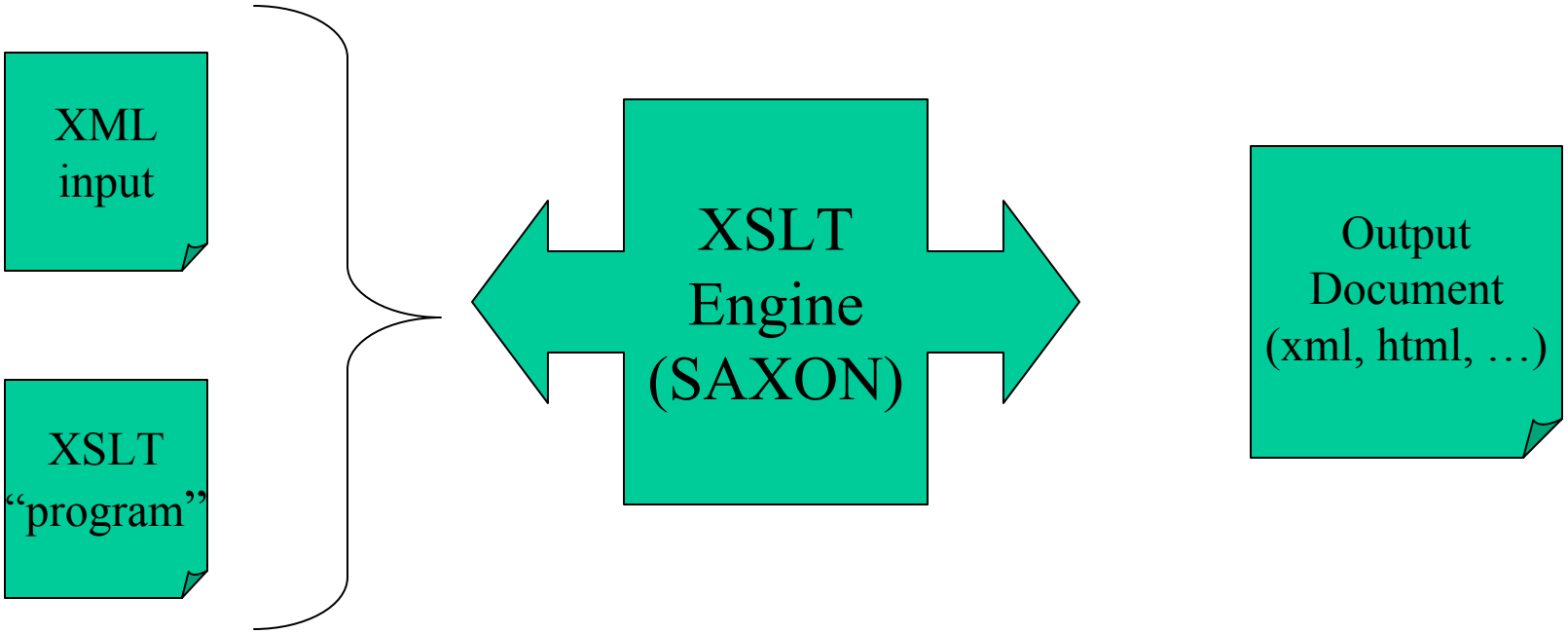
CS 431 – March 1, 2006

Carl Lagoze – Cornell University

XSLT Processing Model



XSLT "engine"


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

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>
```

Template Form

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

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

XSL Execution Model

- Templates represent a set of rules
- Rule matching is done within current tree context
- Rules are not executed in order
- Precedence given to more specific rules
- Default behavior
  - Write element value
  - Reevaluate rules after depth-first tree step
  - Default behavior will ALWAYS happen unless overwritten by specific rule

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
  - Default <apply-templates> action is to walk all but attributes

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

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

Examples of default behavior

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

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>
```

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/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

Variables

- Scoped in normal fashion
  - Global
  - Within tree nesting level
- No static typing – take type of setting
  - string, number, boolean, node-set (set of nodes, sub-tree)

Variables

- Initialization
  - `<xsl:variable name="age" select="25"/>`
  - Distinguish between string literal and xpath
    - `<xsl:variable name="city" select="'ithaca'"/>`
      - set variable to string "ithaca"
    - `<xsl:variable name="city" select="ithaca"/>`
      - set variable to result of xpath expression "ithaca"

Variables

- Initialization
  - Construct temporary tree
    - <xsl:variable name="temptree">
      <foo><bar></bar></foo>
      </xsl:variable>
  - Usage – precede by '$'
    - <xsl:value-of select="$city"/>

Variables (assignment)

- No assignment after initialization
- Think functional programming model (LISP, ML, Scheme)
- Use conditional initialization (<xsl:choose>)
- Use recursion rather than iteration for repetitive tasks

Variables example

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

Various other programming constructs

- Conditionals
- Variables (declaration and use)
  - Once set, can't be reset
  - Functional programming style
  - Use recursion
- Some type conversion
- Parameters
- Sorting

Inputs and outputs

- Inputs - Default is single input document
  - `Document('URL')` function returns root node of document at URL
- Outputs – Default is single XML document
  - `<xsl:output method={"xml" | "html" | "text"}/>` changes output format
  - `<xsl:document href="URL">`
    - Anywhere in xsl file changes the output destination.

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>
```