Stylesheets for XML
XSL

XSL

- eXtensible StyleSheet Language.
- XSL is Language (an XML application) to define the appearance and behaviour of an XML document.

- XSL is a family of recommendations for defining XML document transformation and presentation. It consists of 2 parts:

XSLT & XPath & XSL - FO

XSLT / XPath / XSL-FO

- **XSL Transformations (XSLT)**

Language for describes rules for transforming XML documents.

- **XML Path Language (XPath)**

Expression language used by XSLT to access or refer to parts of an XML document .

- **XSL Formatting Objects (XSL-FO)**

XML vocabulary for specifying formatting semantics. The precise description of page layout.

XSL versus CSS

- XSL uses a XML notation, CSS uses its own
- In CSS, the formatting object tree is almost the same as the source tree.
- In XSL, it provides means to perform operations such as looping, summation ,counting…etc.
- In XSL, these functionalities can be used to restructure a source document to produce a result document, different in structure but based on the source document.

XSL versus CSS (cont.)

- CSS is very poor in manipulating the behavior of an XML document.
- Basic feature of XSL is that it is capable of manipulating both behavior and appearance of the document.
- CSS can't display XML elements in different order than they're given in the XML document.
- CSS can't display attributes.

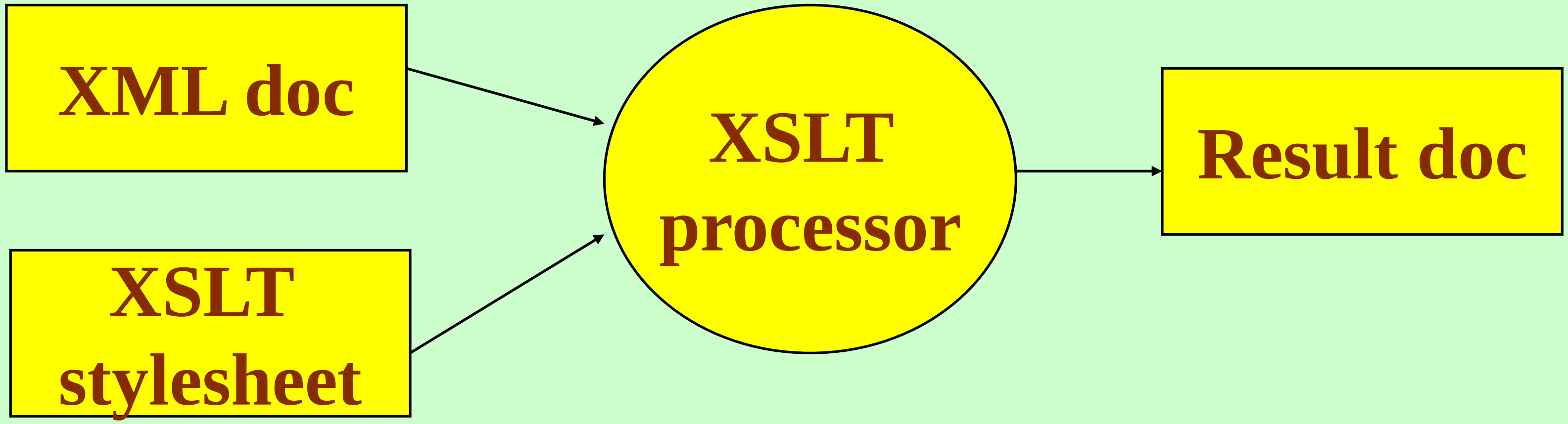
XSL versus CSS (cont.)

	CSS	XSL
Can be used with HTML?	Yes	No
Can be used with XML?	Yes	Yes
Transformation language?	No	Yes
Syntax	CSS	XML

XSLT processors

XSLT processor is the software that transforms an XML file into formatted output.

- **Apache xalan & cocoon**
- **MSXML from Microsoft**



How Does XSLT Transform XML?

- The first stage is a structural transformation, in which the data is converted from the structure of the incoming XML document to a structure that reflects the desired output.

- The second stage is formatting, in which the new structure is output in the required format such as HTML.

The XSLT Language

- XML syntax (using the xsl: namespace) :

  XSL language consists of *directives* (elements in this namespace)

- Rule-based :
  - stylesheets consist of a series of *templates* that contain rules for the processing of a particular element.
  - rules are applied depending upon the logical structure of the document.
- Rules may be conditional
- XSL may contain variables
- XPath & Namespaces are essential in writing XSLT .

The XSLT Language (cont.)

- XSLT-defined elements are distinguished by use of the namespace

  http://www.w3.org/1999/XSL/Transform
- XSLT elements are elements in the XSLT namespace whose syntax and semantics are defined in this specification.
- The transformation is achieved by a set of template rules .
- A template rule associates a pattern, which matches nodes in the source document.

ISO 9001

Simple XSLT stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="XPath Expression">
         <html>
              <body>
                  <xsl:apply-templates />
              </body>
        </html>
</xsl:template>
<xsl:template match=" XPath Expression ">
         <p>
           <xsl:value-of select="name" />
         </p>
</xsl:template></xsl:stylesheet>
```

<xsl:template>

- **The content of an <xsl : template> element in the stylesheet is a sequence of elements and text nodes.**
- **This sequence of elements and text nodes is called a *sequence constructor*, because the result of evaluating it is itself a sequence.**

XPath

- **A Syntax for addressing parts of an XML document.**
- **It defines 2 main components:**

  - Expression syntax used to locate parts of the XML document.

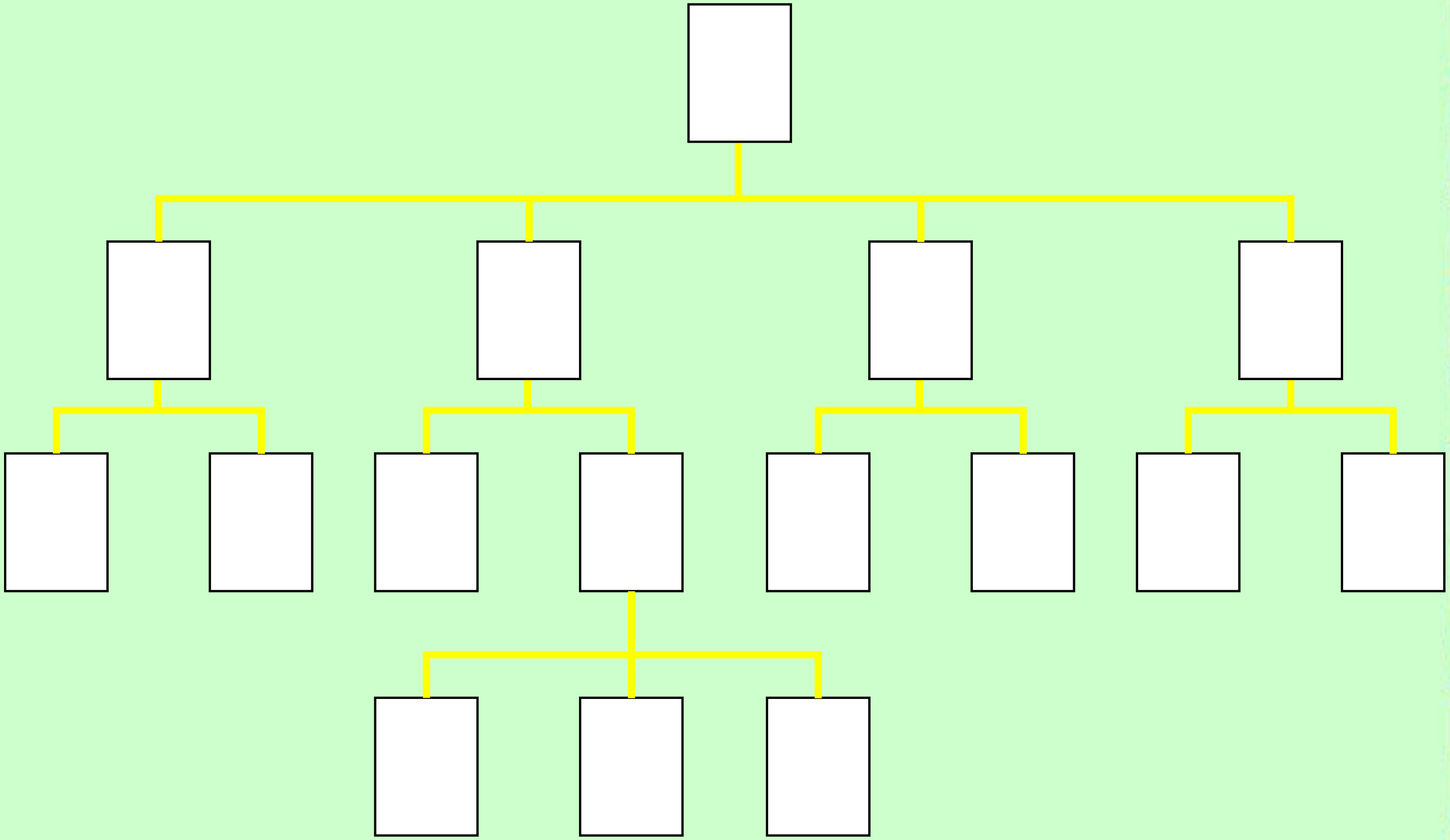
  - Basic set of functions known as "Xpath core library"
- **Supports a tree structure expression**
- **XPath is a foundation for other services in the XML family such as "XSLT, XQuery"**

The Tree Model of XML

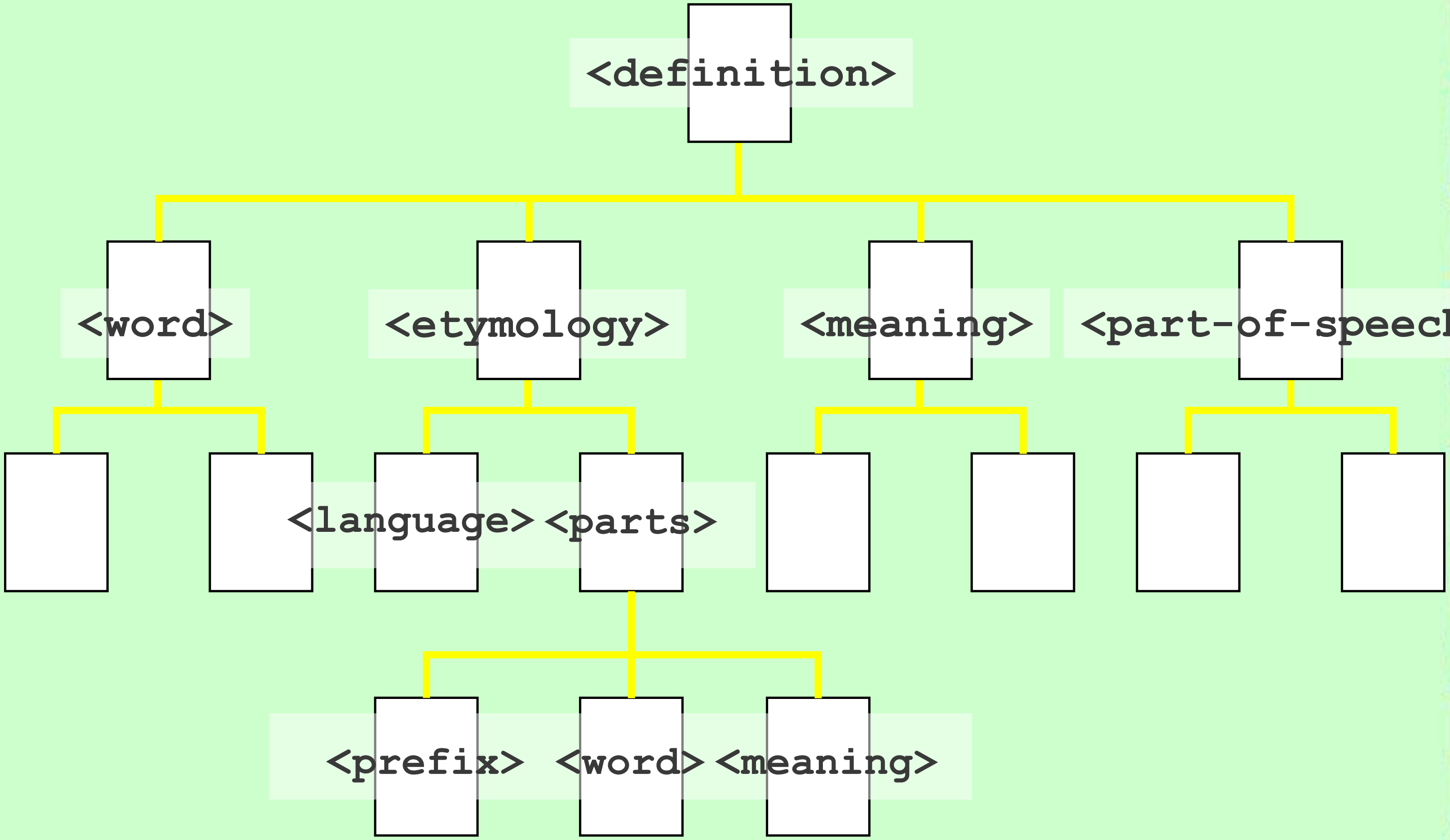
```
<definition>
   <word>export</word>
<etymology>
       <language>Latin</language>
           <parts>
               <word>jjjjj</word>
              <prefix>ex</prefix>
              <meaning>out</meaning>
           </parts>
</etymology>
<meaning> Send out (goods) to another country</meaning>
<part-of-speech>vt</part-of-speech>

</definition>
```

The Tree Model of XML



The Tree Model of XML



XML Node Types

- Root Node

The top level node, not the same as the root element.

- Element Node

An element bound by a start and finish tag (or a single empty-element tag)

- Text Node

A sequence of consecutive characters (PCDATA)

- Attribute Node

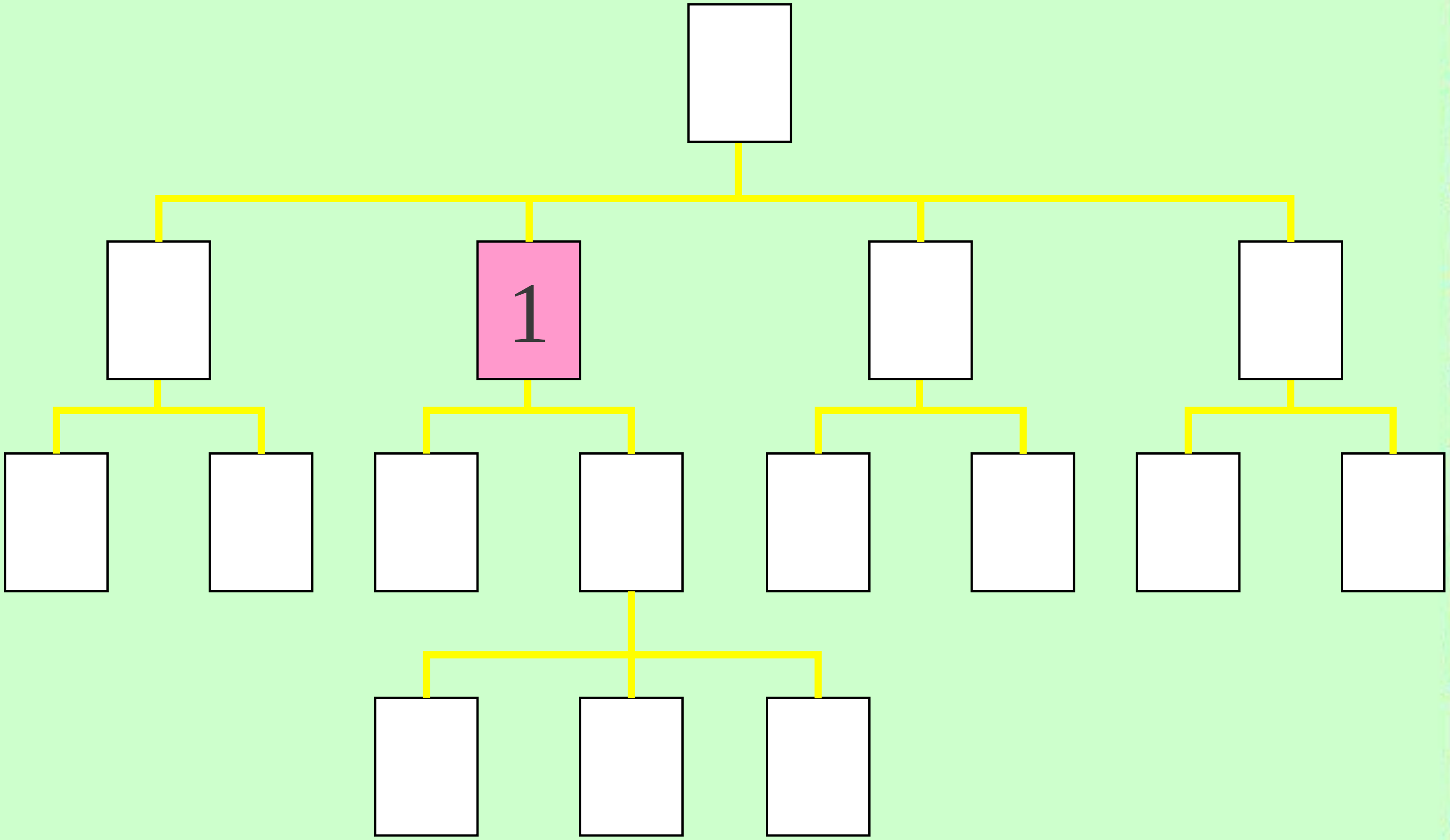
The name *and* value of an attribute inside an element

- Comment Node
- Processing Instruction Node
- Namespace Node

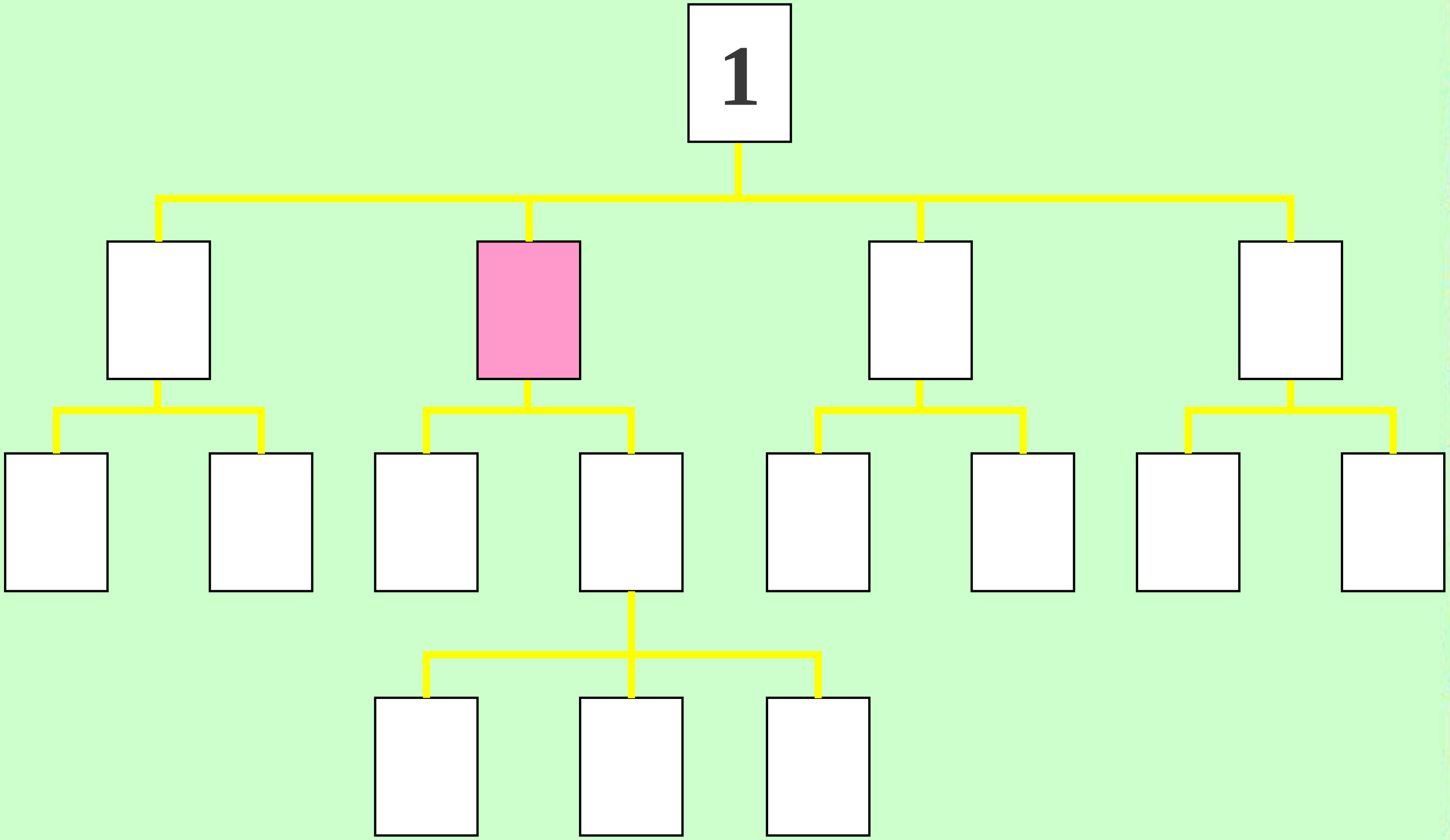
XML Node Relationships

- Self
- Parent
- Ancestor
- Child
- Descendant
- Following
- Following-Sibling
- Preceding
- Preceding-Sibling
- Attribute
- Namespace

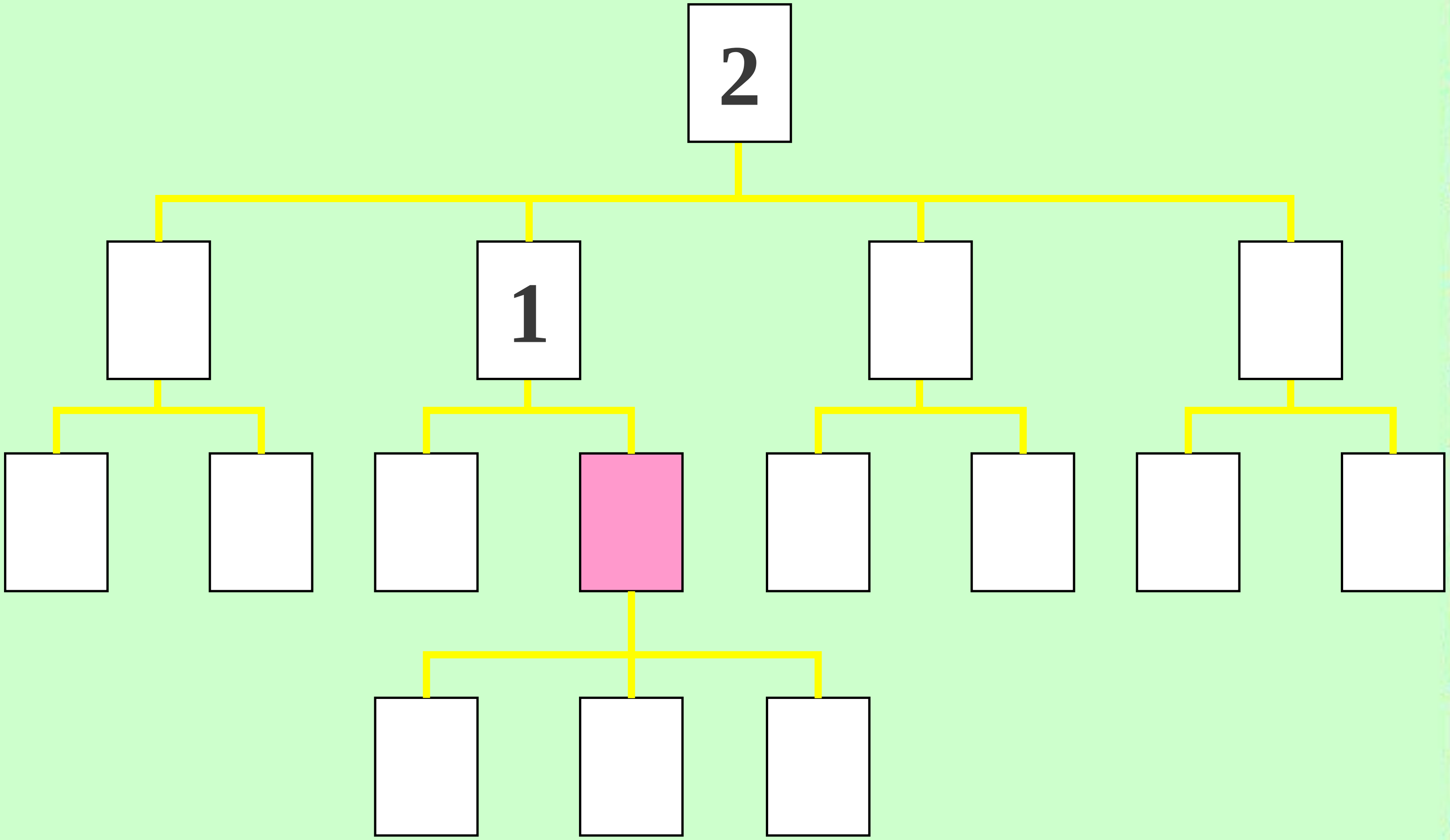
Self



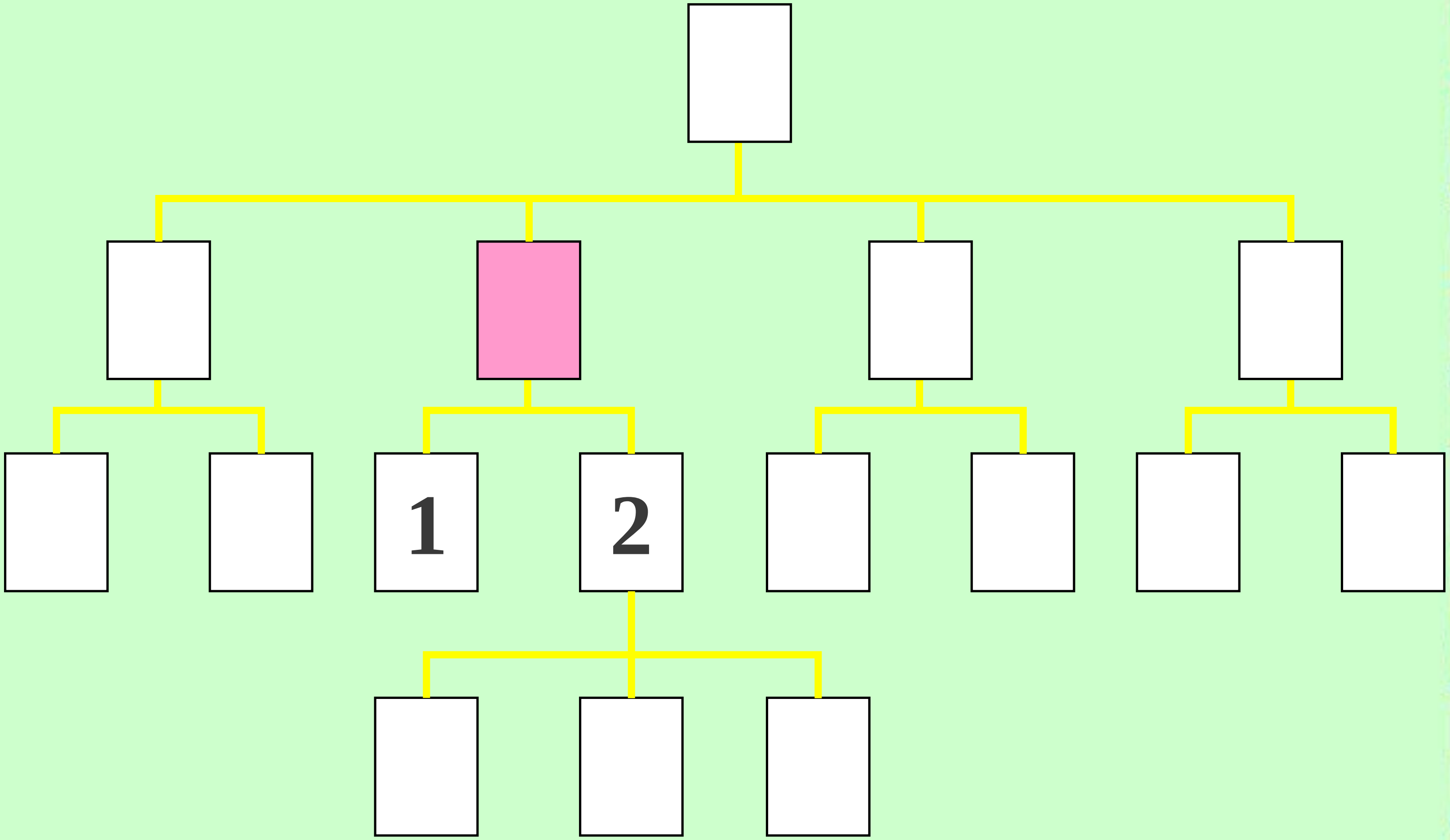
Parent



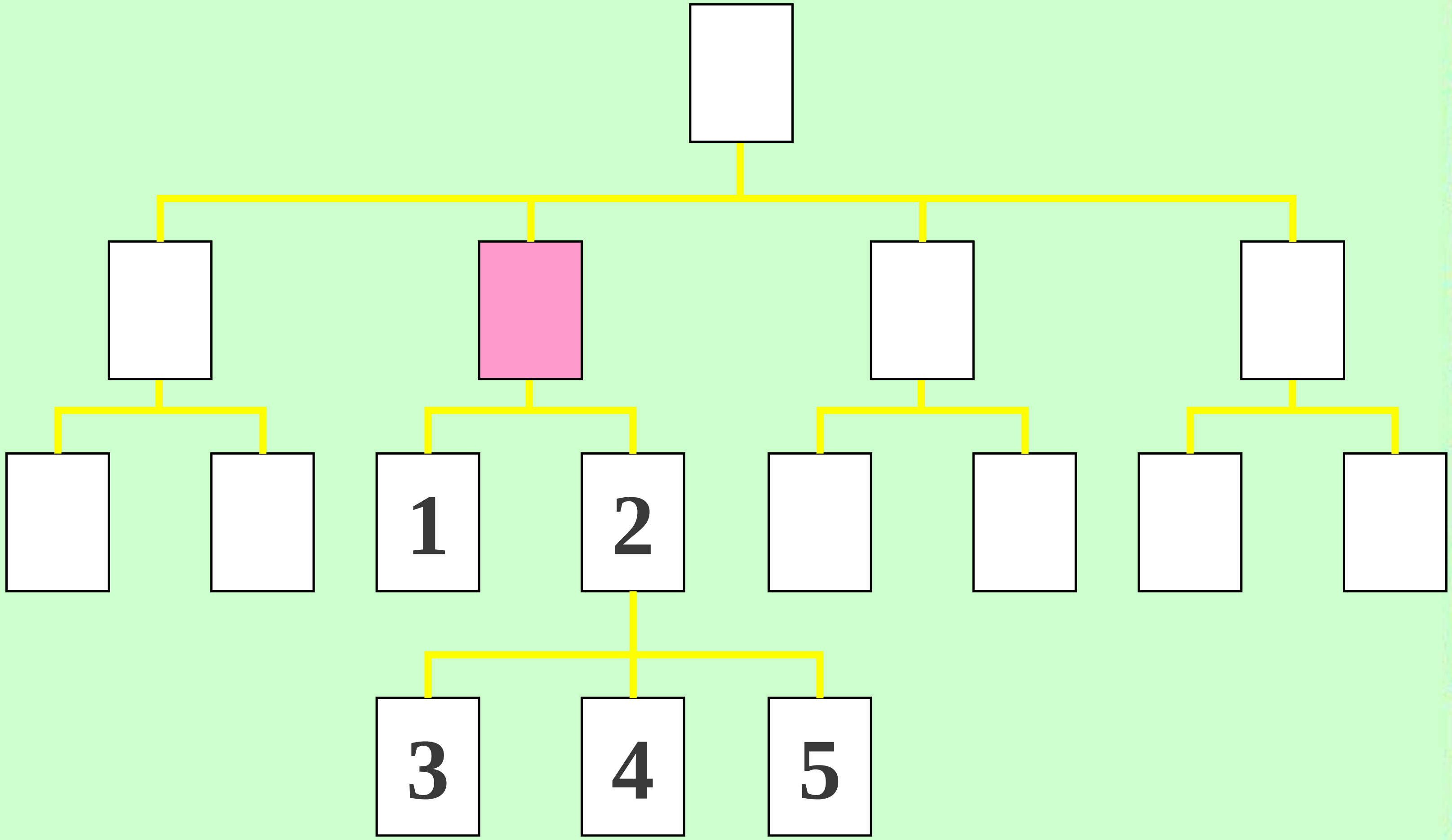
Ancestor



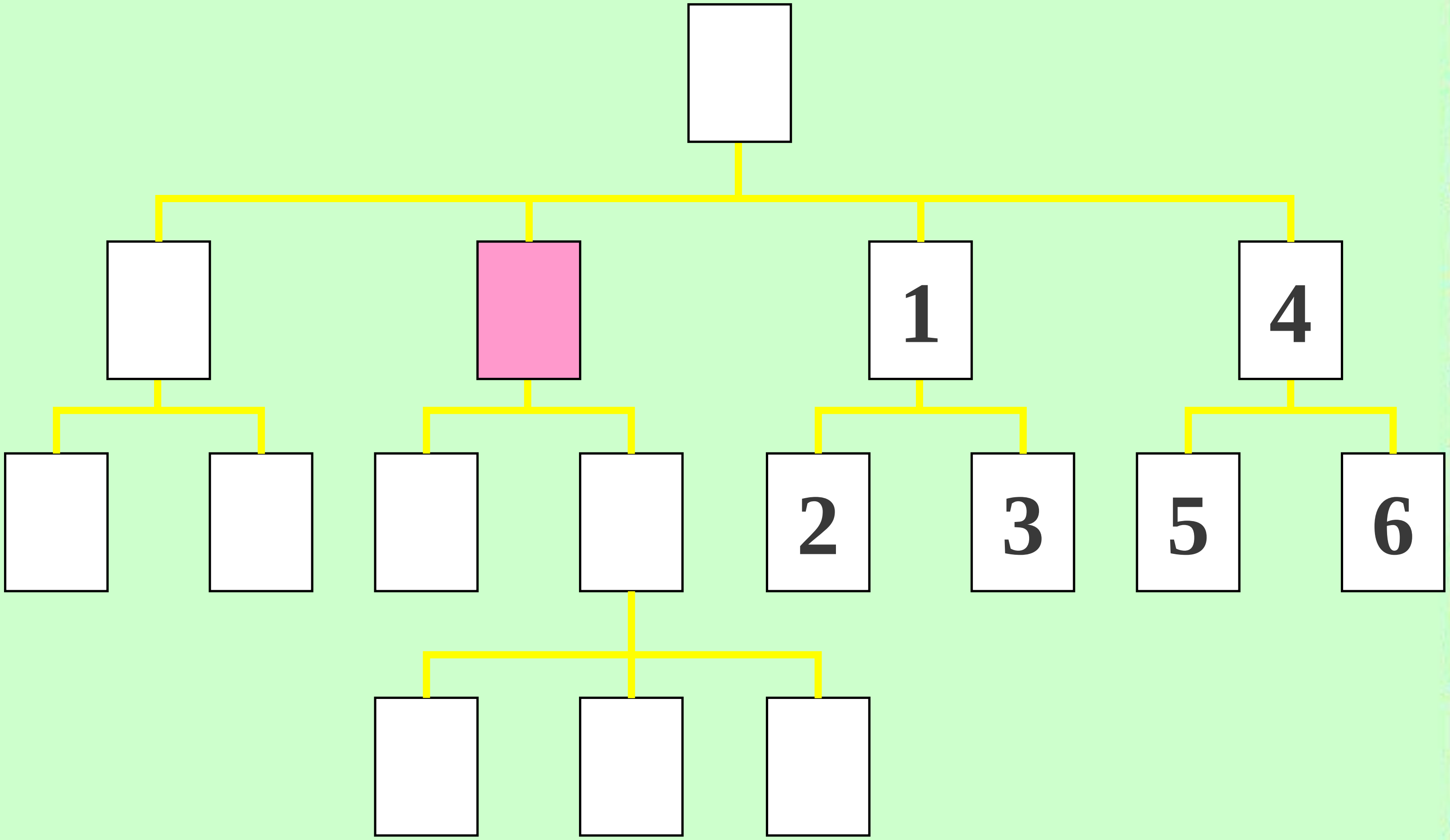
Child



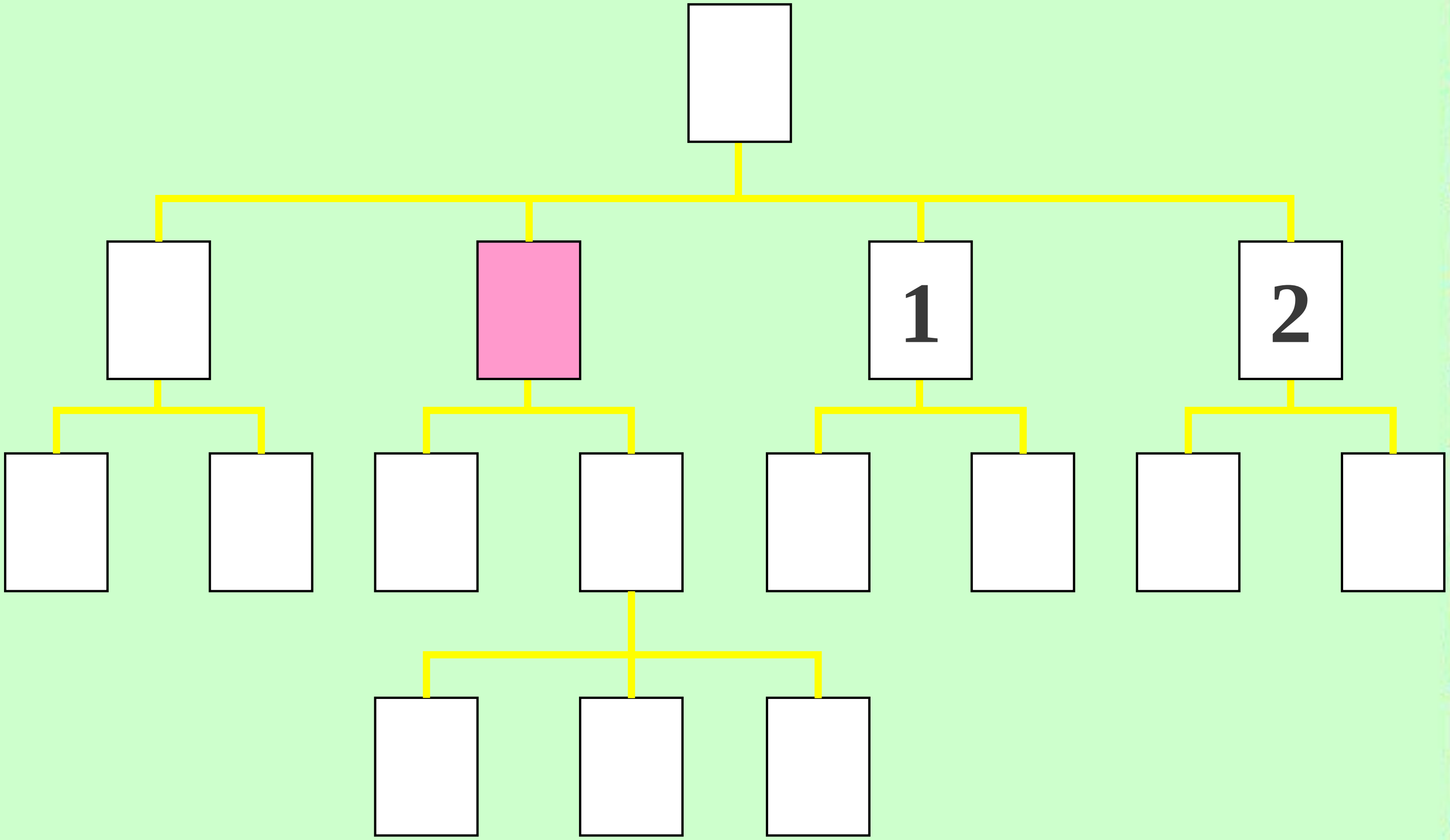
Descendant



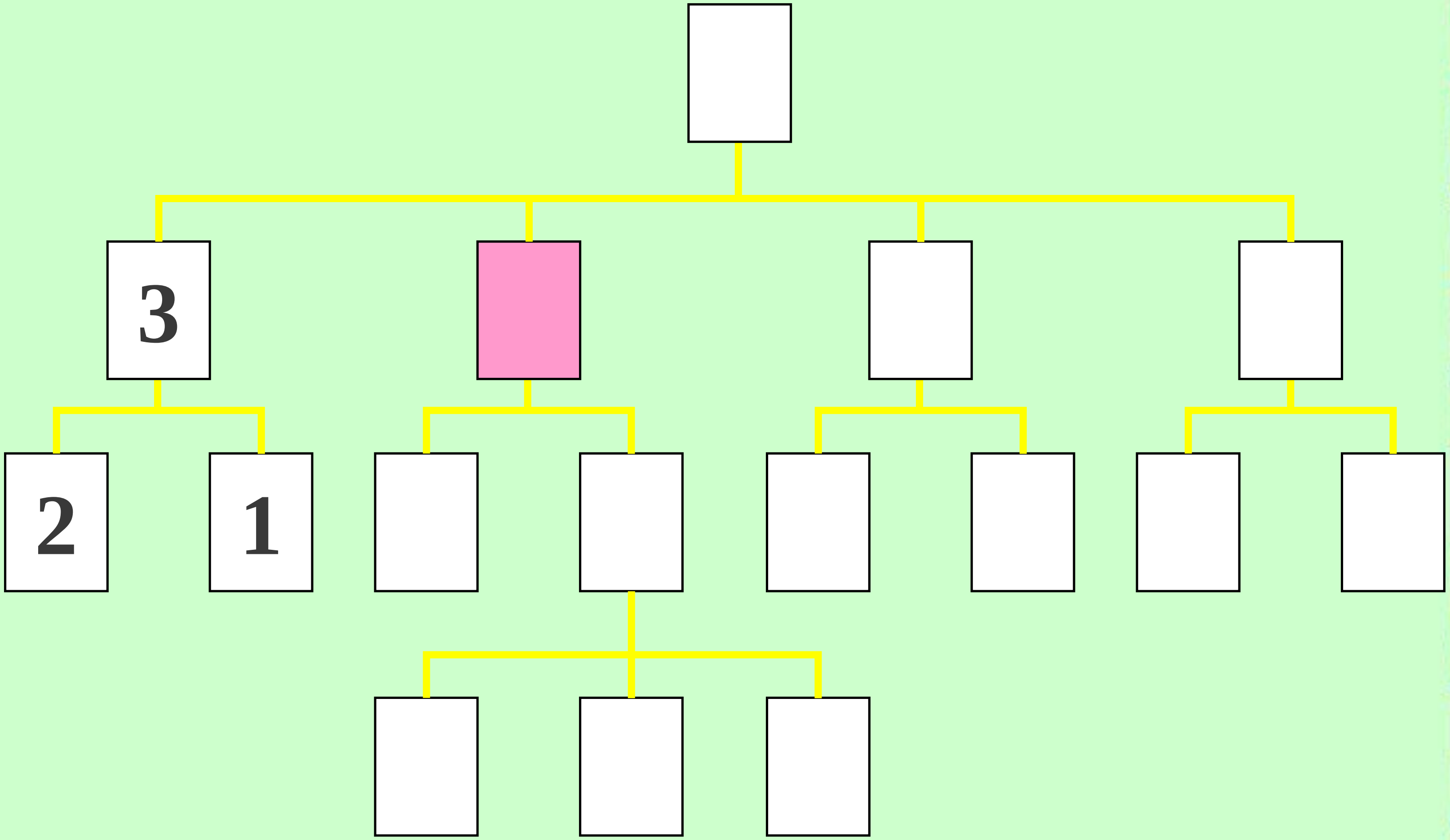
Following



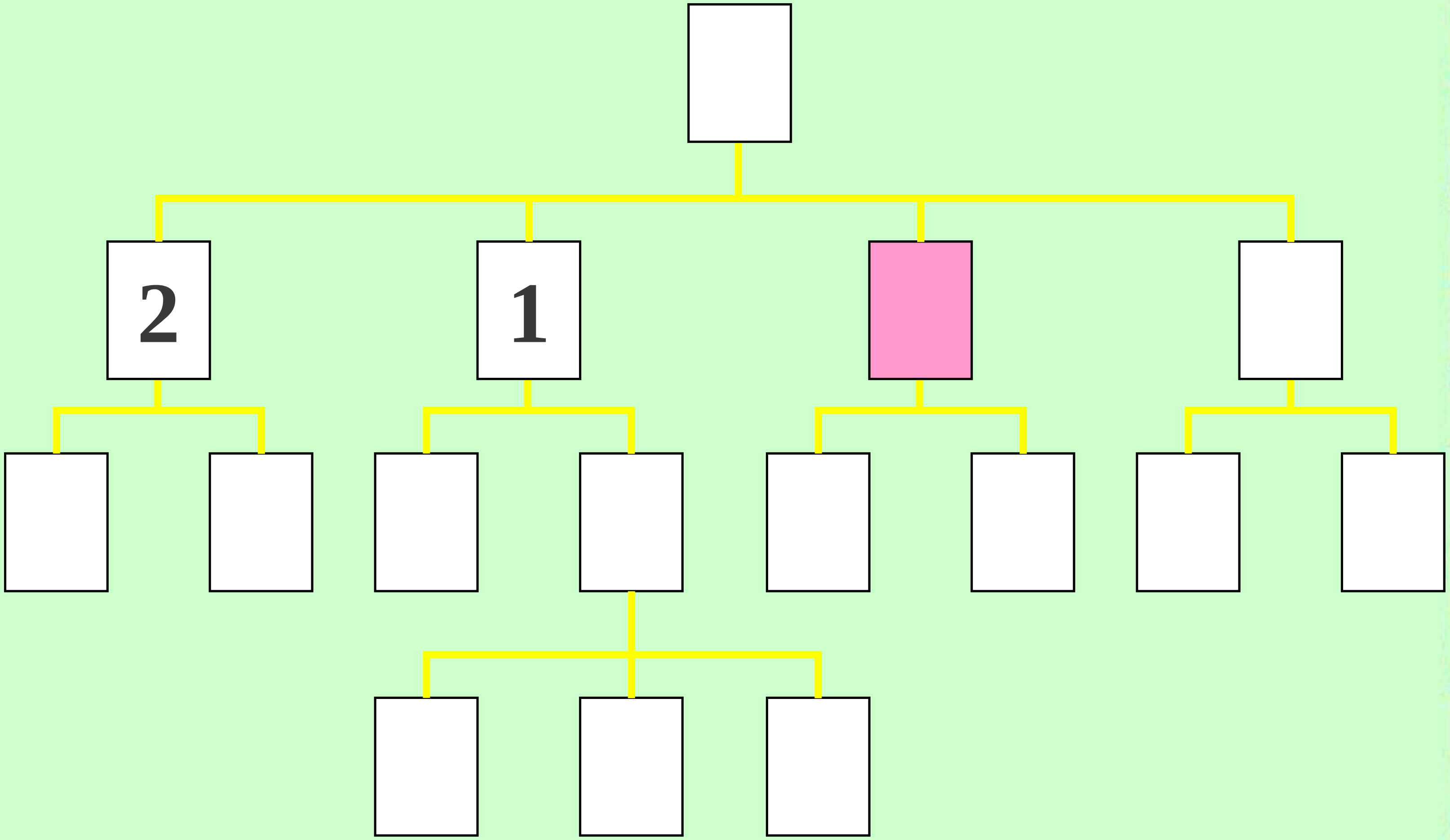
Following-Sibling



Preceding



Preceding-Sibling



Location path

- Node sets are returned by location path(XPath expressions)
- location path is made of location steps
- A location step contains an axis and a node test separated by double colon:

axis::node-test

- A location step may be:
- ➢ ***abbreviated form*** - axis is assumed
- ➢ ***unabbreviated form*** - axis is specified

ISO 9001

XPath Expressions

- An axis (specifies which direction to travel from the context node to look for the next nodes)
- A node test (specifies which node to include along the axis)
- Zero or more predicates (use expressions to further refine the set of nodes selected by the location step)
- The syntax for a location step is:

  axisname::nodetest[predicate]

XPath Expressions (cont.)

- **XPath Boolean Expressions :**

  != ,< ,<=, = , > ,>=

- **XPath node set :**

  last( ), position( ), count( ),boolean( )
  local-name( ),namespace-uri( ),name( )

- **XPath operators :**

  +,-,*,div,mod

EX:

/child::book/child::price[.=9.9]→/book/price[.=9.9]

<xsl:template match="/month[position( ) = 1]">

XPath Expressions (cont.)

- Xpath functions that work with integers:

floor( )➔ returns the largest integer smaller than number you pass to it,floor(4.6)=4

round( )➔round the number you pass to nearest integer ex: round(4.6)=5

sum( )➔sums the numbers you pass to it

ISO 9001

Example

```
<historicaldates>
   <description>some notable dates</description>
   <entry country="Egypt">
    <date day="6" month="oct" year="1973"/>
    <description> Egypt claims back Sinai</description>
   </entry>

   <entry country="Kuwait">
    <date year="1990"/>
    <description> Iraq invaded Kuwait</description>
   </entry>
    .
    .
    .
</historicaldates>
```

XPath Expressions Examples

- **/child::historicaldates/child::description**

  returns the description element of root element (ONLY)

- **/child::historicaldates /child:: entry[position( )=2]/child::date**

  (/historicaldates/entry[position( )=2]/date)

  returns the date element for the 2nd entry

- **/descendant::description (//description)**

  returns all description elements

XPath Expressions Examples (cont.)

- **/ descendant:: entry/child::date/attribute::year**

  (//entry/date/@year)

returns the year attribute value from root node

- **"*"** ➔ *matches all the element children of*

context node

- **../@units** ➔*matches units attribute of the parent of context node.*
- **"*@"** ➔*matches all attributes of the context node.*
- **/states/state[4]/name[3]**➔*matches the third <name> element of the fourth <state>element of the <states>element*

XSL example

```
• <xsl:stylesheet version='1.0`
xmlns:xsl=
'http://www.w3.org/1999/XSL/Transform>
<xsl:template match="/">
<xsl:value-of select="/xsltutorial"/>
</xsl:template>

</xsl:stylesheet>
```

XSL Template

- Templates are used to control the output of an XML document.
- **<xsl:template match="XPath exp.">**

  . . .

  </xsl:template>
- **<xsl:template**

match="/Greeting/msg[@length&lt;=200]/title[.='hi']">

</xsl:template>

XSL Template (cont.)

Templates contain transformation rules (either XSL directives):

- Inserting literal text:

`<xsl:text>Insert your Text Here</xsl:text>`

- Inserting the string representation of an Xpath exp.:

`<xsl:value-of select="XPath Exp."/>`

- Conditional processing (1):

    `<xsl:if test="boolean exp.">`

XSL Templates

- Conditional processing (2):

  <xsl:choose> together with

  <xsl:when test=""> and <xsl:otherwise>
- Loops:

  <xsl:for-each select="node list">
- Sorting (self closing,used within the loops context):

  <xsl:sort select=""/>
- Variables (immutable):

<xsl:variable name="var1" select="value"/>

To send the value of a variable to the o/p tree:

<xsl:copy-of select="$var1">

<xsl:value-of>

ISO 9001

XSL:apply-templates

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

- **The simplest way to process a source tree is thus to write a template rule for each kind of node that can be encountered, and for that template rule to produce any output required, as well as to call <xsl:apply-templates> to process the children of that node.**

ISO 9001

Demos