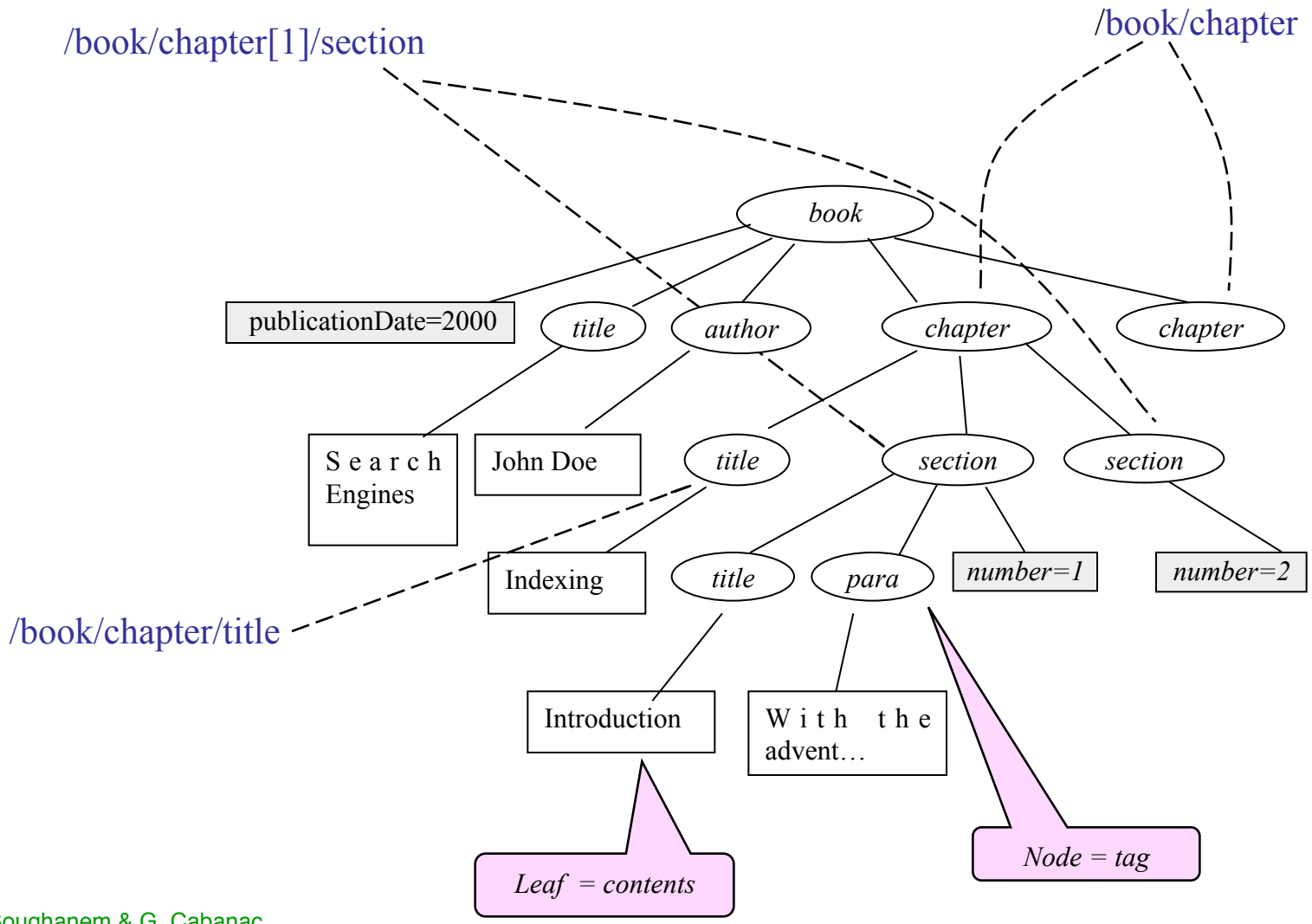
Chapter 4 :
XPath

Introduction

- Document XML = set of tags with a hierarchical organisation (tree-like structure)

- XPath
  - Language that allows the selection of elements in any XML document thanks to path expressions
  - Operates on the tree structure of documents
  - Purpose: XPath references the nodes (elements, attributes, comments, and so on) of an XML document *via* the path from the root to the element

XPath: Examples



Purpose of XPath

- An XPath expression references one or several nodes in an XML document thanks to path expressions

- XPath is used by/for
  - XSLT to select transformation rules
  - XML Schema to handle keys and references
  - XLink to link documents with XML fragments
  - XQuery to query document collections

XPath Expressions

- An XPath expression
  - Specifies a path in the hierarchical structure of the document:
    - From a starting point (a node)
    - … to a set of target nodes
  - Is interpreted as:
    - A set of nodes
    - Or a value that can be numerical, Boolean, or alphanumerical

- An XPath is a sequence of navigation steps concatenated and separated by a slash (/)
  - [/]step1/step2/.../stepN

- Two variants:
  - Absolute XPaths:
    - They start from the root node of the document: /step1/…/stepN
  - Relative XPaths:
    - They start from the current node (a.k.a. context): step1/…/stepN

Steps of XPath Navigation

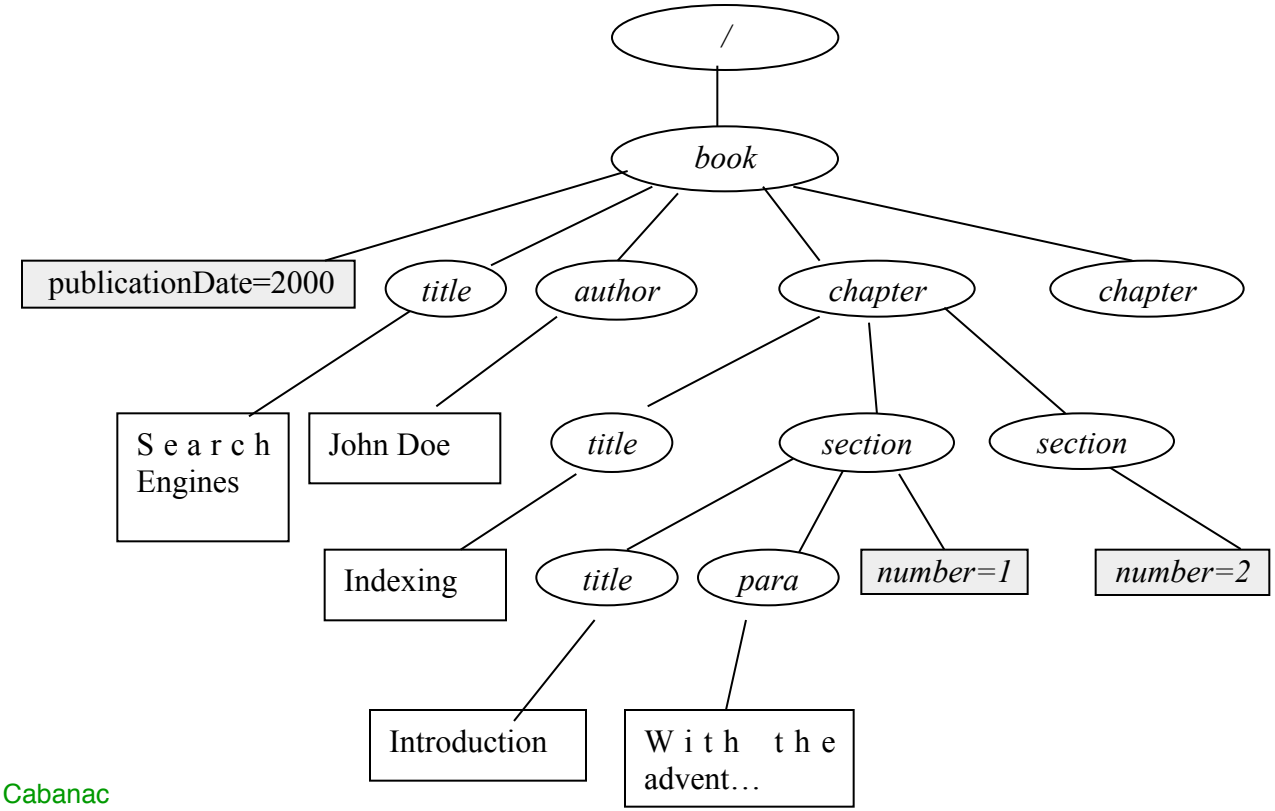
- Each step = an elementary path
  - [Axis::]Filter[condition1][condition2]…
- Location axis
  - Direction of the navigation within nodes (default: child)
- Filter
  - Name of the selected node (element or @attribute)
- Condition (predicates)
  - Selected nodes must comply with these conditions

- Example: /child::book/child::chapter

Step 1 *Step 2*

XPath: Examples

- Selecting a chapter
  - /child::book/child::chapter/ child::section
  - /book/chapter/section
- Text in chapter 1, section 2
  - /descendant::chapter[position() = 1] /child::section[position() = 2]/ child::text()
  - //chapter[1]/section[2]/text()



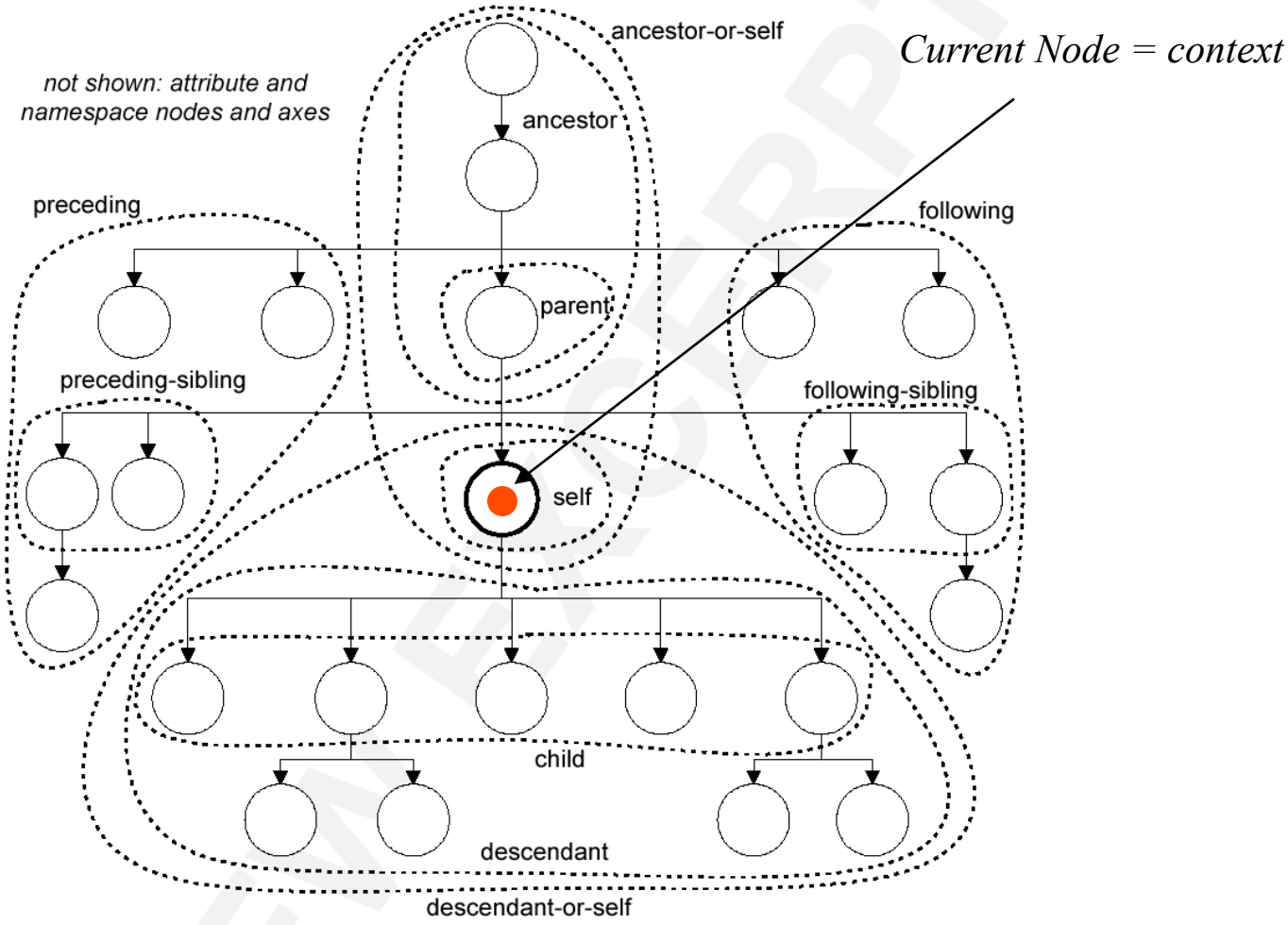
XPath Axes

- An axis defines a node-set relative to the current node (called context):
  - **child**: selects all the children of the current node
  - **descendant**: selects all the descendants (children, grandchildren, etc.) of the current node
  - **ancestor**: selects all the ancestors (parent, grandparent, etc.) of the current node
  - **following-sibling**: selects all the siblings after the current node (or an empty set if the current node is not an element)
  - **preceding-sibling**: selects all the siblings before the current node (or an empty set if the current node is not an element)

XPath Axes (Continued)

- **following**: selects everything in the document after the closing tag of the current node
- **preceding**: selects all the nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
- **attribute**: selects all the attributes of the current node
- **self**: selects the current node
- **descendant-or-self**: selects all the descendants (children, grandchildren, etc.) of the current node and the current node itself
- **ancestor-or-self**: Selects all the ancestors (parent, grandparent, etc.) of the current node and the current node itself

Wrap-Up: XPath Axes



Filters

- A filter is a test that selects some nodes in the axis according to the filter

- Syntax of filters:
  - ***n*** where *n* is a node name: selects the nodes of the axis with name *n*
  - *: selects all the nodes of the axis
  - **node()**: selects all the nodes of the axis
  - **text()**: selects the textual nodes of the axis
  - **comment()**: selects the comment nodes of the axis
  - **processing-instruction(*n*)**: selects the processing instruction nodes of the axis, provided that their name is *n*

A Few Examples

- **child::para** selects the *para* child nodes of the current node
- **child::*** selects all the child nodes of the current node
- **child::text()** select all the textual nodes that are children of the current node
- **child::node()** select all the child nodes of the current node, whatever their type (element or other)
- **attribute::name** selects the *name* attribute of the current node
- **attribute::*** selects all the attributes of the current node
- **descendant::para** selects all the descendant nodes (named *para*) of the current node
- **ancestor::para** selects all the ancestor nodes (named *para)* of the current node
- **ancestor-or-self::section** selects all the ancestor nodes named *section* and the current node itself if it is a *section*
- **descendant-or-self::para** : selects all the descendant nodes named *section* and the current node itself if it is a *section*
- **self::para** selects the current node if it is named *para*, or nothing otherwise
- **child::chapitre/descendant::para** selects the *para* descendants of the *chapter* children associated with the current node
- **child::*/child::para** selects all the *para* grand-children of the current node

Abbreviated Syntax for XPath Expressions

- The following abbreviations are provided to increase the readability of XPath expressions:
  - **child** can be omitted (default axis)
    - Example: child::section/child::para ≡ section/para
  - **attribute** can be replaced by **@**
    - Example: child::para[attribute::type = 'warning'] ≡ para[@type='warning']
  - **// ≡ /descendant-or-self::node()/**
    - Example: //para ≡ /descendant-or-self::node()/child::para
    - //para[1] ≠ /descendant::para[1]
  - **. ≡ self::node()**
  - **..** ≡ parent::node()

Conditions (1)

- Condition:
  - Boolean expression composed of one or many tests combined with the usual connectors: and, or, not
- Test:
  - Any XPath expression whose result is converted into a Boolean type
  - e.g., the result of a comparison, a function call

A Few Examples (1)

- **child::para[position()=1]** selects the first *para* child of the current node
- **child::para[position()=last()]** selects the last *para* child of the current node
- **child::para[position()=last()-1]** selects the last but one *para* child of the current node
- **child::para[position()>1]** selects every *para* children of the current node except from the first one
- **following-sibling::chapter[position()=1]** selects the next *chapter* appearing after the current node
- **preceding-sibling::chapitre[position()=1]** selects the previous *chapter* appearing before the current node
- **/descendant::figure[position()=42]** the 42nd *figure* element in the document
- **/child::doc/child::chapter[position()=5]/child::section[position()=2]** selects the 2nd *section* of the 5th *chapter* in the *doc* element of the document
- **child::para[attribute::type='warning']** selects every *para* child of the current node, provided they have a *type* attribute whose value is 'warning'

A Few Examples (2)

- **child::para[attribute::type='warning'][position()=5]** selects the 5th *para* child of the current node having a *type* attribute with the 'warning' value
- **child::para[position()=5][attribute::type='warning']** selects the 5th *para* child of the current node if it has a *type* attribute with the 'warning' value
- **child::chapitre[child::title='Introduction']** selects the *chapter* children c of the current node, provided that c has a *title* child node whose value is 'Introduction'
- **child::chapitre[child::title]** selects the *chapter* children of the current node having at least one child node called *title*
- **child::*[self::chapitre or self::appendix]** selects the *chapter* children or *appendix* children of the current node
- **child::*[self::chapitre or self::appendix][position()=last()]** selects the last children of the current node with name *chapter* or *appendix*
- **/A/B/descendant::text()[position()=1]** selects the first textual node that is a descendant of /A/B

Conditions (2)

- There are 4 ways to express conditions:
  - *axis*::*filter*[*number*]
  - *axis*::*filter*[*XPATH*_expression]
  - *axis*::*filter*[*Boolean_expression*]
  - Compound conditions

axis::*filter*[*number*]

- Selects nodes according to their position
  - Example:
  - /book/chapter/section[2]
  - //section[position()=last()]

    … which is evaluated the same way as
    - //section[last()]

axis::filter[XPATH_expression]

- Selects nodes for which the XPATH_expression results in a non empty node-set
  - Examples
  - Chapters with text
    - /book/chapter[text()]
  - Sections with a num attribute
    - //chapter/section[@num]

axis::filter[Boolean_expression]

- Conditions may apply to two operands tested with the boolean operators =, !=, <, <=, >, >=

- *value1 operator value2*
  condition1 and *condition2*
  condition1 or *condition2*
  not(*condition*)
  true()
  false()
  boolean(*object*)
  - Chapters featuring a section with an attribute num = 1
    - chapter[section/@num = '1']

  - //chapter/section[@num != '1' and text()]
  - //chapter/section[@num > 1 and title/text()='Introduction']
  - //chapter[following::section[@num=1]]

XPATH: Functions & operations (1)

- Boolean expressions may also use the following functions:
  - Values of the following types:
    - Boolean, string, real number, node-set
  - Numerical operators:
    - +, -, *, div, mod
  - last():
    - Returns true if the current node is the last node among its siblings
  - position():
    - Returns the position of the current node.
      Example: item[(position() mod 2) = 0]
  - id(*name*) :
    - Returns the node identified by *name*

XPATH: Functions & operations (2)

- Other functions:
  - local-name(*nodes*) namespace-uri(*nodes*) name(*nodes*)
  - string(*object*)
  - concat(*string1*, ..., *stringN*)
  - string-length(*string*)
  - normalize-space(*string*)
  - translate(*s1*, *s2*, *s3*)
  - substring-before(*s1*, *s2*) returns the string *res* such that *s1 = res + s2 + miscellaneous*
  - substring-after(*s1*, *s2*) return the string *res* such that *s1 = miscellaneous + s2 + res*
  - substring(s, *start*)
  - substring(*s*, *start*, *length*)

XPATH: Functions & operations (3)

- Other functions:
  - starts-with(*s1*, *s2*) is true if *s1* starts with *s2*
  - contains(*s1*, *s2*) is true if *s1* contains *s2*
  - number(*object*) converts *o* to a number
  - sum(*ns*) returns the sum of all nodes in the node-set *ns*.
    Each node is first converted to a number value before summing
  - count(*ns*) returns the number of nodes in the node-set *ns*
  - floor(*n*) returns the largest integer that is not greater than *n*
  - ceiling(*n*) returns the smallest integer that is not less than *n*
  - round(*number*) returns an integer closest in value to *n*

Functions: Recap (1)

- **For nodes**
  - *number* **last()**
  - *number* **position()**
  - *number* **count(***nodes****)**
  - *nodes** **id(***object***)**
    - `id("foo")/child::para[position()=5]`

- **For strings**
  - *string* **string**(*object*?)
  - *string* **concat(***string*, *string, string****)**
  - string **starts-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*?)

Functions: Recap (2)

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

- **For numbers**
  - *number* **number**(*object*?)
  - *number* **sum(***noeuds****)**
  - number **floor(***number***)**
  - *number* **ceiling(***number***)**
  - *number* **round(***number***)**

Compound Conditions

- *axis*::*filter*[*condition1*][*condition2*]...
  Selects the nodes identified by *filter* when all the conditions are satisfied.
  Beware: these two expressions are different
  - chapitre[2][para]
    selects the chapter nodes appearing at position 2, provided they have a para child node.
  - chapitre[text()][2]
    selects the second chapter node that has a textual child node.

The End

- Exercises
  - select
    - Titles of all sections
    - Chapters that have sections
    - Sections with attributes
    - Contents of section titles
    - Sections entitled “introduction”
    - Titles that contain the word “introduction”