

Web Data Models

XPath: Syntax, Semantics

Silviu Maniu



Comprendre le monde,
construire l'avenir

université
PARIS-SACLAY

XPath

- once we have an XML document, we want to **query it and retrieve data**
- **XPath**: query language to **select a sequence of nodes from an XML**
- **W3C standard** — also used in other standards (XQuery, XSLT, XPointer, XLink, ...); **supported by every modern browser**

XPath

- XPath is a **navigational language** — specifies how the XML documents should be traversed
- **Main issue:** big volume of nodes can be extracted via XPath, so **efficient processing is still an ongoing challenge**

Lecture Outline

- XPath *axes*
- XPath *tests*
- XPath *nodes*

XPath: Quick Example

Q1 = `/bib/book/title`

```
<bib>
```

```
  <book>
```

```
    <author>Abiteboul</author>
```

```
    <author>Hull</author>
```

```
    <author>Vianu</author>
```

```
    <title>Foundations of Databases</title>
```

```
    <year>1995</year>
```

```
  </book>
```

```
  <book>
```

```
    <author>Ullmann</author>
```

```
    <title>Principles of Database and Knowledge Base Systems</title>
```

```
    <year>1998</year>
```

```
  </book>
```

```
</bib>
```

XPath: Quick Example

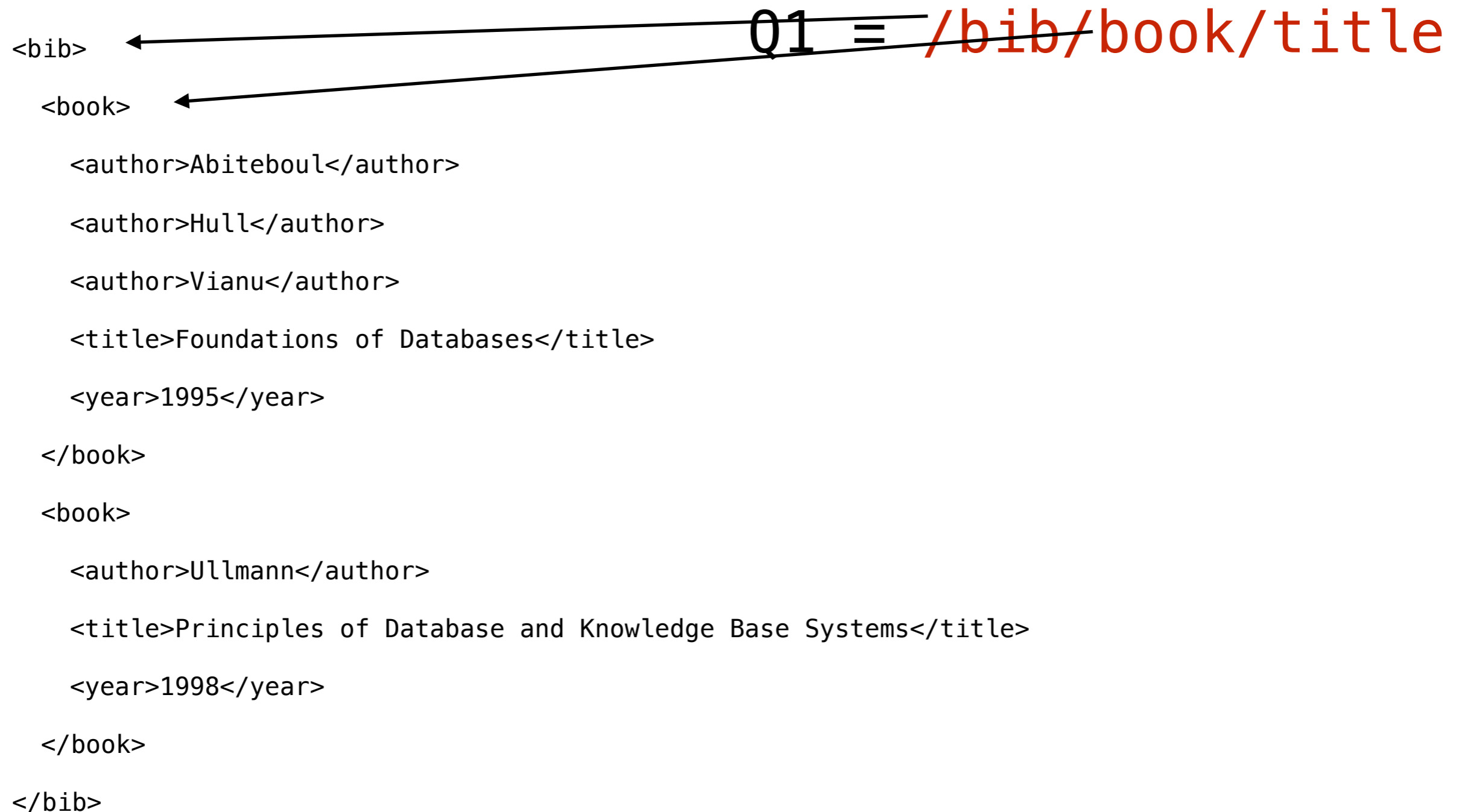
`<bib>` ← **Q1 = `/bib/book/title`**

```
<book>
  <author>Abiteboul</author>
  <author>Hull</author>
  <author>Vianu</author>
  <title>Foundations of Databases</title>
  <year>1995</year>
</book>
<book>
  <author>Ullmann</author>
  <title>Principles of Database and Knowledge Base Systems</title>
  <year>1998</year>
</book>
</bib>
```

XPath: Quick Example

```
<bib>
  <book>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <title>Foundations of Databases</title>
    <year>1995</year>
  </book>
  <book>
    <author>Ullmann</author>
    <title>Principles of Database and Knowledge Base Systems</title>
    <year>1998</year>
  </book>
</bib>
```

Q1 = ~~/bib/book/title~~



XPath: Quick Example

`<bib>` ← **Q1 = /bib/book/title**

`<book>` ←

`<author>Abiteboul</author>`

`<author>Hull</author>`

`<author>Vianu</author>`

`<title>Foundations of Databases</title>` ←

`<year>1995</year>`

`</book>`

`<book>`

`<author>Ullmann</author>`

`<title>Principles of Database and Knowledge Base Systems</title>`

`<year>1998</year>`

`</book>`

`</bib>`

The diagram illustrates the XPath expression `Q1 = /bib/book/title` in red text. Three black arrows originate from the expression: one points to the opening `<bib>` tag, another points to the opening `<book>` tag of the first book, and the third points to the `<title>` element of that same book. The XML document contains two book entries within a `<bib>` container. The first book has authors Abiteboul, Hull, and Vianu, and the title 'Foundations of Databases'. The second book has author Ullmann and the title 'Principles of Database and Knowledge Base Systems'.

XPath: Quick Example

`<bib>`

`<book>`

`<author>Abiteboul</author>`

`<author>Hull</author>`

`<author>Vianu</author>`

`<title>Foundations of Databases</title>`

`<year>1995</year>`

`</book>`

`<book>`

`<author>Ullmann</author>`

`<title>Principles of Database and Knowledge Base Systems</title>`

`<year>1998</year>`

`</book>`

`</bib>`

Q1 = `/bib/book/title`

R1 = `<title>Foundations of Databases</title>`

R2 = `<title>Principles of Database and Knowledge Base Systems</title>`

XPath

XPath is composed of a sequence of context nodes and a step:

$$cs_0 / \text{step}$$

- after the context sequence, take a step in a given direction
- can also have multi-steps:

$$cs_0 / \text{step}_1 / \text{step}_2 / \dots$$
$$cs_1 / \text{step}_2 / \dots$$

XPath: Quick Example

Q1 = `/bib/book/title`

```
<bib>
```

```
  <book>
```

```
    <author>Abiteboul</author>
```

```
    <author>Hull</author>
```

```
    <author>Vianu</author>
```

```
    <title>Foundations of Databases</title>
```

```
    <year>1995</year>
```

```
  </book>
```

```
  <book>
```

```
    <author>Ullmann</author>
```

```
    <title>Principles of Database and Knowledge Base Systems</title>
```

```
    <year>1998</year>
```

```
  </book>
```

```
</bib>
```

XPath: Quick Example

Q1 = **`/bib/book/title`**

```
<bib>
```

```
  <book>
```

```
    <author>Abiteboul</author>
```

```
    <author>Hull</author>
```

```
    <author>Vianu</author>
```

```
    <title>Foundations of Databases</title>
```

```
    <year>1995</year>
```

```
  </book>
```

```
  <book>
```

```
    <author>Ullmann</author>
```

```
    <title>Principles of Database and Knowledge Base Systems</title>
```

```
    <year>1998</year>
```

```
  </book>
```

```
</bib>
```

XPath: Quick Example

Q1 = `/bib/book/title`

```
<bib>
```

```
  <book>
```

```
    <author>Abiteboul</author>
```

```
    <author>Hull</author>
```

```
    <author>Vianu</author>
```

```
    <title>Foundations of Databases</title>
```

```
    <year>1995</year>
```

```
  </book>
```

```
  <book>
```

```
    <author>Ullmann</author>
```

```
    <title>Principles of Database and Knowledge Base Systems</title>
```

```
    <year>1998</year>
```

```
  </book>
```

```
</bib>
```

XPath: Quick Example

Q1 = `/bib/book/title`

```
<bib>
```

```
  <book>
```

```
    <author>Abiteboul</author>
```

```
    <author>Hull</author>
```

```
    <author>Vianu</author>
```

```
    <title>Foundations of Databases</title>
```

```
    <year>1995</year>
```

```
  </book>
```

```
  <book>
```

```
    <author>Ullmann</author>
```

```
    <title>Principles of Database and Knowledge Base Systems</title>
```

```
    <year>1998</year>
```

```
  </book>
```

```
</bib>
```

XPath: Syntax

Each step has the syntax:

$$\text{axis} :: \text{n\textit{test}}[\text{pred}_1] \cdots [\text{pred}_n]$$

- **axis** specifies the **direction of navigation to be followed**,
- **n\textit{test}** specifies the **node test** (used to **navigate only nodes of a certain kind**),
- the optional **pred**, which allows **filters on the sequence of nodes** we navigate to.

XPath: Axes

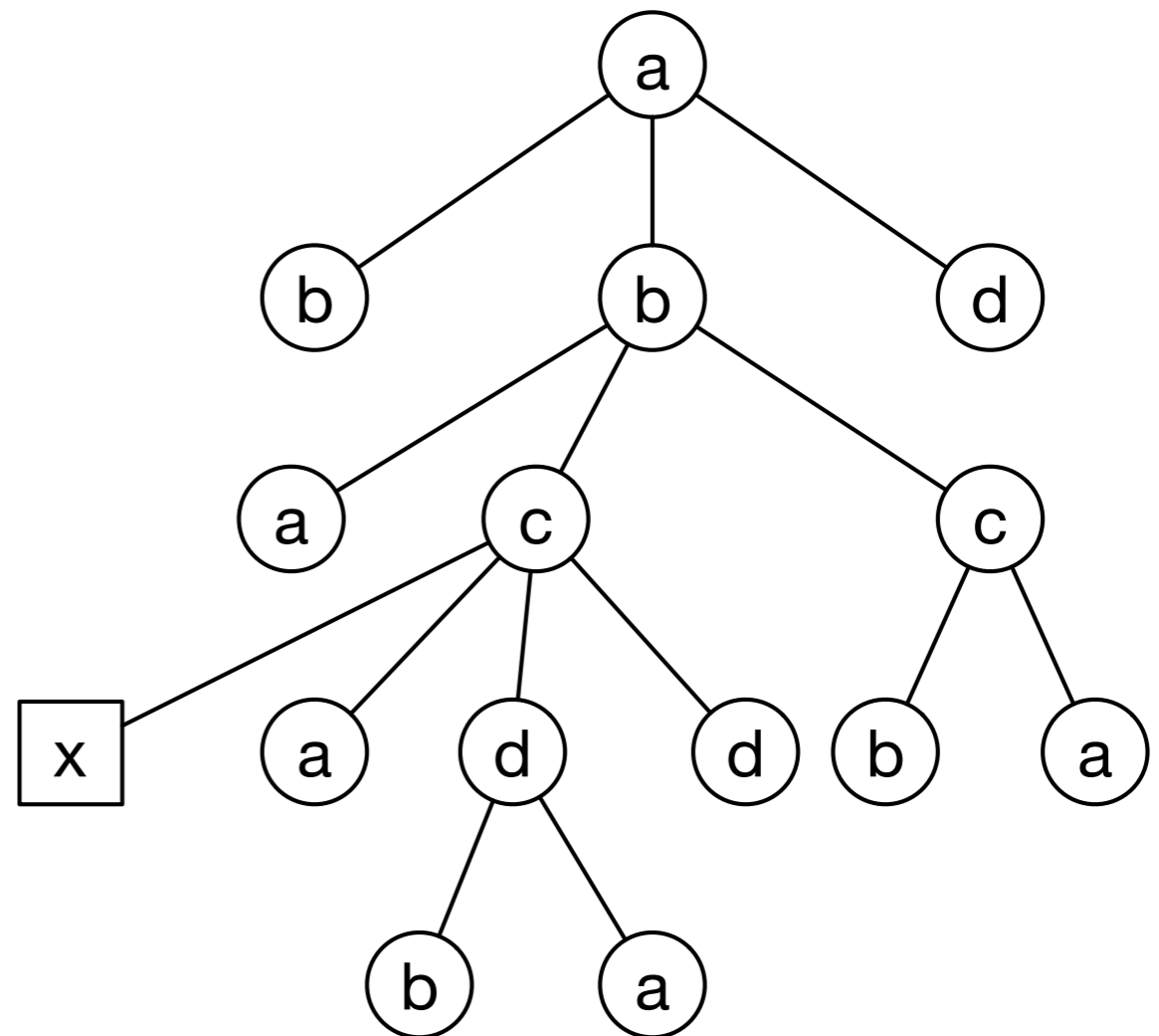
$\text{axis} :: \text{nTest}[\text{pred}_1] \cdots [\text{pred}_n]$

XPath has a family of **12 axes** allowing for flexible navigation within the node hierarchy

- forward axes (in document order)
- backward axes (in reverse document order)

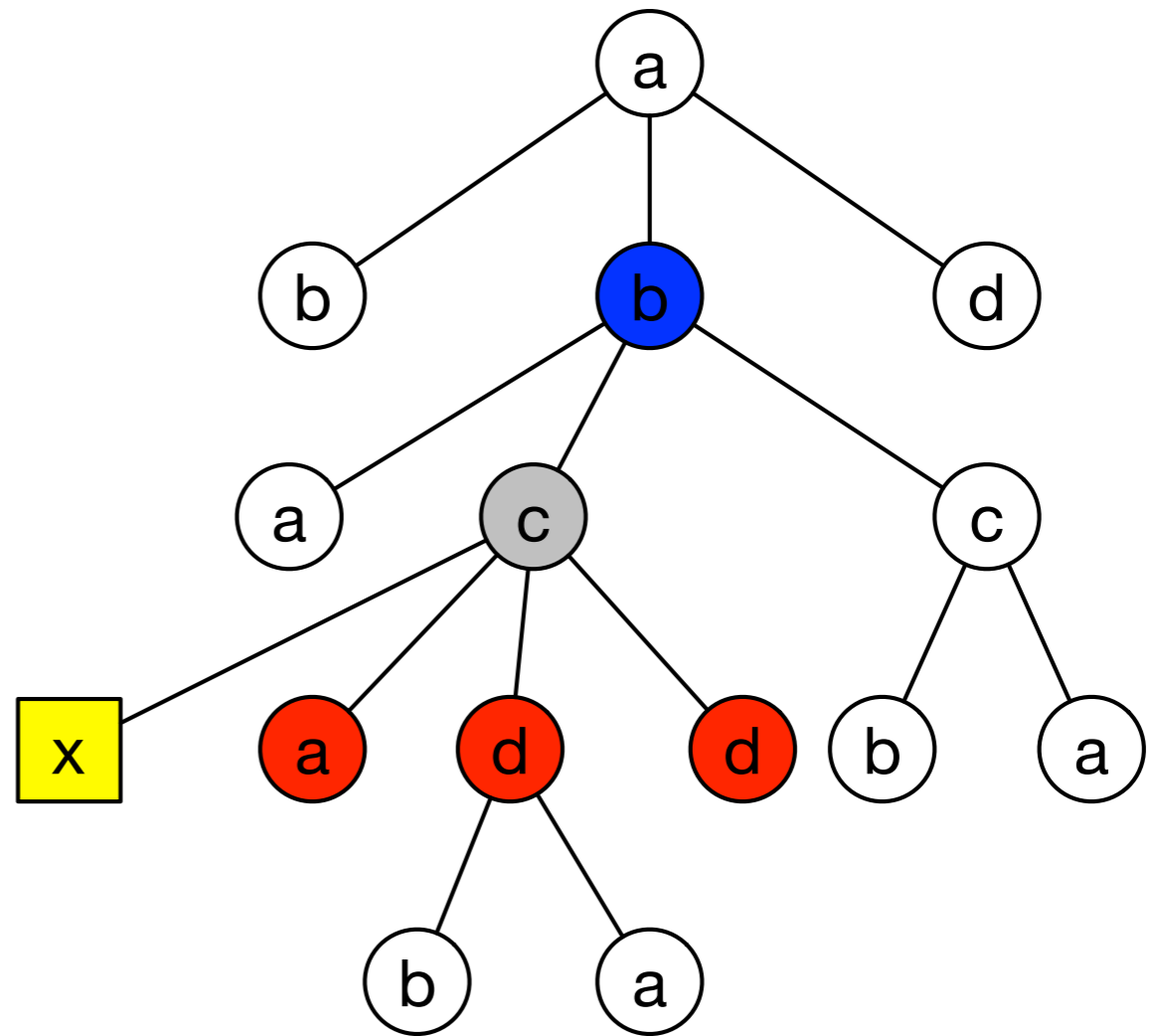
XPath: Axes

```
<a>  
  <b />  
  <b>  
    <a />  
    <c x="1.0">  
      <a>text1</a>  
      <d>  
        <b />  
        <a />  
      </d>  
    </c>  
    <c>  
      <b />  
      <a>text2</a>  
    </c>  
  </b>  
  <d />  
</a>
```



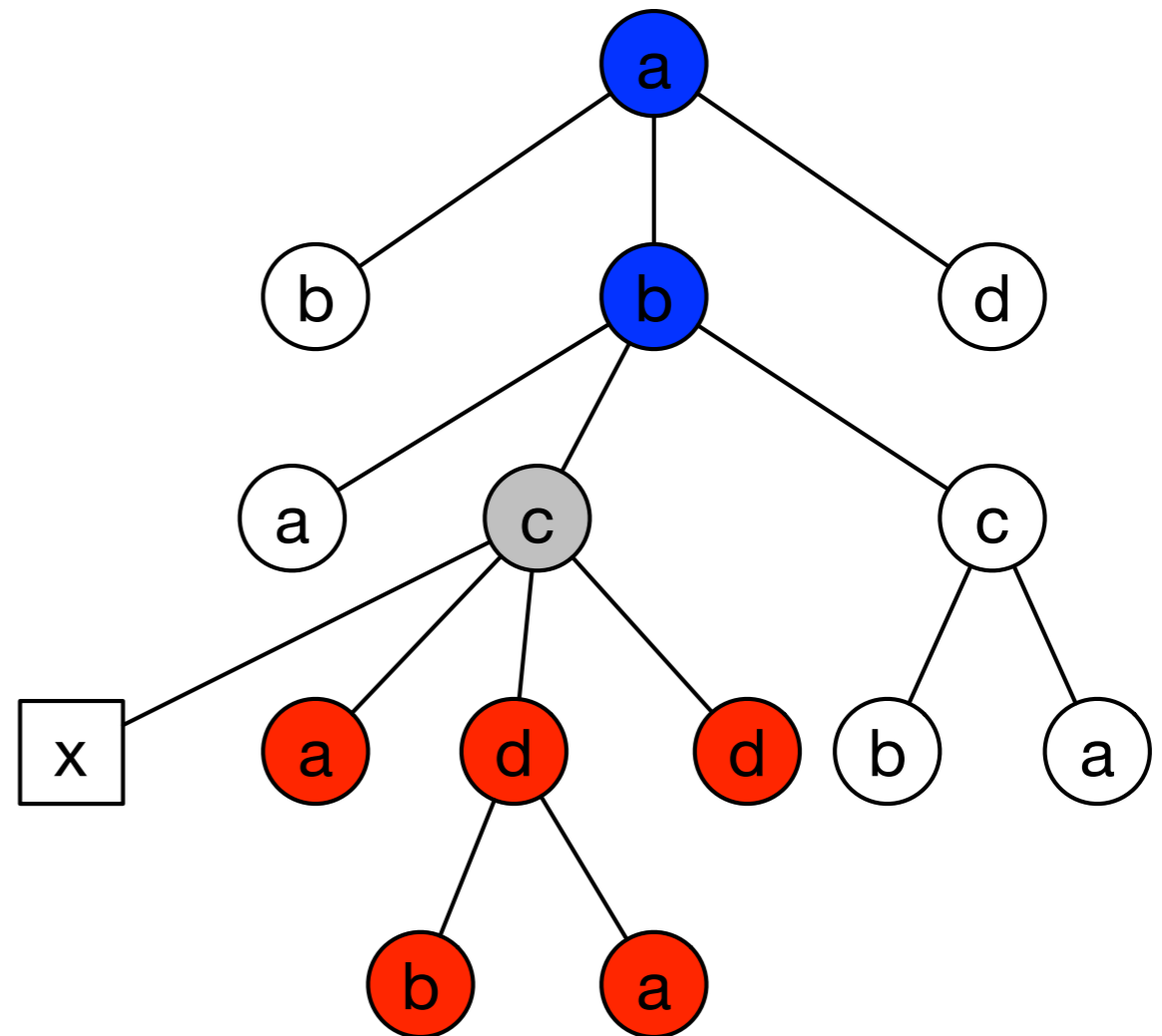
XPath: Axes

- self (fwd)
- child (fwd)
- parent (back)
- attribute (fwd)



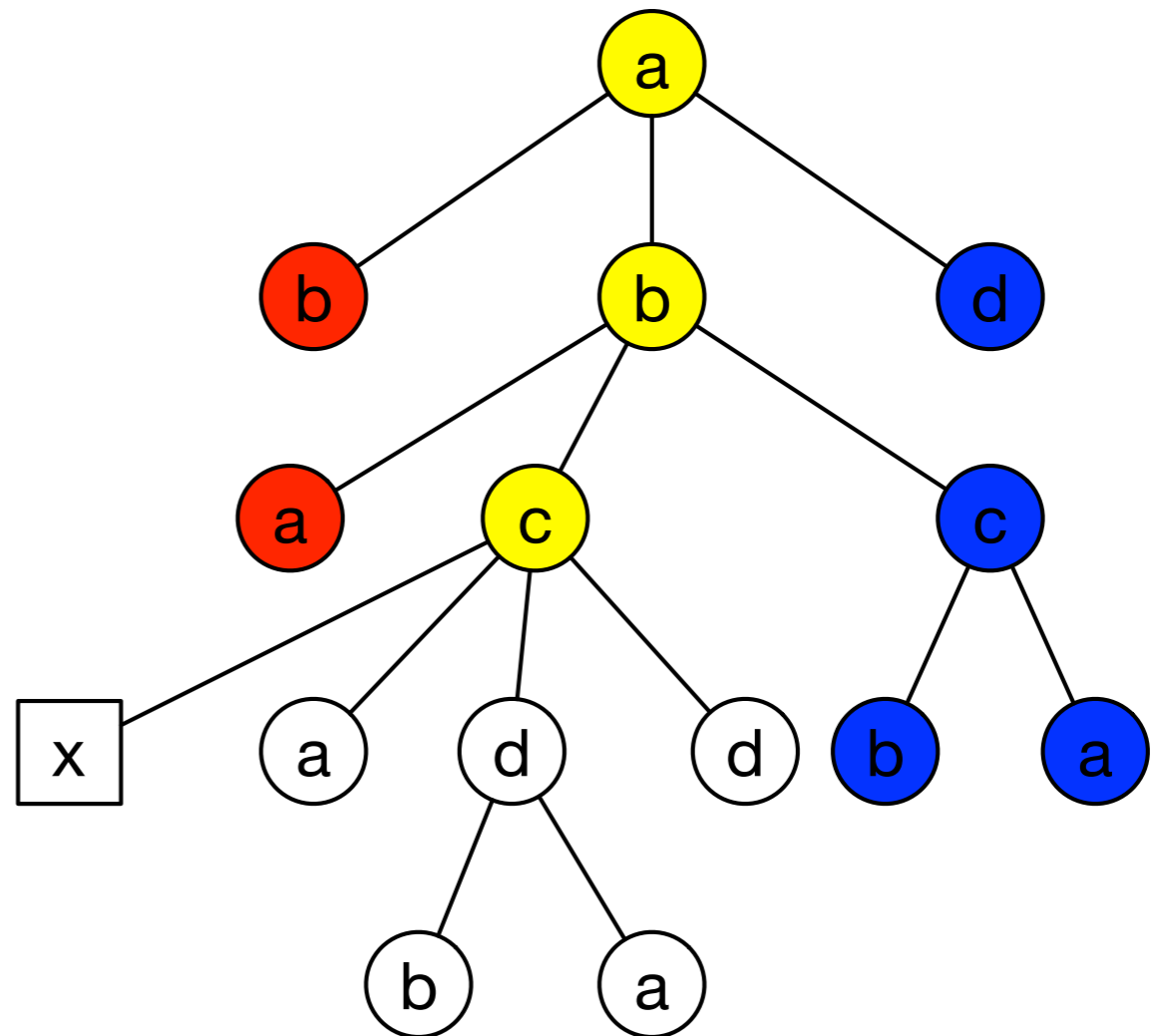
XPath: Axes

- self (fwd)
- descendant (fwd)
- ancestor (back)



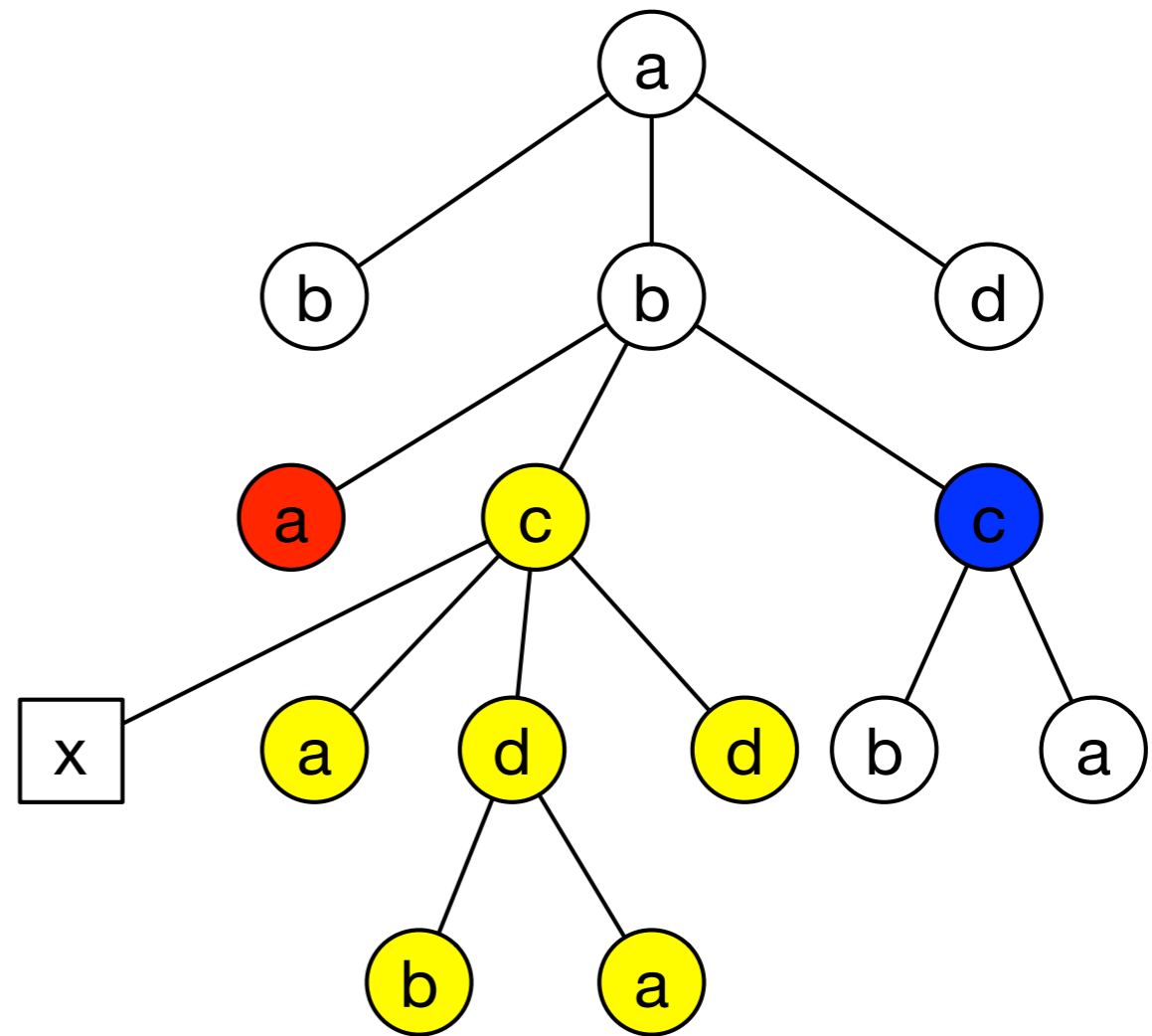
XPath: Axes

- preceding (back)
- following (fwd)
- ancestor-or-self (back)



XPath: Axes

- preceding-sibling (back)
- following-sibling (fwd)
- descendant-or-self (fwd)



XPath: Result Order

The result node sequence of any XPath navigation is returned in document order with no duplicate nodes.

XPath: Node Tests

axis :: ntest[pred₁] ··· [pred_n]

Can also apply a **node test** to filter nodes based on **kind** and name.

Test	Semantics
node()	let any node pass
text()	preserve only text nodes
comment()	preserve only comment nodes
processing-instruction()	preserve processing instructions
processing-instruction(p)	preserve processing instructions of the form <?p ...p>
document-node()	preserve the document root node

XPath: Node Tests

axis :: nptest[pred₁] ··· [pred_n]

Can also apply a **node test** to filter nodes based on kind and **name**.

Test	Semantics
<i>name</i>	preserve element node with tag <i>name</i> only
*	preserve element nodes with arbitrary tag names

XPath: Abbreviations

There are a few [abbreviations](#) in XPath:

`/a` `/child::a`

`//a` `/descendant-or-self::node()/child::a`

`//` `/descendant-or-self::node()`

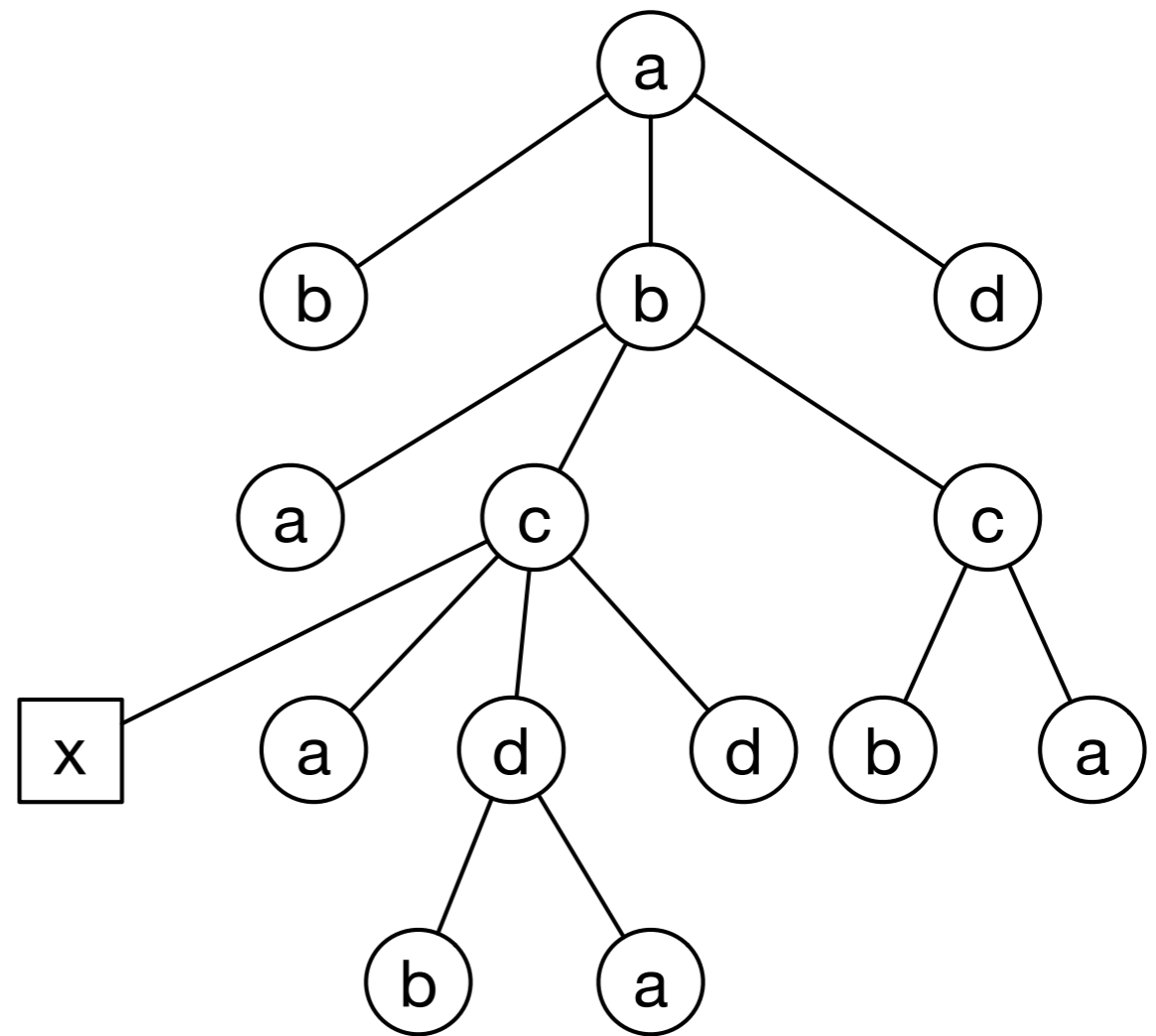
`.` `/self::node()`

`..` `/parent::node()`

`@` `attribute::`

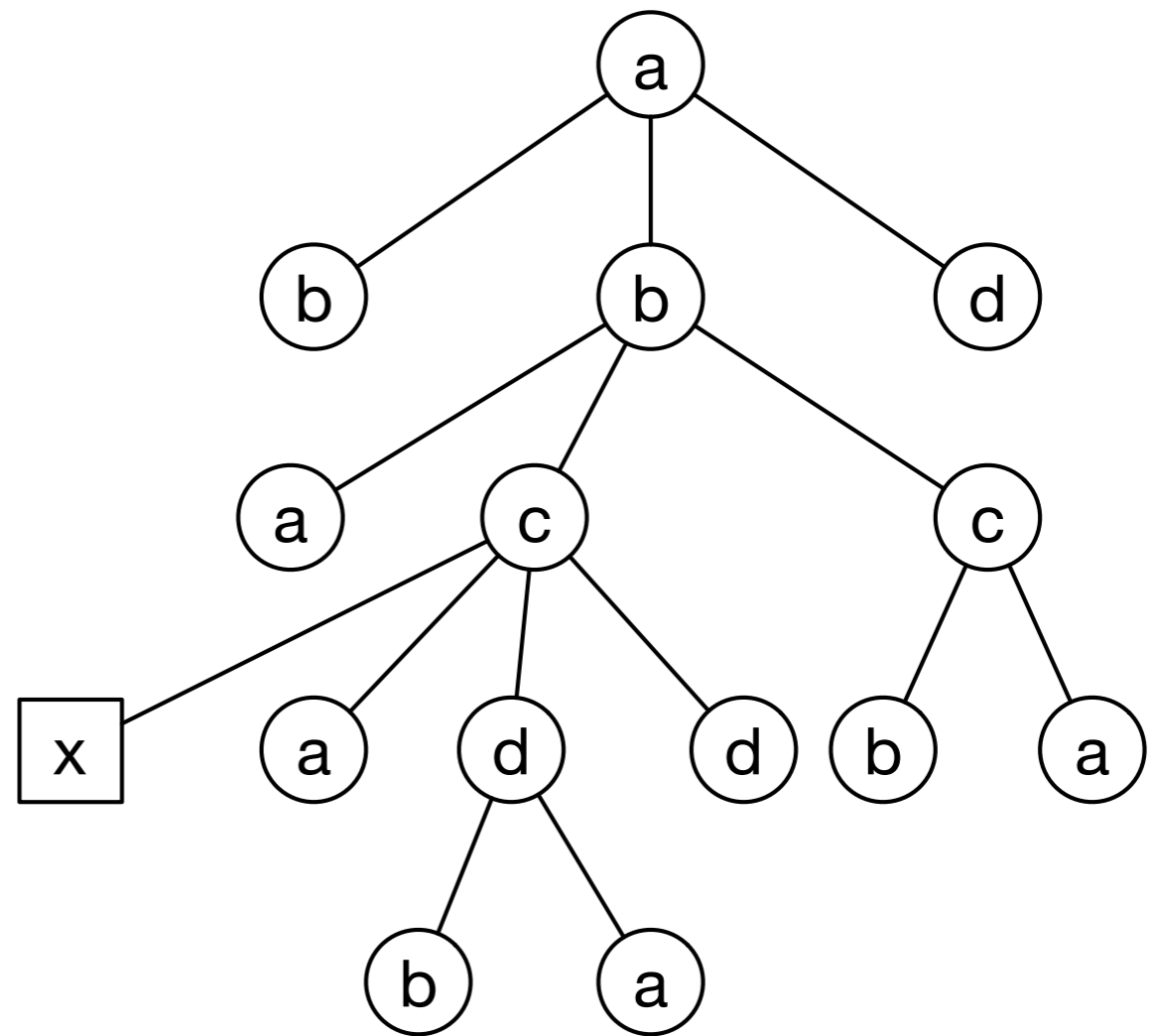
XPath: Examples

Q1 := /a/b/a



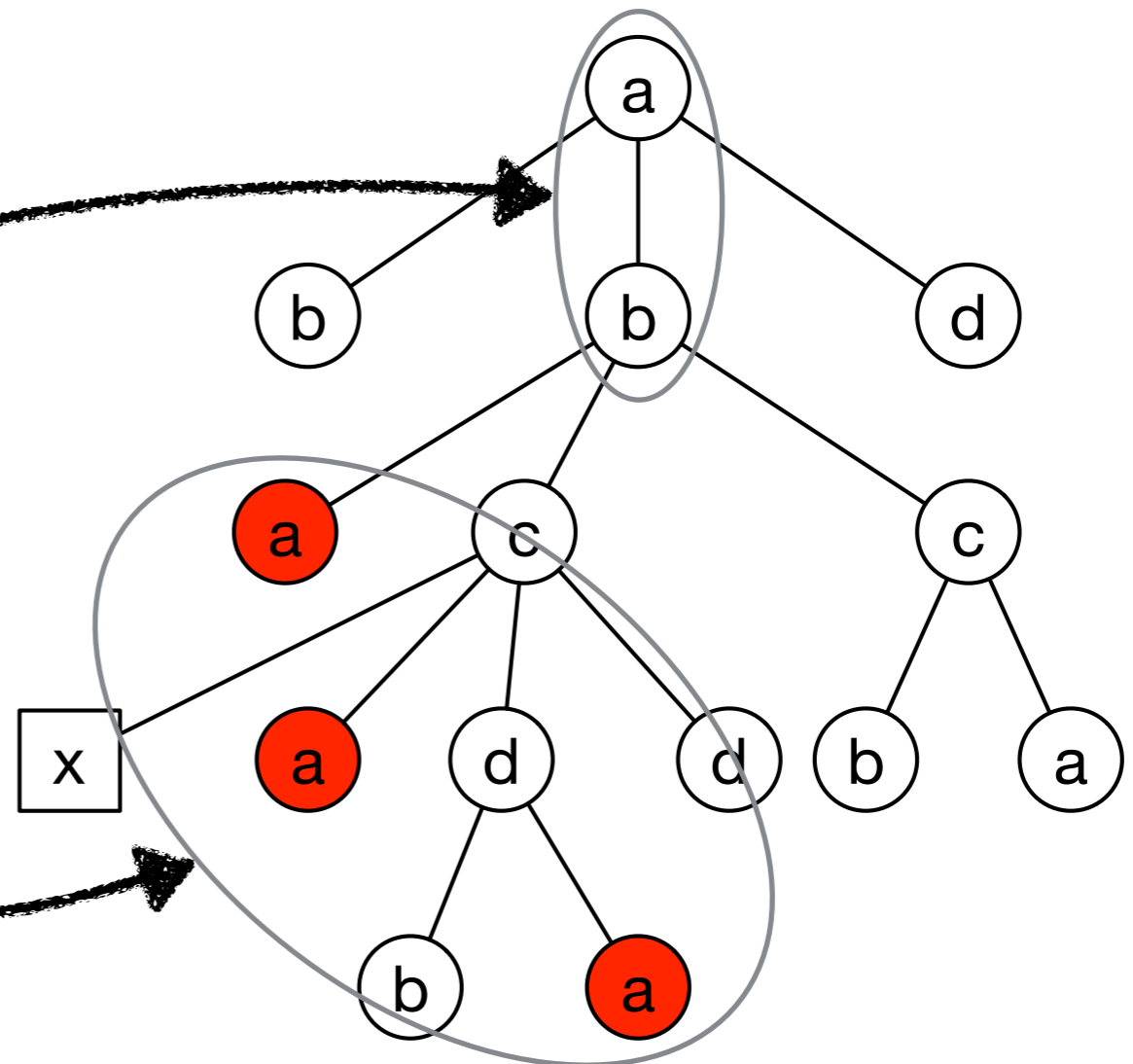
XPath: Examples

Q2 := /a/b//a



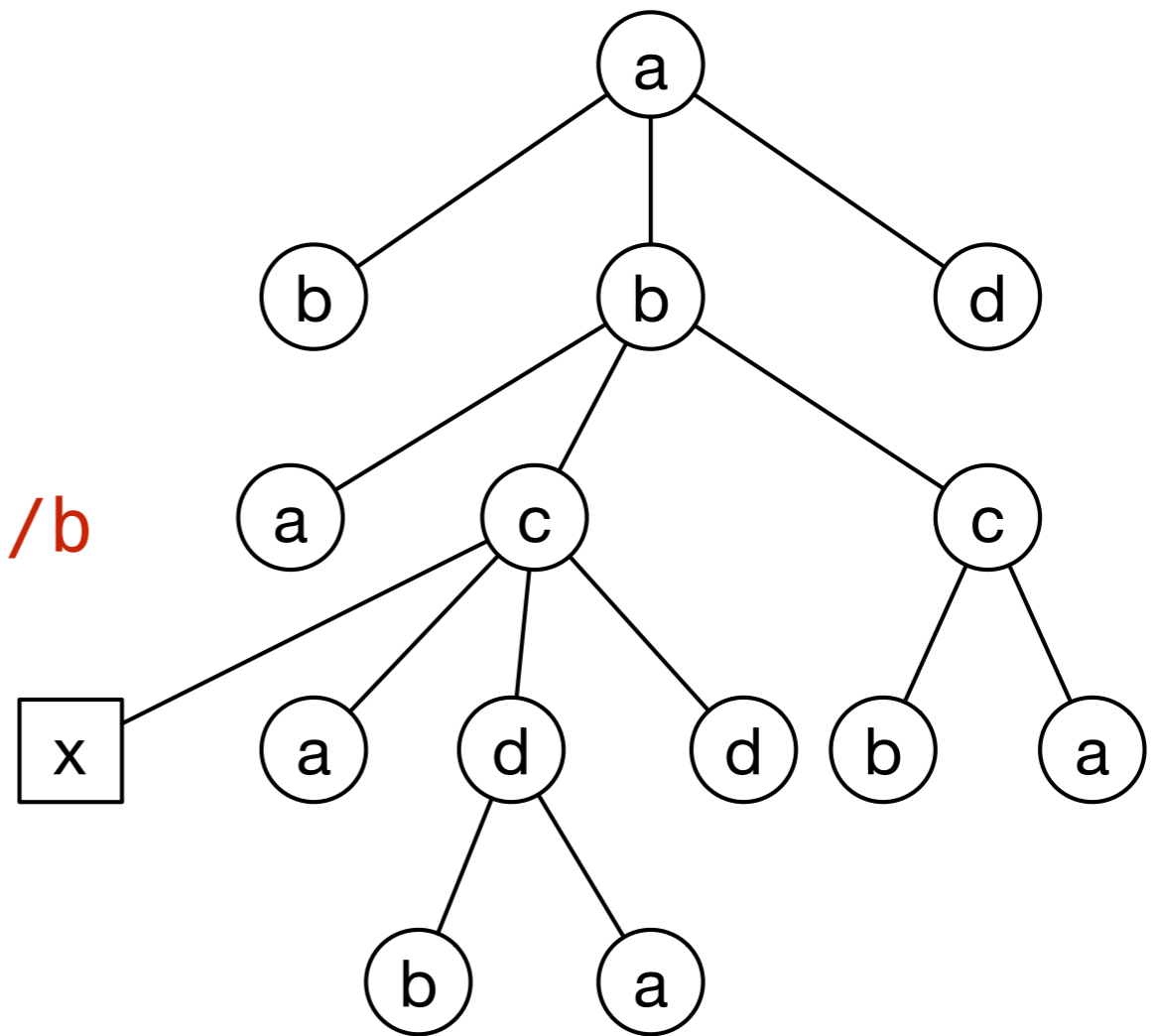
XPath: Examples

Q2 := /a/b//a



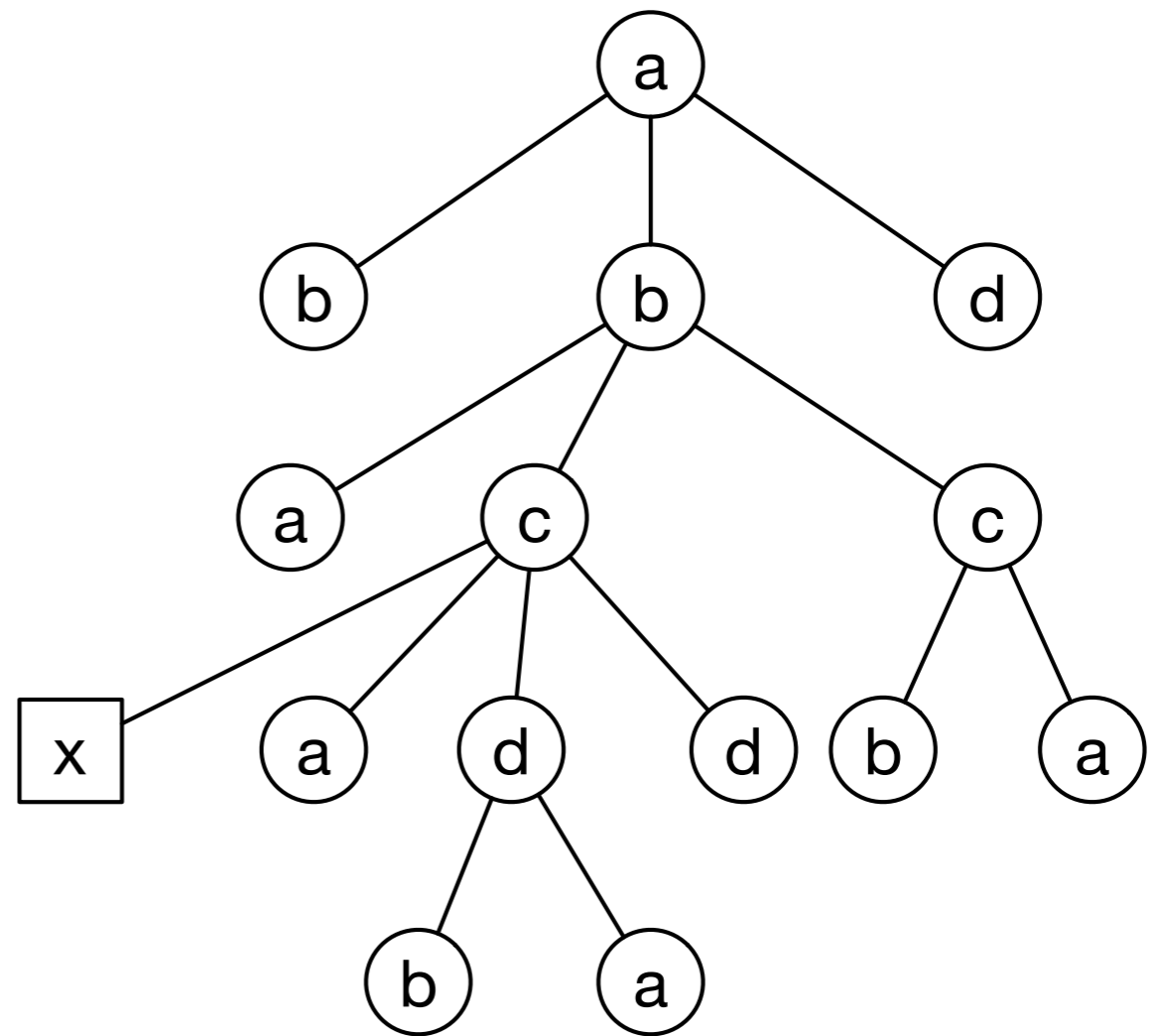
XPath: Examples

Q3 := /a/b/child::node()//b



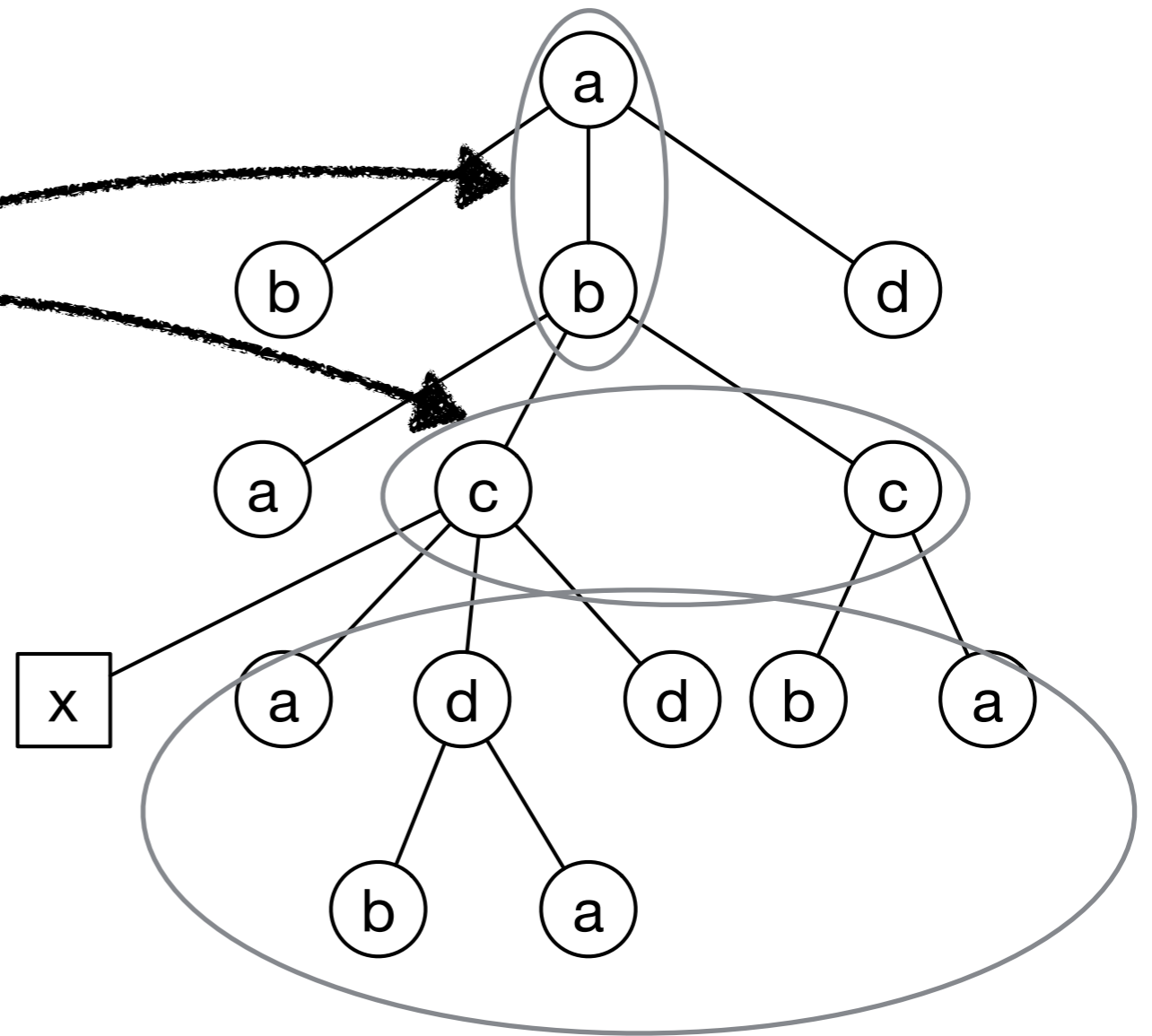
XPath: Examples

Q4 := /a/b/c//c



XPath: Examples

Q4 := /a/b/c//c



XPath: Predicates

axis :: ntest[pred₁] ··· [pred_n]

Predicates are **tested against a node**, and are optional.

- they have **higher precedence than the XPath step**,
- are **evaluated left to right**,
- they may be **any XQuery expression** which evaluates to a value v

XPath: Predicates

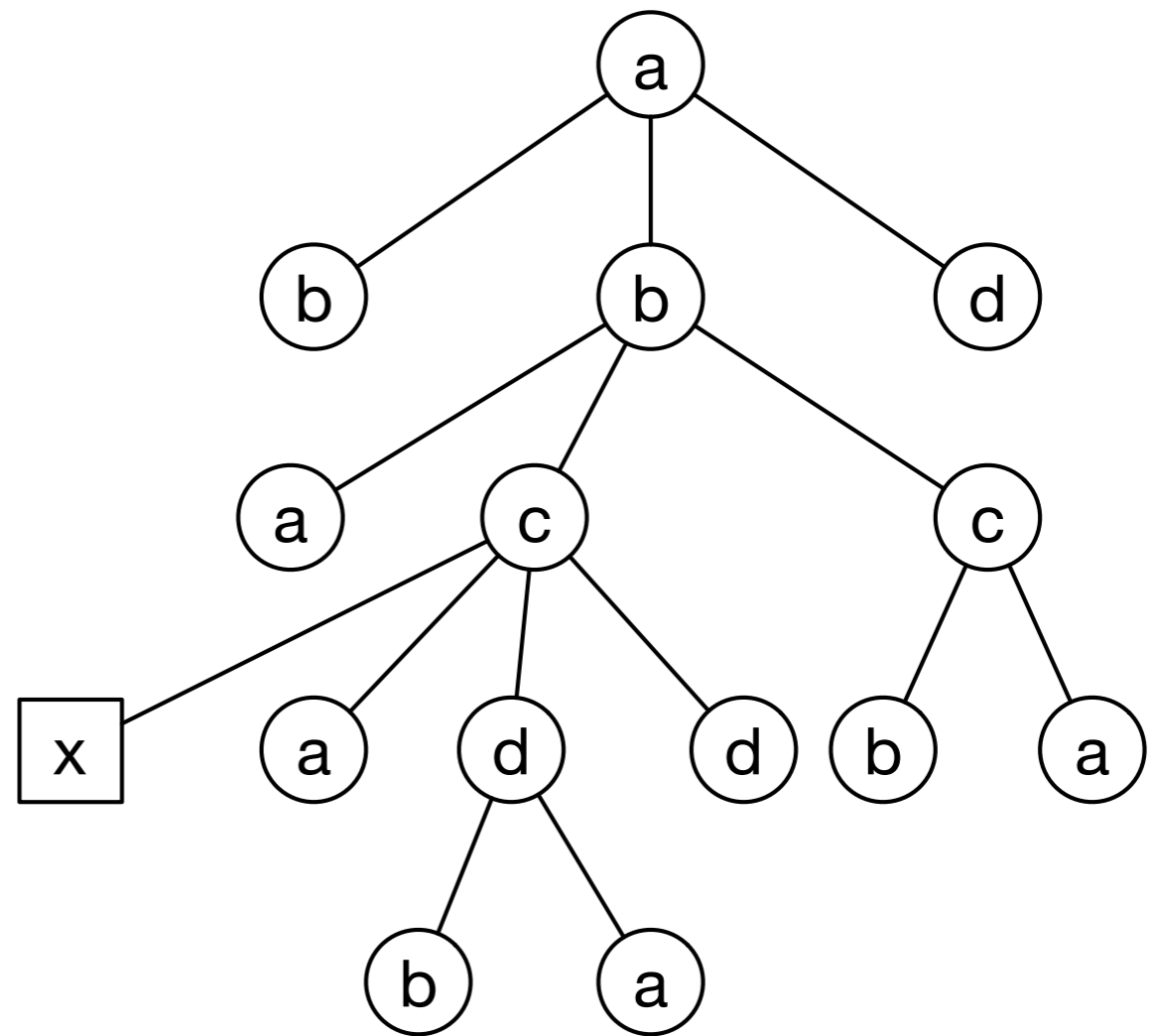
axis :: ntest[pred₁] ··· [pred_n]

XPath calculates an effective boolean value $ebv(v)$, depending on the value v :

v	ebv(v)
()	false()
0, NaN	false()
""	false()
false()	false()
x	true()
(x1, x2, ..., xn)	true()

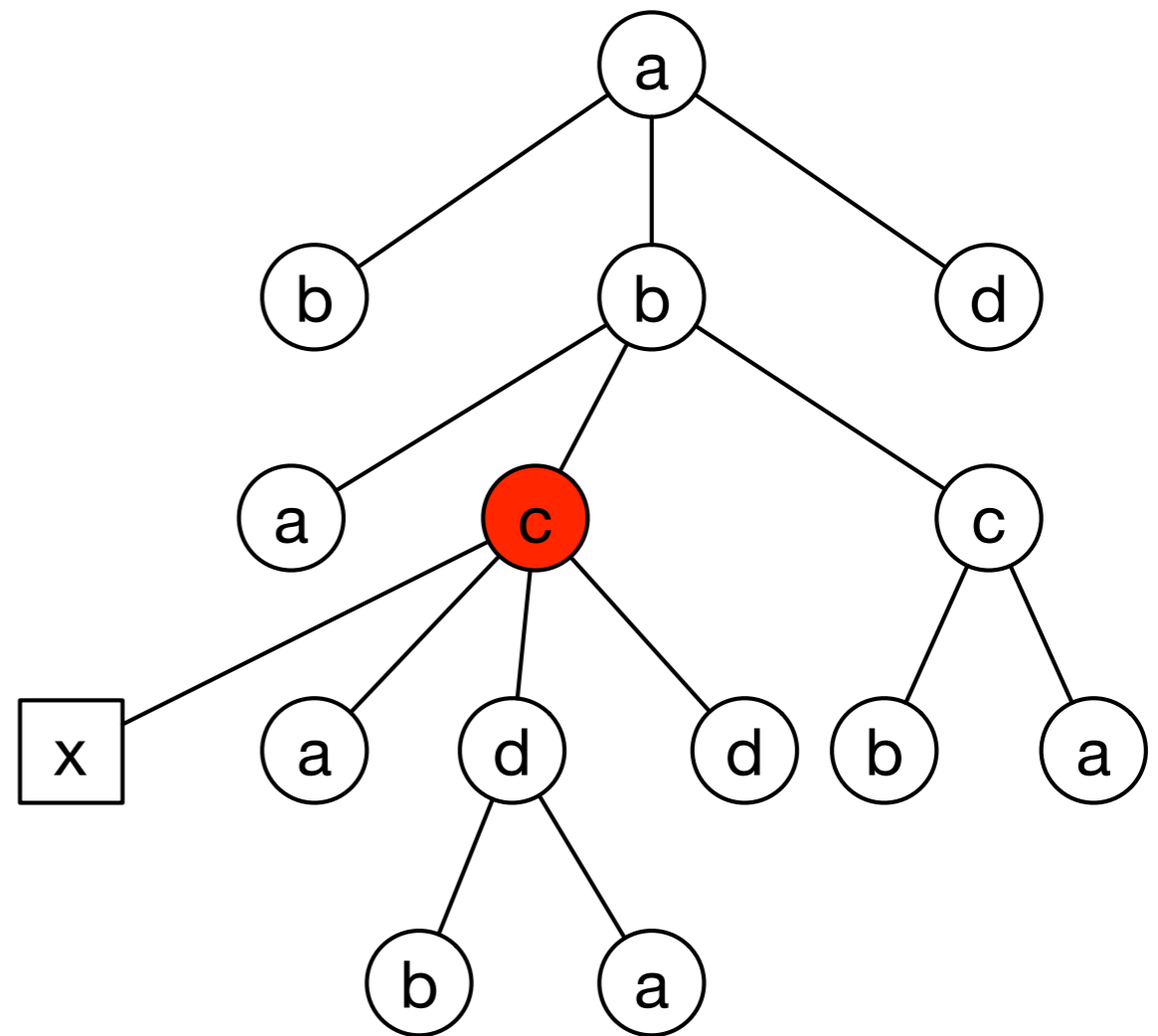
XPath: Examples

Q5 := /a/b/c[./d]



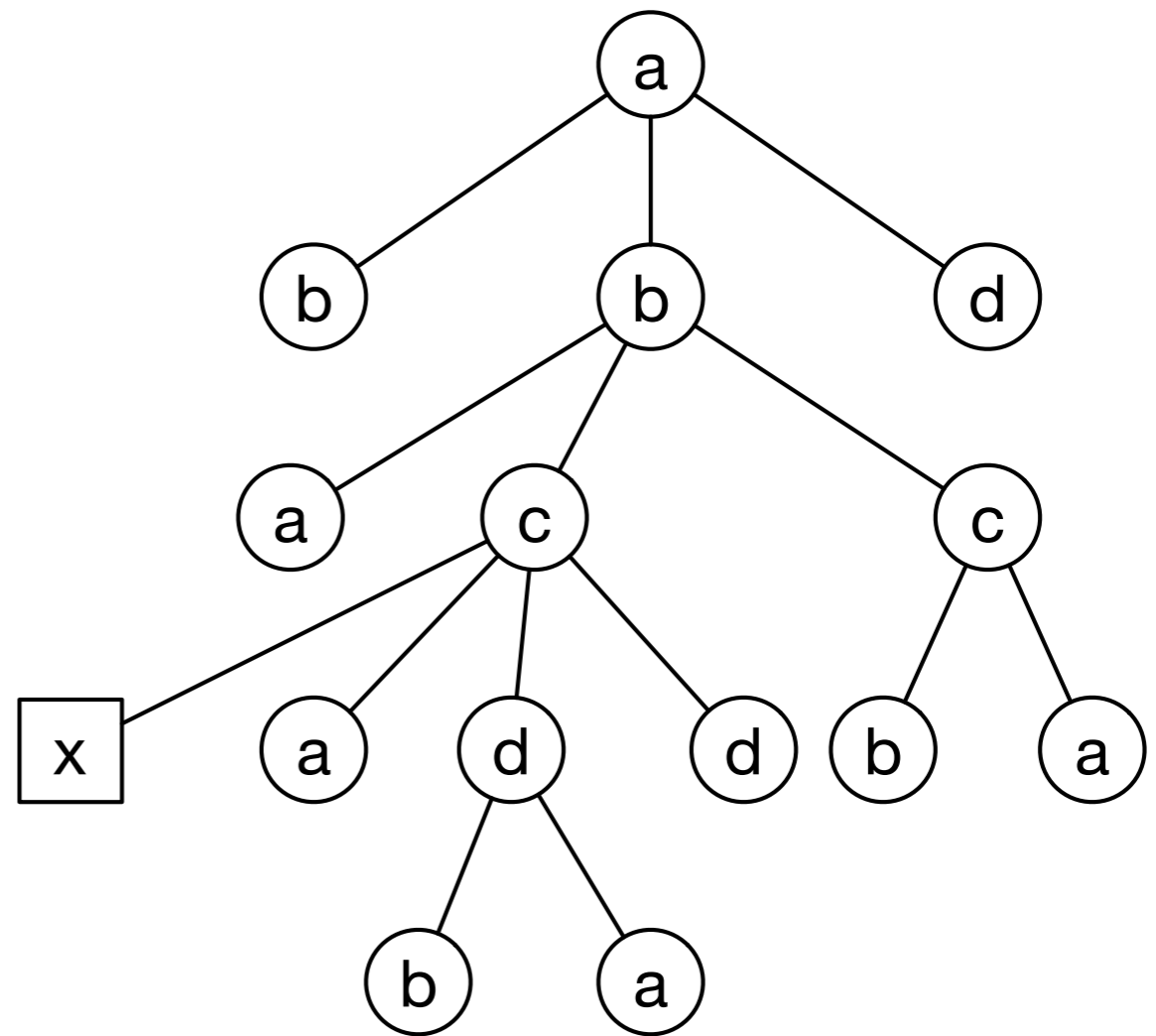
XPath: Examples

Q5 := /a/b/c[./d]



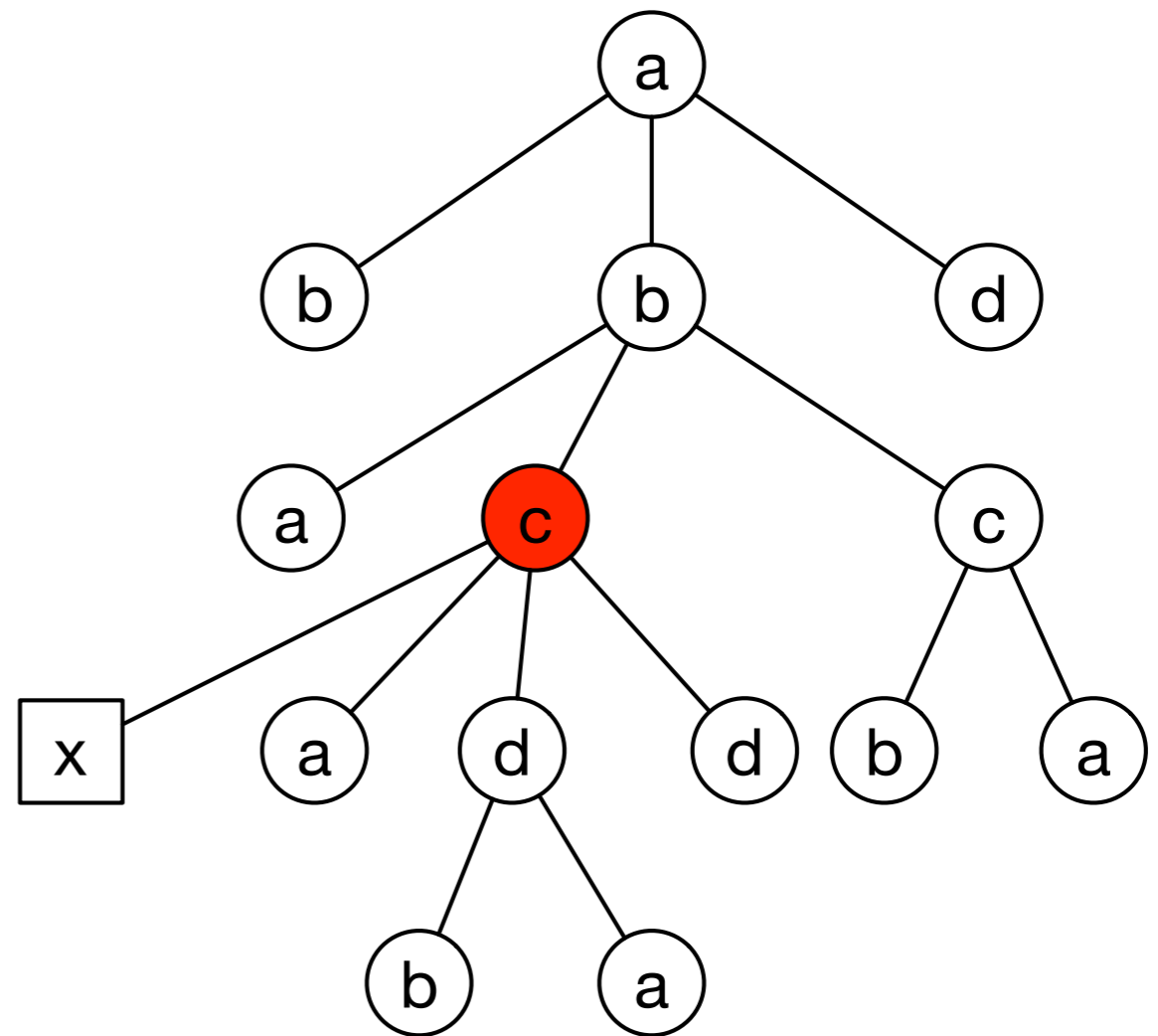
XPath: Examples

Q6 := /a/b/c[a and d]



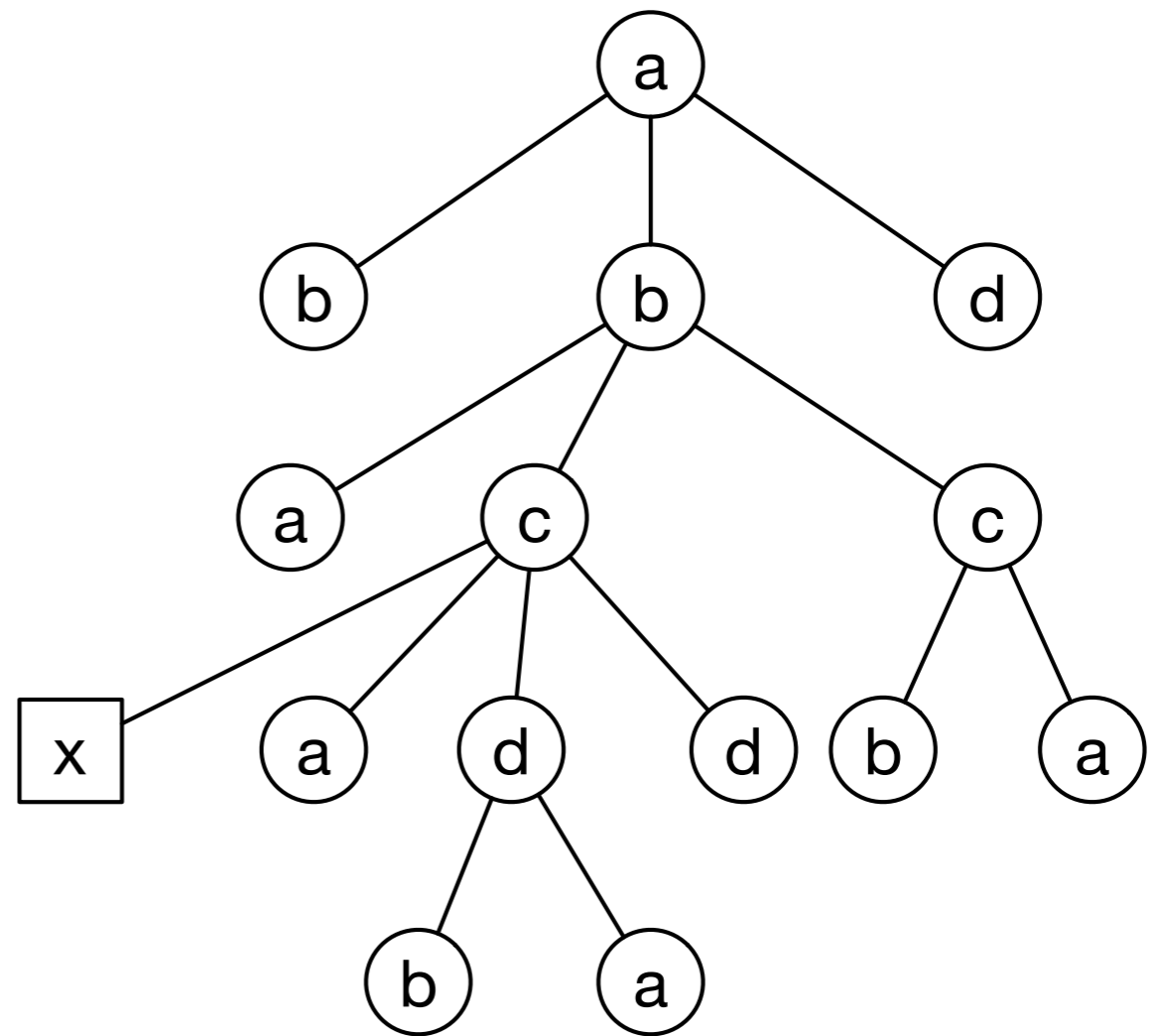
XPath: Examples

Q6 := /a/b/c[a and d]



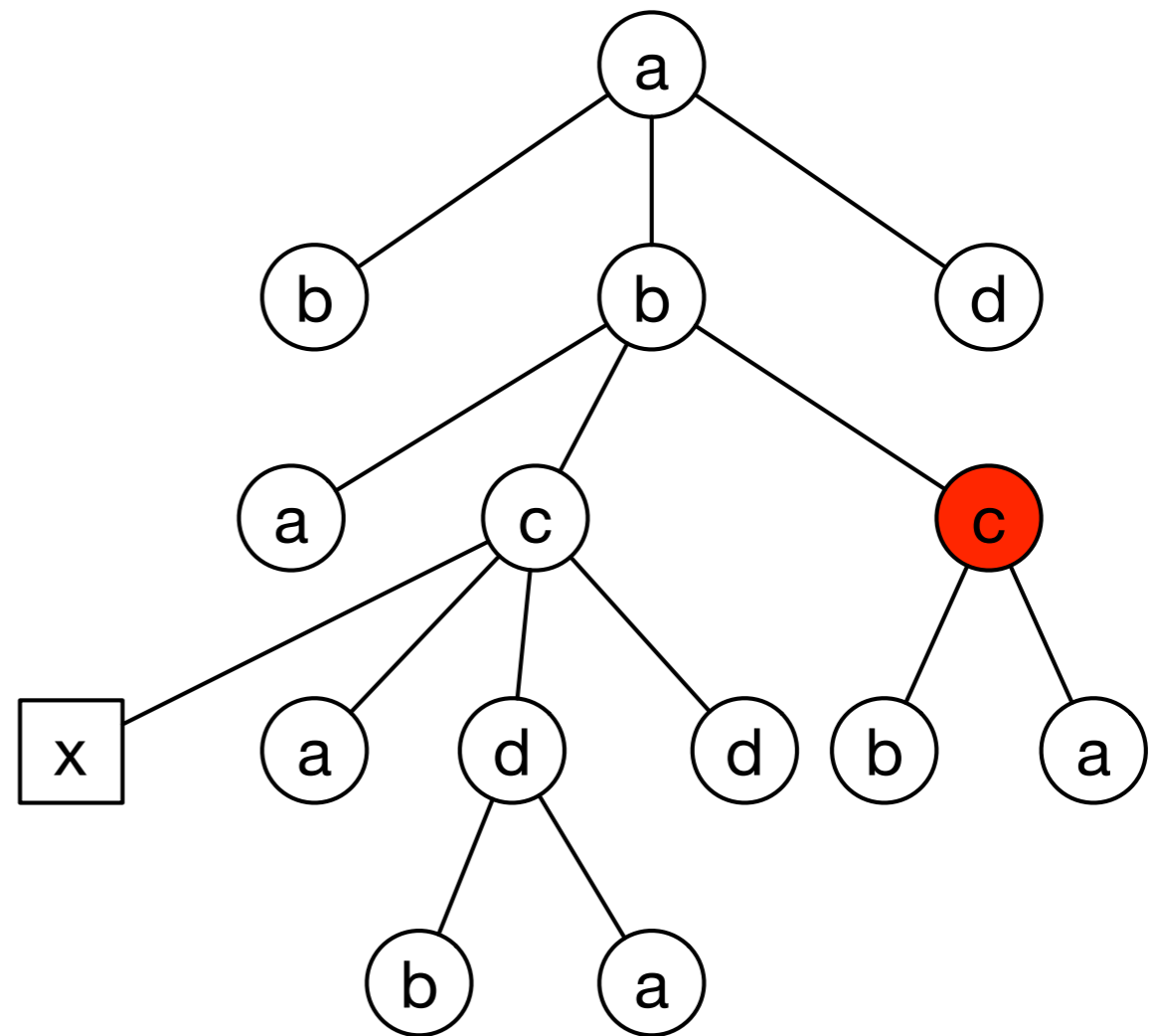
XPath: Examples

Q7 := /a/b/c[not(d)]



XPath: Examples

Q7 := /a/b/c[not(d)]



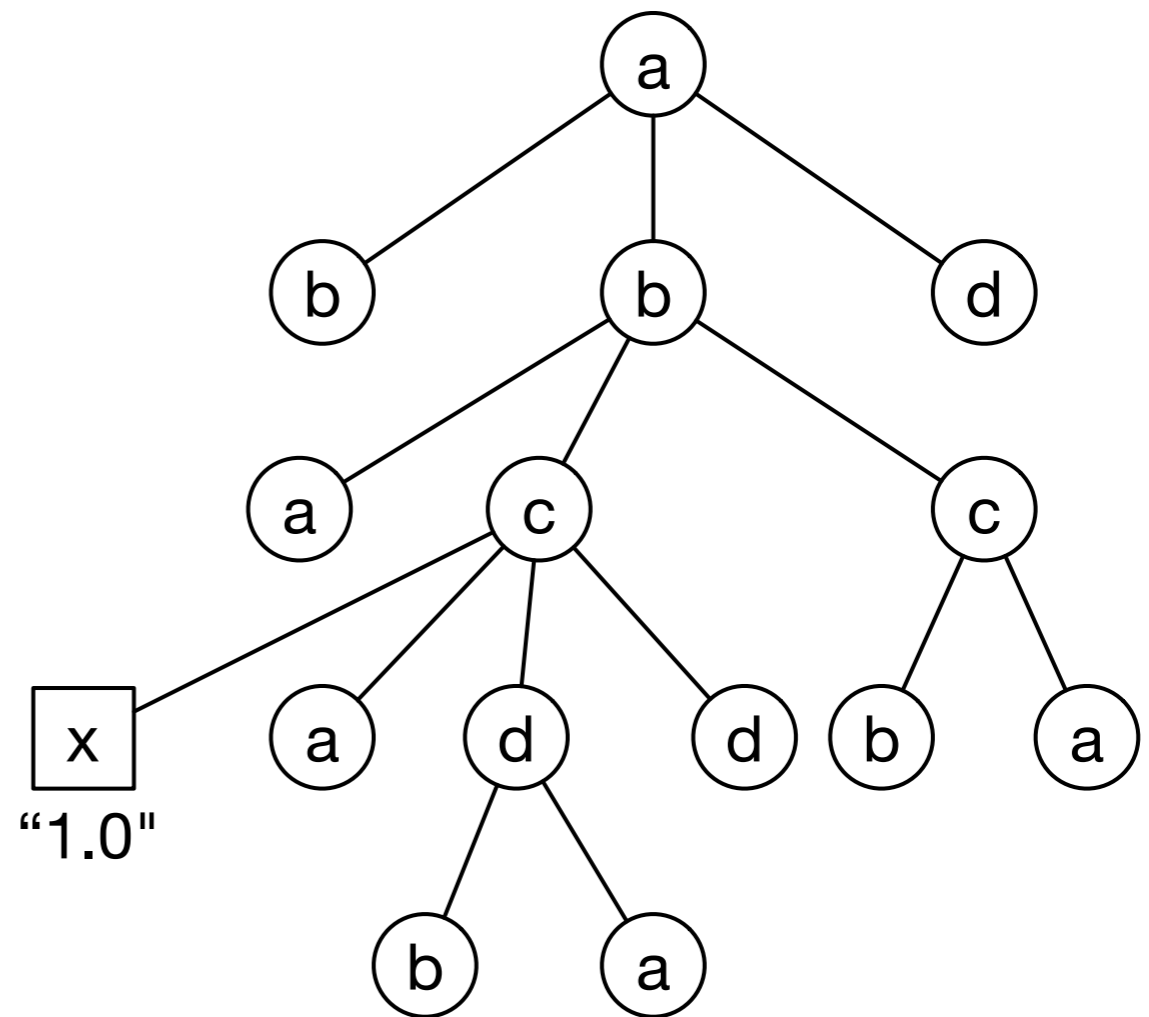
XPath: Predicates

axis :: ntest[pred₁] ··· [pred_n]

Predicates allow testing for attribute values.

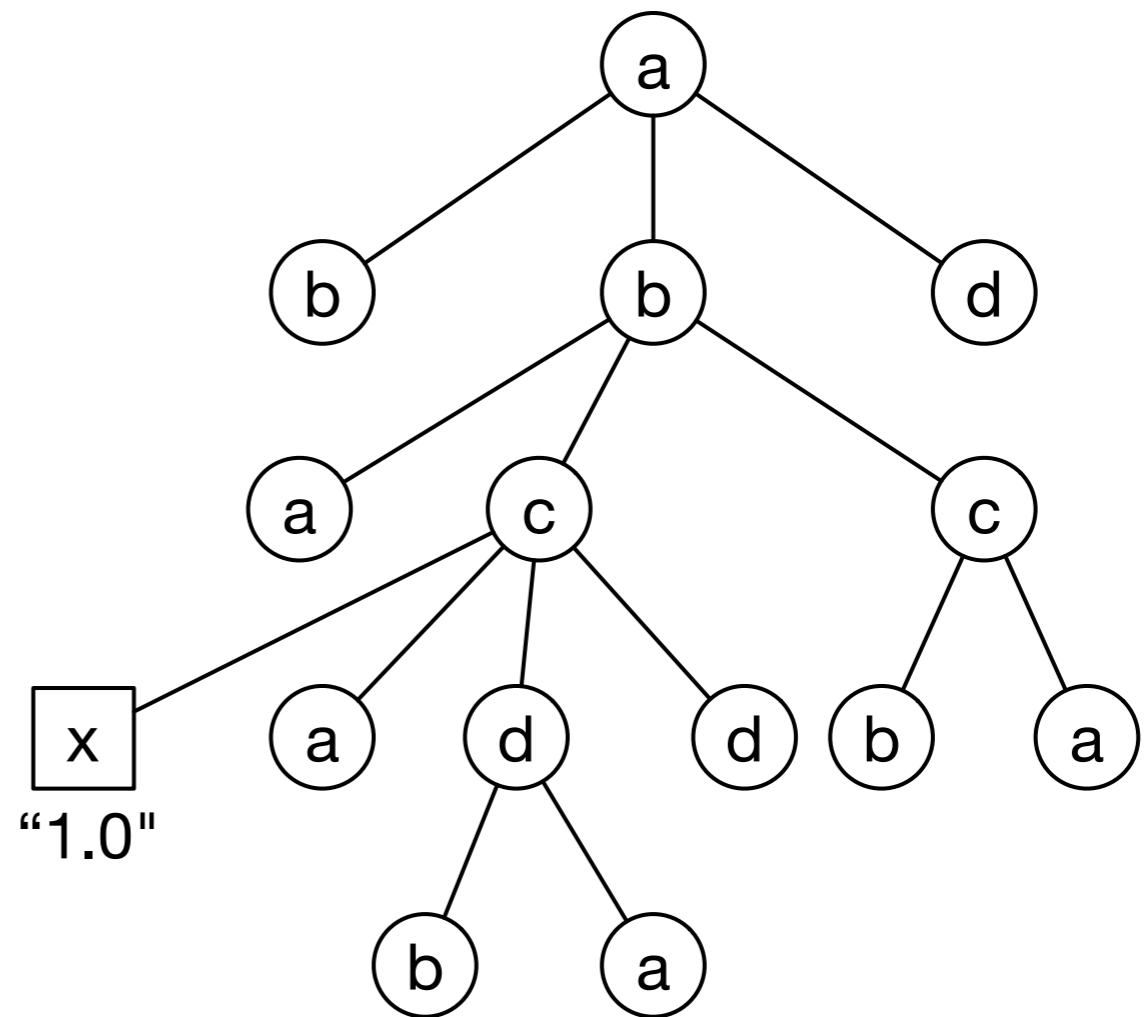
XPath: Examples

Q8 := /a/b/c[@x=1]



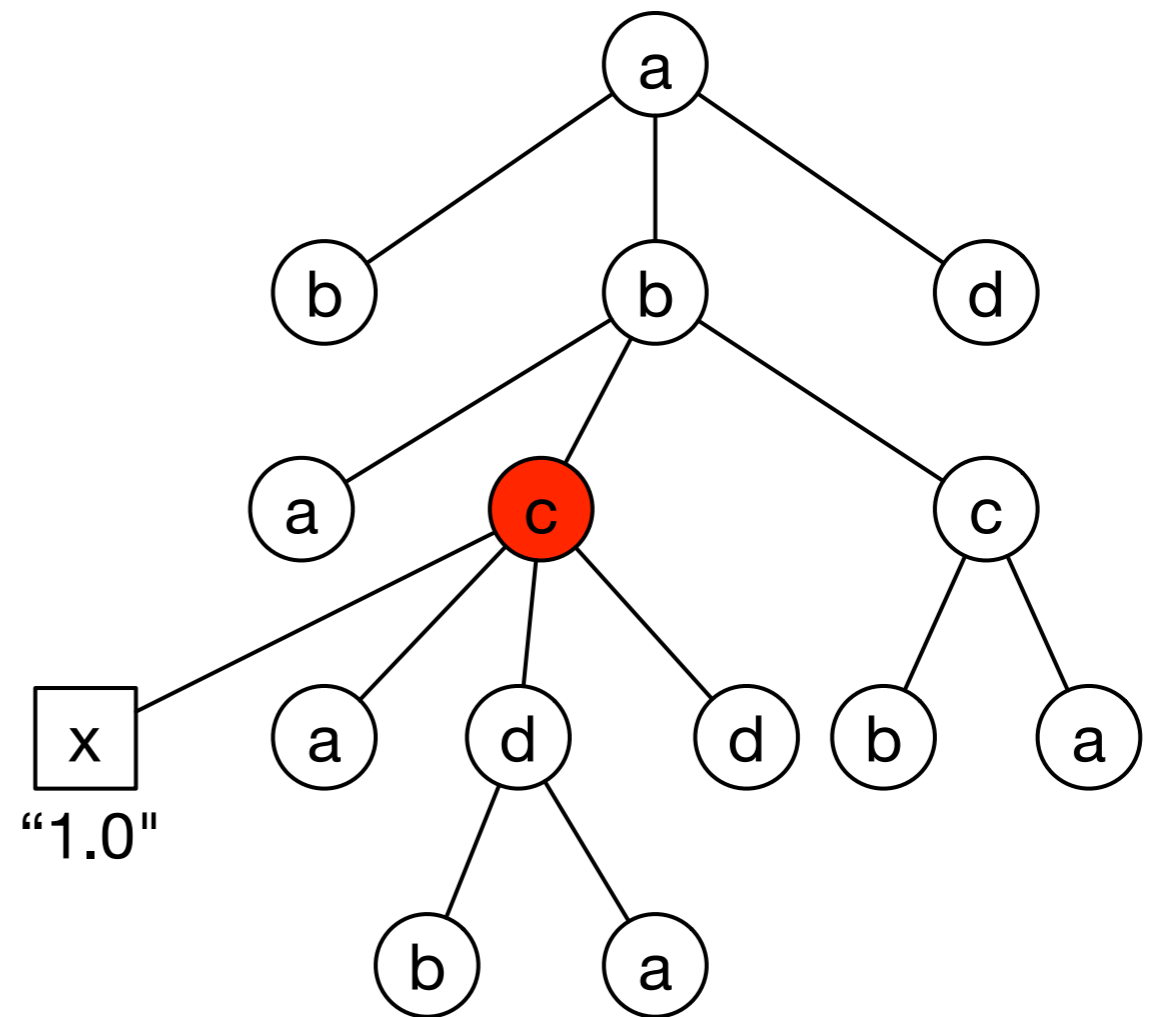
XPath: Examples

Q9 := /a/b/c[@x="1.0"]



XPath: Examples

Q9 := /a/b/c[@x="1.0"]



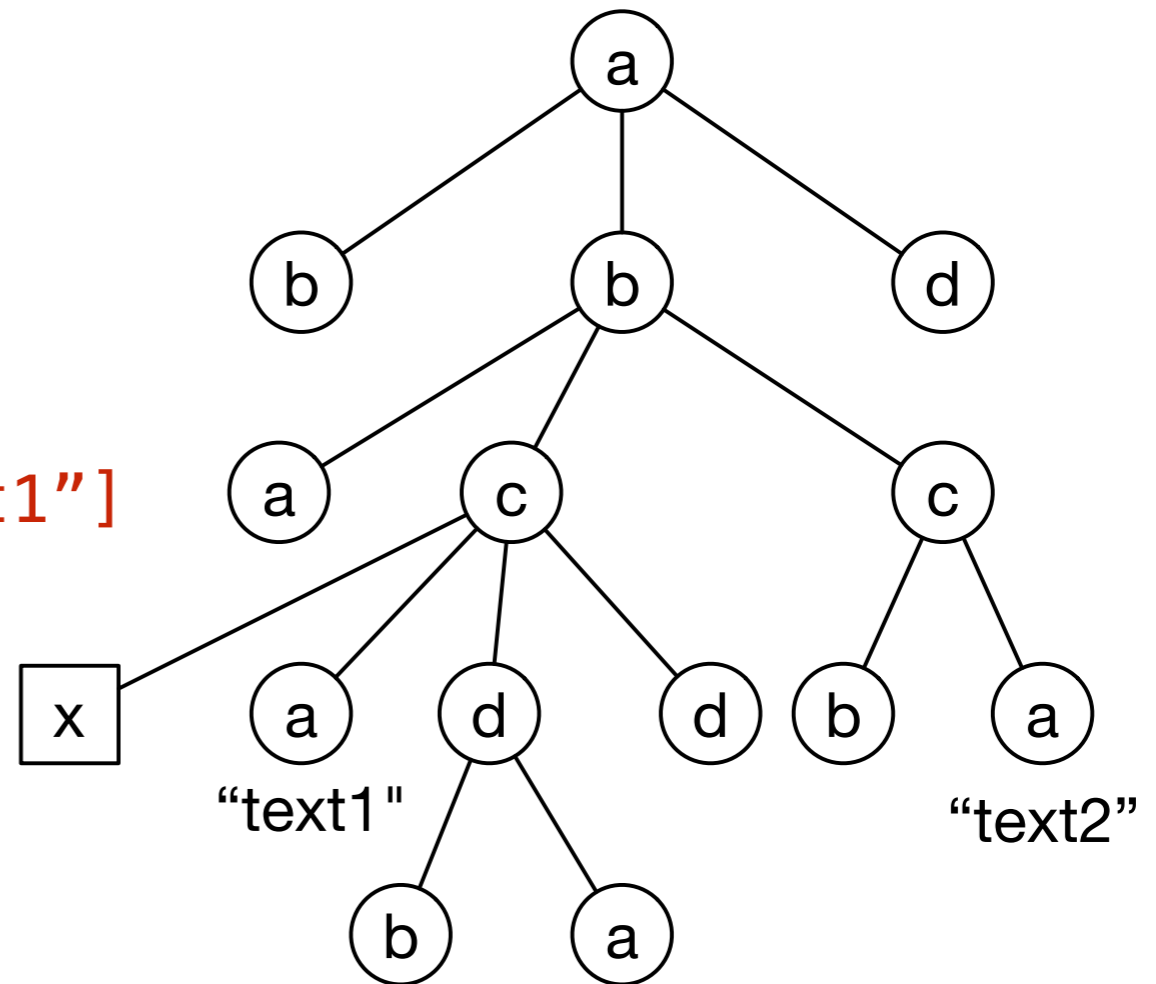
XPath: Predicates

axis :: ntest[*pred*₁] ··· [*pred*_{*n*}]

Predicates allow testing for text in nodes.

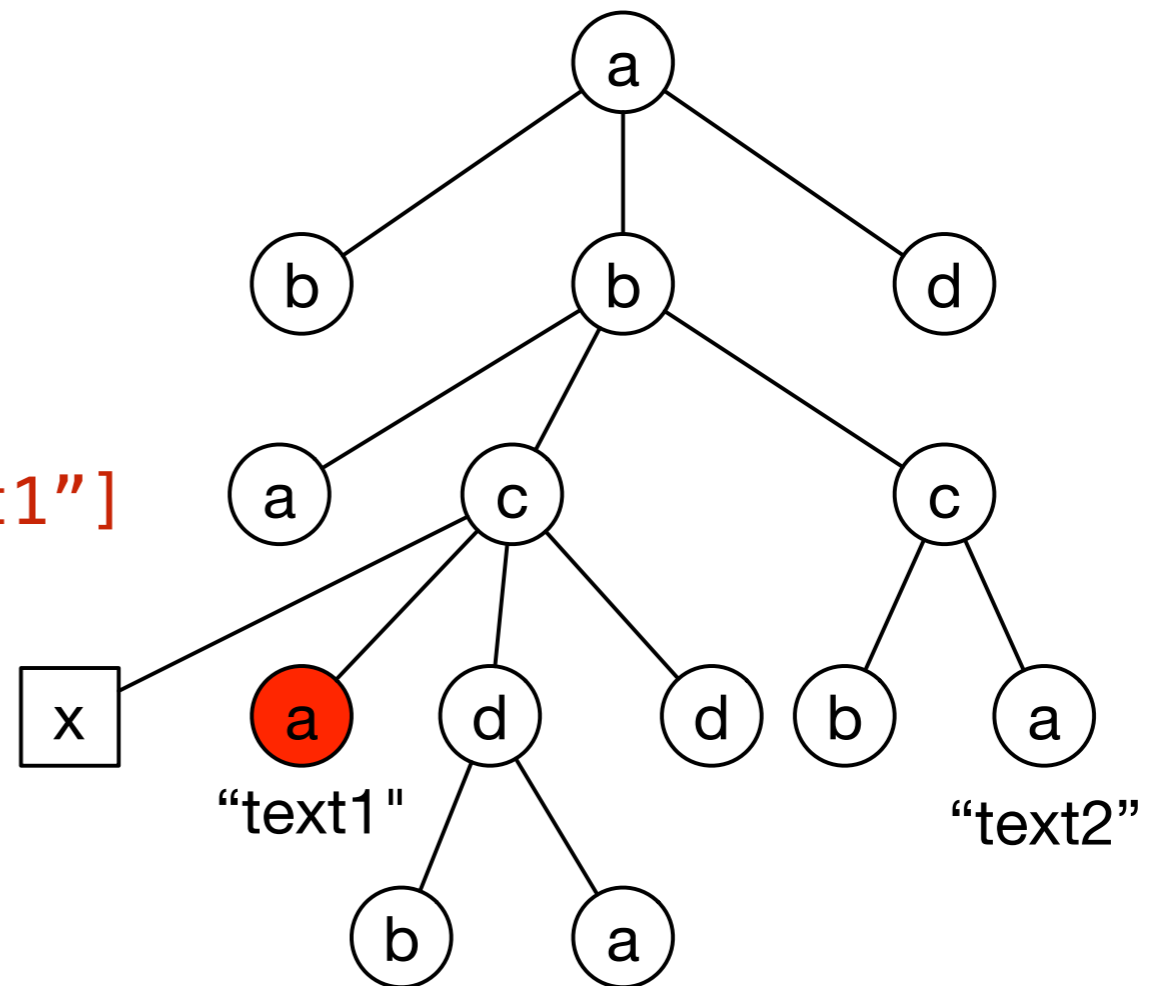
XPath: Examples

Q10 := /a/b/c//*[text()='text1']



XPath: Examples

Q10 := /a/b/c//*[text()='text1']



XPath: Useful Predicate Functions

$\text{axis} :: \text{nTest}[\text{pred}_1] \cdots [\text{pred}_n]$

Predicates allow **functions for testing on node sets:**

Function	Semantics
<code>count(ex)</code>	counts number of results
<code>last()</code>	returns context size from the evaluation context
<code>position()</code>	returns context position from the evaluation context
...	

XPath: Useful Predicate Functions

axis :: ntest[pred₁] ··· [pred_n]

Predicates allow **functions for testing on strings**:

Function	Return value
<code>concat(s1,...,sn)</code>	concatenated string
<code>startswith(str,pre)</code>	<code>true()</code> if <code>str</code> starts with <code>pre</code>
<code>contains(str,substr)</code>	<code>true()</code> if <code>str</code> contains <code>substr</code>
<code>substring(str,i,j)</code>	the substring of <code>str</code> from <code>i</code> to <code>j</code>
<code>stringlength(str)</code>	the length of <code>str</code>
...	

Useful Reading

- XPath Reference <http://www.w3.org/TR/xpath/>
- March Schol's slides on XPath <http://www.inf.uni-konstanz.de/dbis/teaching/ws0506/database-xml/P8.pdf>
- Sebastian Maneth's slides on XPath <http://www.cse.unsw.edu.au/~cs4317/10s1/lectures/06.pdf>