Web scraping and crawling, open data, markup languages and data shaping

Paolo Boldi
Dipartimento di Informatica
Università degli Studi di Milano

Data Analysis – Three steps

Data Analysis – Three steps

- In every data analysis process we can single out three phases:

Data Analysis – Three steps

- In every data analysis process we can single out three phases:
  - harvesting

Data Analysis – Three steps

- In every data analysis process we can single out three phases:
  - harvesting
  - indexing

Data Analysis – Three steps

- In every data analysis process we can single out three phases:
  - harvesting
  - indexing
  - querying / analyzing

Data Analysis – Three steps

- In every data analysis process we can single out three phases:
  - harvesting
  - indexing
  - querying / analyzing
- Example: search engine (harvesting=crawling)

Data: structured vs. unstructured

Data: structured vs. unstructured

- Data can come in with different degrees of *structure*

Data: structured vs. unstructured

- Data can come in with different degrees of *structure*
- Structured datasets (e.g. databases)

Data: structured vs. unstructured

- Data can come in with different degrees of *structure*
- Structured datasets (e.g. databases)
- Unstructured datasets (e.g. pure text)

Data: structured vs. unstructured

- Data can come in with different degrees of *structure*
- Structured datasets (e.g. databases)
- Unstructured datasets (e.g. pure text)
- Semi-structured dataset (e.g. XML)

Structured data

Typical example: relational databases

Structured data

Typical example: relational databases

User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Structured data

"Find me the e-mail address of all users that ever ordered potatoes"

User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Structured data

"Find me the e-mail address of all users that ever ordered potatoes"

User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

```
SELECT DISTINCT Email FROM User
NATURAL JOIN Order
NATURAL JOIN LineItem
NATURAL JOIN Product
WHERE Product.Description LIKE "%potato%"
```

Structured data

"Find me the e-mail address of all users that ever ordered potatoes"

User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	
99	Zach	99 South St.		zach@	

Order

OrderID	
1234	1
5678	1
...	...
5588	99

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Structure => easy to specify exactly what you need

```
SELECT DISTINCT Email FROM User
NATURAL JOIN Order
NATURAL JOIN LineItem
NATURAL JOIN Product
WHERE Product.Description LIKE "%potato%"
```

Unstructured data

Unstructured data

- Unstructured data: text, audio, images, video…

Unstructured data

- Unstructured data: text, audio, images, video…
- Information Retrieval (a.k.a., IR)

Unstructured data

- Unstructured data: text, audio, images, video…
- Information Retrieval (a.k.a., IR)
- Dates back to the 70s

Unstructured data

- Unstructured data: text, audio, images, video…
- Information Retrieval (a.k.a., IR)
- Dates back to the 70s
- Boosted by the advent of search engines

Semi-structured data

Semi-structured data

- Less structured than a database, more structured than poor text

Semi-structured data

- Less structured than a database, more structured than poor text
- Typical formats:

Semi-structured data

- Less structured than a database, more structured than poor text
- Typical formats:
  - markup languages (most notably: XML)

Semi-structured data

- Less structured than a database, more structured than poor text
- Typical formats:
  - markup languages (most notably: XML)
  - JSON

Markup languages

Definition

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*
- The added information can be of various kinds

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*
- The added information can be of various kinds
  - presentational (explains how the text should be rendered / visualized / reproduced)

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*
- The added information can be of various kinds
  - presentational (explains how the text should be rendered / visualized / reproduced)
  - procedural (provides instructions to tools that will process the text)

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*
- The added information can be of various kinds
  - presentational (explains how the text should be rendered / visualized / reproduced)
  - procedural (provides instructions to tools that will process the text)
  - descriptive (explains what the text means)

Definition

- A **markup language** is a language used to annotate text with *extra-textual information*
- The added information can be of various kinds
  - presentational (explains how the text should be rendered / visualized / reproduced)
  - procedural (provides instructions to tools that will process the text)
  - descriptive (explains what the text means)
- Sometimes the markup structure takes over

SGML example (docbook)

```
<article>
== A paper about ducks
    <articleinfo>
=== This paper talks about ducks
        <author>
            <firstname>Daffy</firstname>
            <surname>Duck</surname>
        </author>
        <volumenum>1234</volumenum>
    <chapter>
=== Chapter on how ducks are born
    === Female ducks
            Blablabla
        === Anatomy of female ducks
                Blablabla
    === Male ducks
            Blablabla
            <mediaobject>
                    <imageobject>
                    image::duck.png[]
                    </imageobject>
                    <textobject>
                        <phrase>This is a nice duck</phrase>
                    </textobject>
            </mediaobject>
……
```

XML example

```
<note>
  <date>
    <day>12</day>
    <month>11</month>
    <year>99</year>
  </date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

HTML example

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

LaTeX example

```
\paragraph{Sets, integers, keys.} For every natural number
$n$, I let $[n]=\{\,0,1,\ldots,n-1\,\}$; I occasionally use the same notation
when $n$ is a real number, omitting a ceiling operation.

In the following, I will always assume that a universe $U$ of $u=|U|$ items
called \emph{keys} is fixed; this set may in many applications be infinite, but
unless otherwise specified I will suppose that it is finite. Occasionally, I assume
that $U$ is endowed with a total order $\leq$.
```

Pre-history of markup

Pre-history of markup

- First examples of markup languages date back to the 60s (*GenCode, troff* and *nroff*)

Pre-history of markup

- First examples of markup languages date back to the 60s (*GenCode, troff* and *nroff*)
- Knuth's *TeX* is one of the first typesetting systems (especially aimed at mathematics)

SGML

SGML

- The first general-purpose markup language (SGML=Standard Generalized Markup Language) [1986]

SGML

- The first general-purpose markup language (SGML=Standard Generalized Markup Language) [1986]
- The language of choice for military, aerospace, technical, industrial applications

SGML

- The first general-purpose markup language (SGML=Standard Generalized Markup Language) [1986]
- The language of choice for military, aerospace, technical, industrial applications
- HTML used to be (until HTML 5) a *DTD of SGML*

SGML

- The first general-purpose markup language (SGML=Standard Generalized Markup Language) [1986]
- The language of choice for military, aerospace, technical, industrial applications
- HTML used to be (until HTML 5) a *DTD of SGML*
- By now largely substituted by XML

HTML

HTML

- A markup language used to specify hypertexts (HTML=HyperText Markup Language) [1993]

HTML

- A markup language used to specify hypertexts (HTML=HyperText Markup Language) [1993]
- Born as a *formatting language*

HTML

- A markup language used to specify hypertexts (HTML=HyperText Markup Language) [1993]
- Born as a *formatting language*
- Progressively transformed into a language that only specifies the *logical structure* of a document (formatting is specified separately, typically through CSS)

HTML, SGML, XML

HTML, SGML, XML

- Originally HTML was a DTD (a special case) of SGML

HTML, SGML, XML

- Originally HTML was a DTD (a special case) of SGML
- Now HTML and SGML are different (non-related) markup languages

HTML, SGML, XML

- Originally HTML was a DTD (a special case) of SGML
- Now HTML and SGML are different (non-related) markup languages
- XML is different from both, but there exists a XML version of HTML called XHTML (that browsers support)

HTML, SGML, XML

- Originally HTML was a DTD (a special case) of SGML
- Now HTML and SGML are different (non-related) markup languages
- XML is different from both, but there exists a XML version of HTML called XHTML (that browsers support)
- In a way, though, HTML can be thought of as X(H)TML

A brief introduction to XML and XPath

XML

XML

- XML=Extensible Markup Language

XML

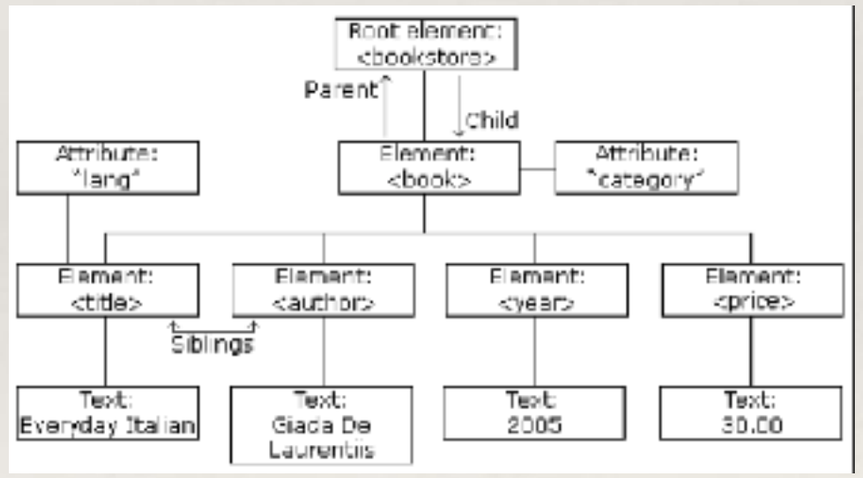
- XML=Extensible Markup Language
- A way to specify a (semi-structured) document

XML

- XML=Extensible Markup Language
- A way to specify a (semi-structured) document
- A document is the textual representation of a *tree*

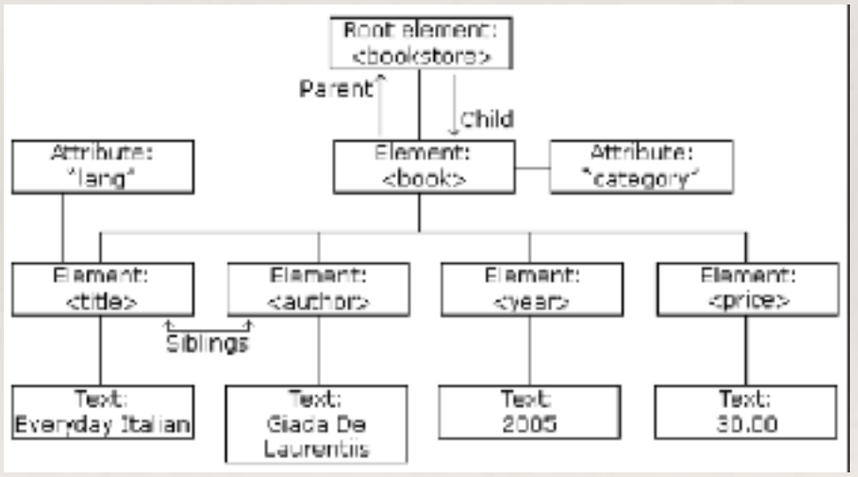
XML (tree view)

XML (tree view)



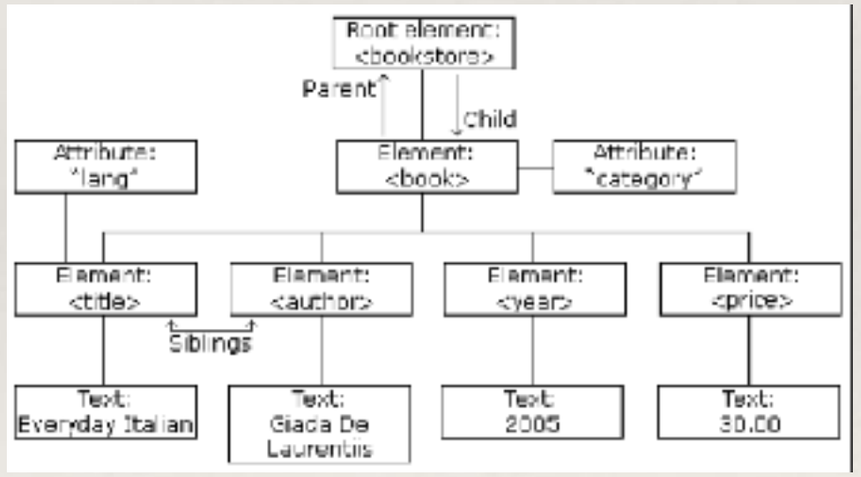
XML (tree view)

- Nodes of the tree are called *elements*



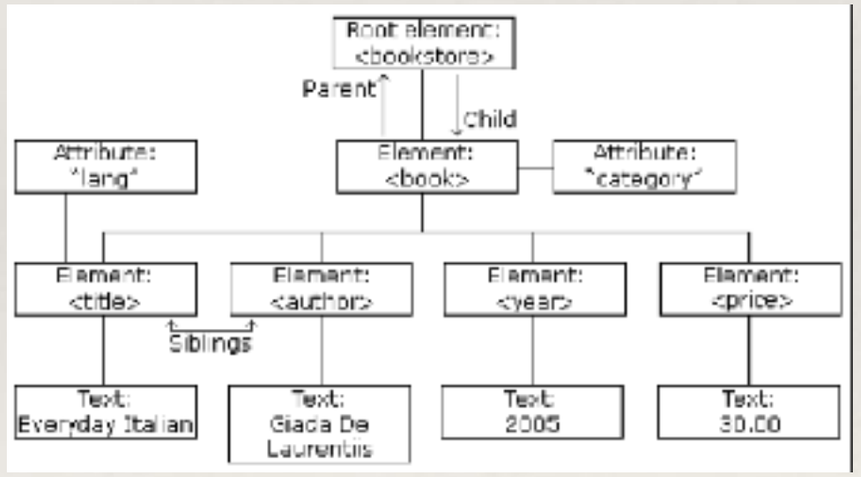
XML (tree view)

- Nodes of the tree are called *elements*
- The starting point of the tree is called *root*



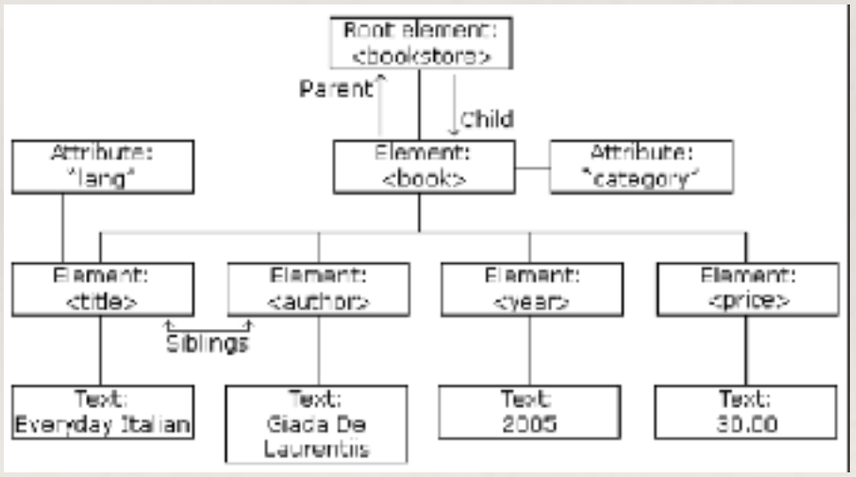
XML (tree view)

- Nodes of the tree are called *elements*
- The starting point of the tree is called *root*
- Each element may have one or more *children:*



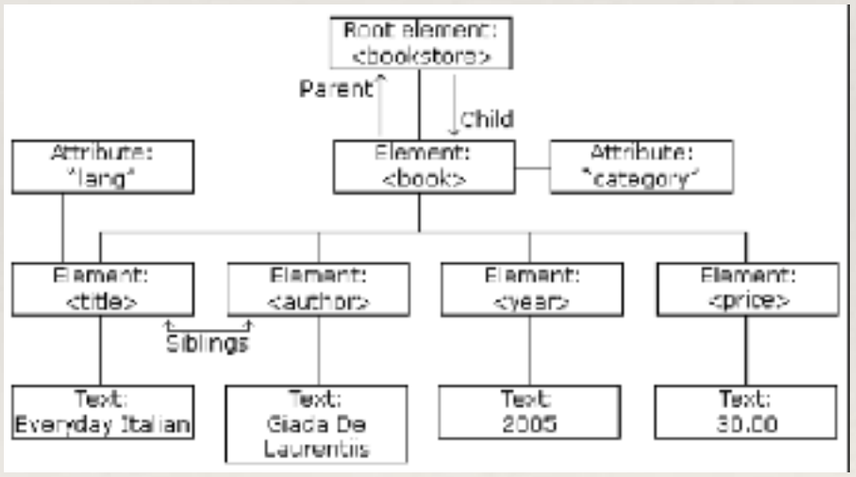
XML (tree view)

- Nodes of the tree are called *elements*
- The starting point of the tree is called *root*
- Each element may have one or more *children:*
  - children can be other elements…



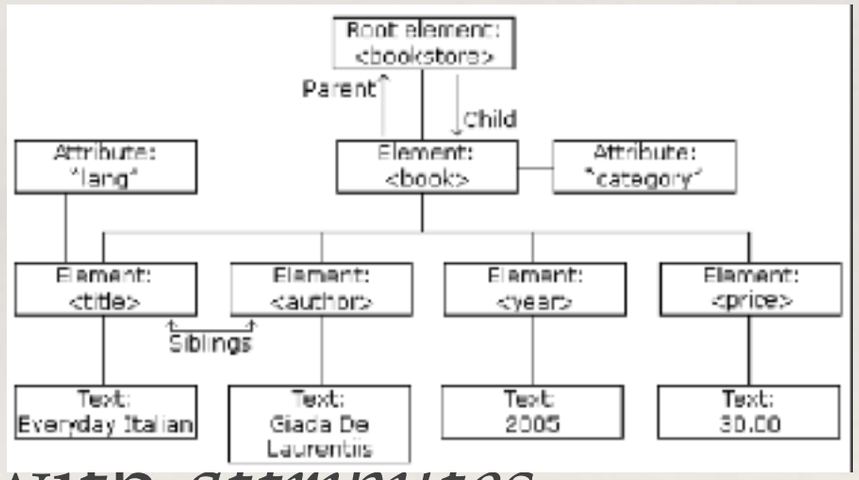
XML (tree view)

- Nodes of the tree are called *elements*
- The starting point of the tree is called *root*
- Each element may have one or more *children:*
  - children can be other elements…
  - …or text



XML (tree view)

- Nodes of the tree are called *elements*
- The starting point of the tree is called *root*
- Each element may have one or more *children:*
  - children can be other elements…
  - …or text
- Elements may further be decorated with *attributes* (name/value pairs)



XML (text view)

XML (text view)

- Every element is enclosed between a start-tag and an end-tag:

<bookstore>

.....

</bookstore>

XML (text view)

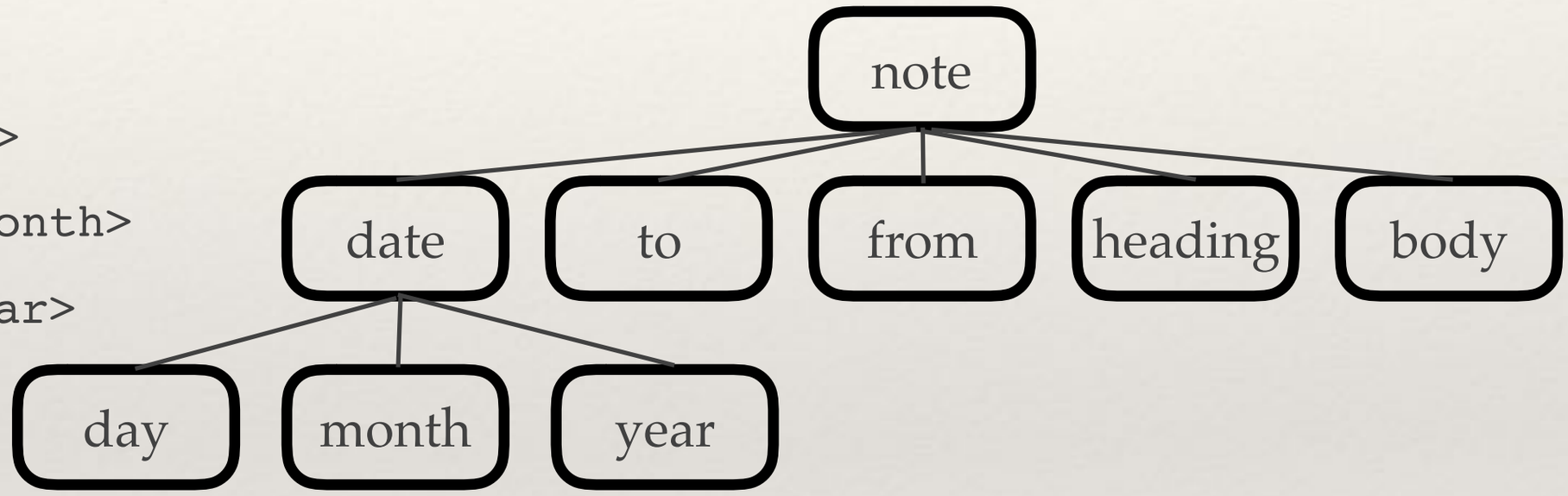
- Every element is enclosed between a start-tag and an end-tag:
  <bookstore>
  .....
  </bookstore>
- Attributes are specified in the start-tag:
  <person gender="male" ethnicity="caucasian">
  .....
  </person>

Example

```
<?xml version="1.0"?>
<note>
   <date>
      <day>12</day>
      <month>11</month>
      <year>99</year>
   </date>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```

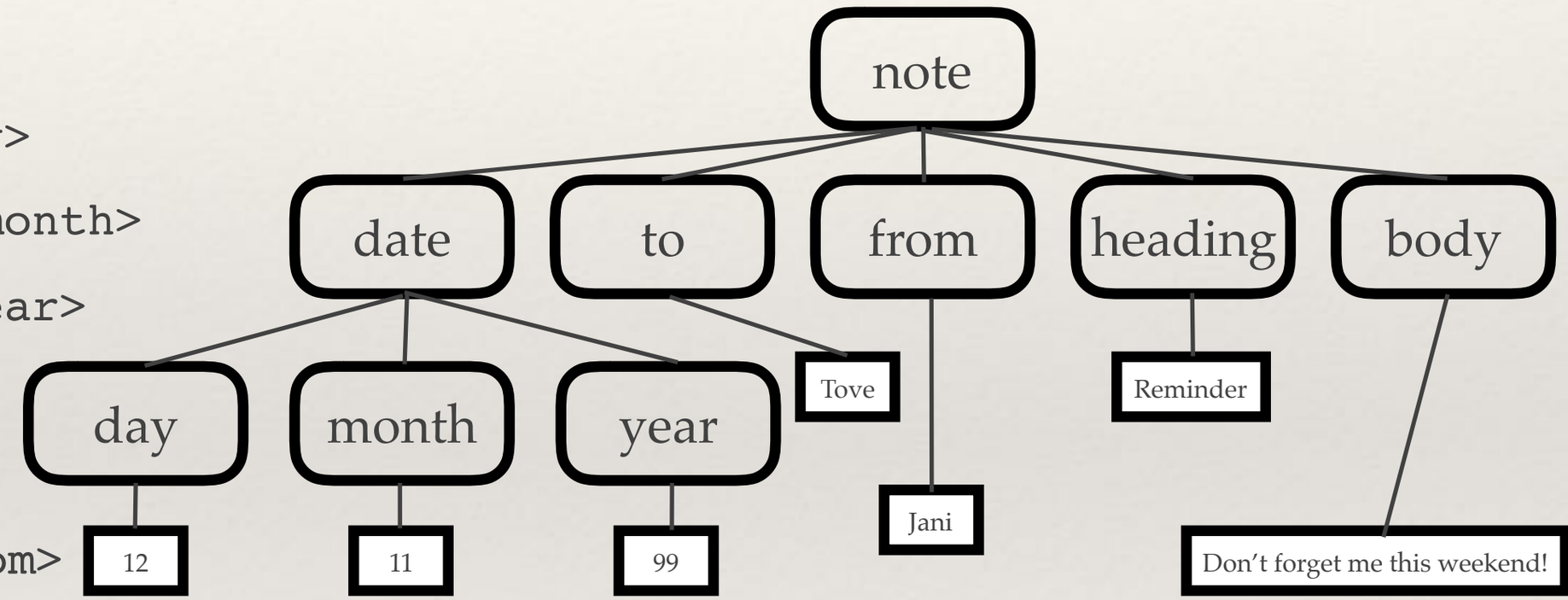
Example

```
<?xml version="1.0"?>
<note>
   <date>
      <day>12</day>
      <month>11</month>
      <year>99</year>
   </date>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```



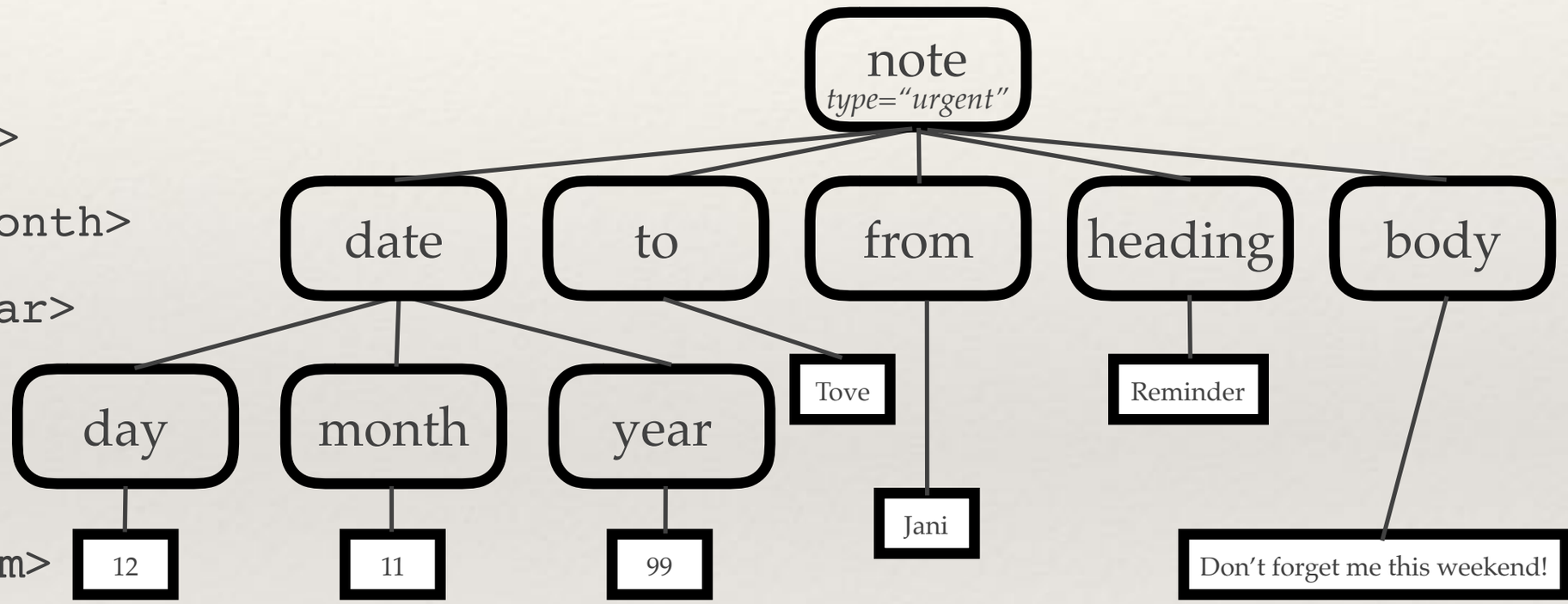
Example

```
<?xml version="1.0"?>
<note>
   <date>
      <day>12</day>
      <month>11</month>
      <year>99</year>
   </date>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```



Example (with attributes)

```
<?xml version="1.0"?>
<note type=“urgent”>
   <date>
      <day>12</day>
      <month>11</month>
      <year>99</year>
   </date>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```



XML elements and attributes

XML elements and attributes

- XML distinguishes between well-formedness and validity

XML elements and attributes

- XML distinguishes between well-formedness and validity
- A *well-formed XML document* doesn't need:

XML elements and attributes

- XML distinguishes between well-formedness and validity
- A *well-formed XML document* doesn't need:
  - a list of allowed elements and/or attributes

XML elements and attributes

- XML distinguishes between well-formedness and validity
- A *well-formed XML document* doesn't need:
  - a list of allowed elements and/or attributes
  - a specification of which elements can be enclosed in which other elements etc…

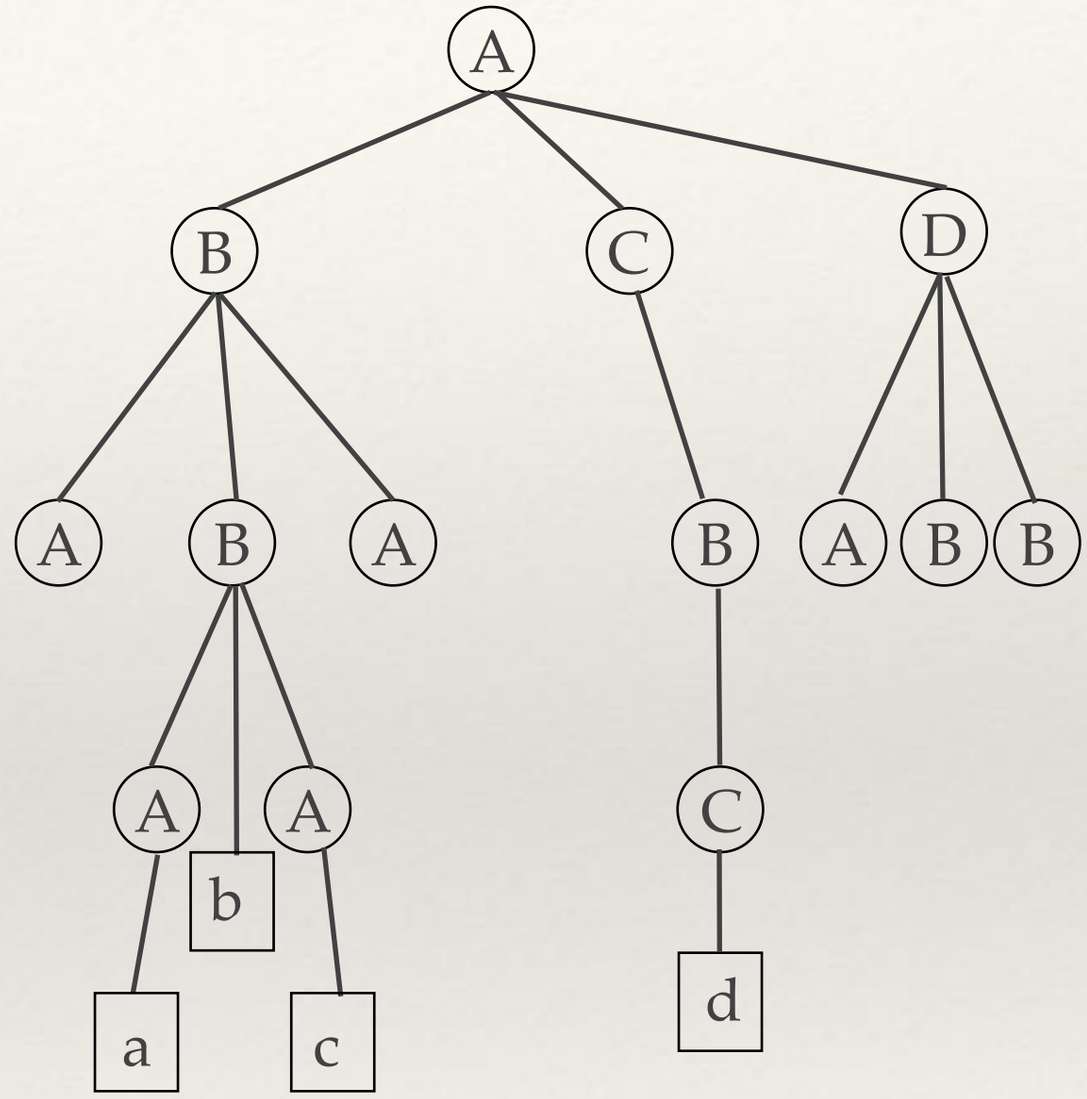
XML elements and attributes

- XML distinguishes between well-formedness and validity
- A *well-formed XML document* doesn't need:
  - a list of allowed elements and/or attributes
  - a specification of which elements can be enclosed in which other elements etc…
- A *valid* document requires that one specifies a *schema* (typically, a DTD=Document Type Definition)

XML elements and attributes

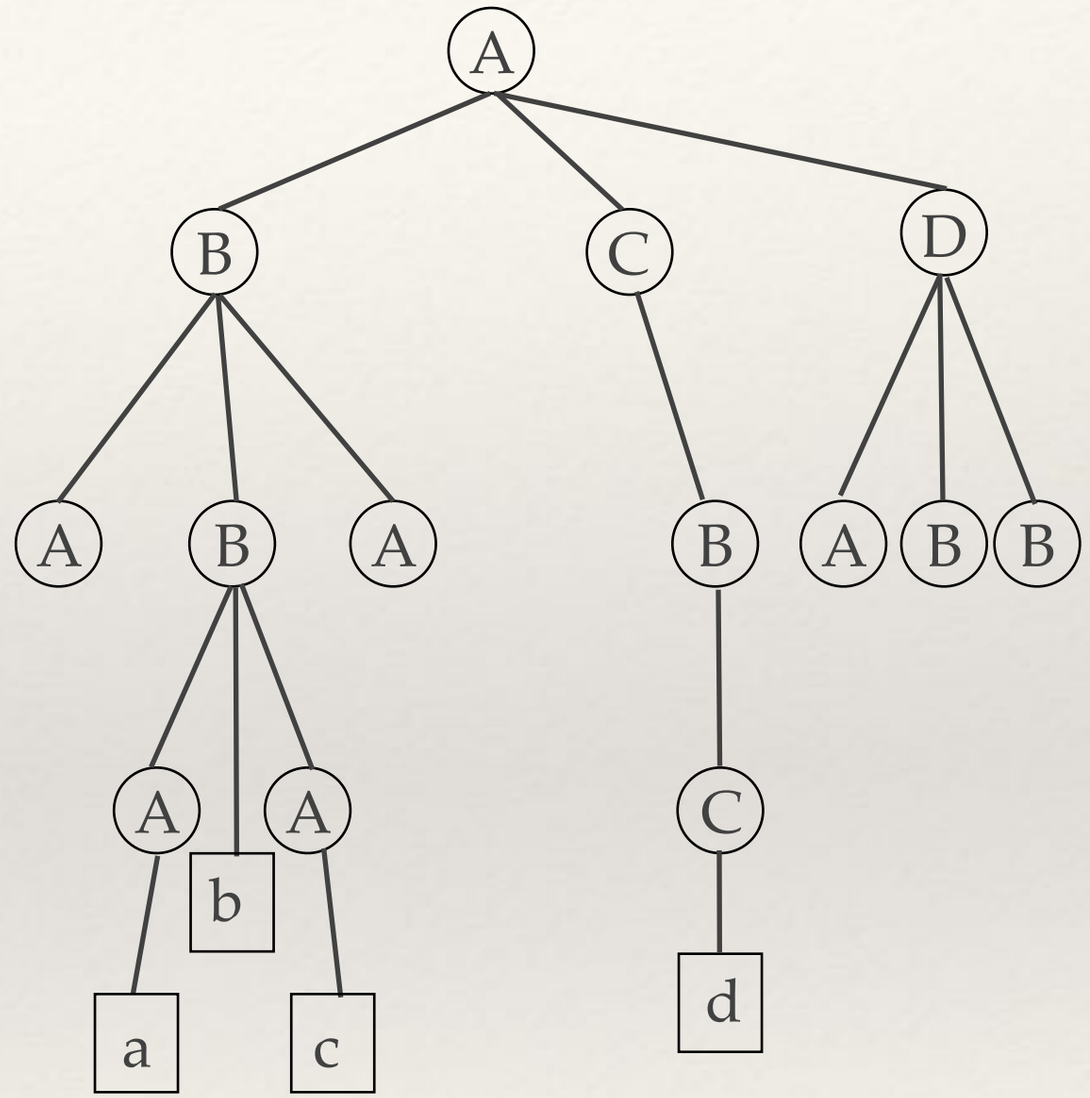
- XML distinguishes between well-formedness and validity
- A *well-formed XML document* doesn't need:
  - a list of allowed elements and/or attributes
  - a specification of which elements can be enclosed in which other elements etc…
- A *valid* document requires that one specifies a *schema* (typically, a DTD=Document Type Definition)
- We will only look at well-formedness, not validity

Example: an XML document

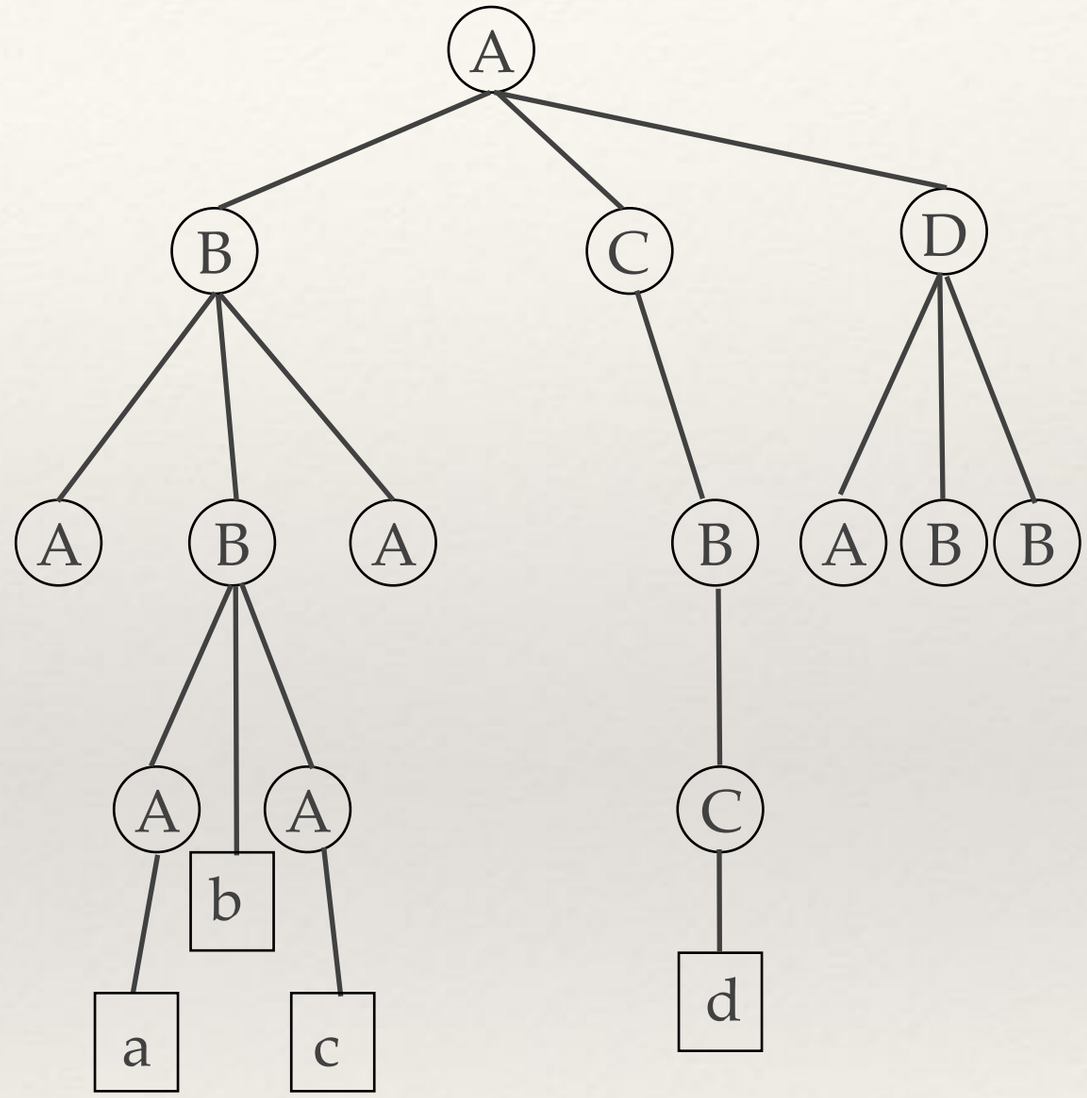


Example: an XML document

```
<A>
      <B>
            <A></A>
            <B>
                  <A>a</A>
                  b
                  <A>c</A>
            </B>
            <A></A>
      </B>
      <C>
            <B>
                  <C>d</C>
            </B>
      </C>
      <D>
            <A></A>
            <B></B>
            <B></B>
      </D>
</A>
```

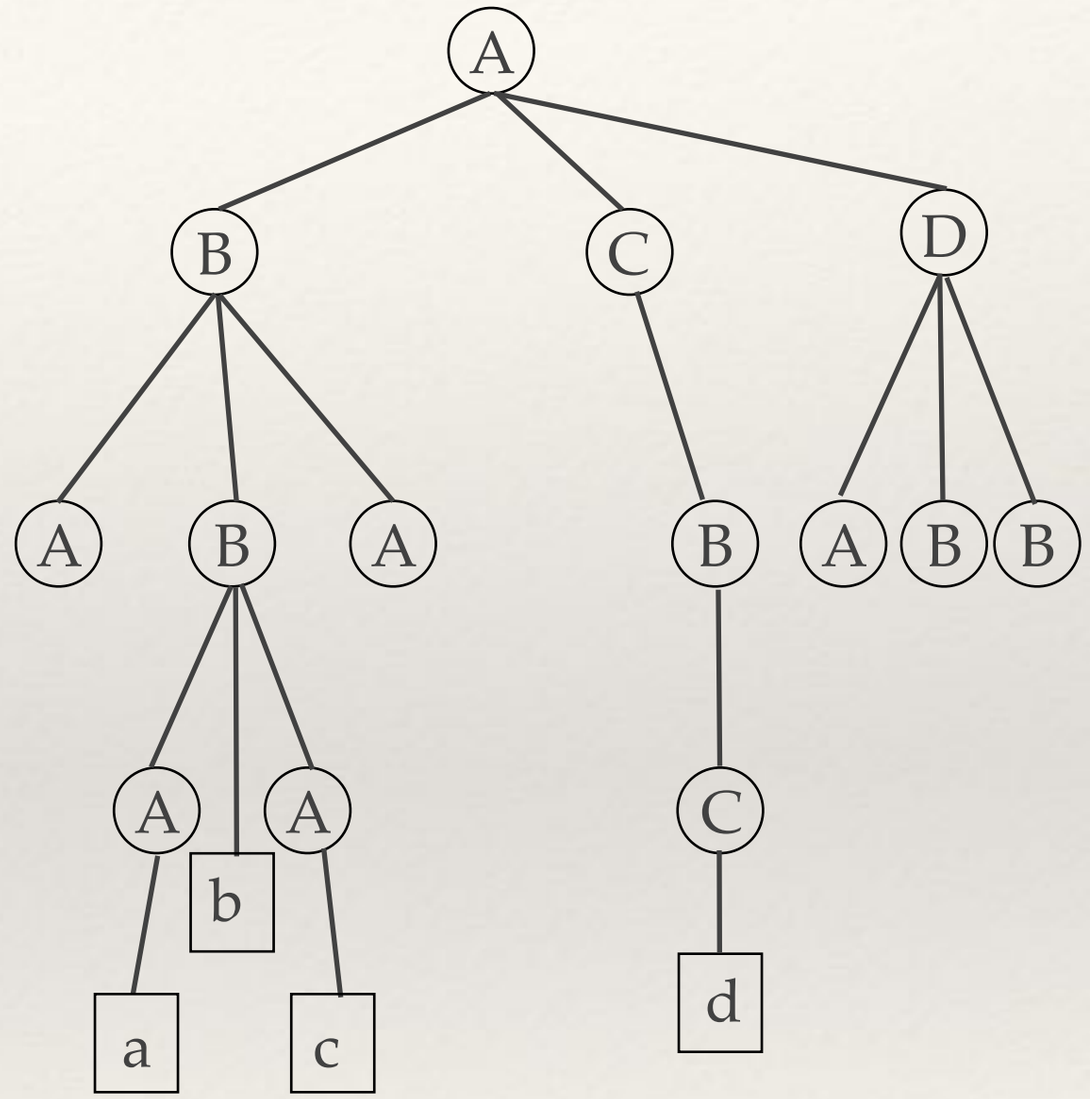


Example: an XML document (with empty tags for empty elements)



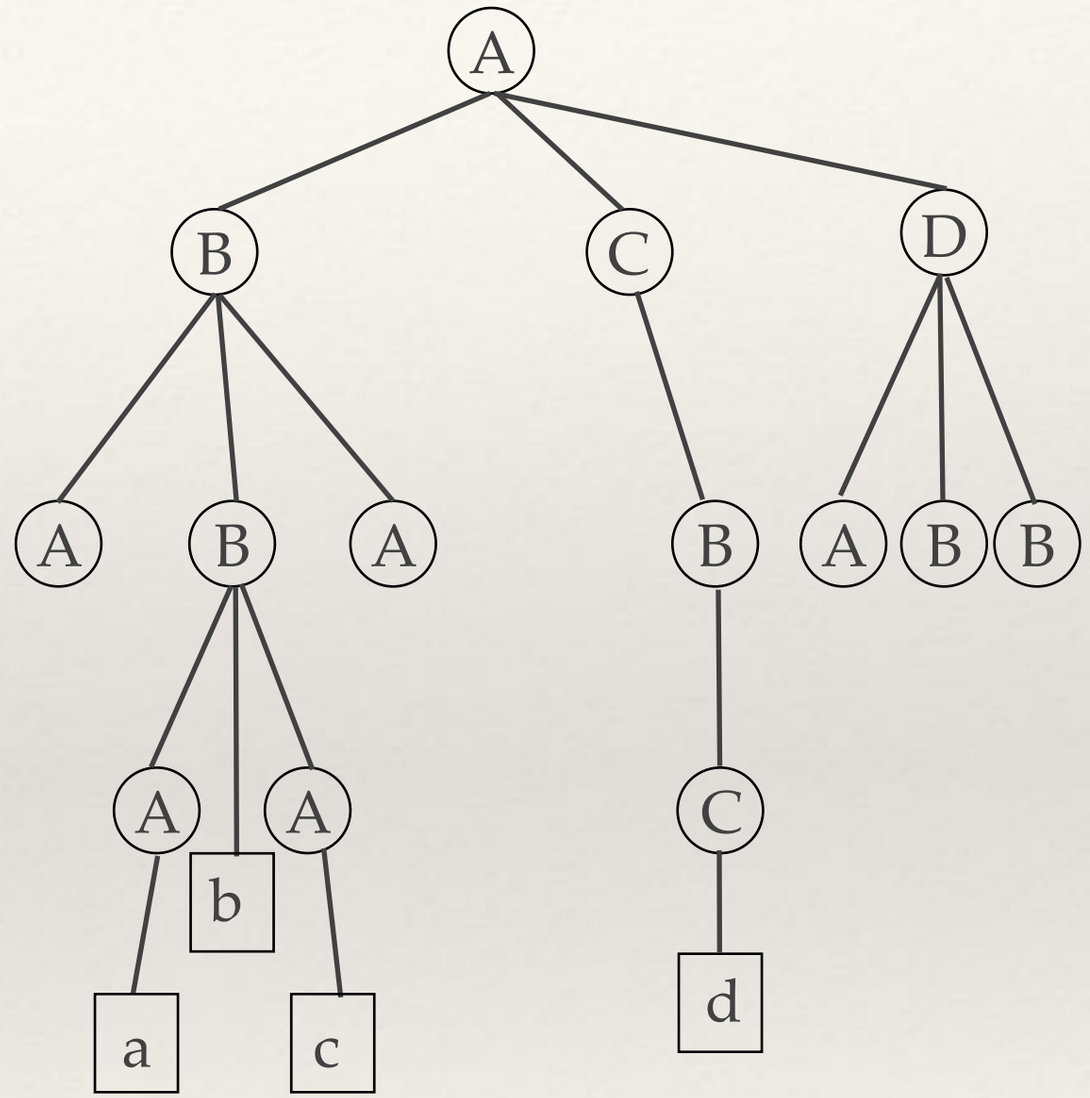
Example: an XML document (with empty tags for empty elements)

```
<A>
      <B>
            <A></A>
            <B>
                  <A>a</A>
                  b
                  <A>c</A>
            </B>
            <A></A>
      </B>
      <C>
            <B>
                  <C>d</C>
            </B>
      </C>
      <D>
            <A></A>
            <B></B>
            <B></B>
      </D>
</A>
```



Example: an XML document (with empty tags for empty elements)

```
<A>
      <B>
            <A/>
            <B>
                  <A>a</A>
                  b
                  <A>c</A>
            </B>
            <A/>
      </B>
      <C>
            <B>
                  <C>d</C>
            </B>
      </C>
      <D>
            <A/>
            <B/>
            <B/>
      </D>
</A>
```



Online resource

`http://countwordsfree.com/xmlviewer`

Online resource

http://countwordsfree.com/xmlviewer

```
<A>
     <B>
          <A></A>
          <B>
               <A>a</A>
               b
               <A>c</A>
          </B>
          <A></A>
     </B>
     <C>
          <B>
               <C>d</C>
          </B>
     </C>
     <D>
          <A></A>
          <B></B>
          <B></B>
     </D>
</A>
```

Online resource

`http://countwordsfree.com/xmlviewer`

XML

XML

- An XML document has more structure than a simple text document

XML

- An XML document has more structure than a simple text document
- One can write *queries* using suitable query languages

XML

- An XML document has more structure than a simple text document
- One can write *queries* using suitable query languages
- XPath is one such language

XPath idea

XPath idea

- Many types of queries; the most important is called a *location path*

XPath idea

- Many types of queries; the most important is called a *location path*
- Given an XPath location path query and an XML document, the query *selects* (zero, one or many) nodes in the document

XPath idea

- Many types of queries; the most important is called a *location path*
- Given an XPath location path query and an XML document, the query *selects* (zero, one or many) nodes in the document
- Other types of queries return strings or other values

XPath location paths

XPath location paths

- We only consider *absolute location paths*

XPath location paths

- We only consider *absolute location paths*
- An absolute path identifies a set of nodes in the XML tree

XPath location paths

- We only consider *absolute location paths*
- An absolute path identifies a set of nodes in the XML tree
- Syntax (full):

XPath location paths

- We only consider *absolute location paths*
- An absolute path identifies a set of nodes in the XML tree
- Syntax (full):
  - /step/step/step…

XPath location paths

- We only consider *absolute location paths*
- An absolute path identifies a set of nodes in the XML tree
- Syntax (full):
  - /step/step/step…
  - where step is

XPath location paths

- We only consider *absolute location paths*
- An absolute path identifies a set of nodes in the XML tree
- Syntax (full):
  - /step/step/step…
  - where step is
  - axis::node_test[predicate][predicate]… (the predicate part is optional)

Meaning of a step

Meaning of a step

- You can think of a location path as a way to identify a set of (zero, one or more) nodes, called *current nodes*

Meaning of a step

- You can think of a location path as a way to identify a set of (zero, one or more) nodes, called *current nodes*
- Each step modifies the set of current nodes, depending on

Meaning of a step

- You can think of a location path as a way to identify a set of (zero, one or more) nodes, called *current nodes*
- Each step modifies the set of current nodes, depending on
  - axis: direction where we should move

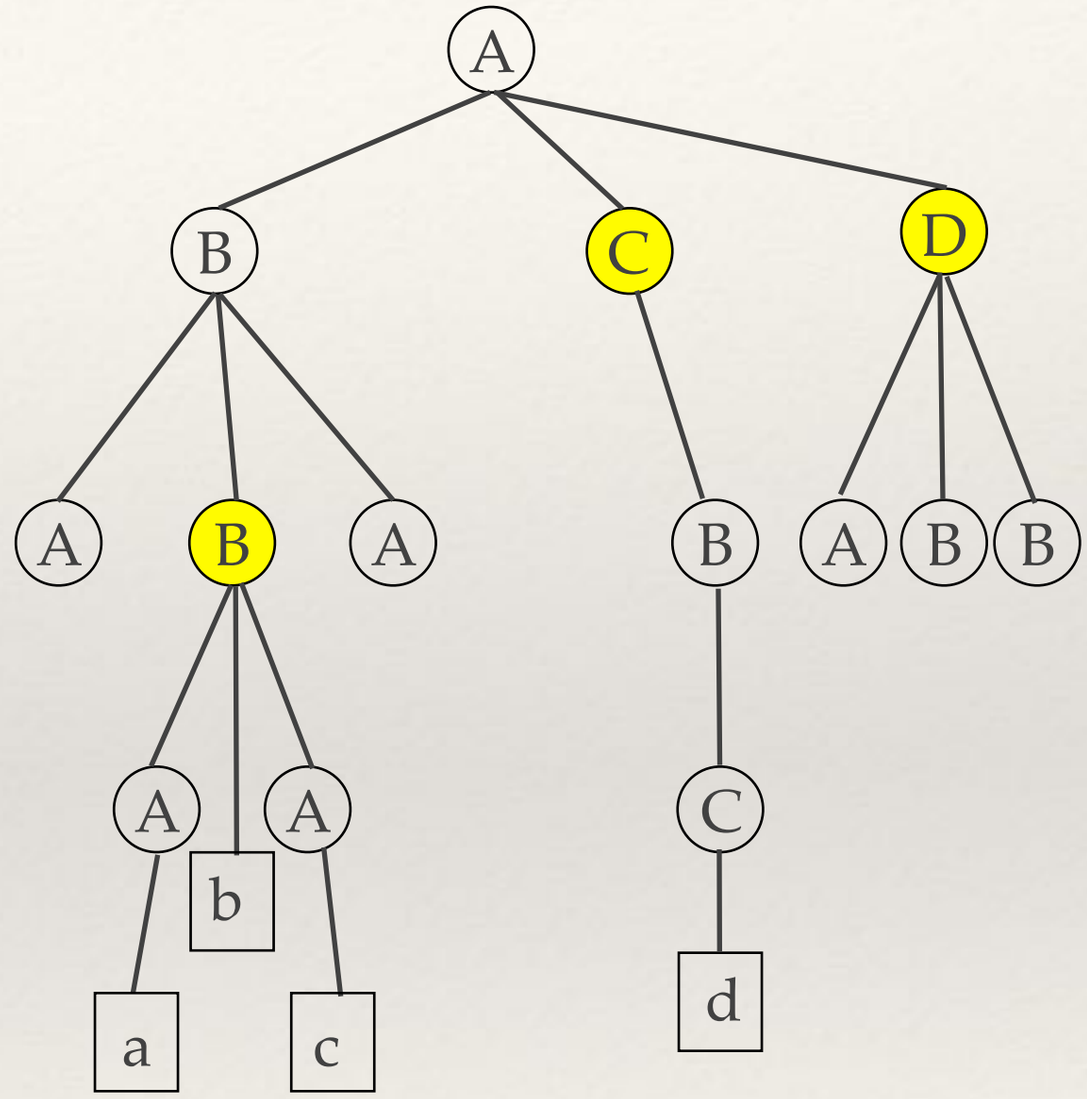
Meaning of a step

- You can think of a location path as a way to identify a set of (zero, one or more) nodes, called *current nodes*
- Each step modifies the set of current nodes, depending on
  - axis: direction where we should move
  - node_test: select only nodes with a specific name

Meaning of a step

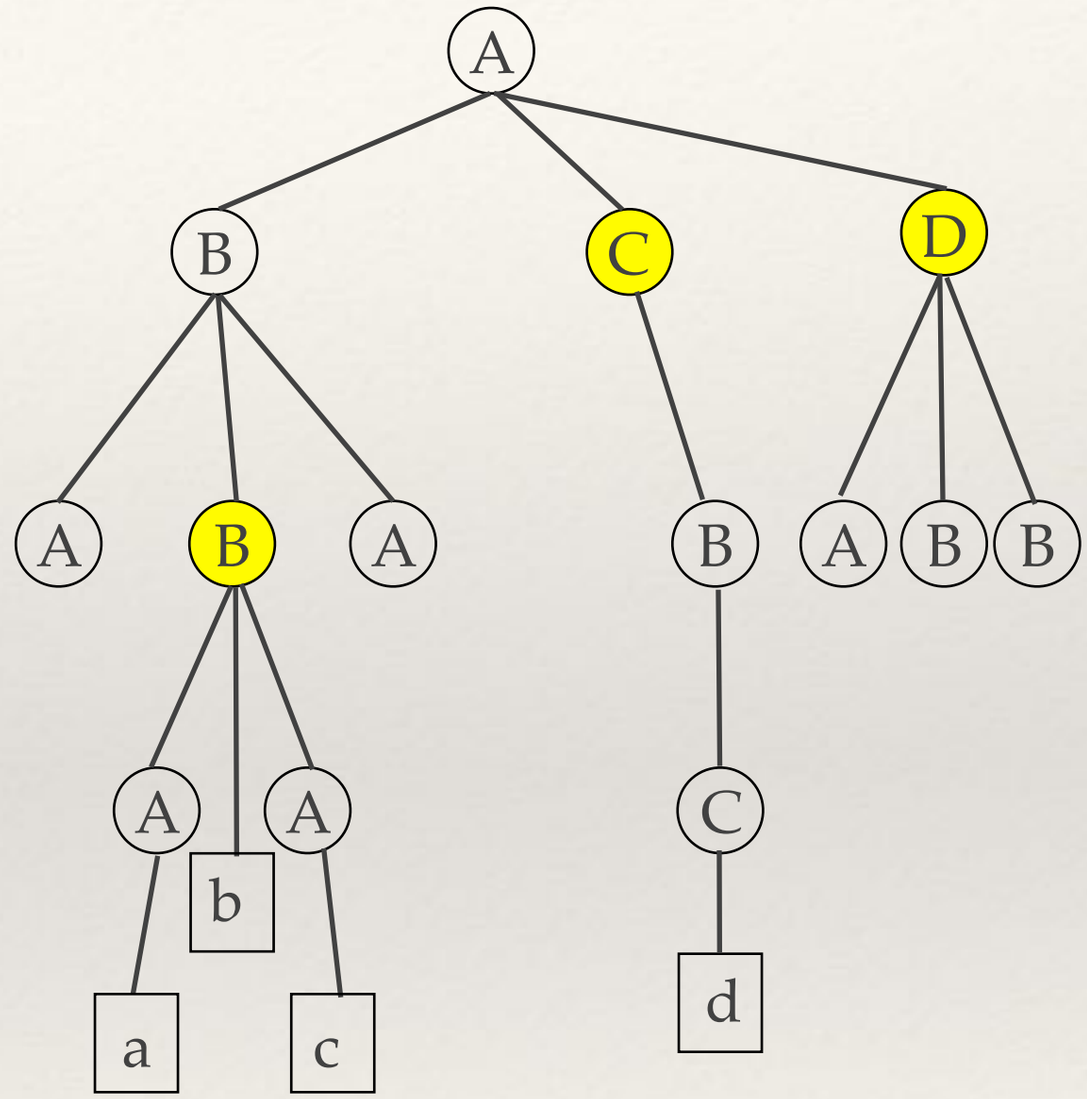
- You can think of a location path as a way to identify a set of (zero, one or more) nodes, called *current nodes*
- Each step modifies the set of current nodes, depending on
  - axis: direction where we should move
  - node_test: select only nodes with a specific name
  - predicate: further filter the nodes according to a certain boolean function

Axis: child (1)

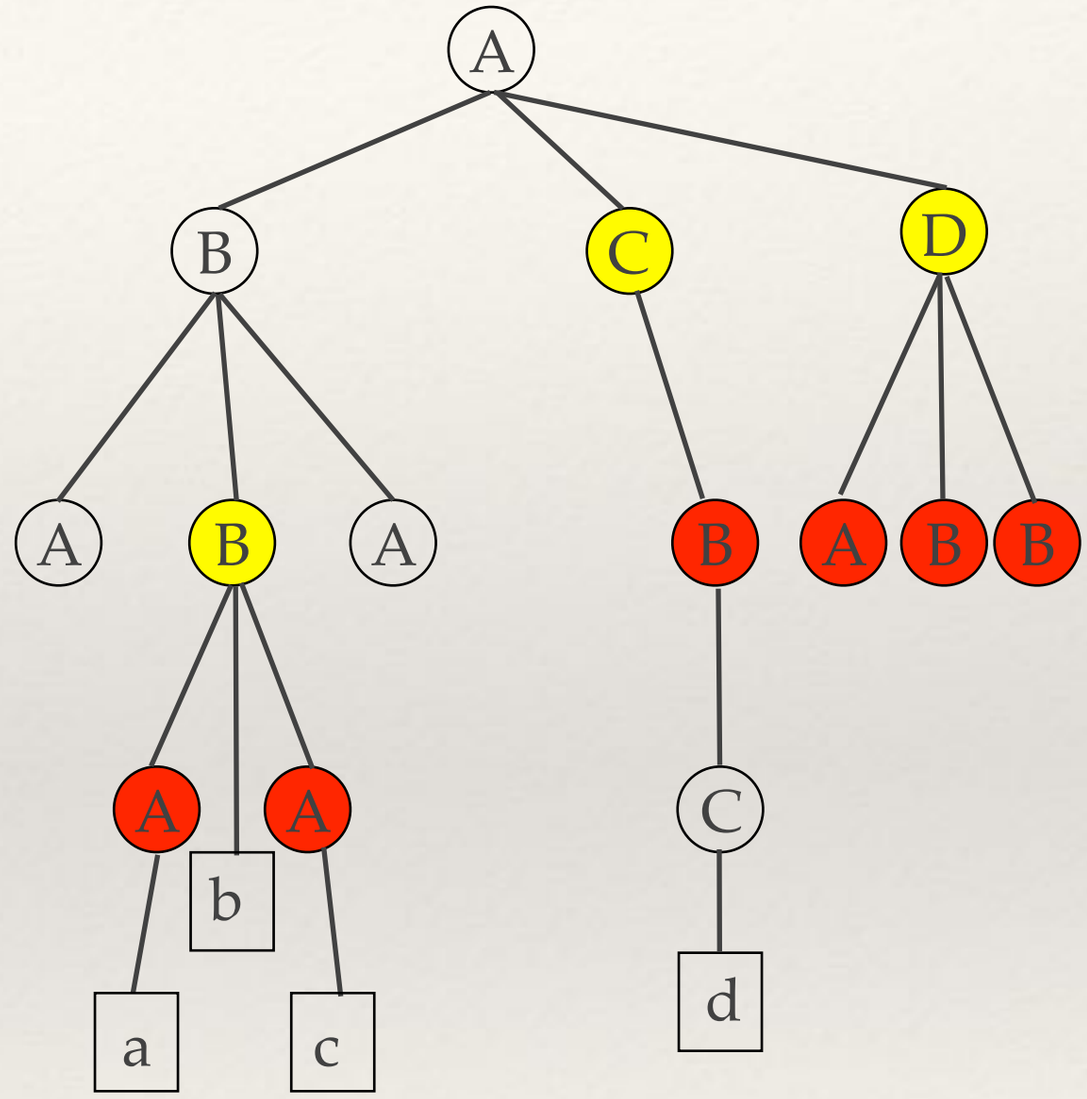


Axis: child (1)

```
Step


child::*

(“select all children nodes”)
```

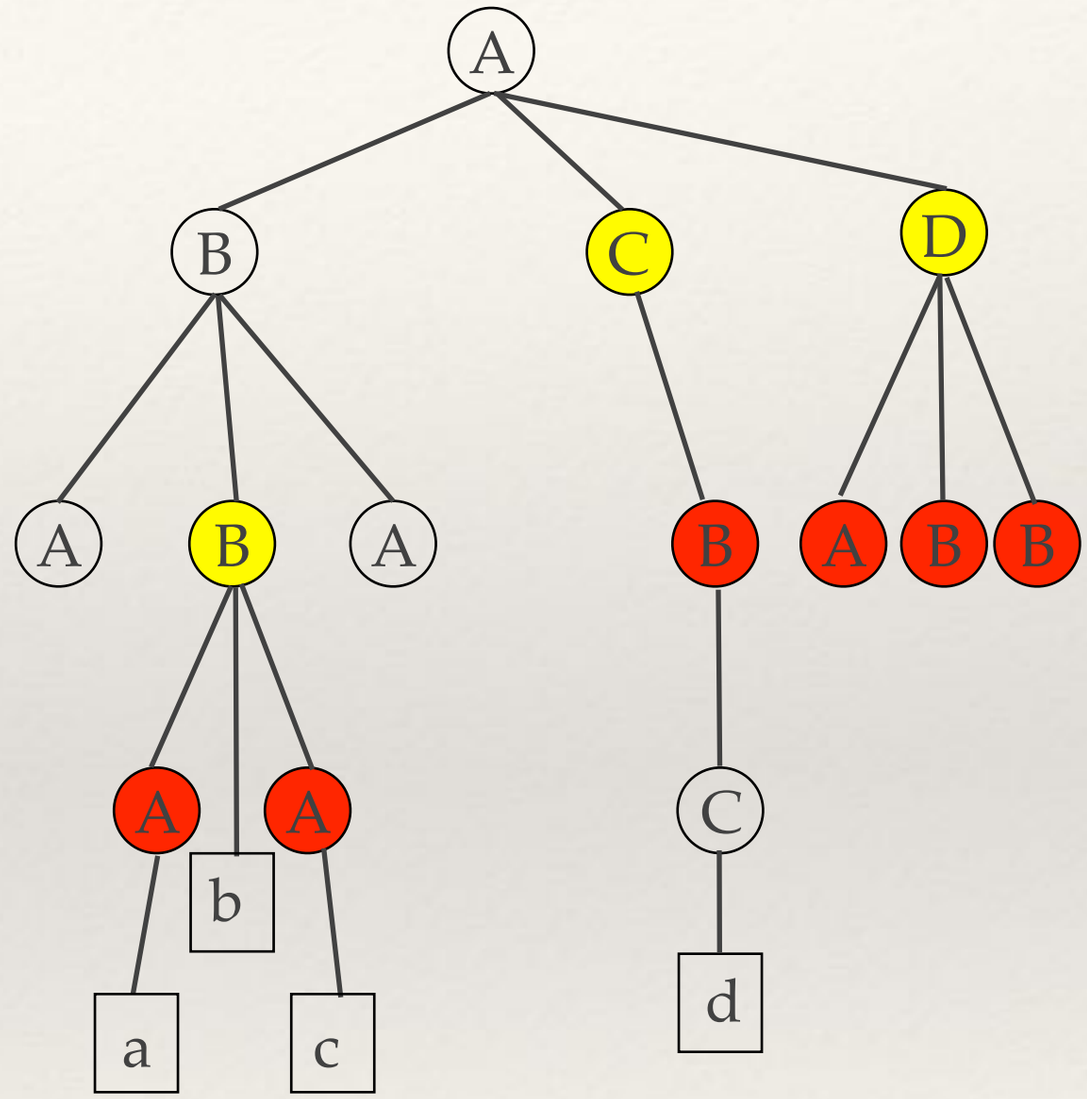


Axis: child (1)

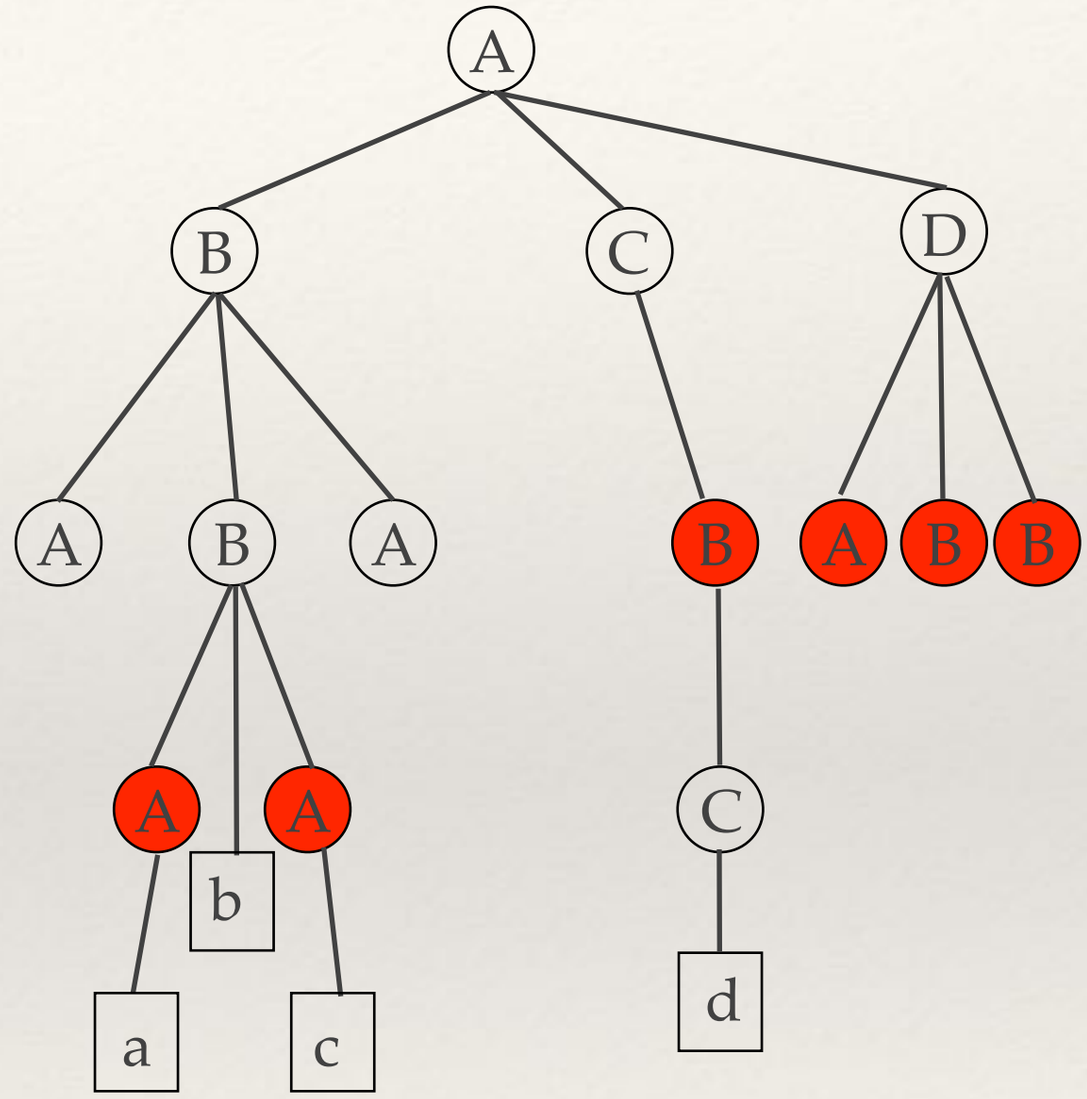


Axis: child (1)

```
Step


child::*

(“select all children nodes”)
```



Axis: child (1)

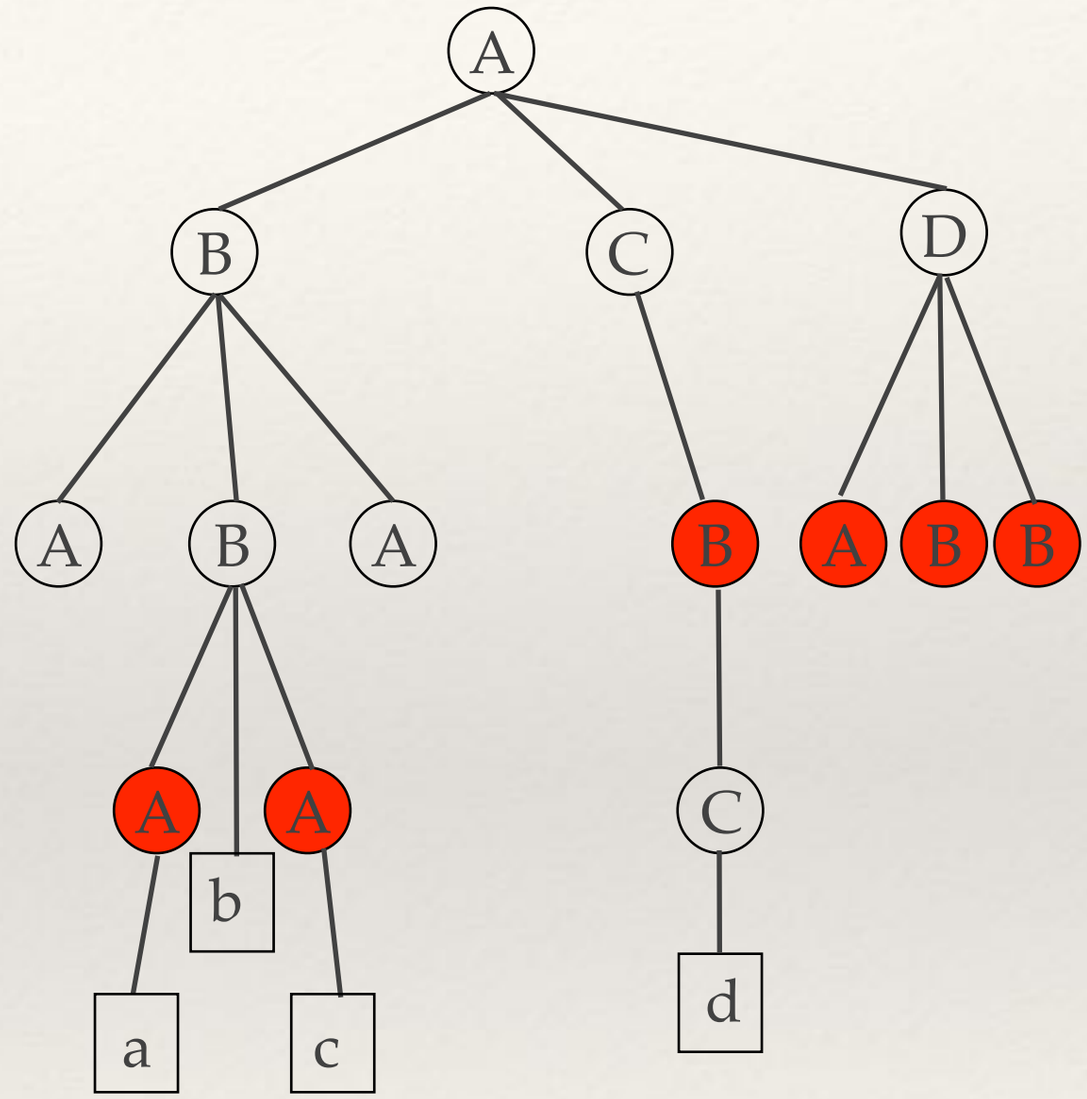


Axis: child (1)

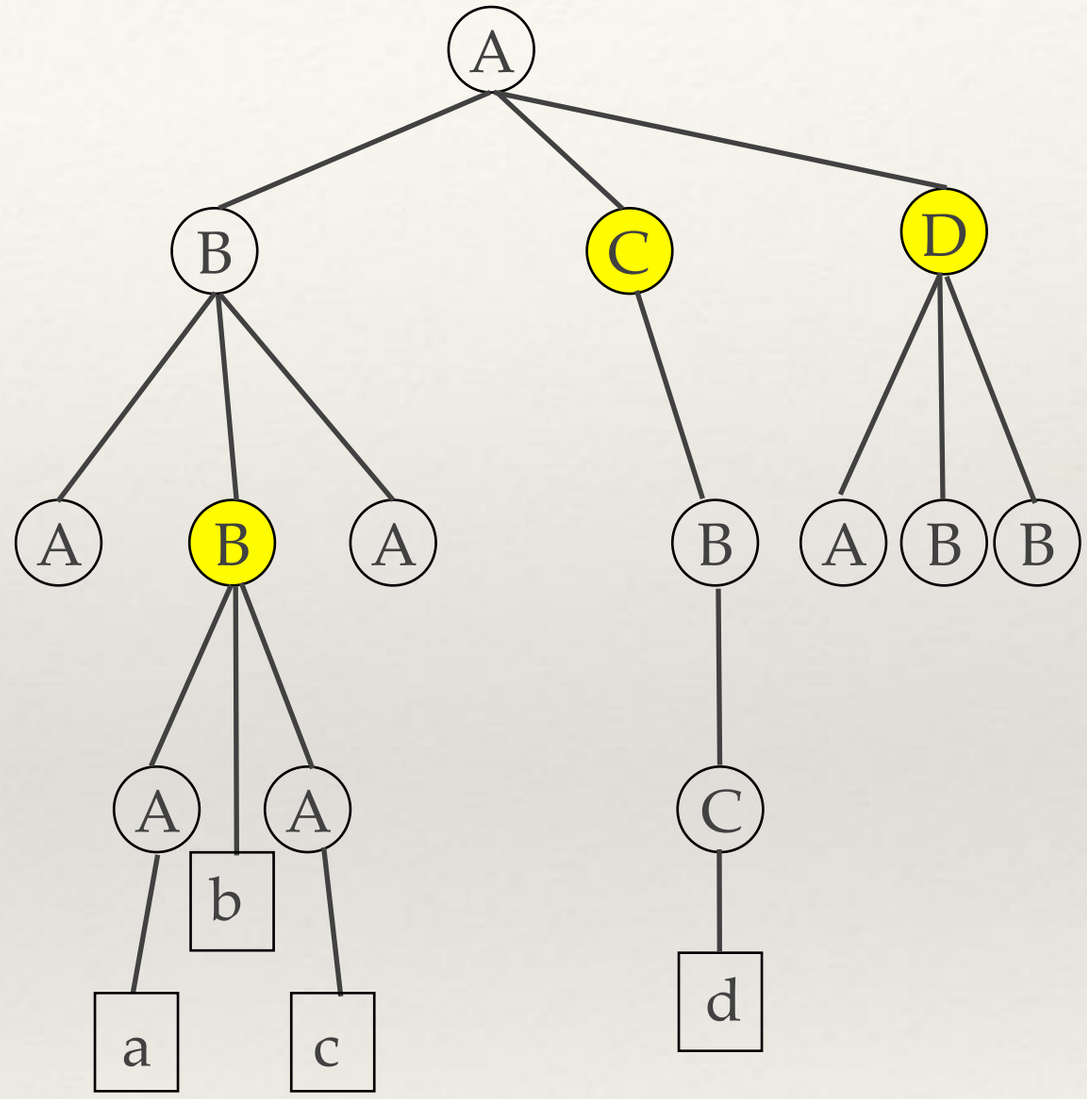
Step

child::*

(“select all children nodes”)

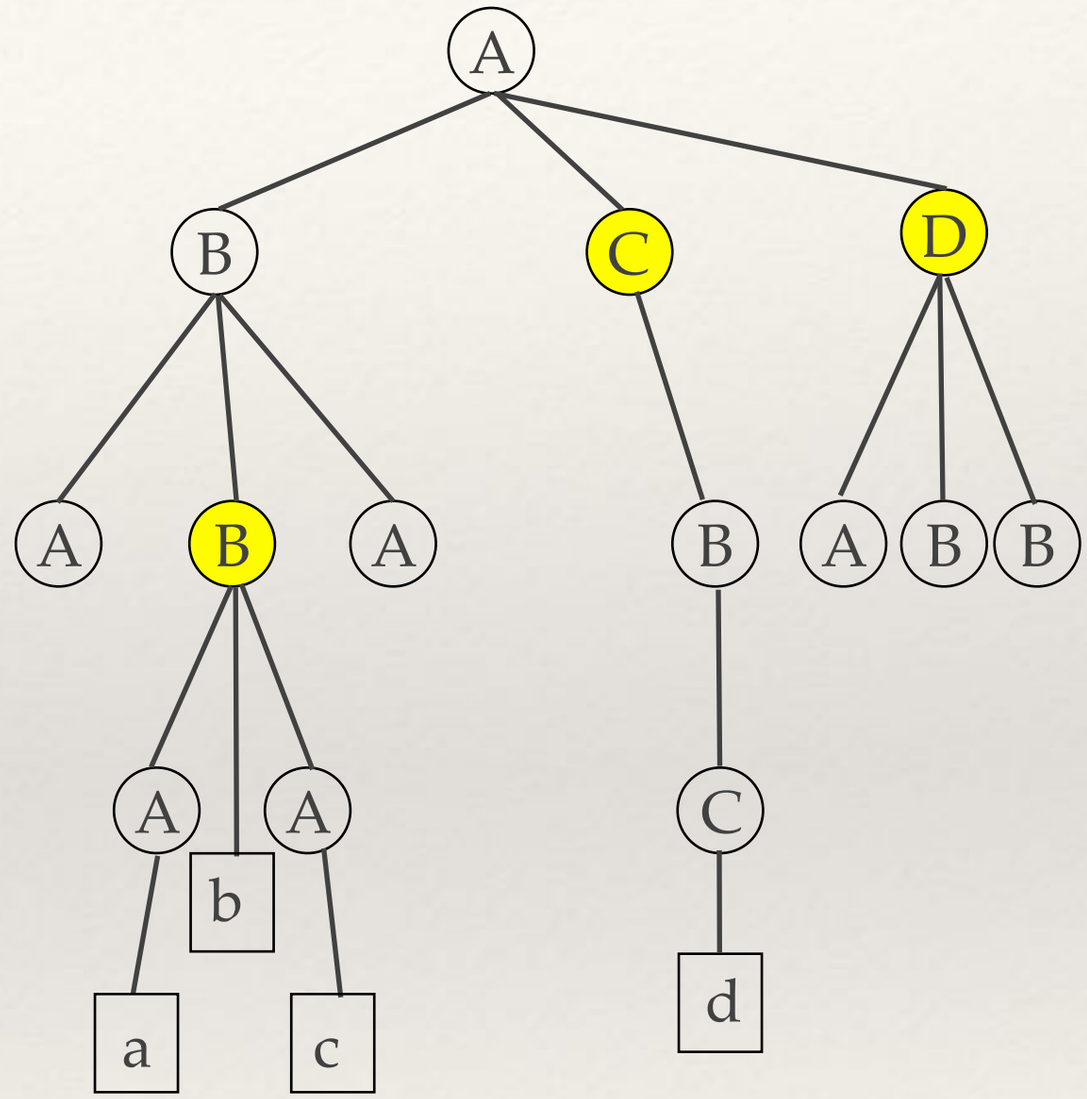


Axis: child (2)

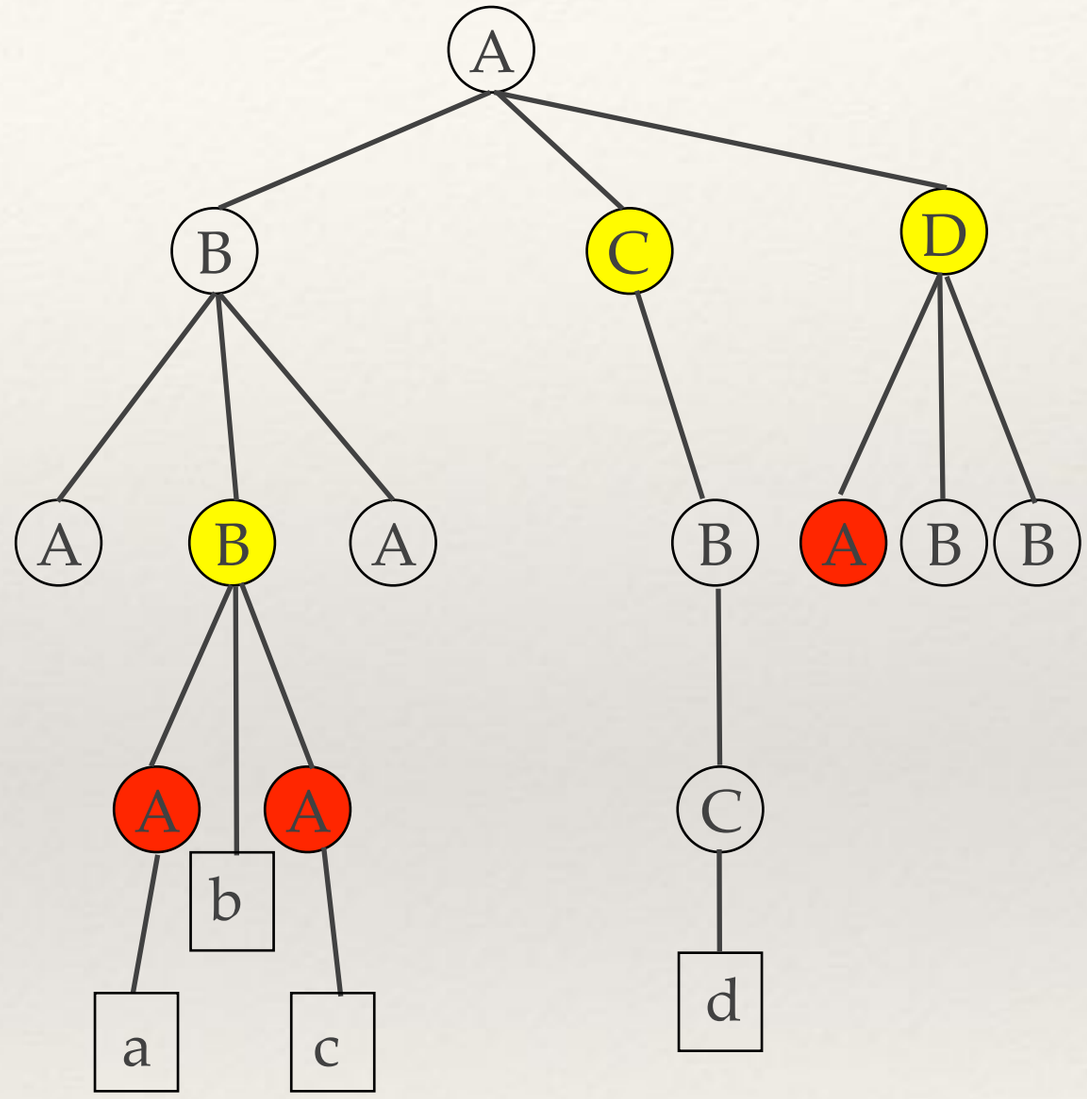


Axis: child (2)

```
Step


child::A

(“select all A children nodes”)
```

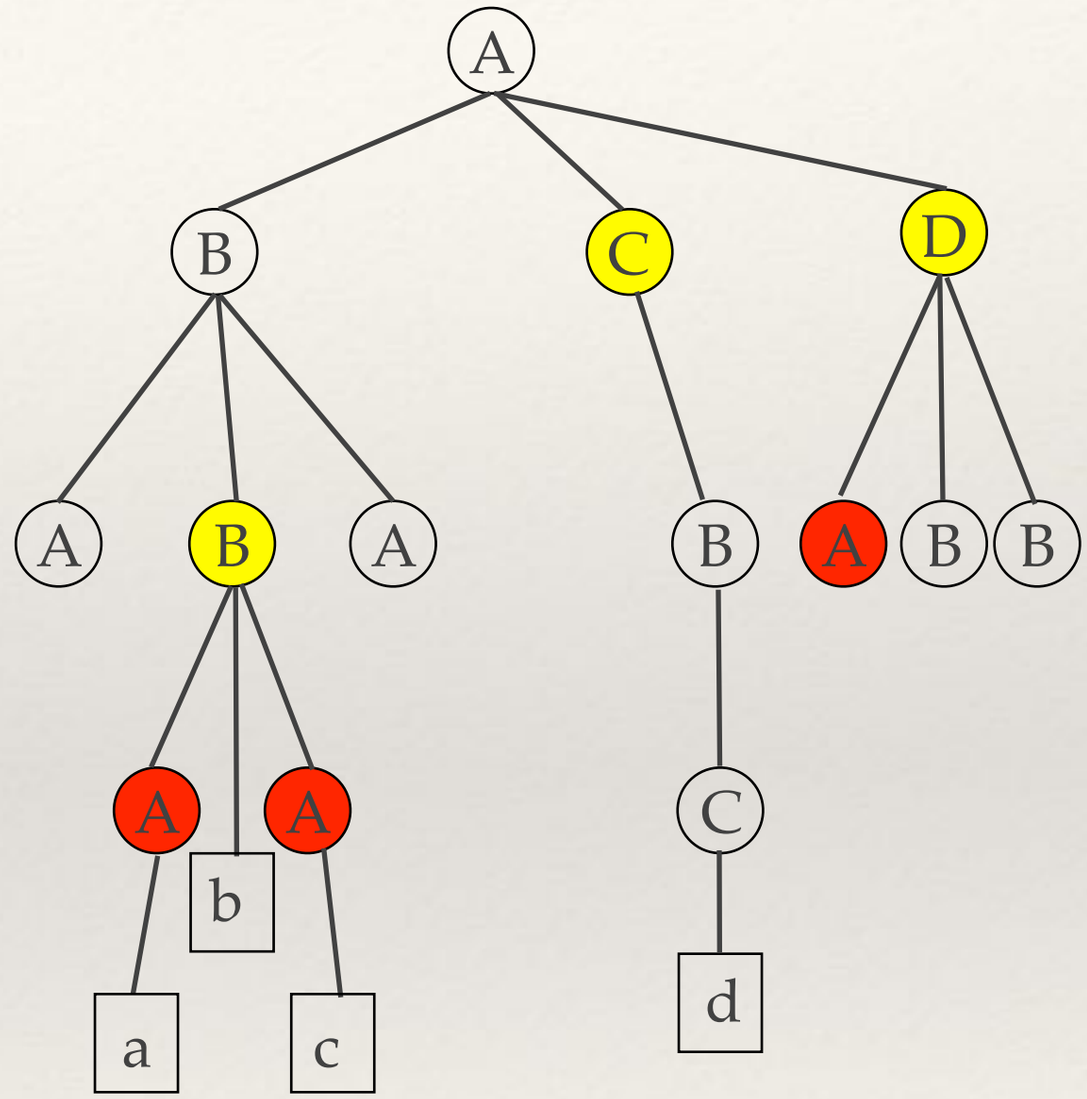


Axis: child (2)

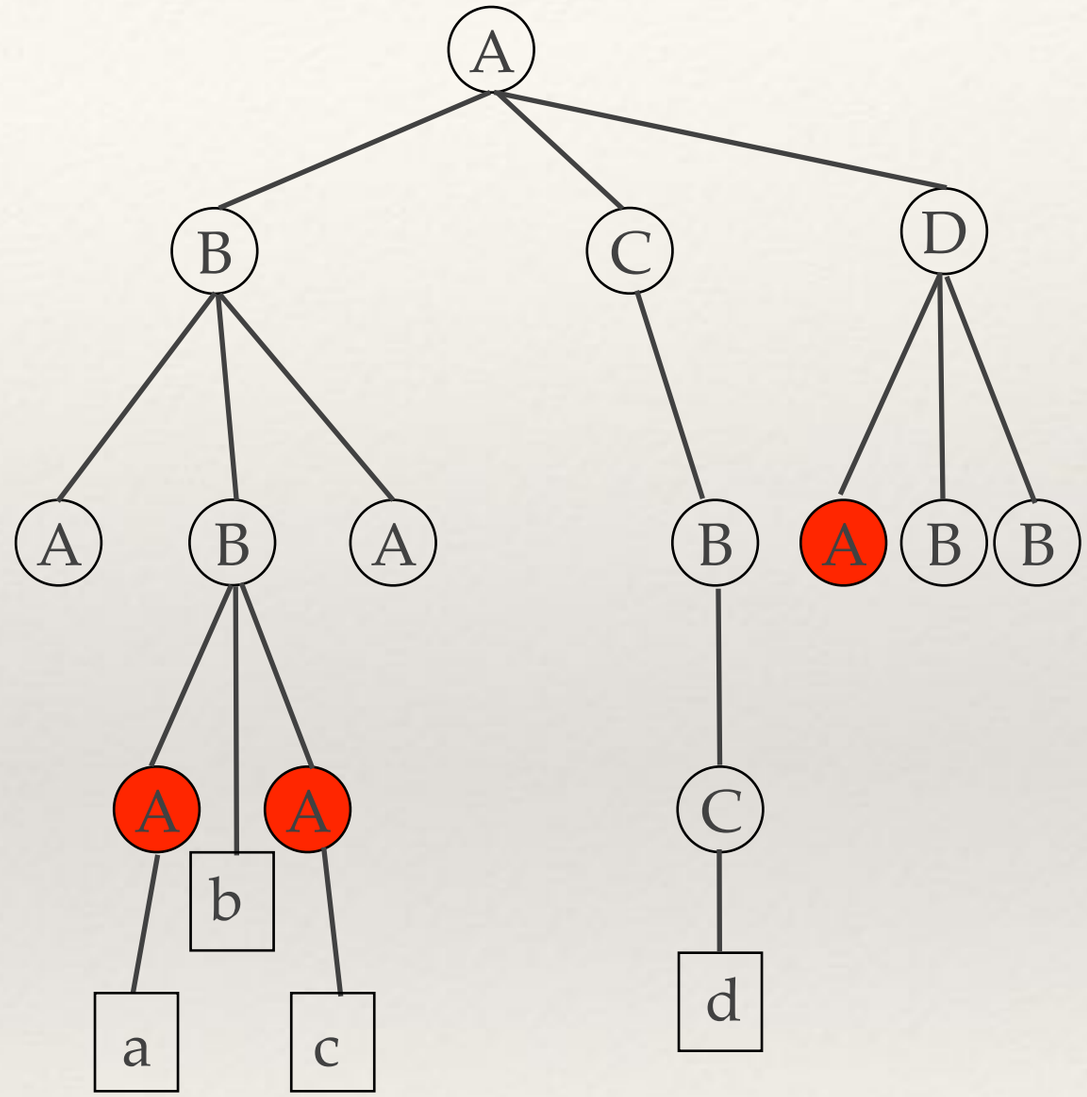


Axis: child (2)

```
Step


child::A

(“select all A children nodes”)
```

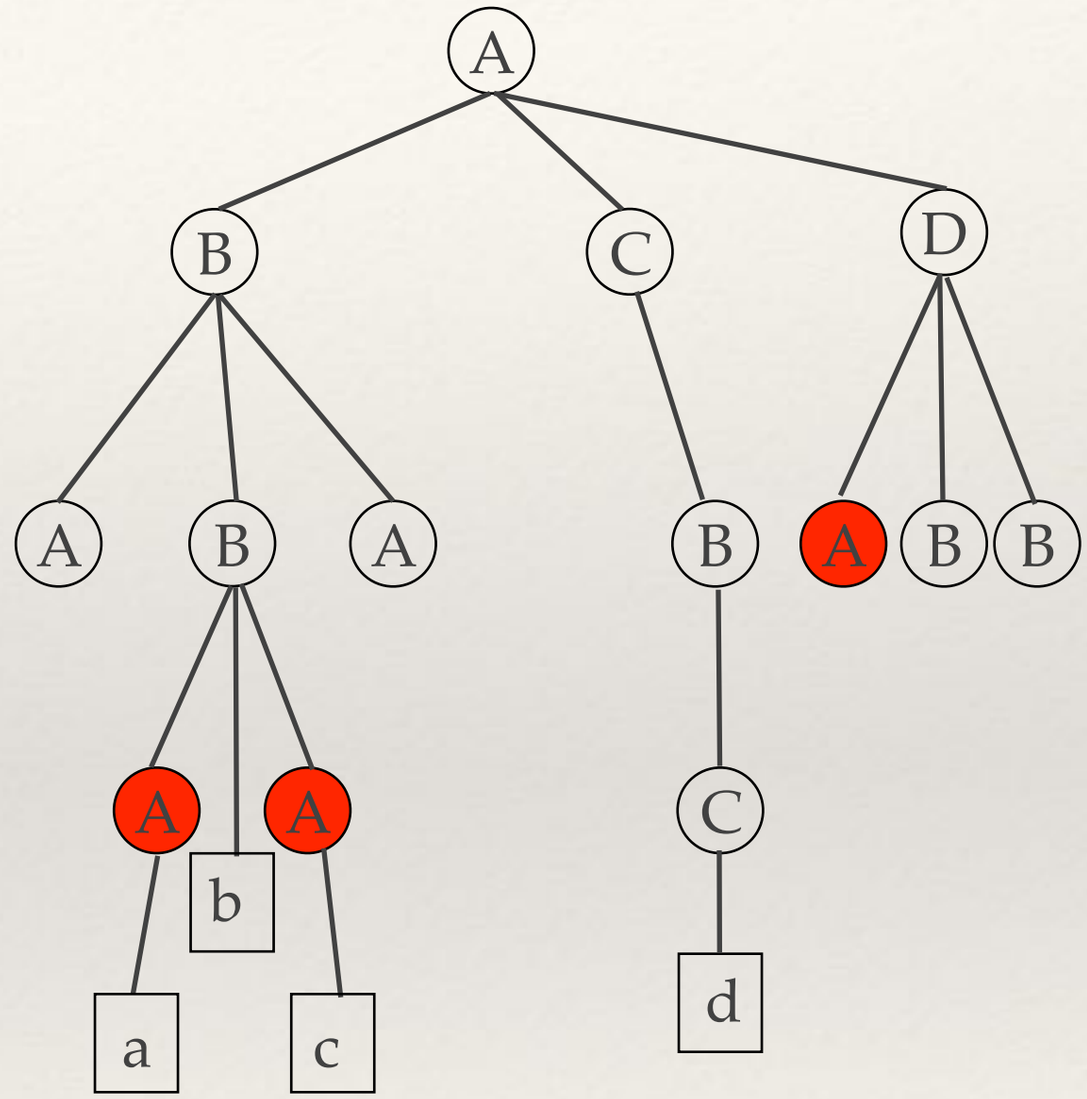


Axis: child (2)

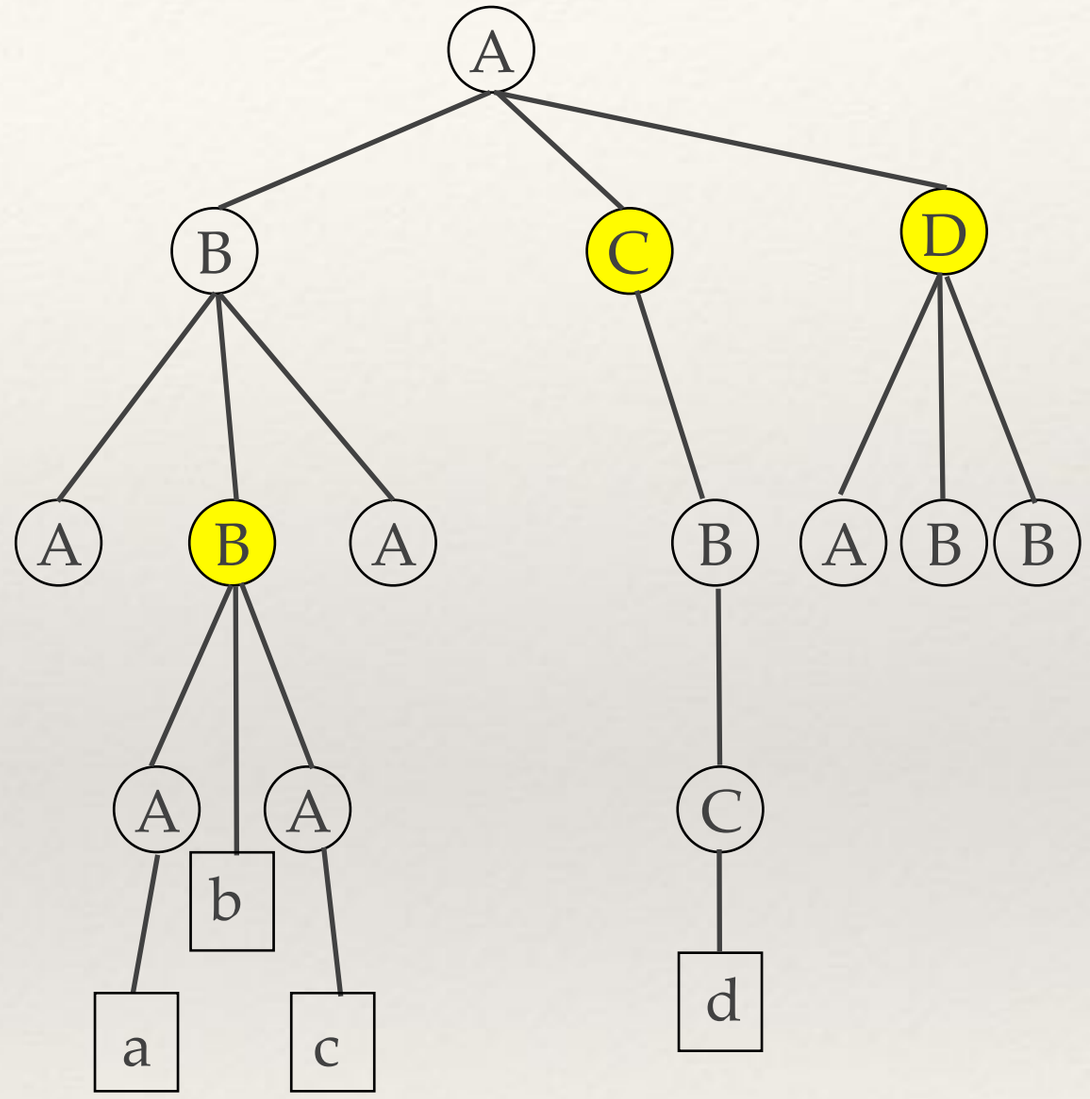


Axis: child (2)

```
Step


child::A

(“select all A children nodes”)
```



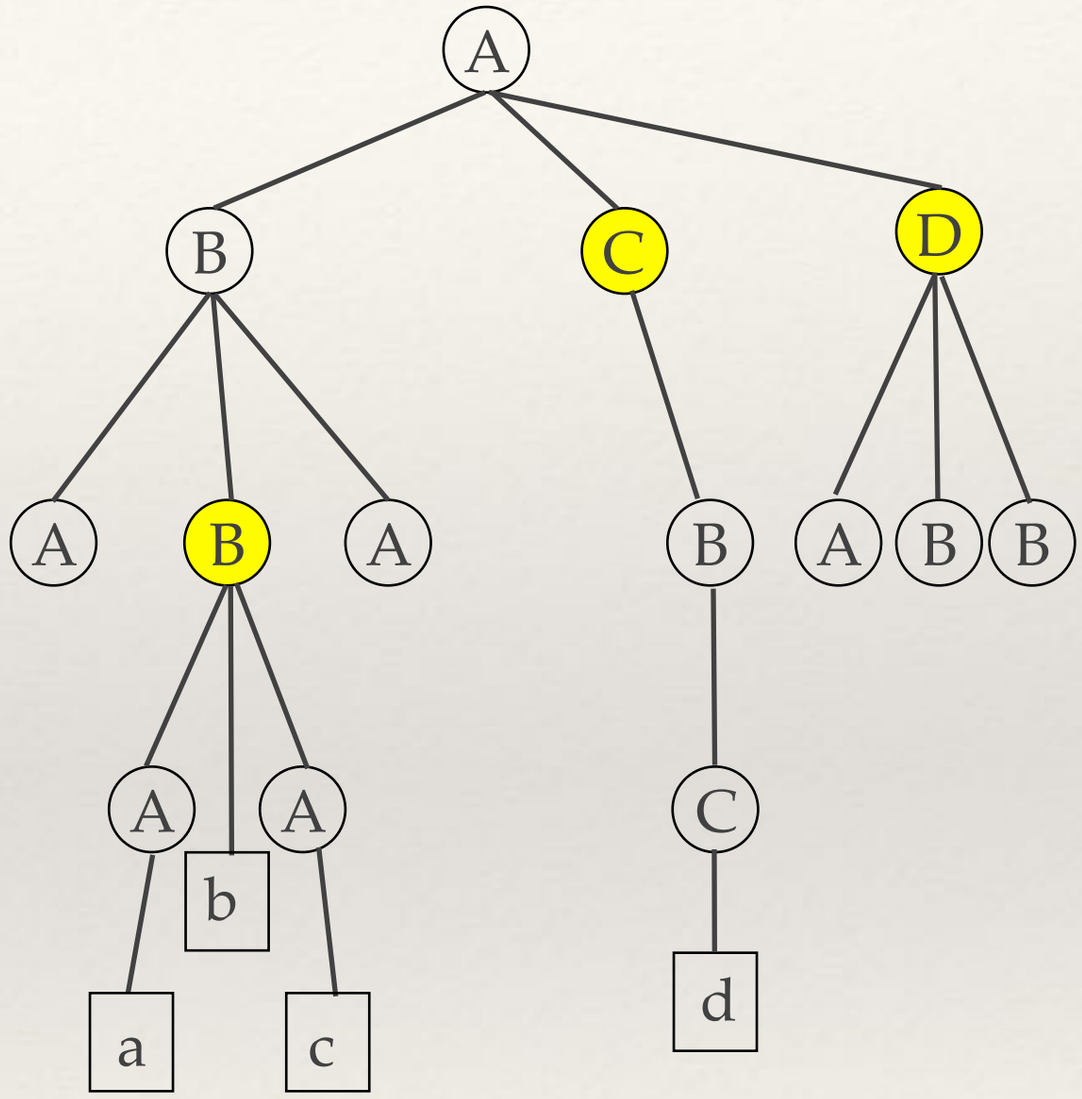
Axis: child (3)



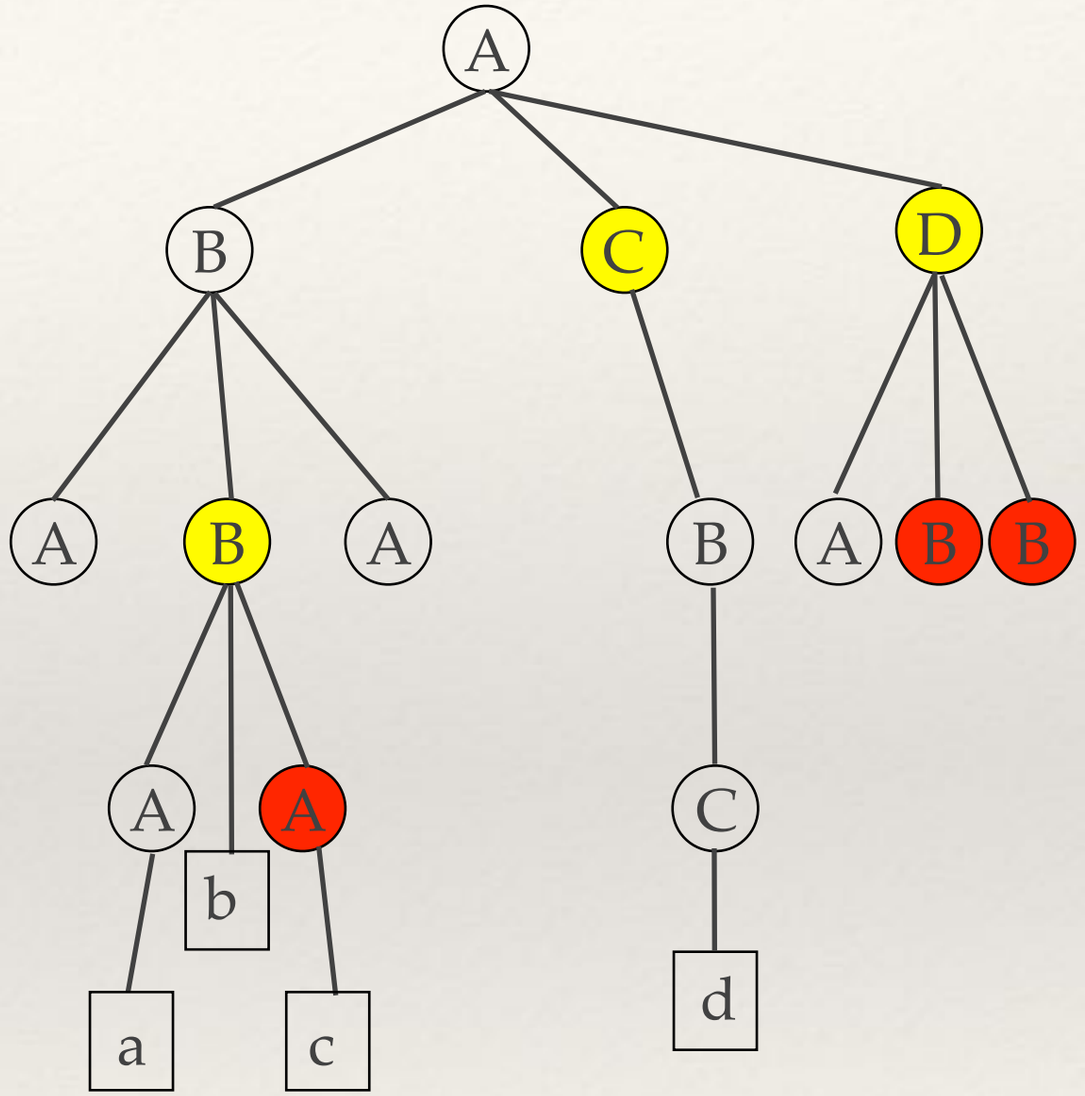
Axis: child (3)

Step

```
child::*[position()>1]

(“select all children nodes in
positions larger than 1 (i.e.,
all except the first child)”)
```



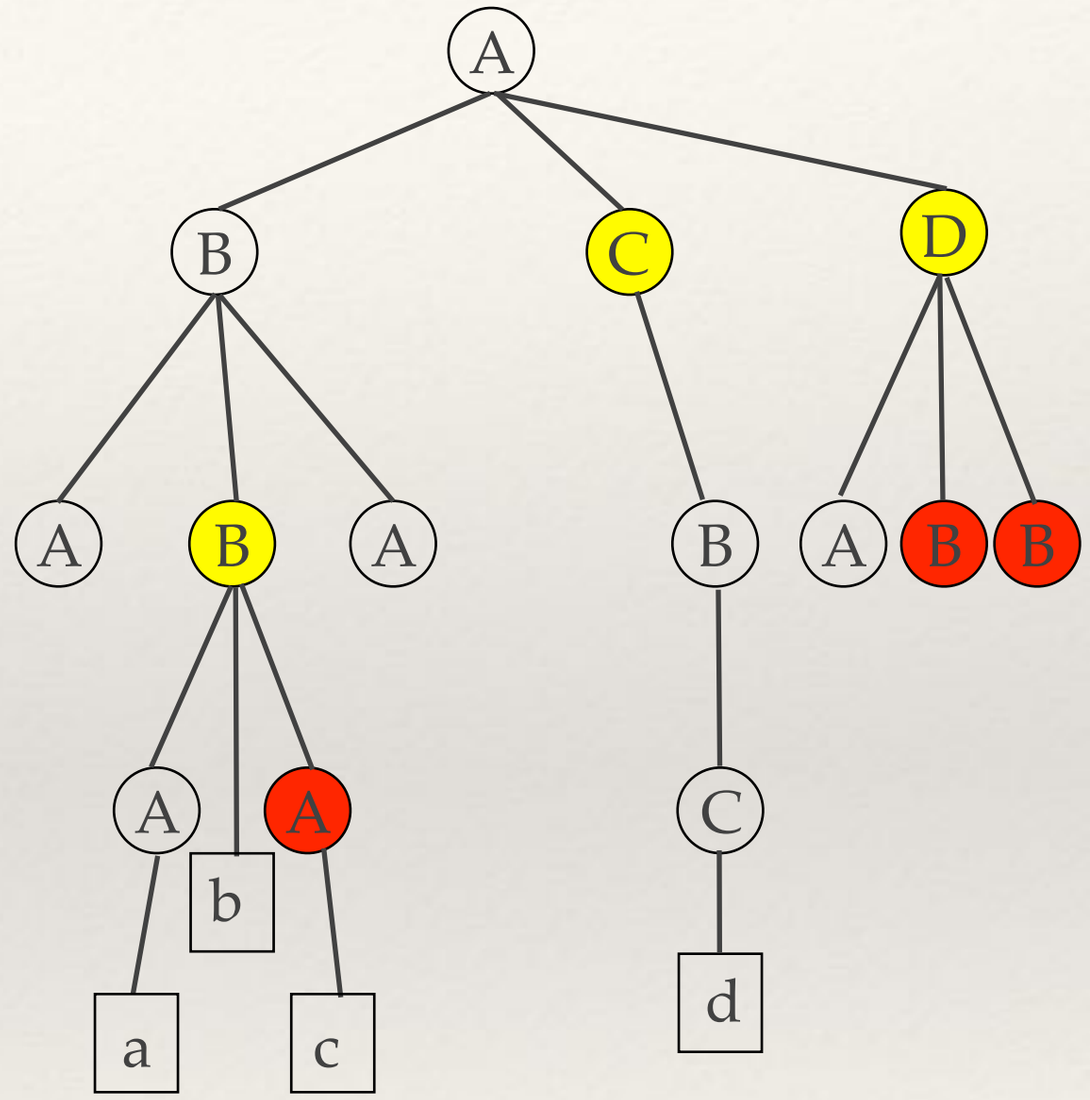
Axis: child (3)



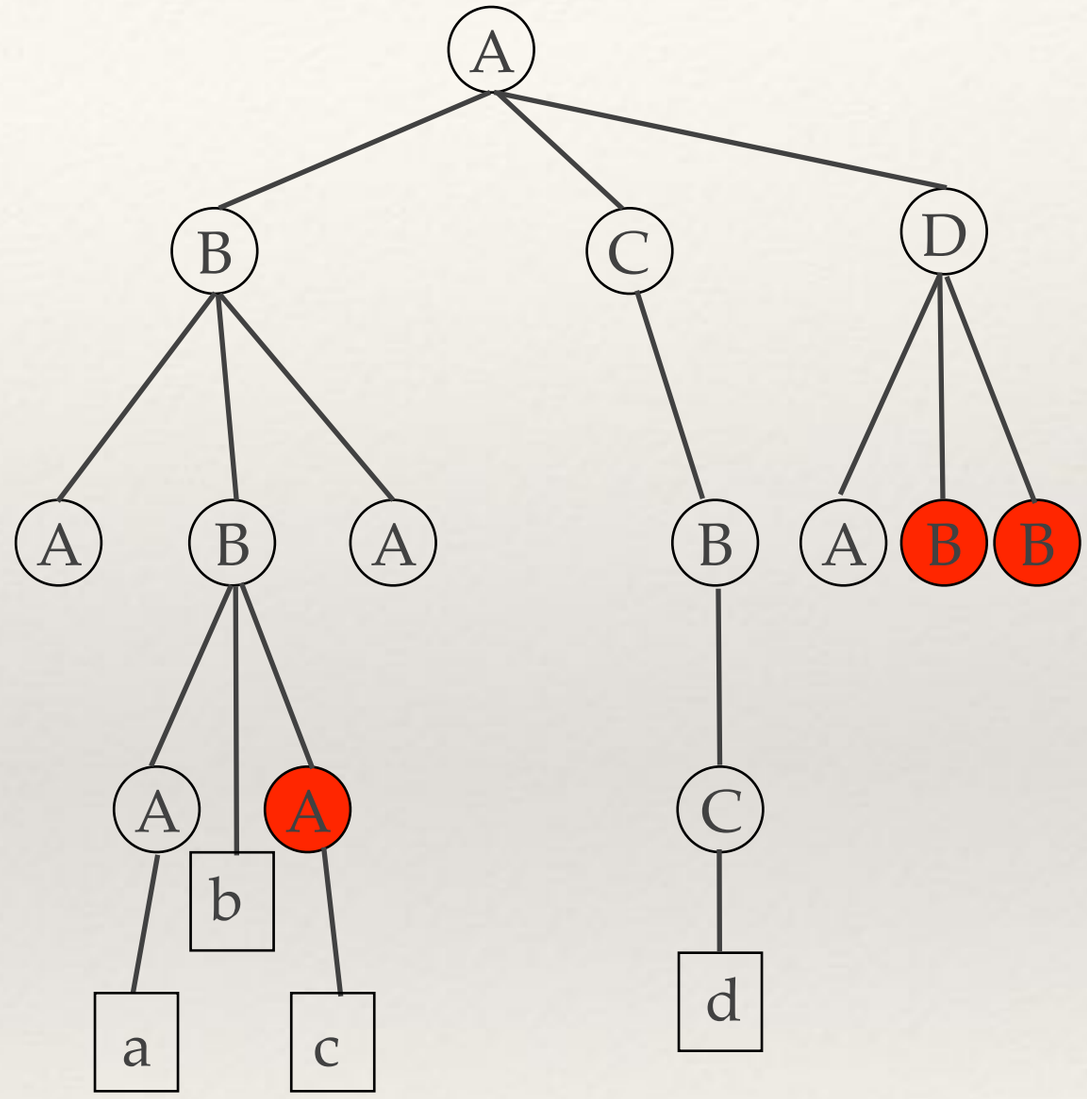
Axis: child (3)

Step

```
child::*[position()>1]

(“select all children nodes in
positions larger than 1 (i.e.,
all except the first child)”)
```



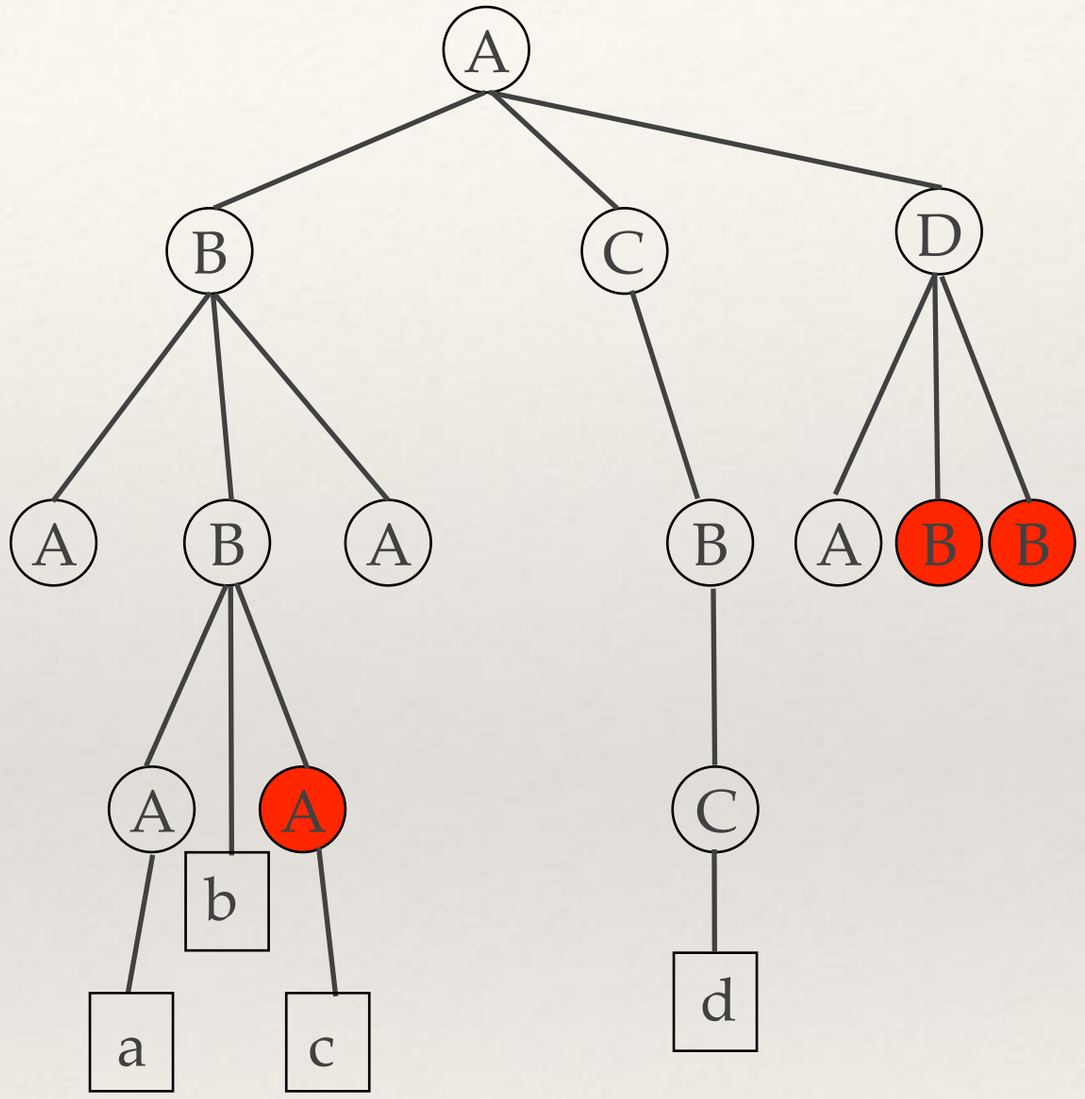
Axis: child (3)



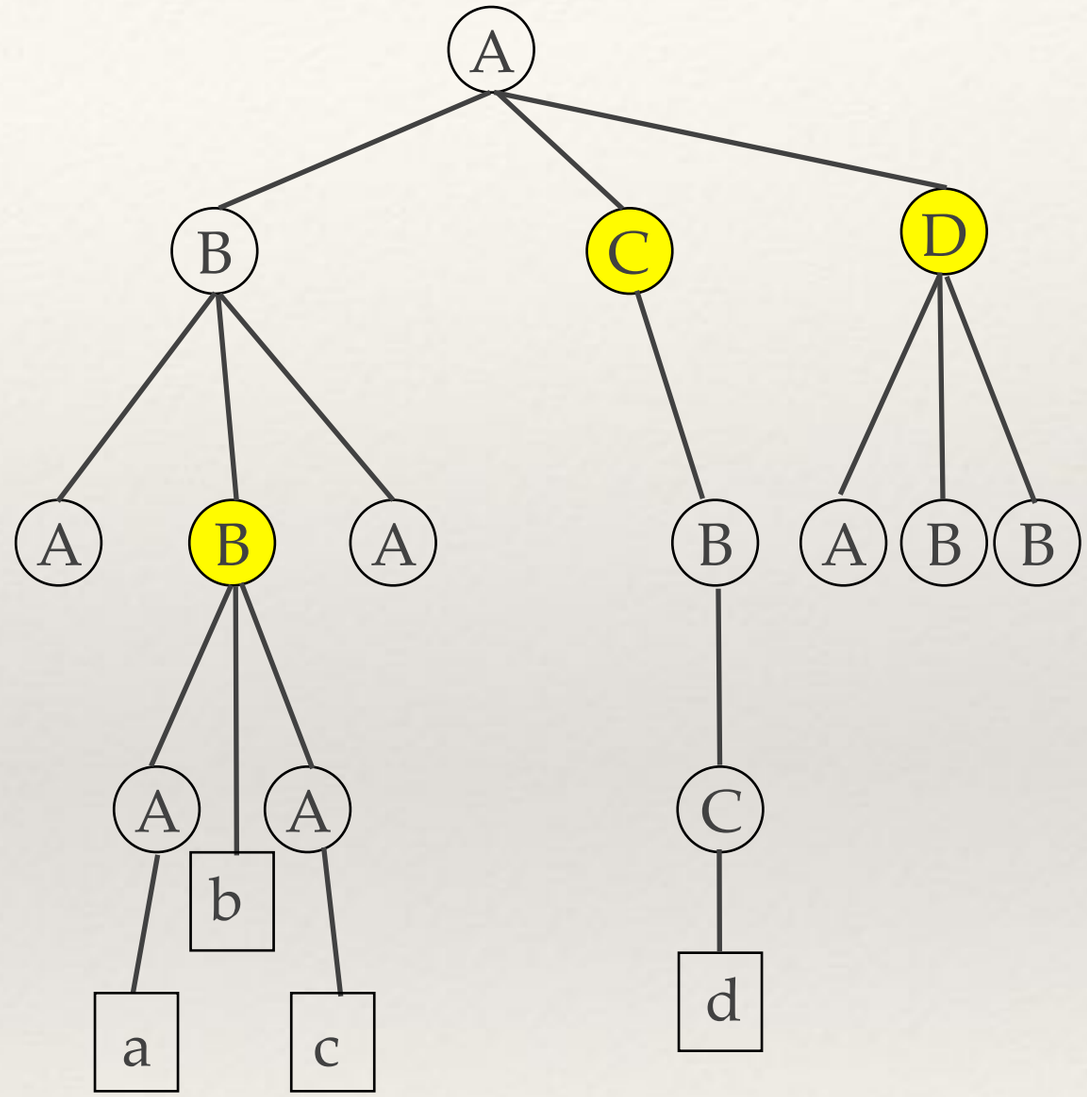
Axis: child (3)

Step

```
child::*[position()>1]

(“select all children nodes in
positions larger than 1 (i.e.,
all except the first child)”)
```

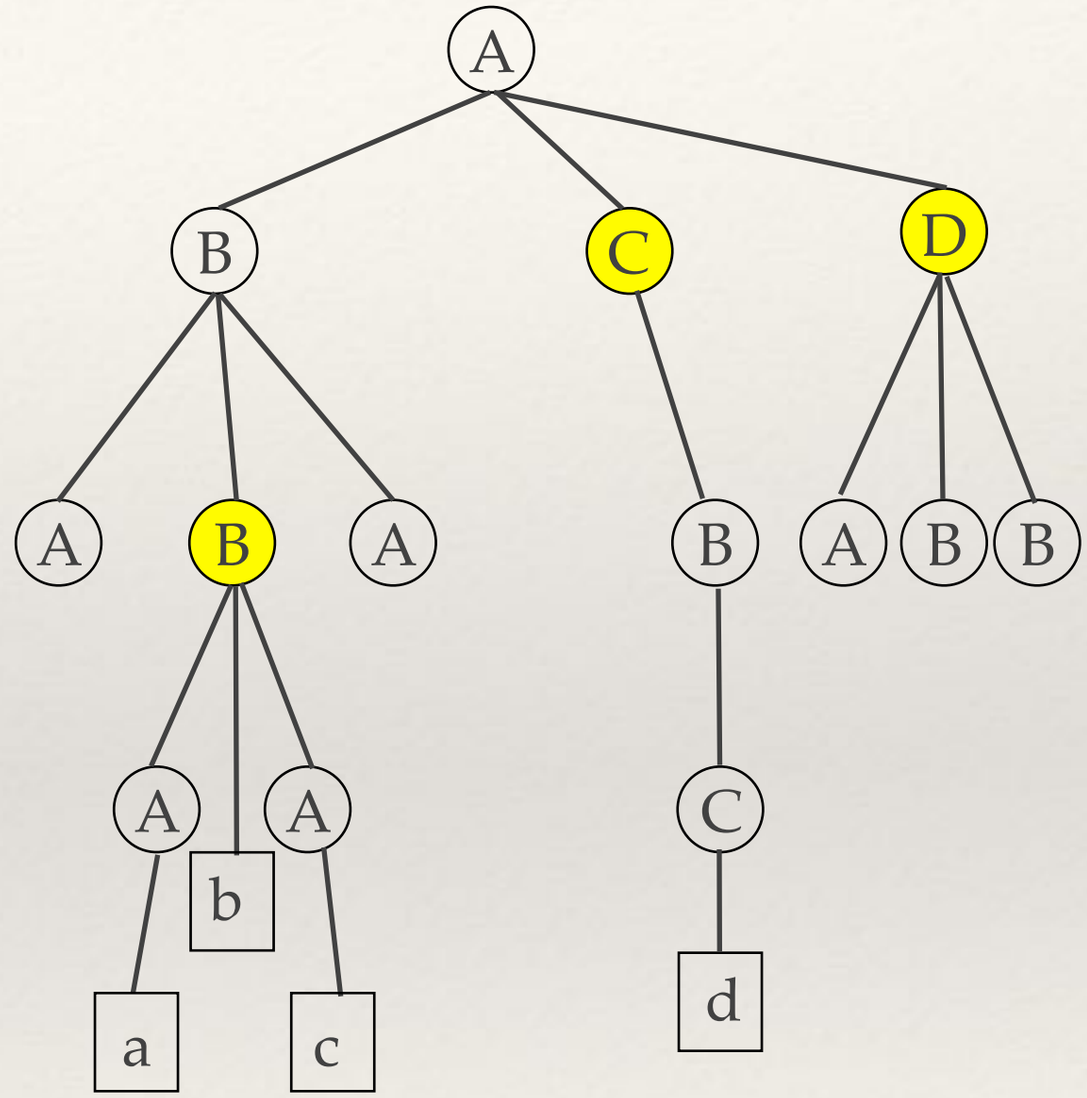


Axis: descendant

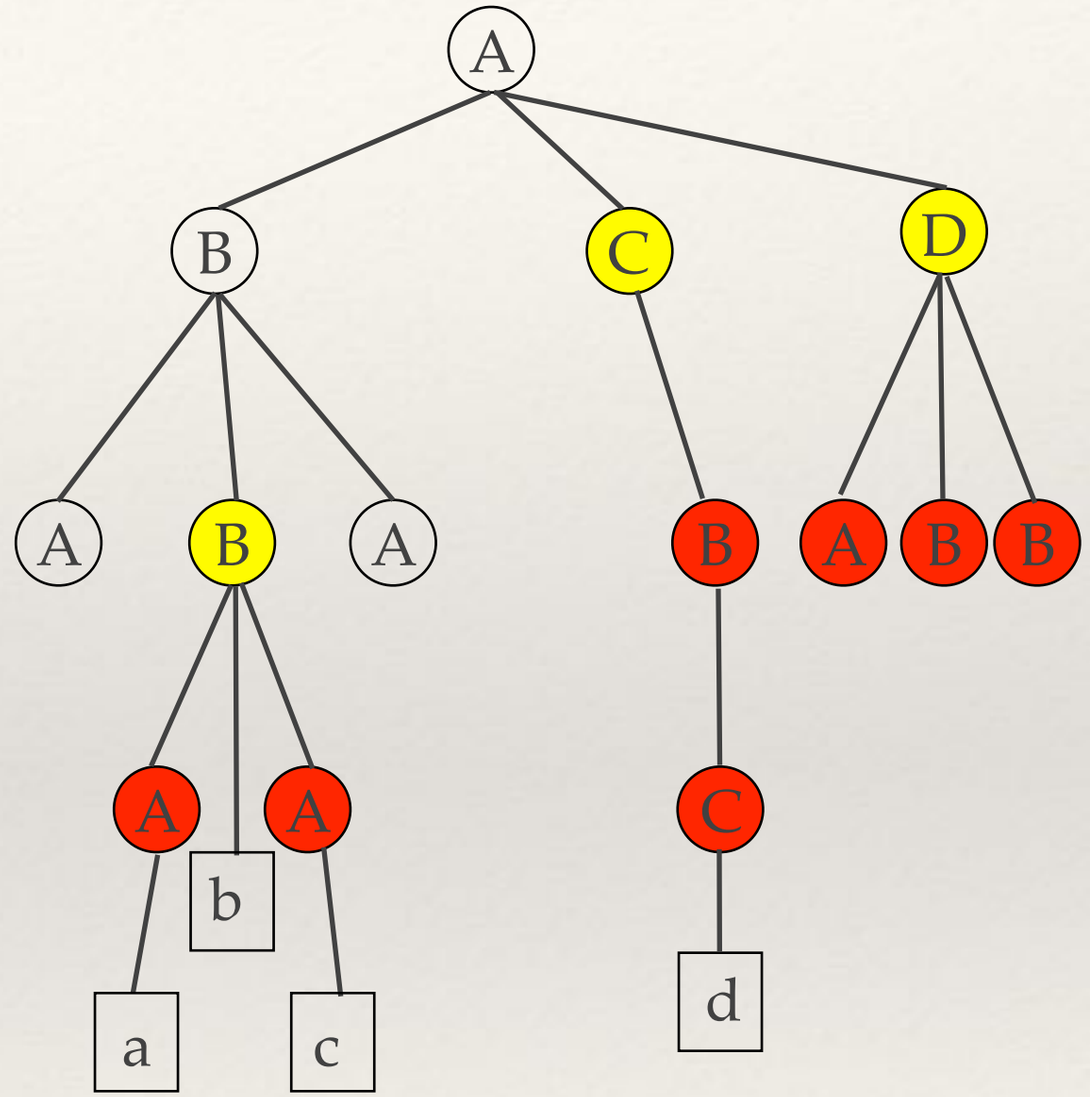


Axis: descendant

```
Step


descendant::*

(“select all descendant nodes”)
```



Axis: descendant

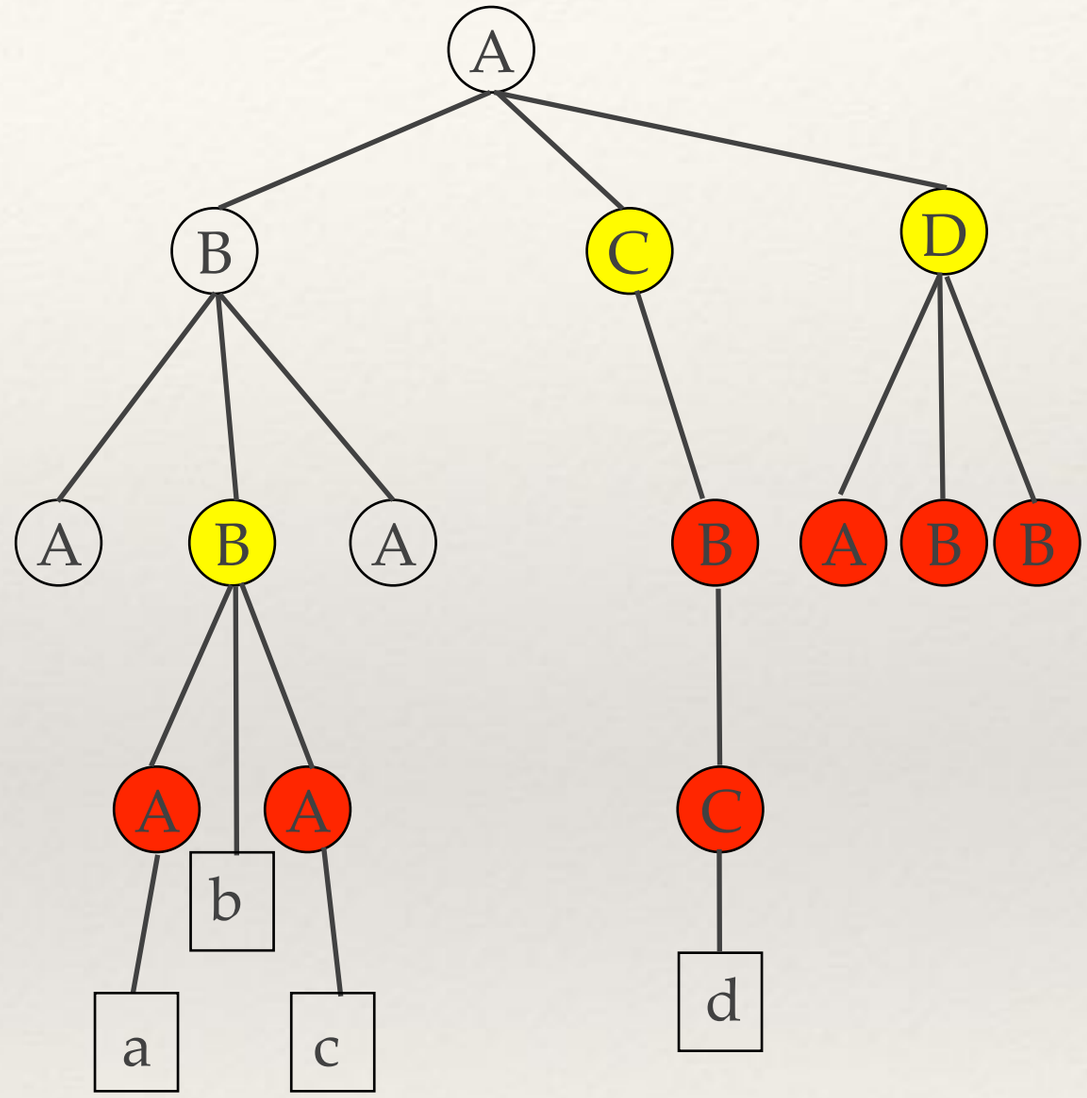


Axis: descendant

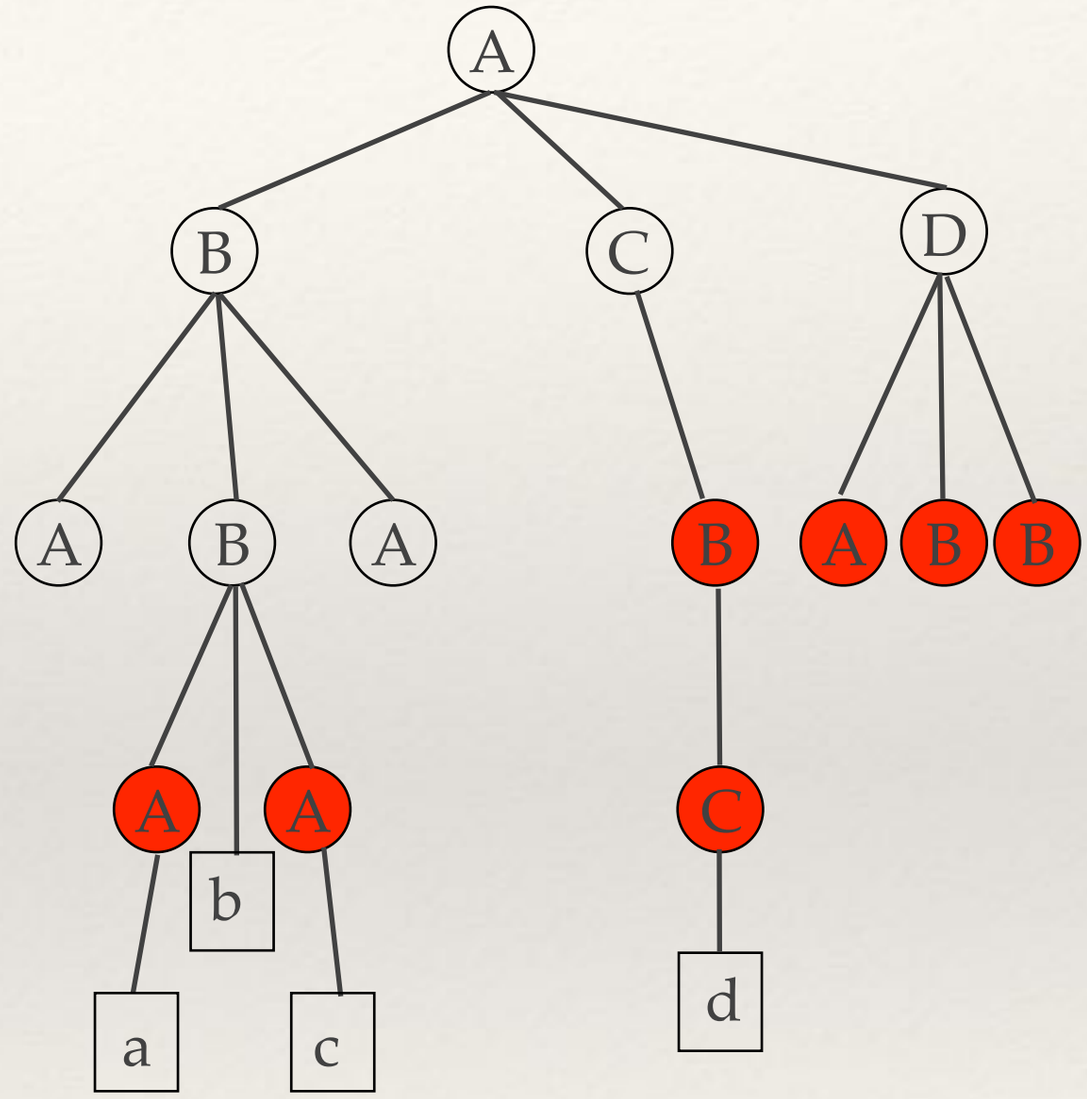
Step

descendant::*

(“select all descendant nodes”)



Axis: descendant

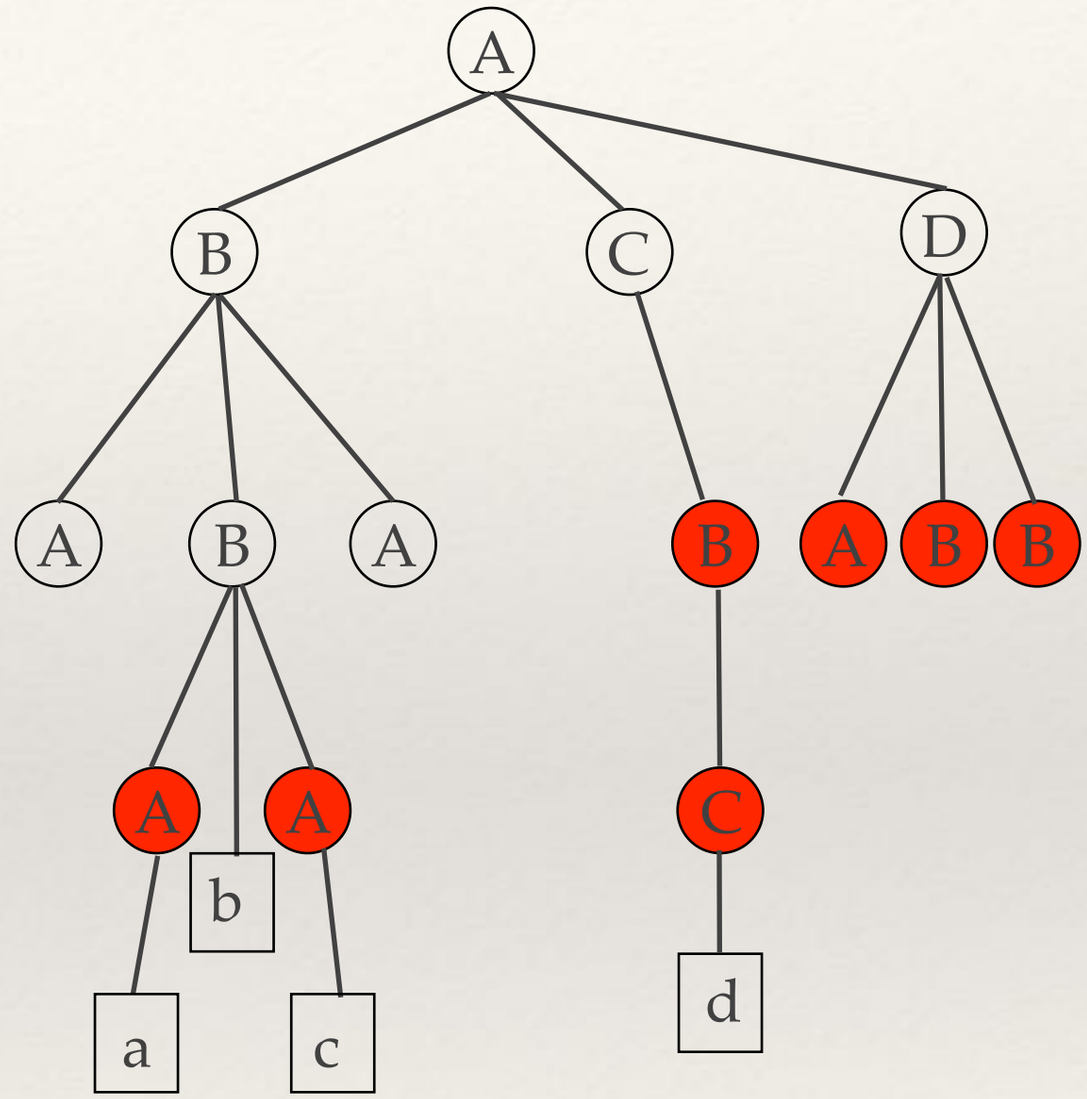


Axis: descendant

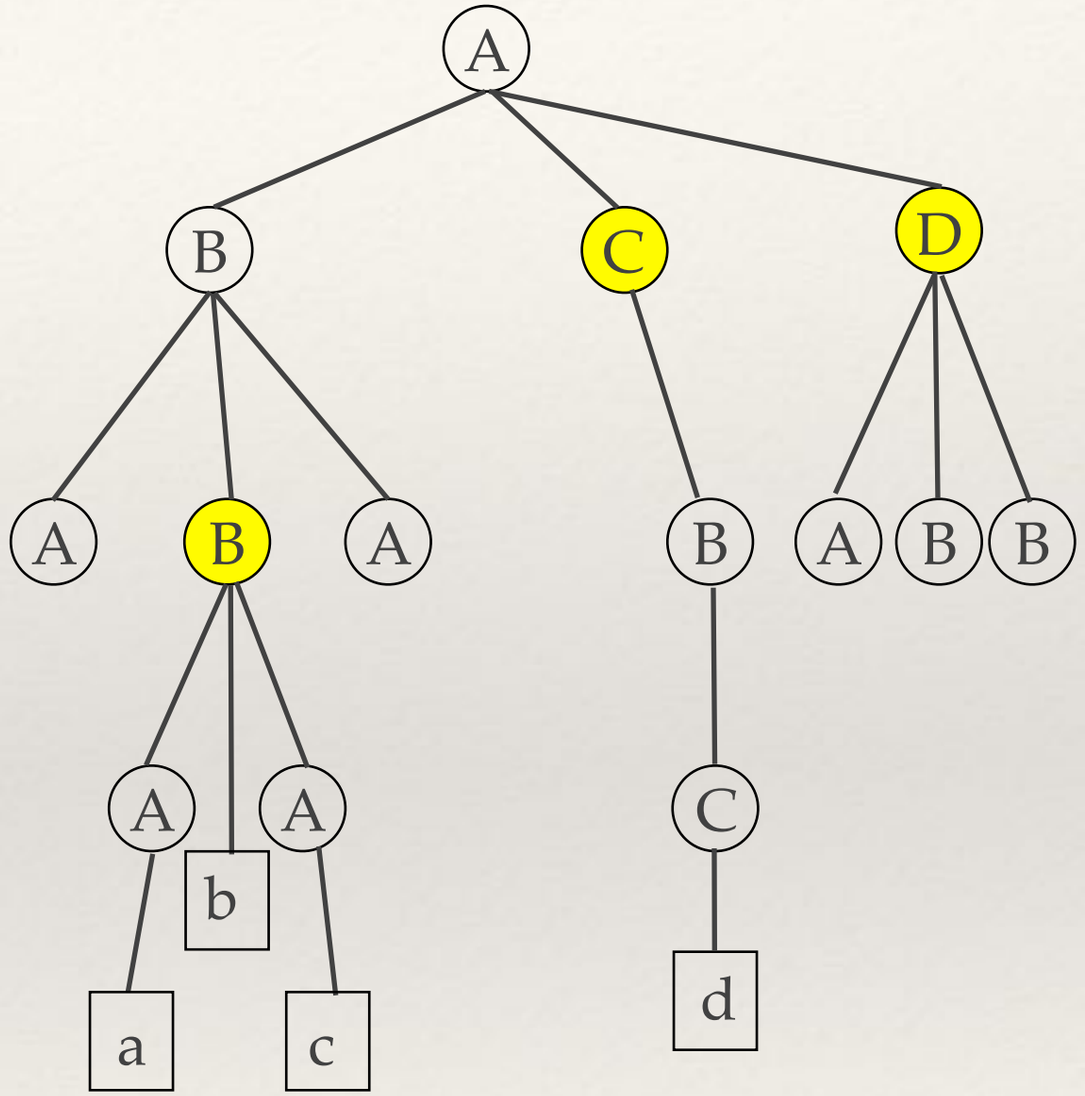
Step

descendant::*

(“select all descendant nodes”)



Axis: ancestor

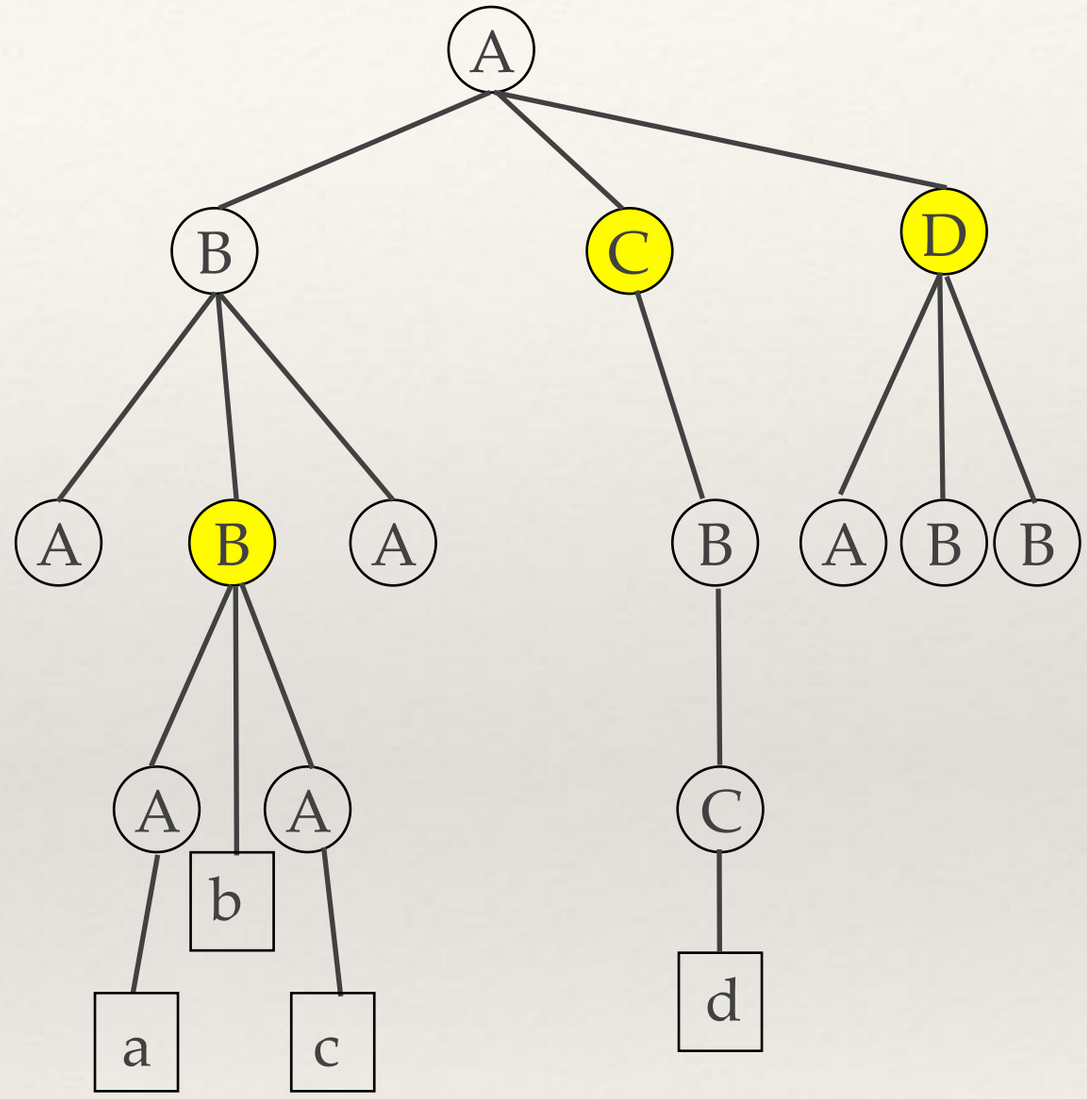


Axis: ancestor

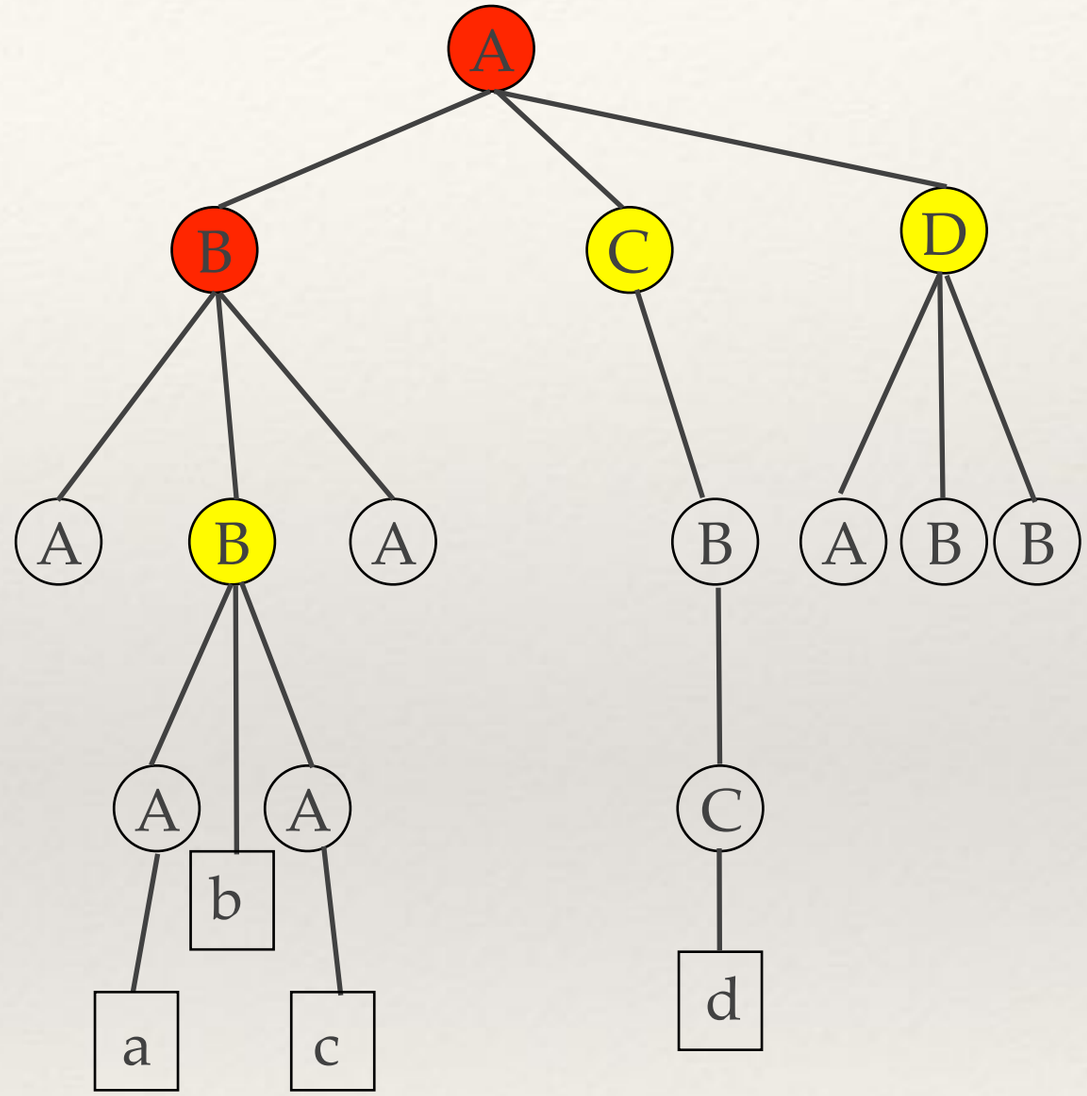
Step

ancestor::*

(“select all descendant nodes, including the current node”)

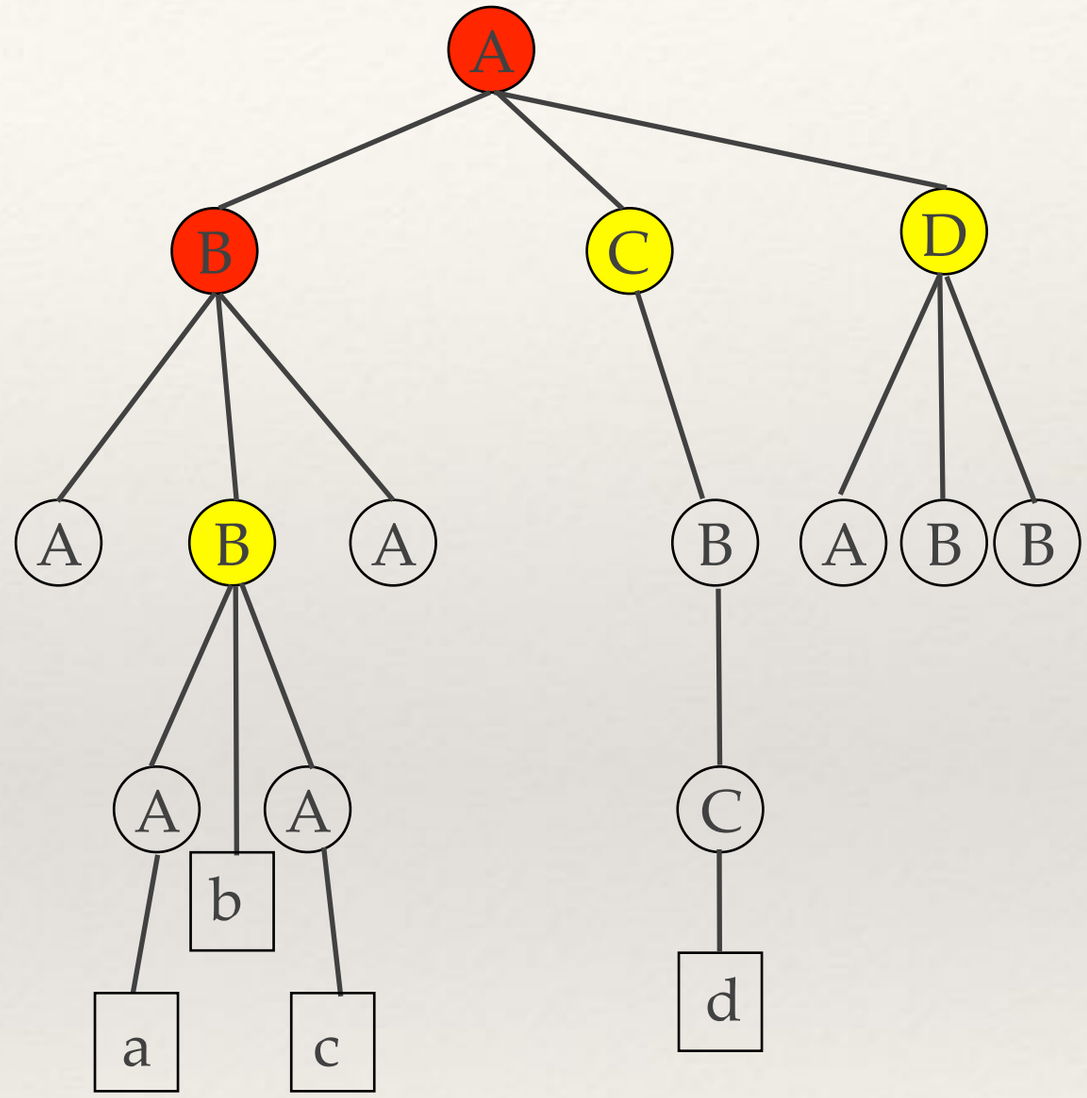


Axis: ancestor

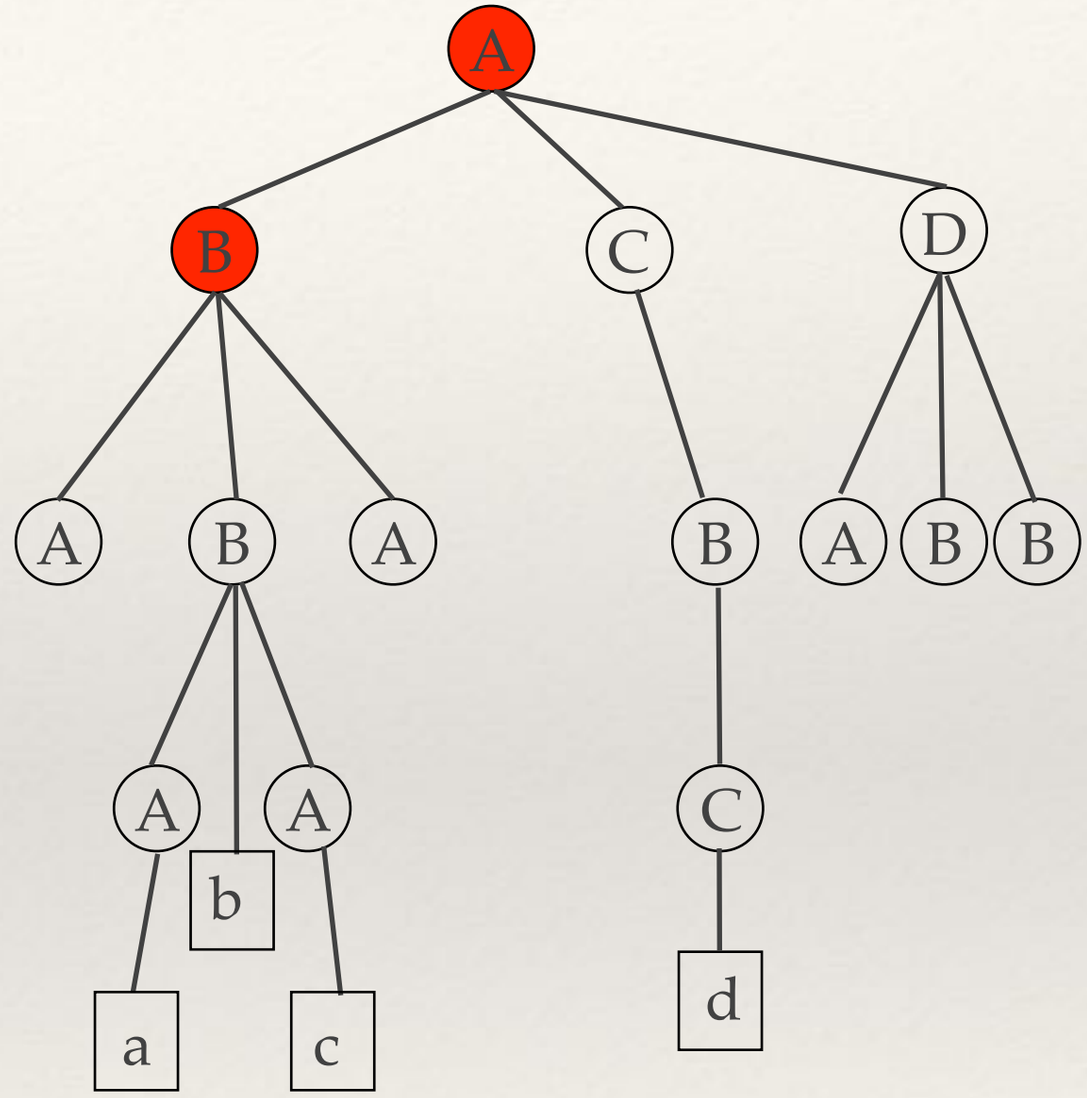


Axis: ancestor

```
Step


ancestor::*

(“select all ancestor nodes”)
```



Axis: ancestor

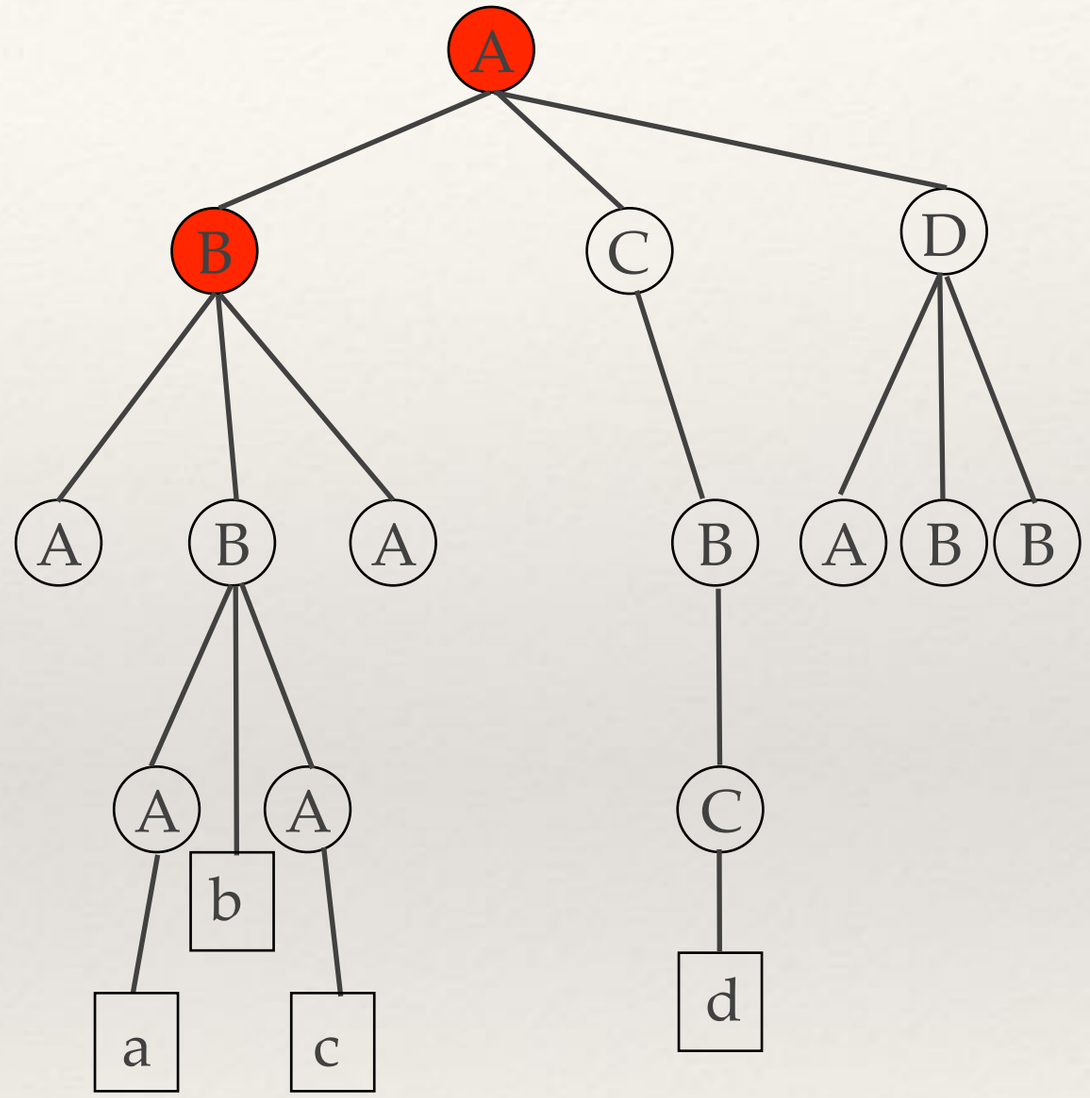


Axis: ancestor

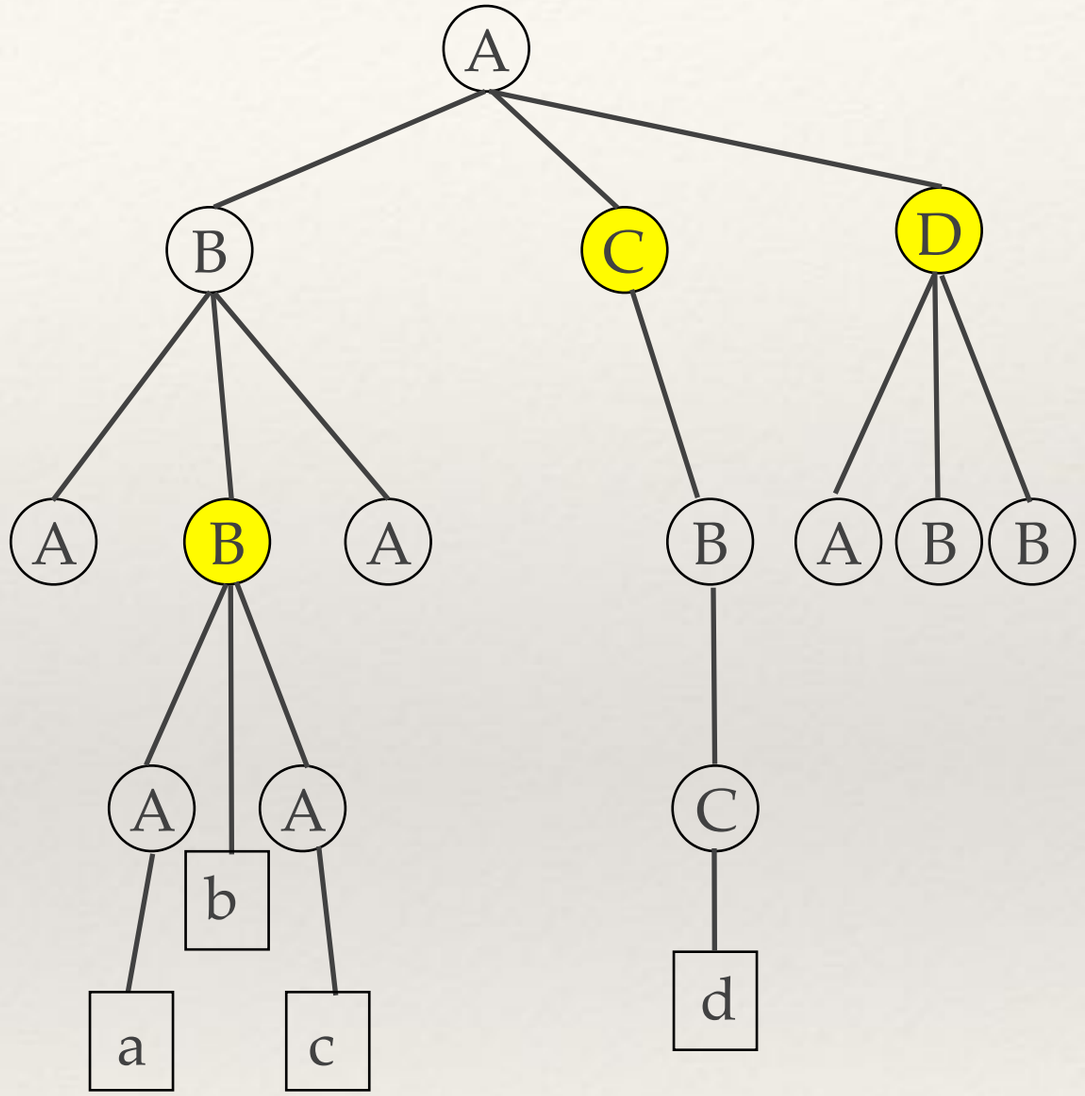
Step

ancestor::*

(“select all ancestor nodes”)



Axis: descendant-or-self

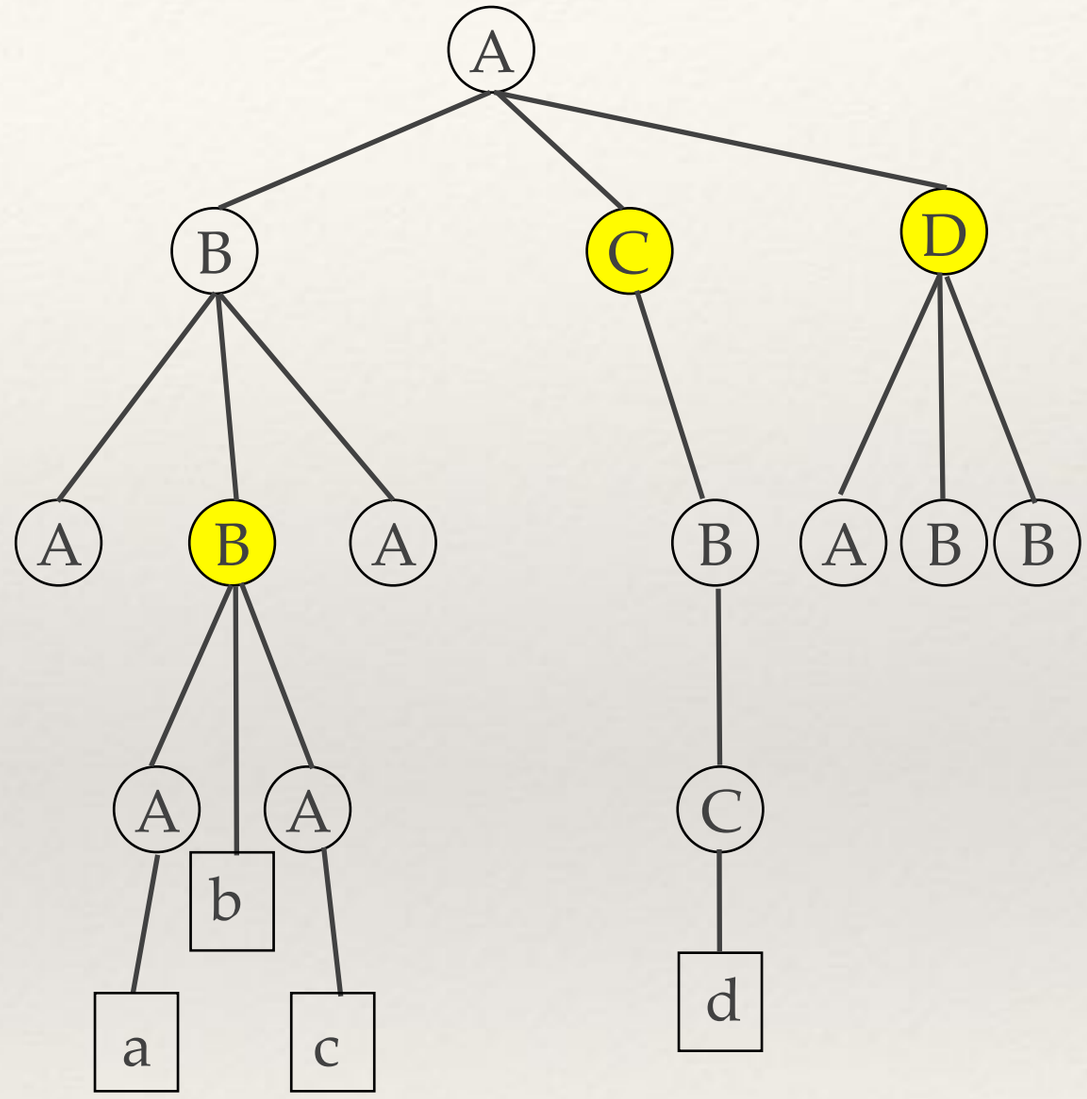


Axis: descendant-or-self

Step

descendant-or-self::*

(“select all descendant nodes, including the current node”)

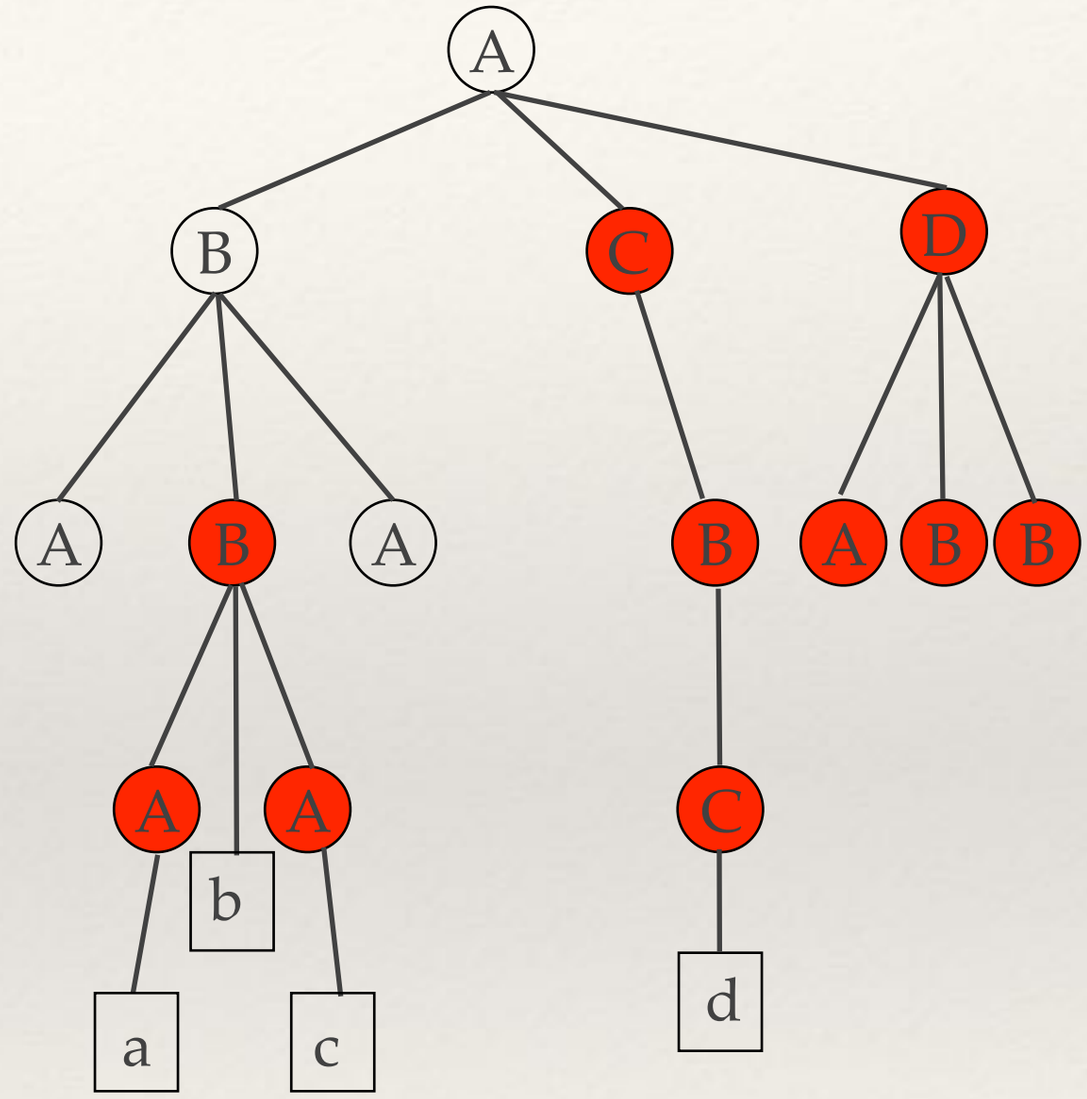


Axis: descendant-or-self

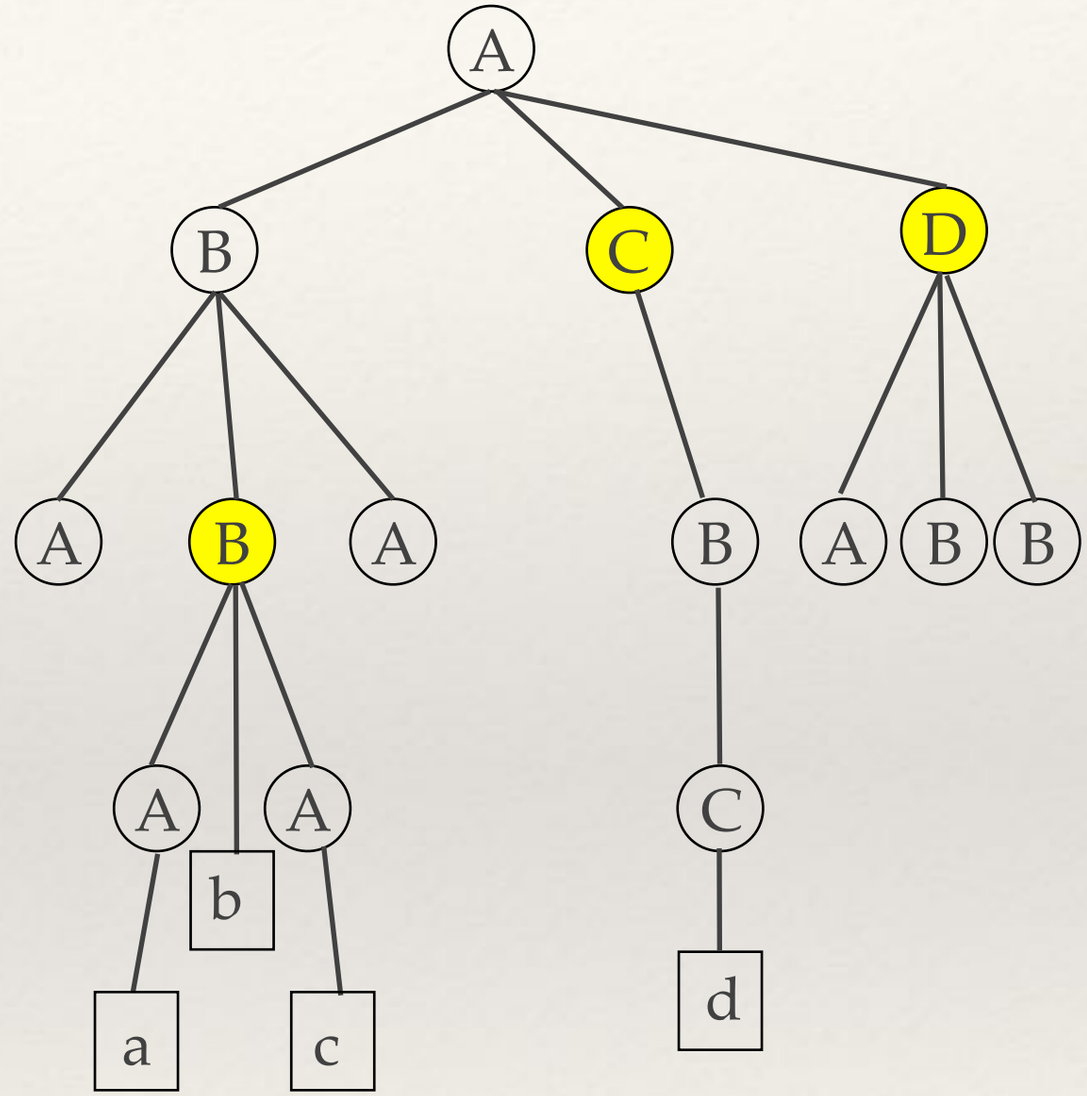
Step

descendant-or-self::*

(“select all descendant nodes, including the current node”)



Axis: following-sibling

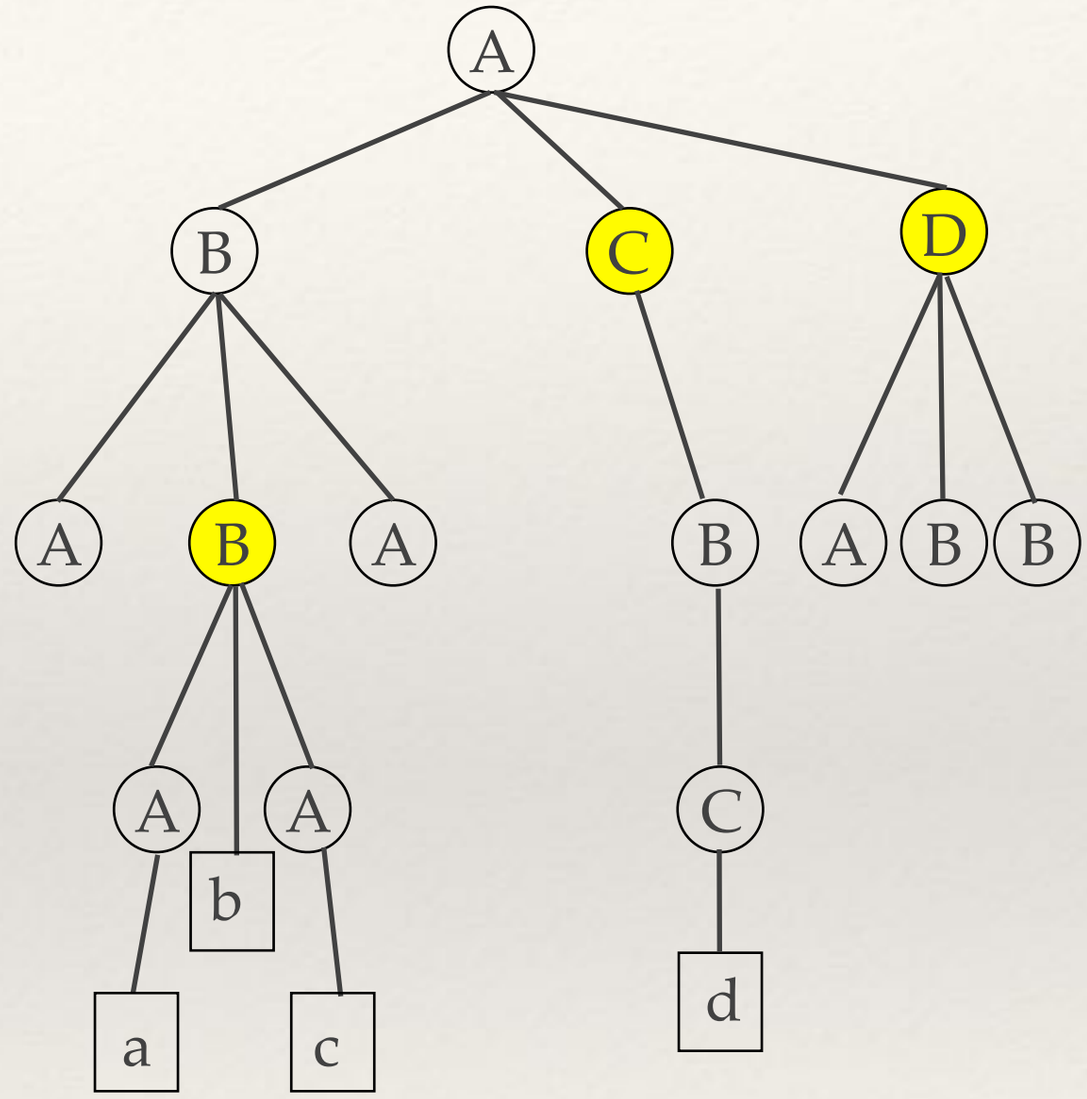


Axis: following-sibling

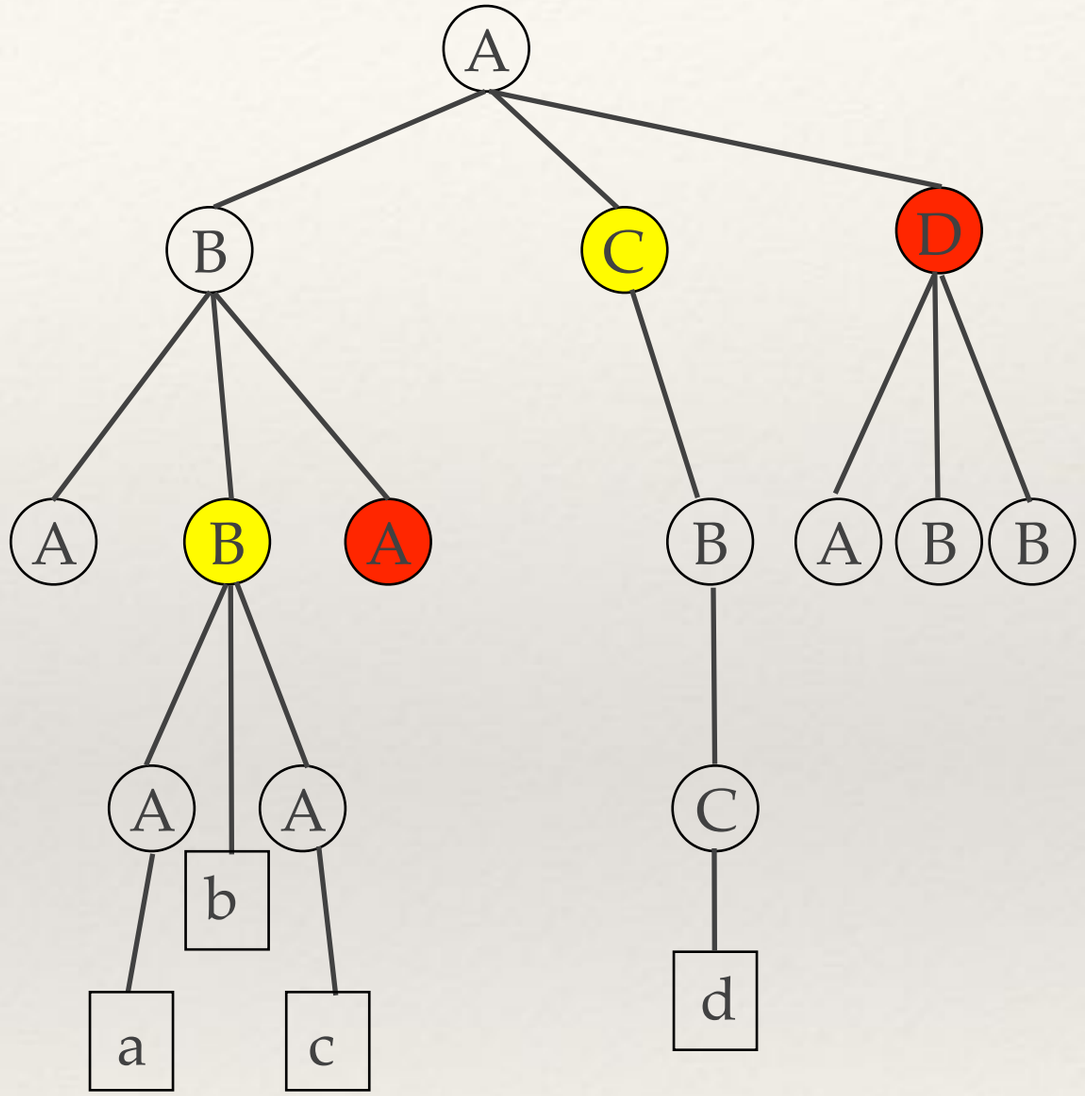
Step

following-sibling::*

(“select all following sibling nodes”)



Axis: following-sibling

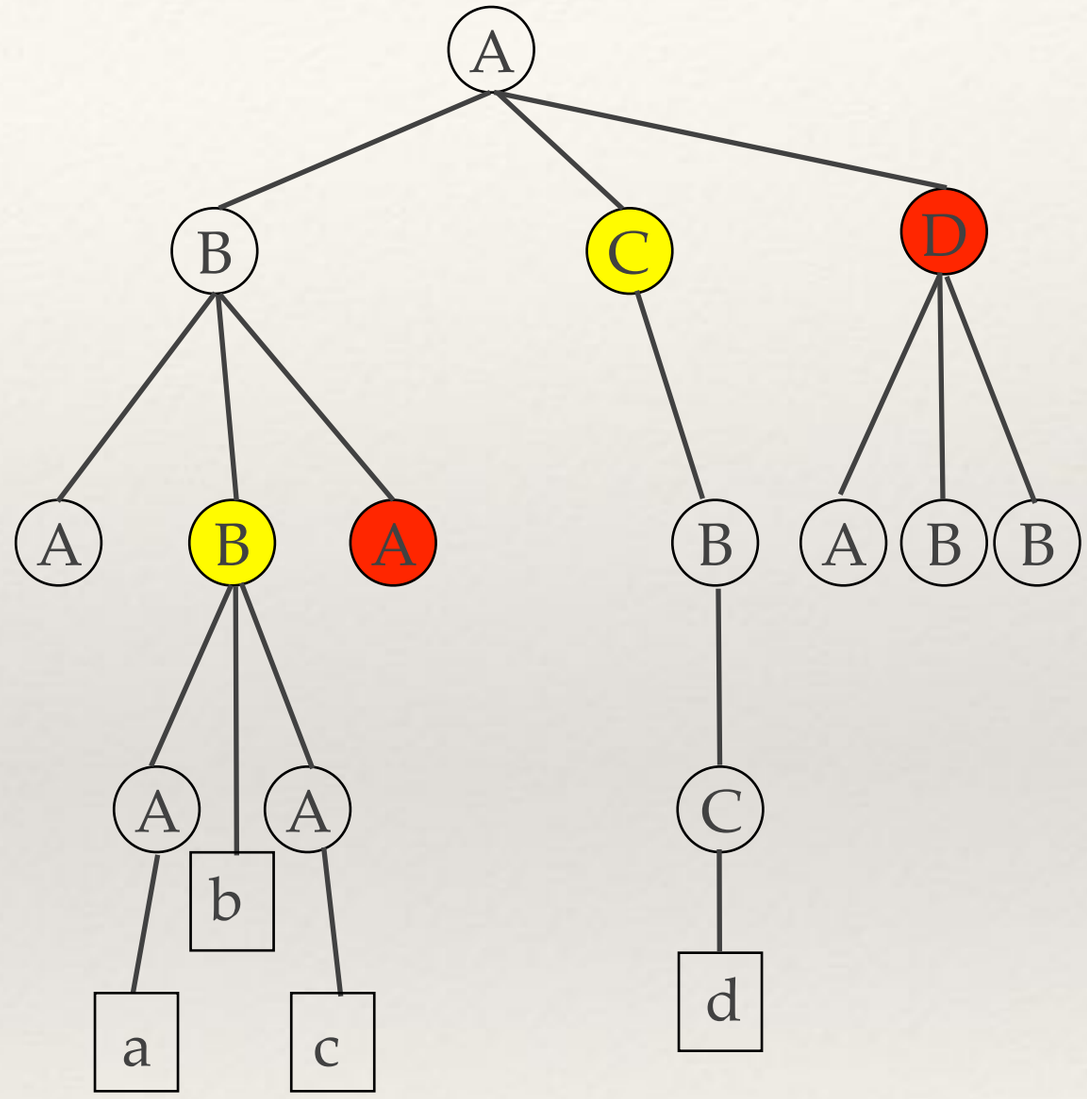


Axis: following-sibling

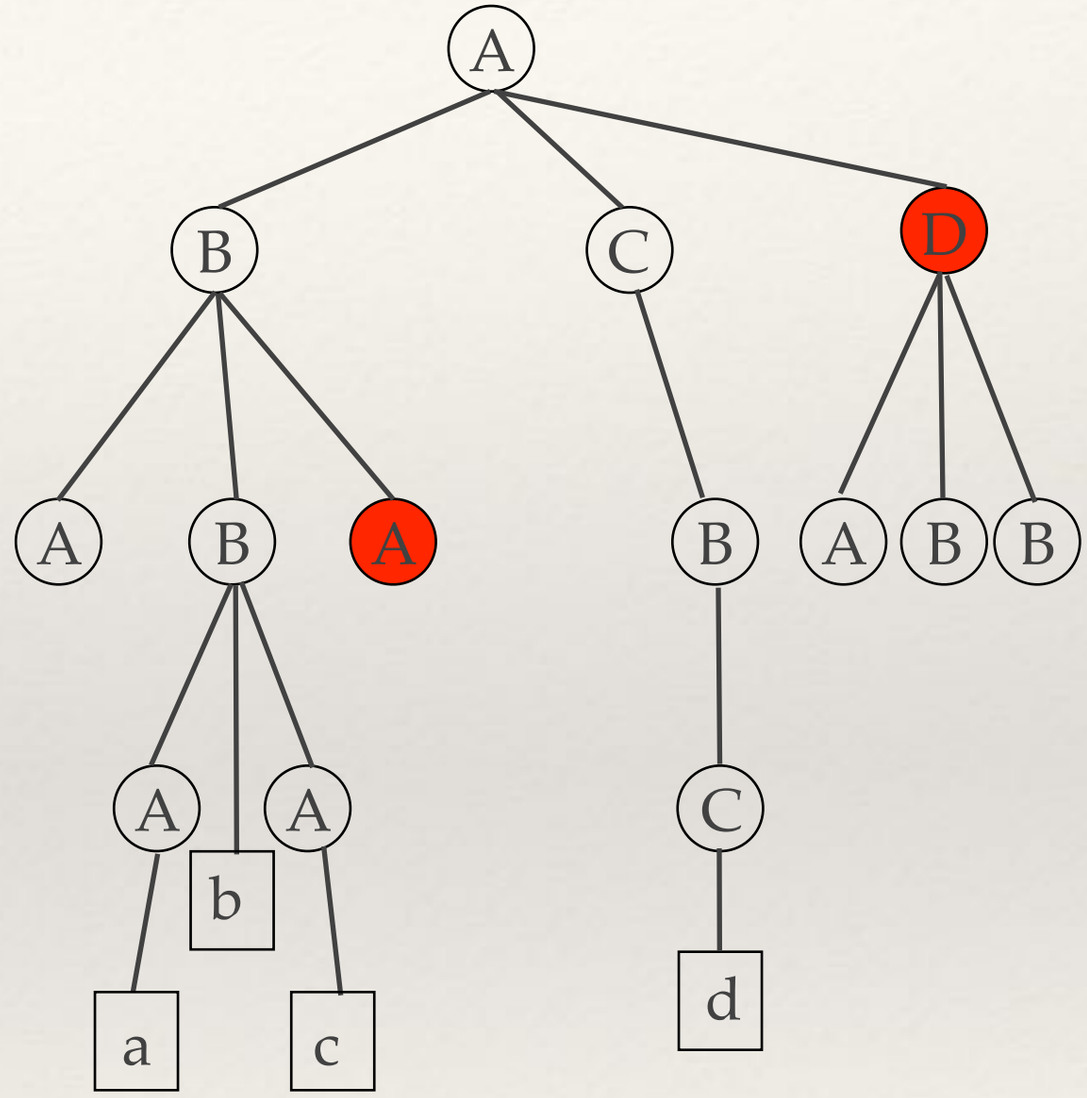
Step

following-sibling::*

(“select all following sibling nodes”)



Axis: following-sibling

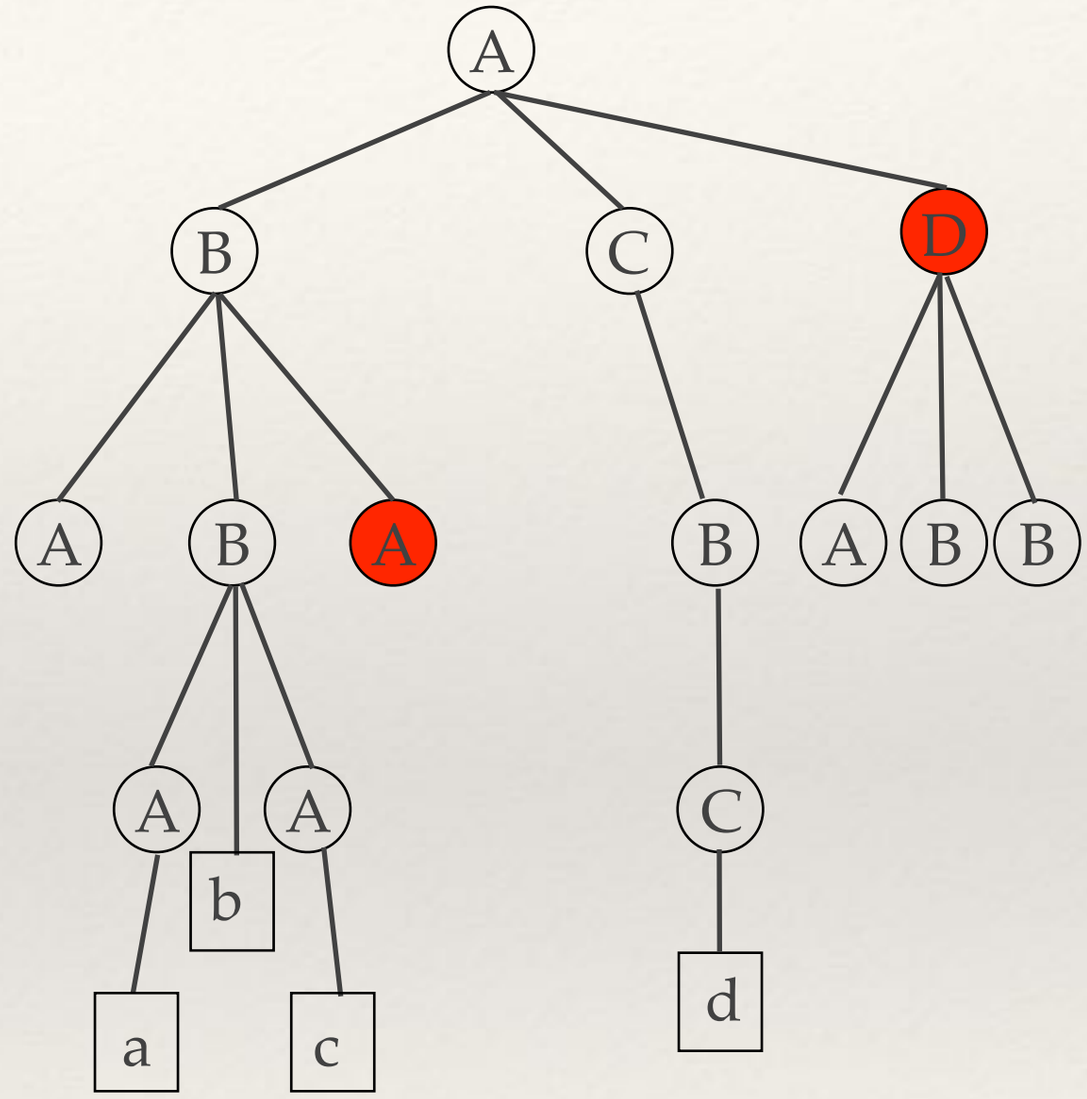


Axis: following-sibling

Step

following-sibling::*

(“select all following sibling nodes”)



XPath example (1)

XPath example (1)

Query:

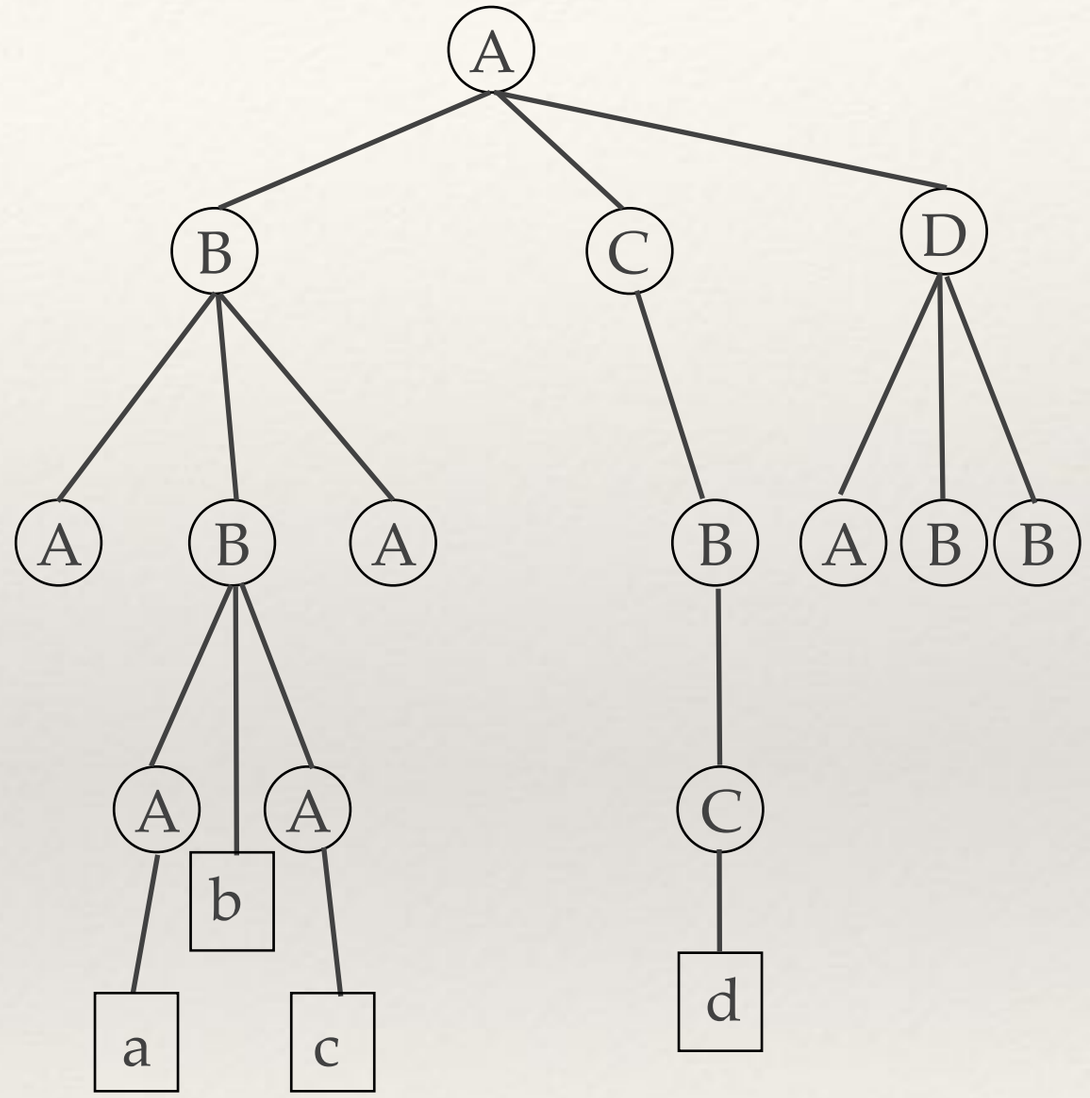
```
/

(“select the root node”)
```

XPath example (1)

Query:

```
/
```

(“select the root node”)

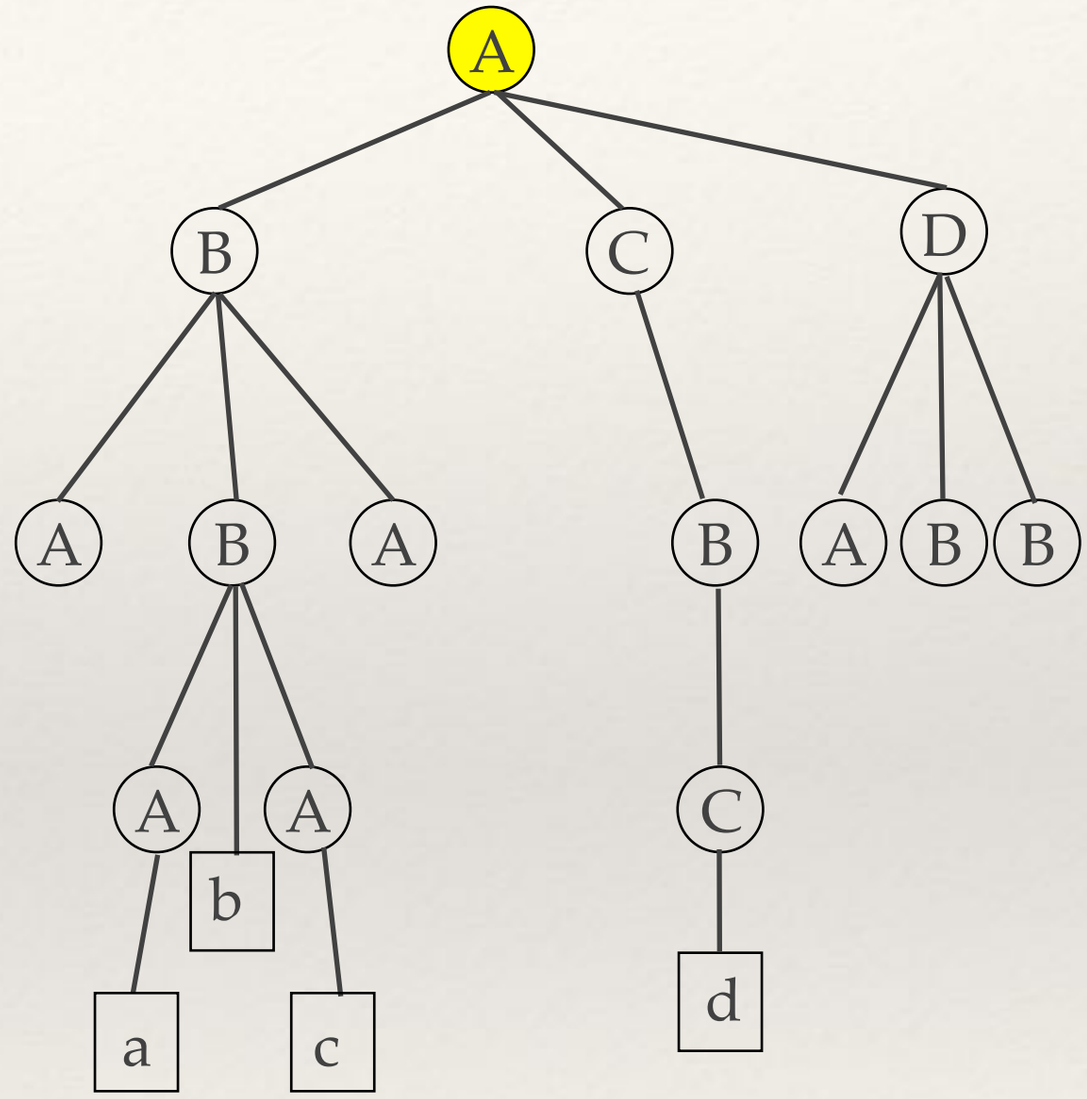


XPath example (1)

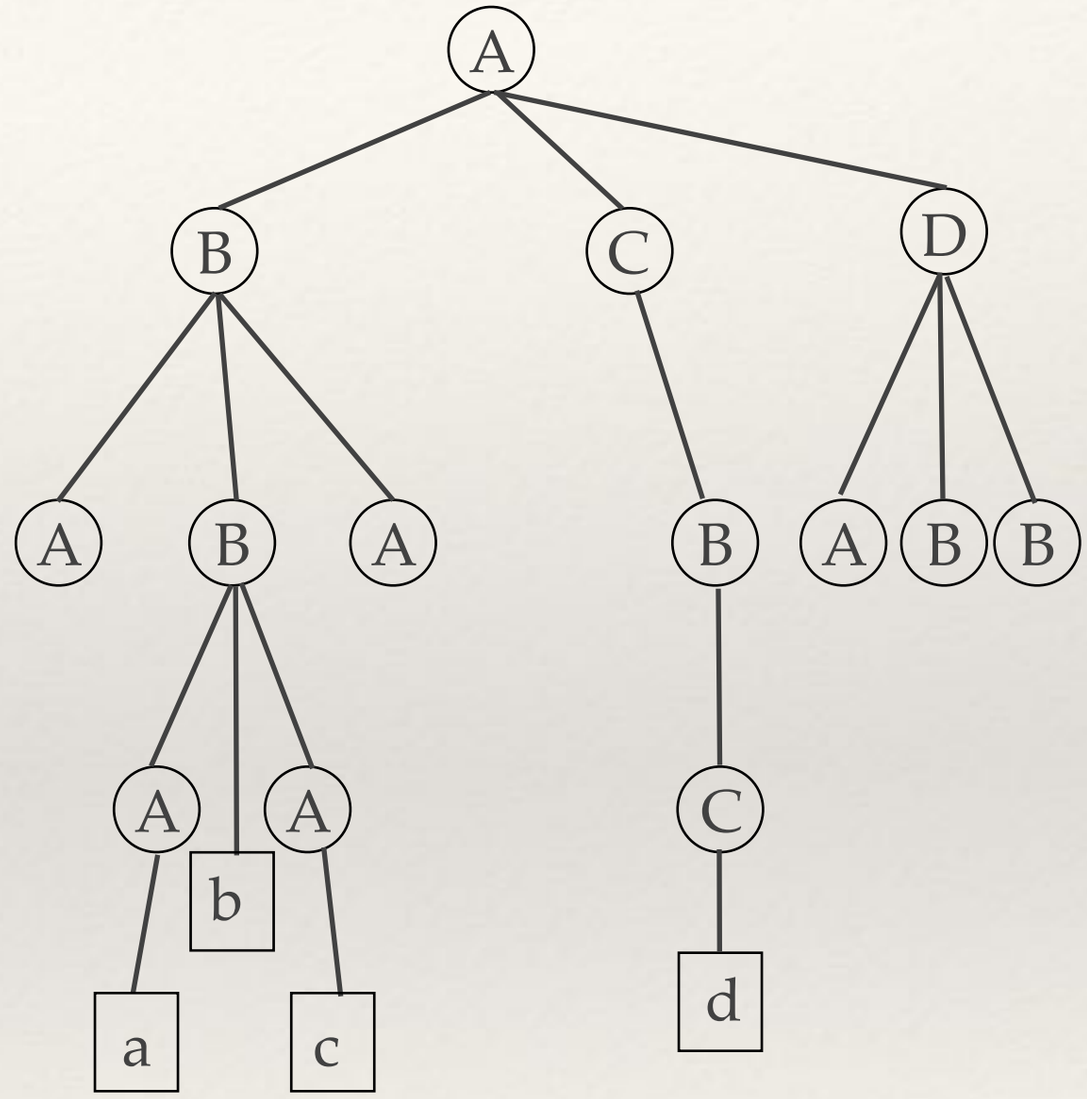
Query:

/

(“select the root node”)



XPath example (2)

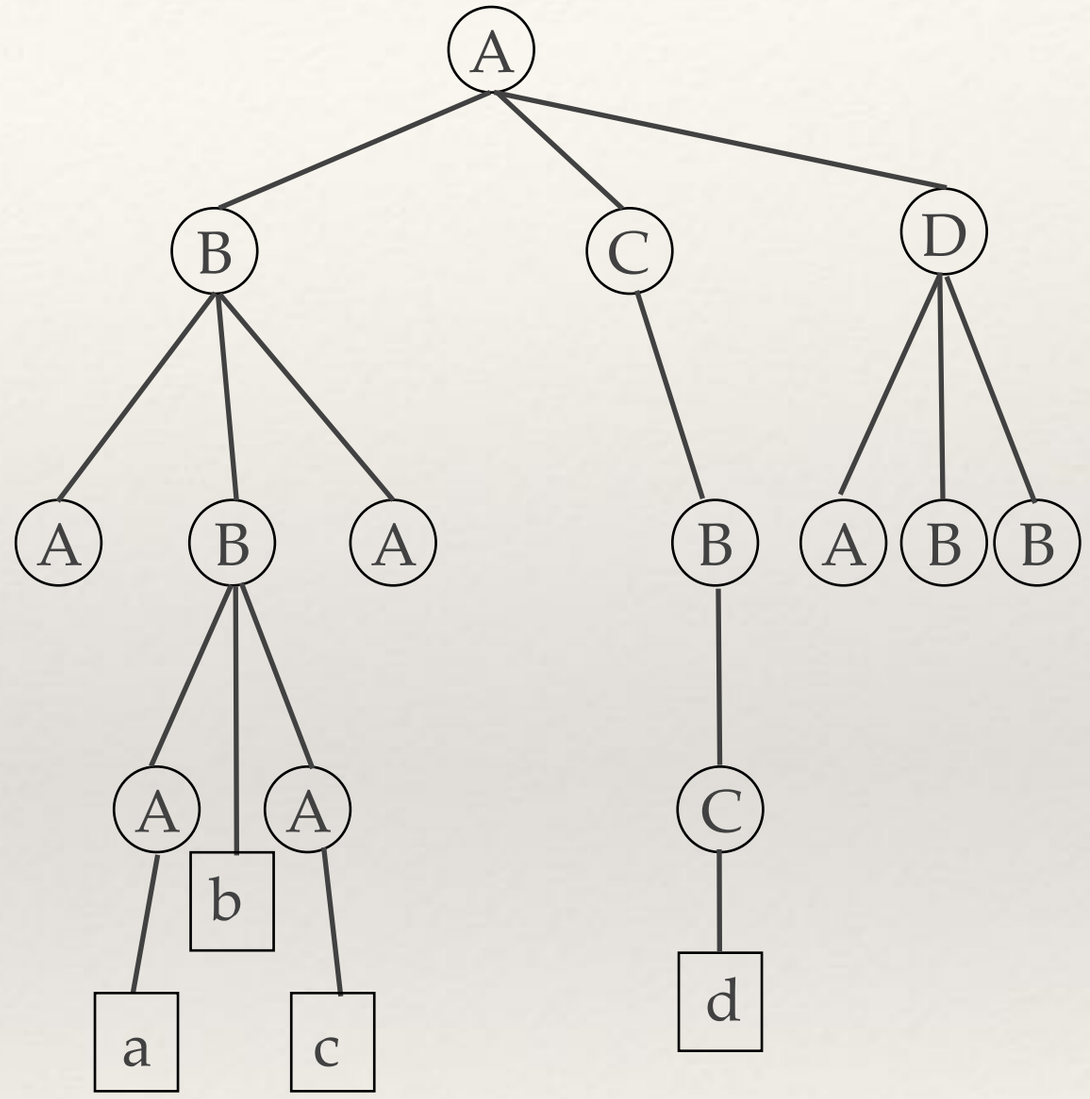


XPath example (2)

Query:

```
/descendant::B
```

(“select all B descendants of the root”)

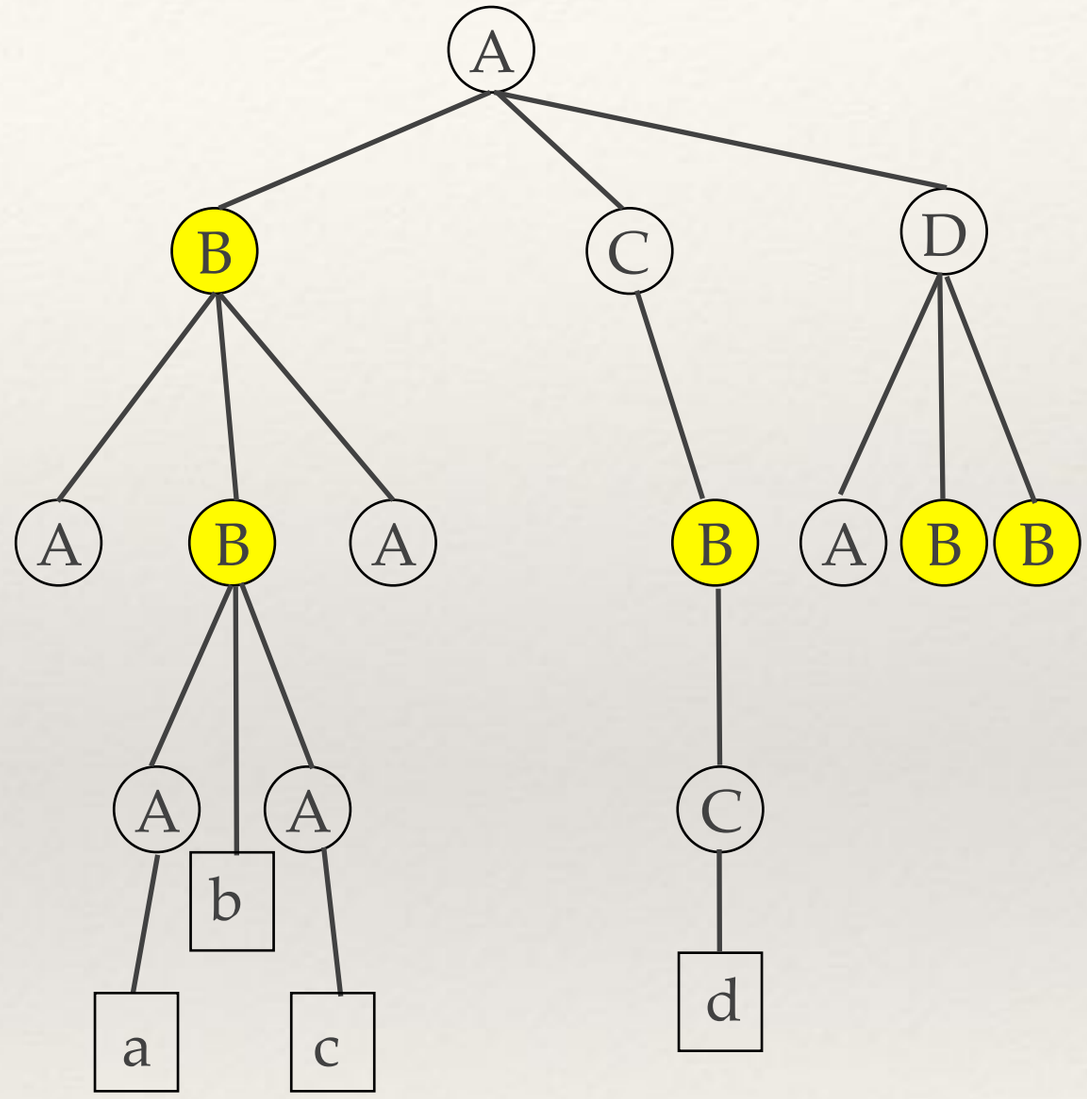


XPath example (2)

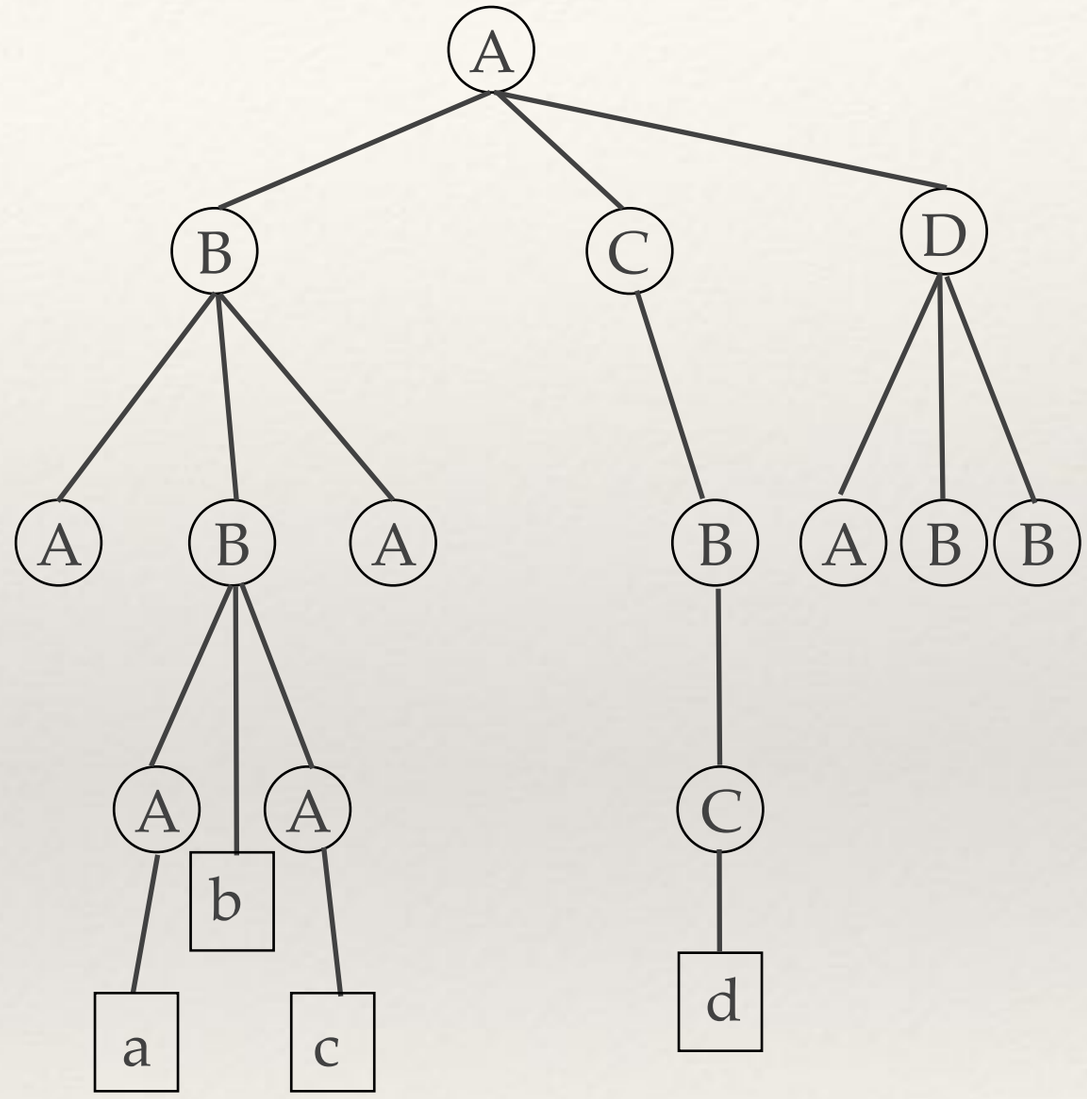
Query:

```
/descendant::B
```

(“select all B descendants of the root”)



XPath example (3)

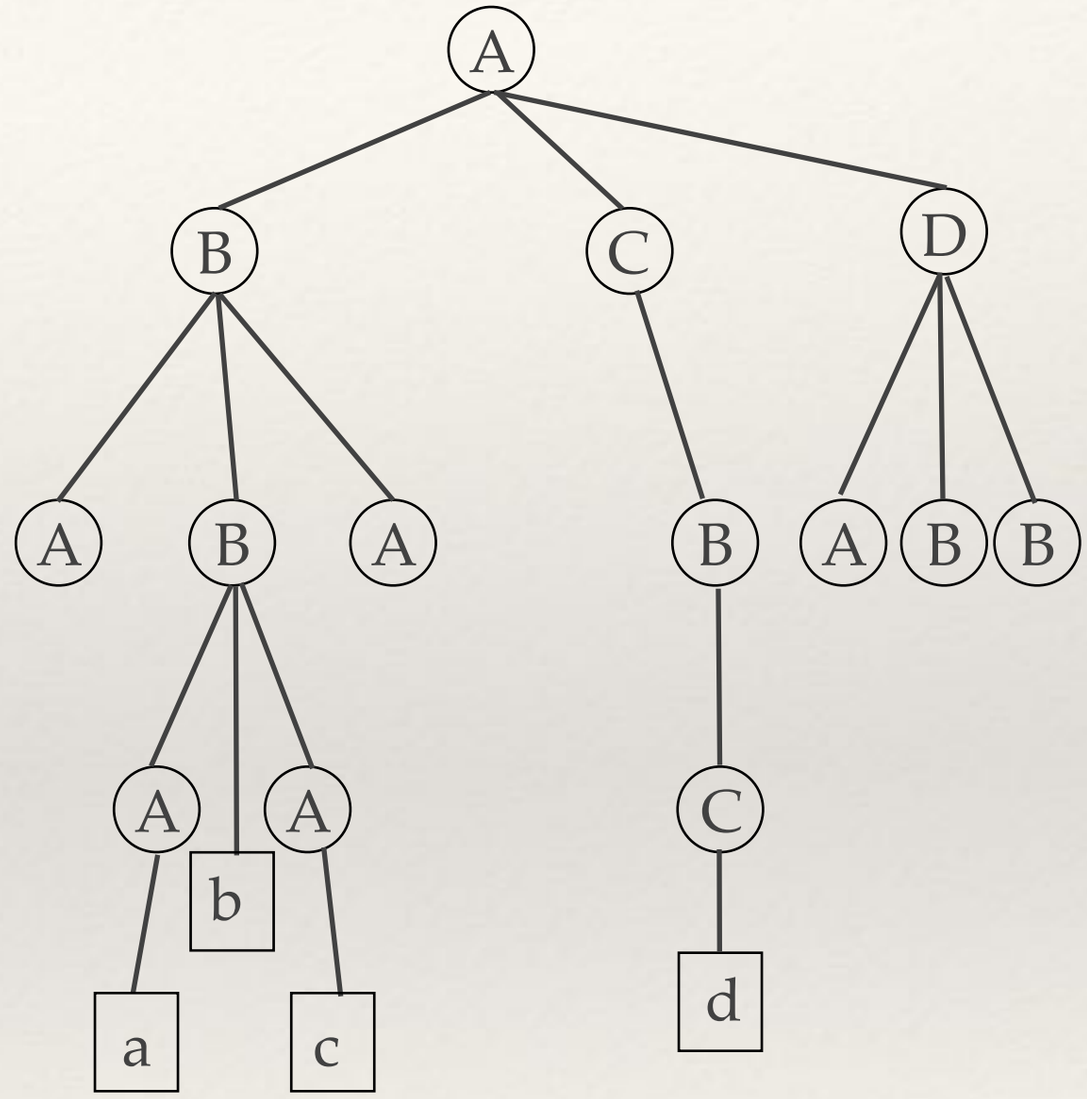


XPath example (3)

Query:

```
/descendant::B/child::A
```

(“select all A children of a B descendant of the root”)

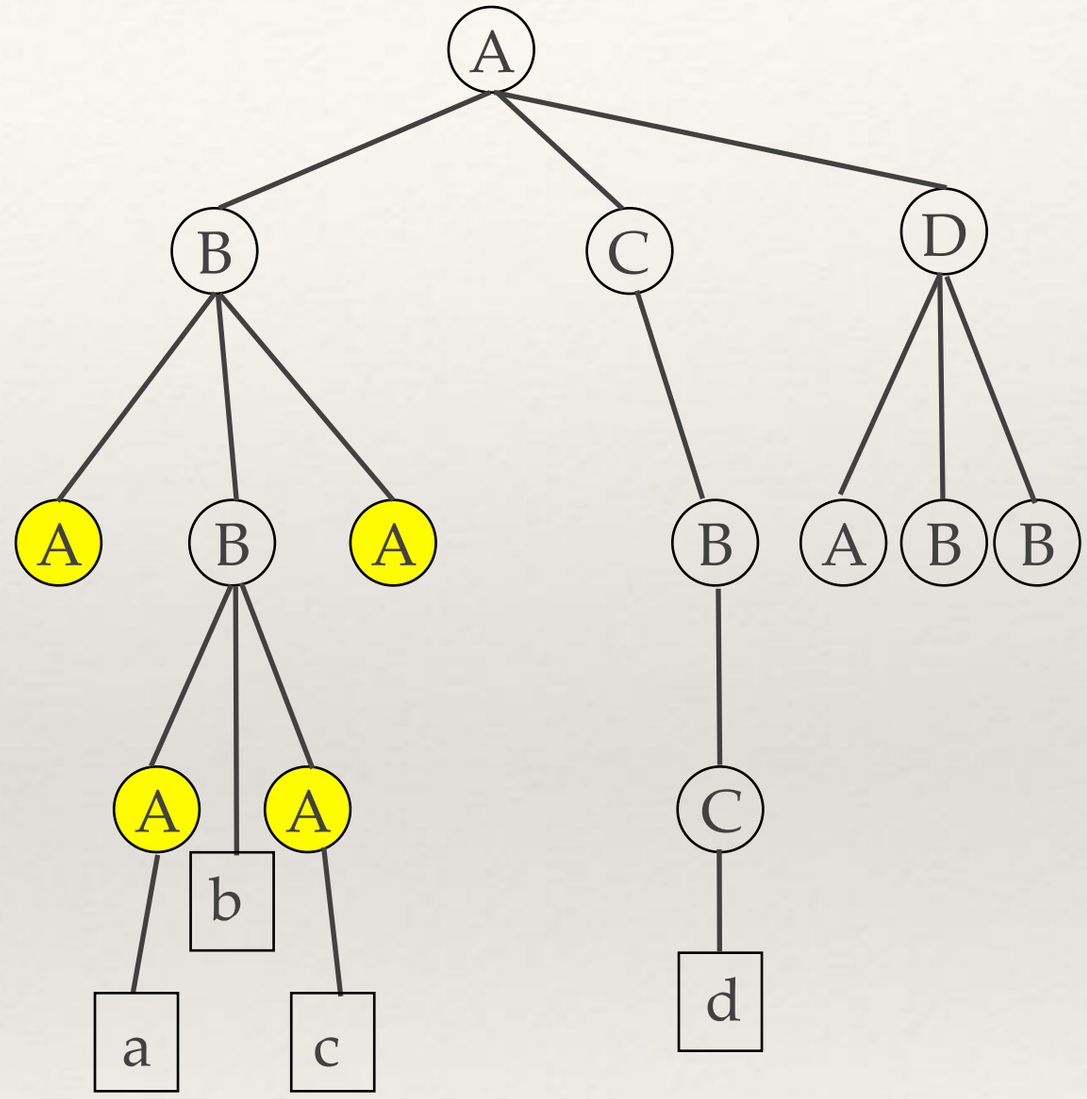


XPath example (3)

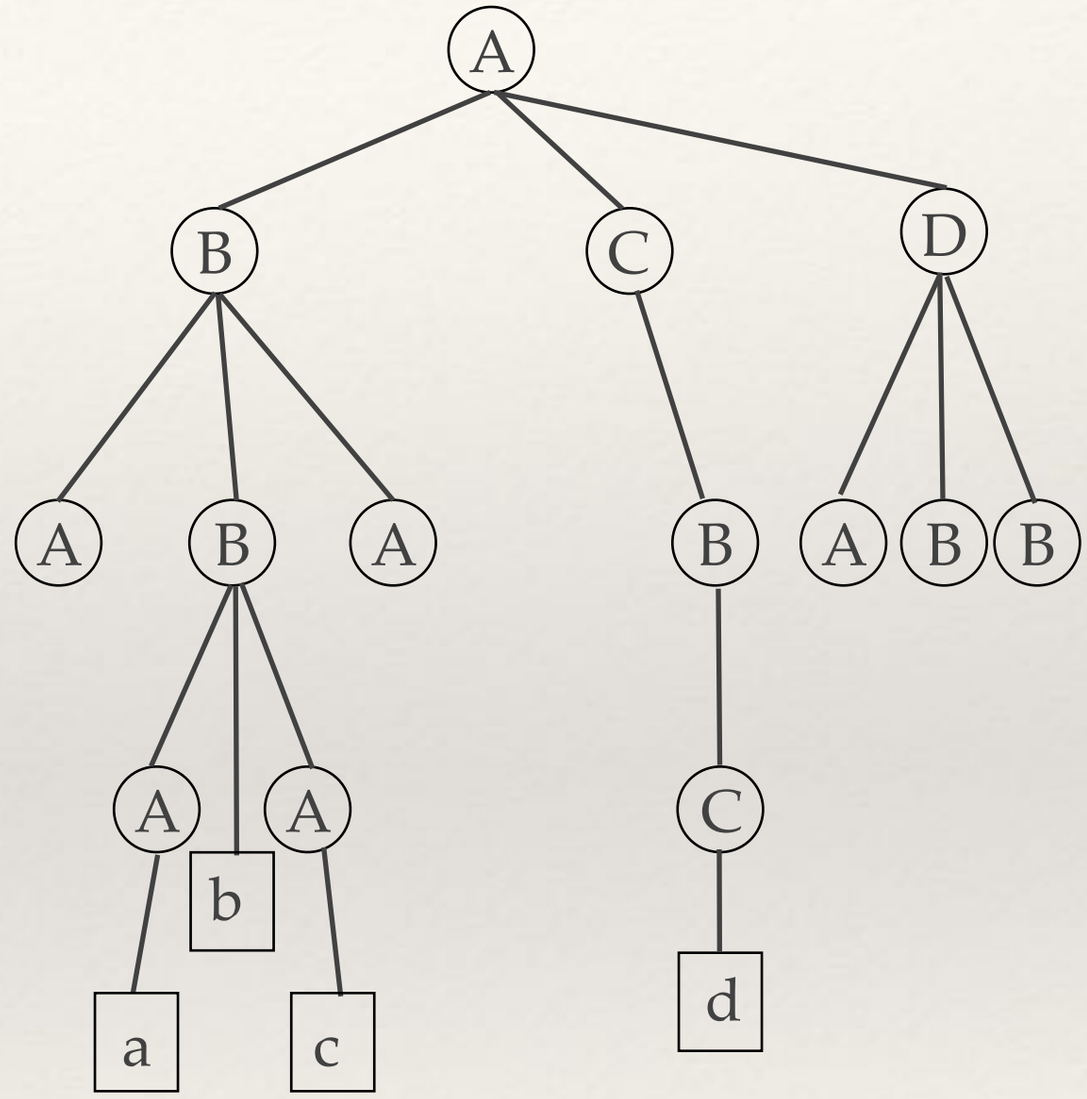
Query:

```
/descendant::B/child::A
```

(“select all A children of a B descendant of the root”)



XPath example (4)

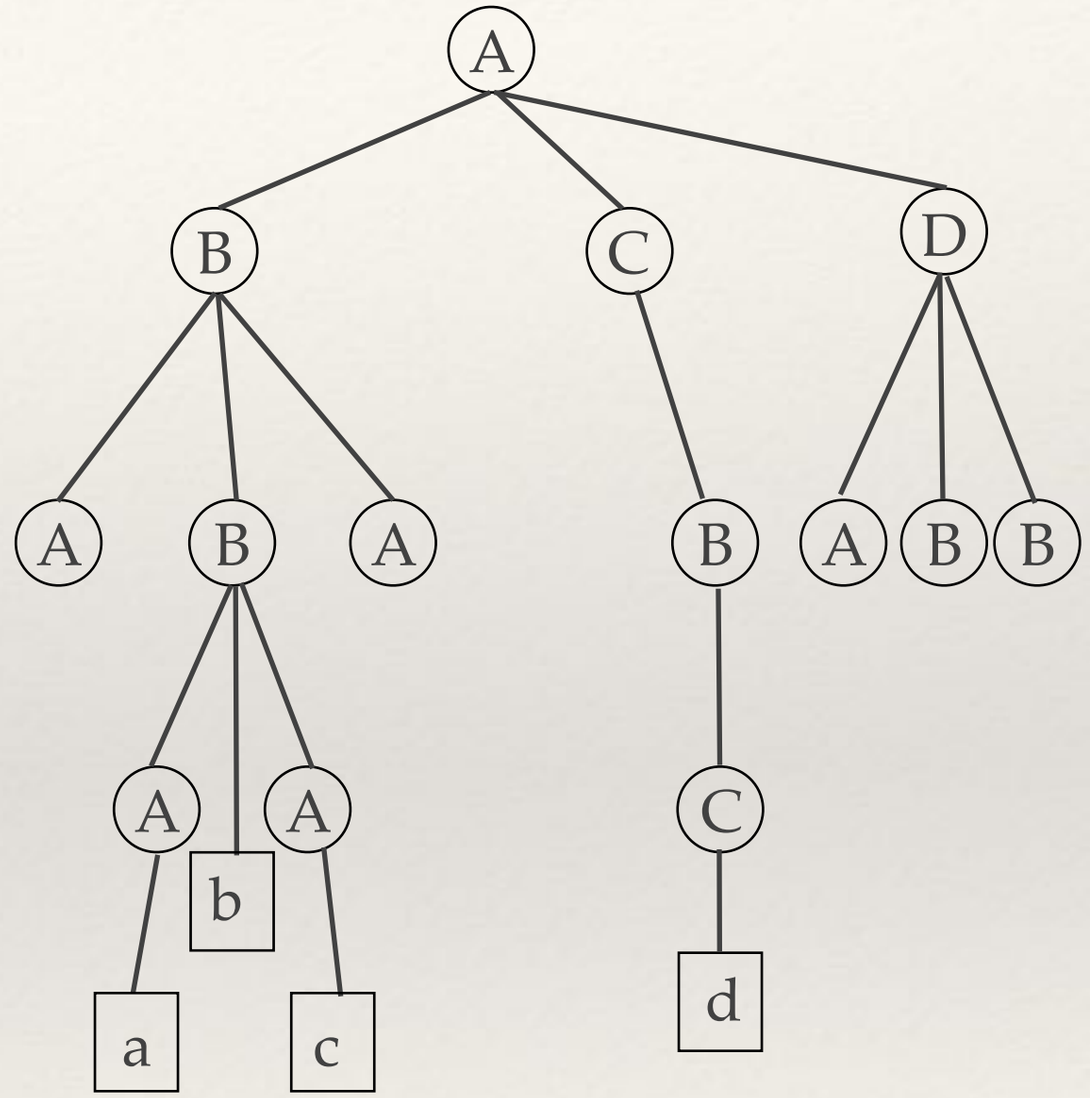


XPath example (4)

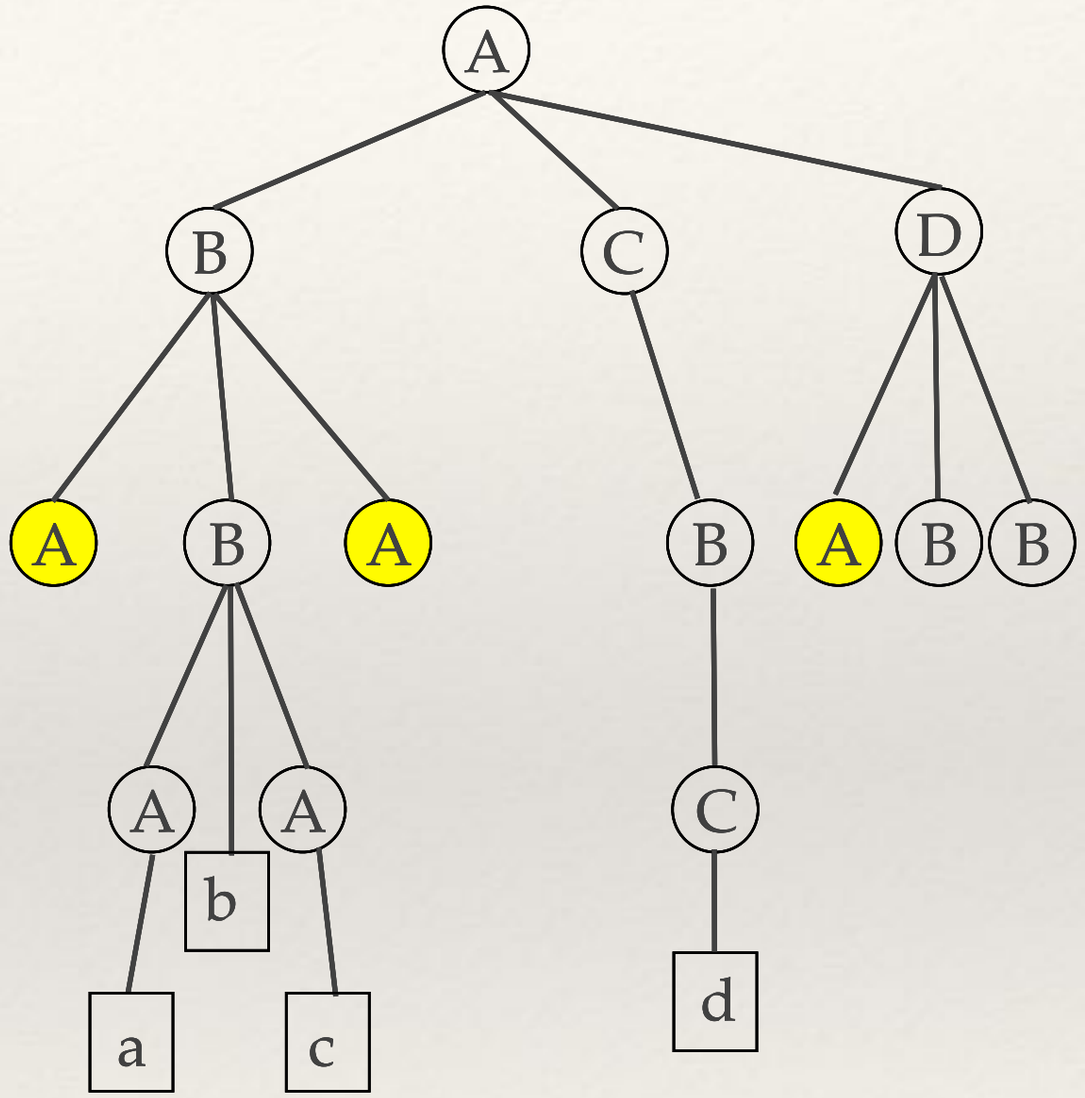
Query:

```
/child::*/child::*/child::A
```

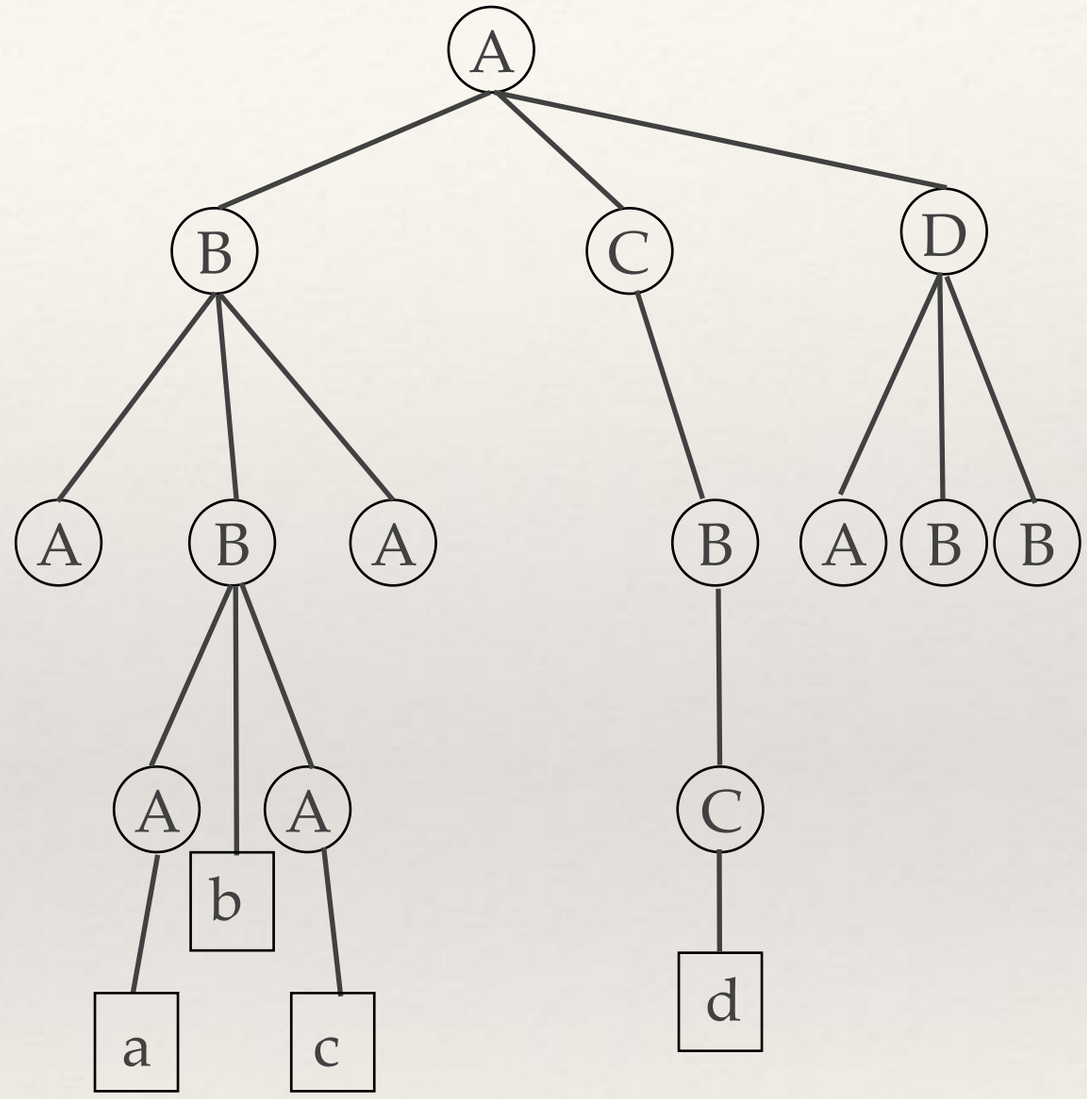
(“select all A grandchildren of the root”)



XPath example (4)



XPath example (5)

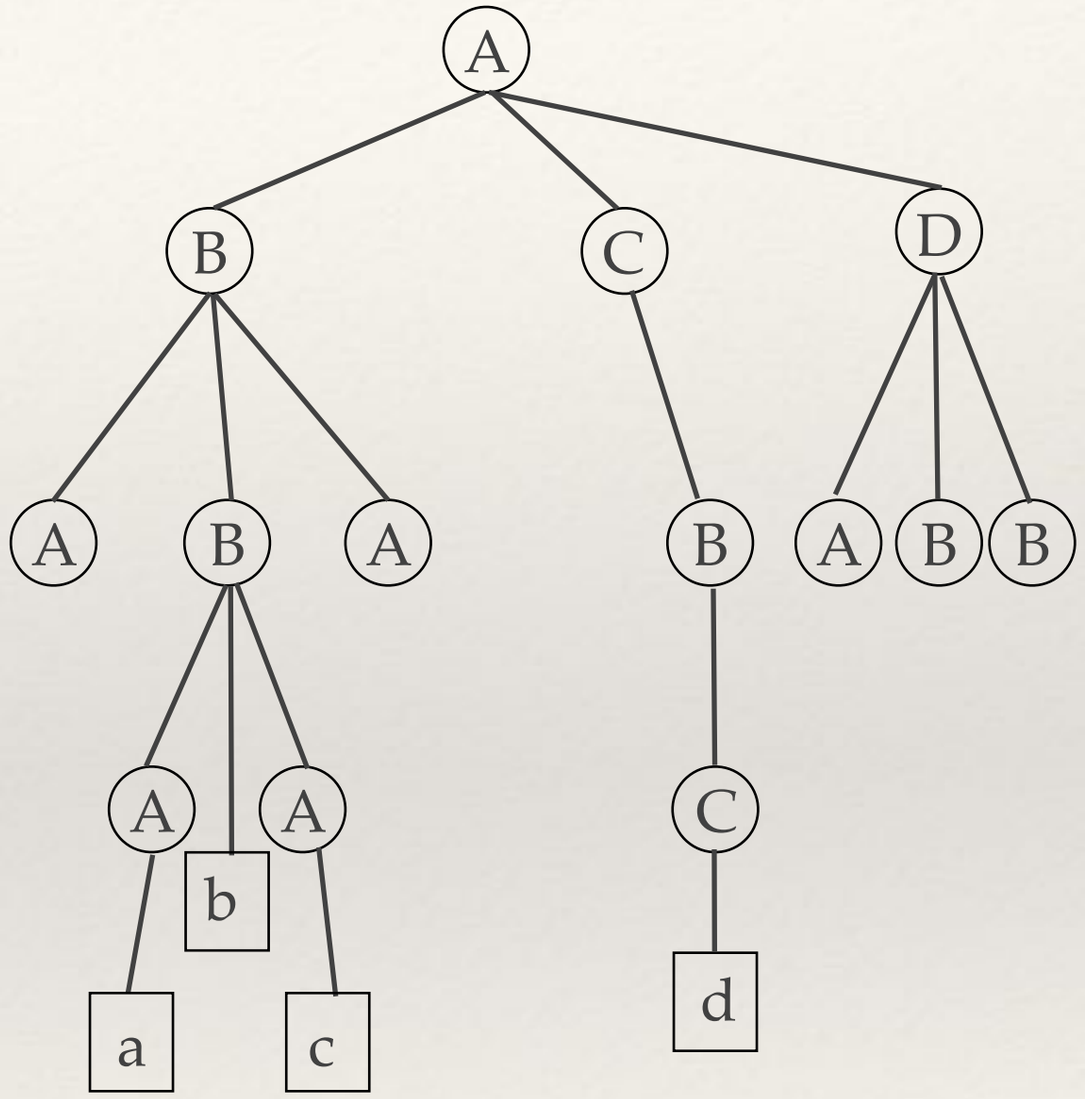


XPath example (5)

Query:

```
/descendant::B/child::*[position()=1]
```

(“select all first children of a B node that is a descendant of the root”)

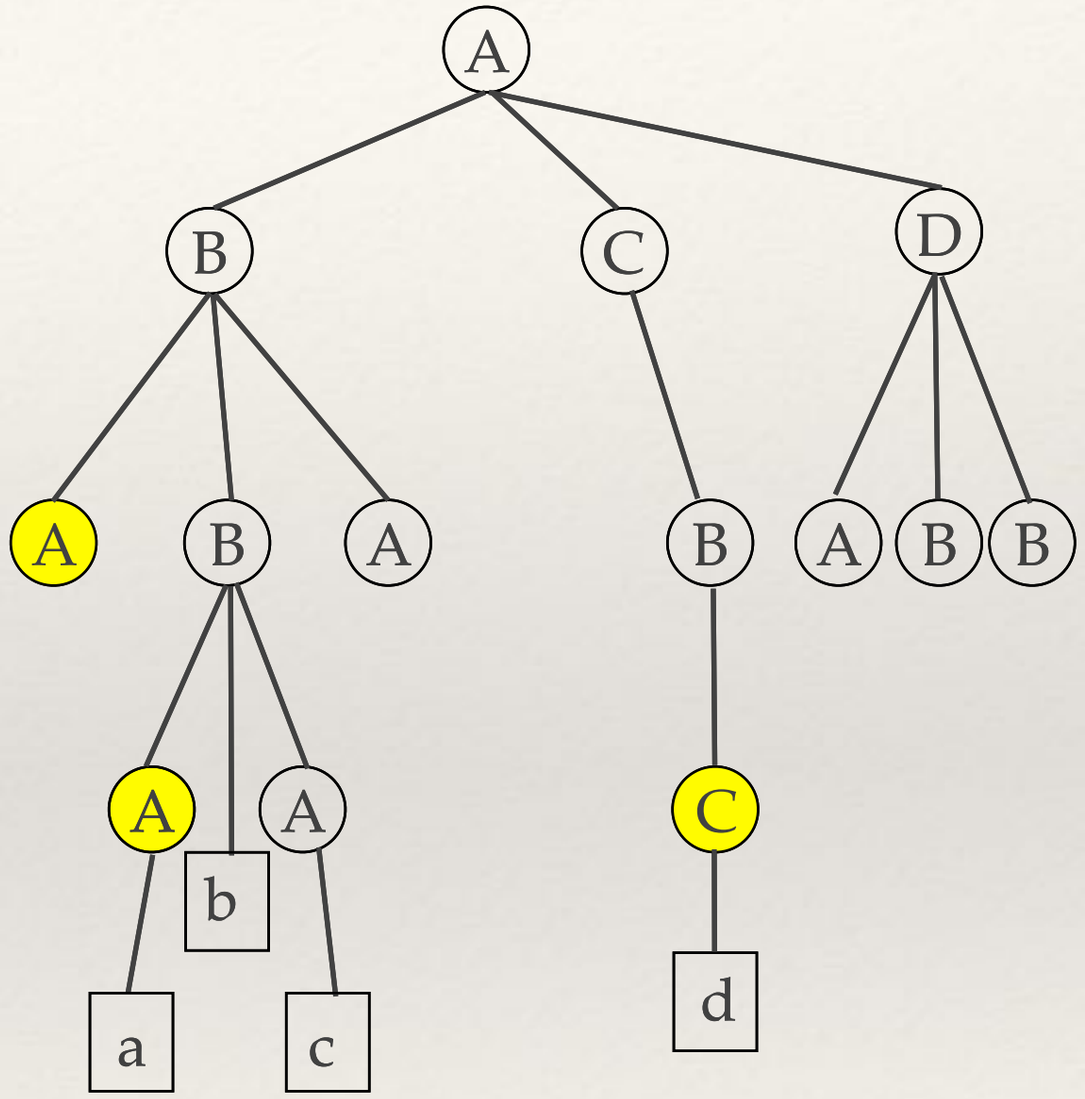


XPath example (5)

Query:

```
/descendant::B/child::*[position()=1]
```

(“select all first children of a B node that is a descendant of the root”)



Main axes

self

parent

child

ancestor

descendant

ancestor-or-self

descendant-or-self

following-sibling

preceding-sibling

Abbreviated syntax

Abbreviated syntax

- “child” axes can be omitted

Abbreviated syntax

- “child” axes can be omitted
- ::* can be omitted

Abbreviated syntax

- “child” axes can be omitted
- ::* can be omitted
- // stands for /descendant-or-self::*/

Abbreviated syntax

- “child” axes can be omitted
- ::* can be omitted
- // stands for /descendant-or-self::*/
- /../ stands for /parent::*/

Abbreviated syntax

- “child” axes can be omitted
- ::* can be omitted
- // stands for /descendant-or-self::*/
- /../ stands for /parent::*/
- [x] stands for [position()=x]

Abbreviated syntax

Regular syntax	Abbreviated syntax
`/descendant::B/child::A`	`/descendant::B/A`
`/descendant-or-self::olist/child::item`	`//olist/item`
`/child::doc/child::chapter[position()=5]/child::section[position()=2]`	`/doc/chapter[5]/section[2]`

After locating a node…

After locating a node…

- …one can get its attribute(s), with the syntax

  attribute::xyz (abbreviated: @xyz)

After locating a node…

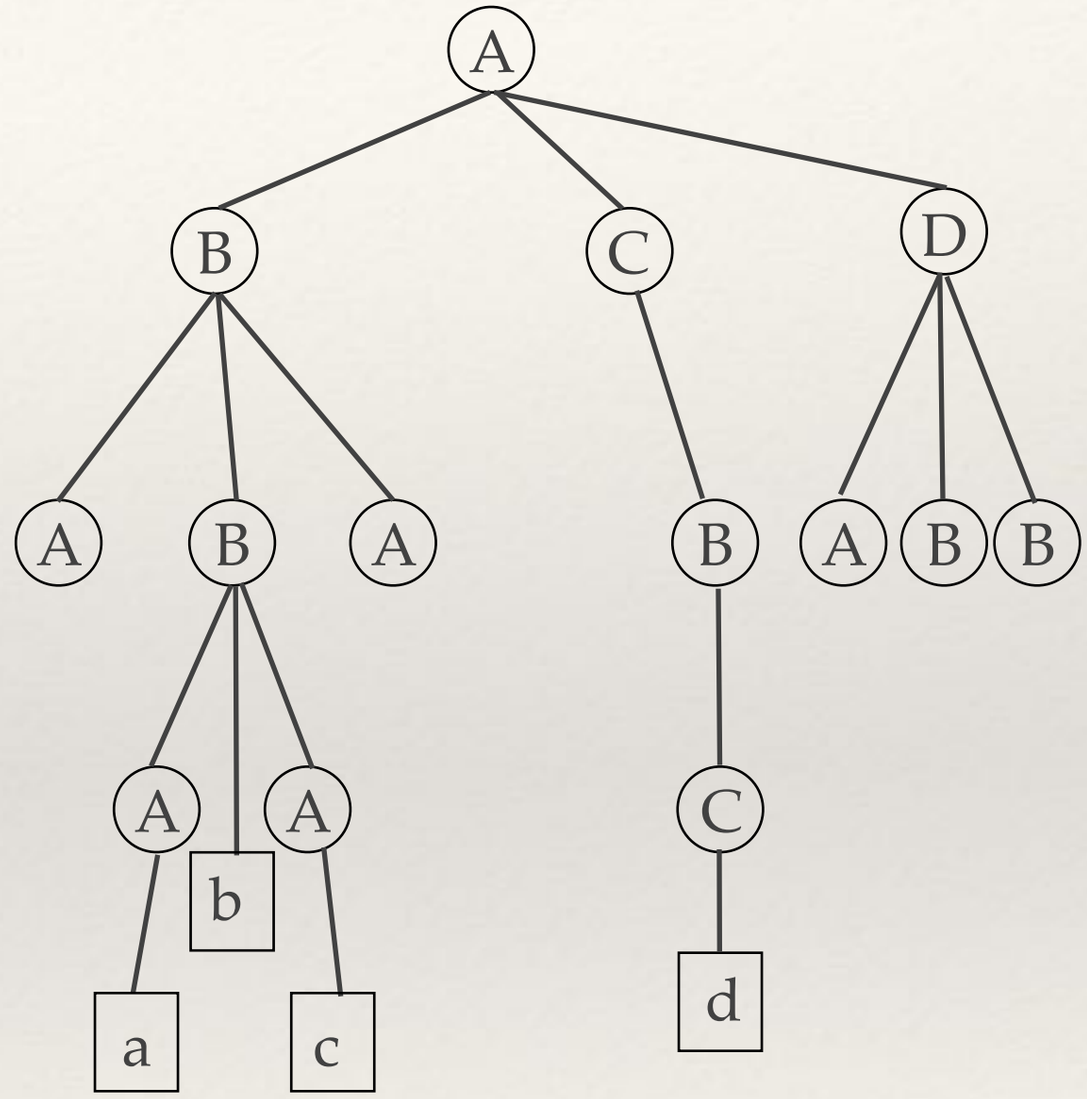
- …one can get its attribute(s), with the syntax

  attribute::xyz (abbreviated: @xyz)

- …or its text children, with the syntax

  text()

Getting text

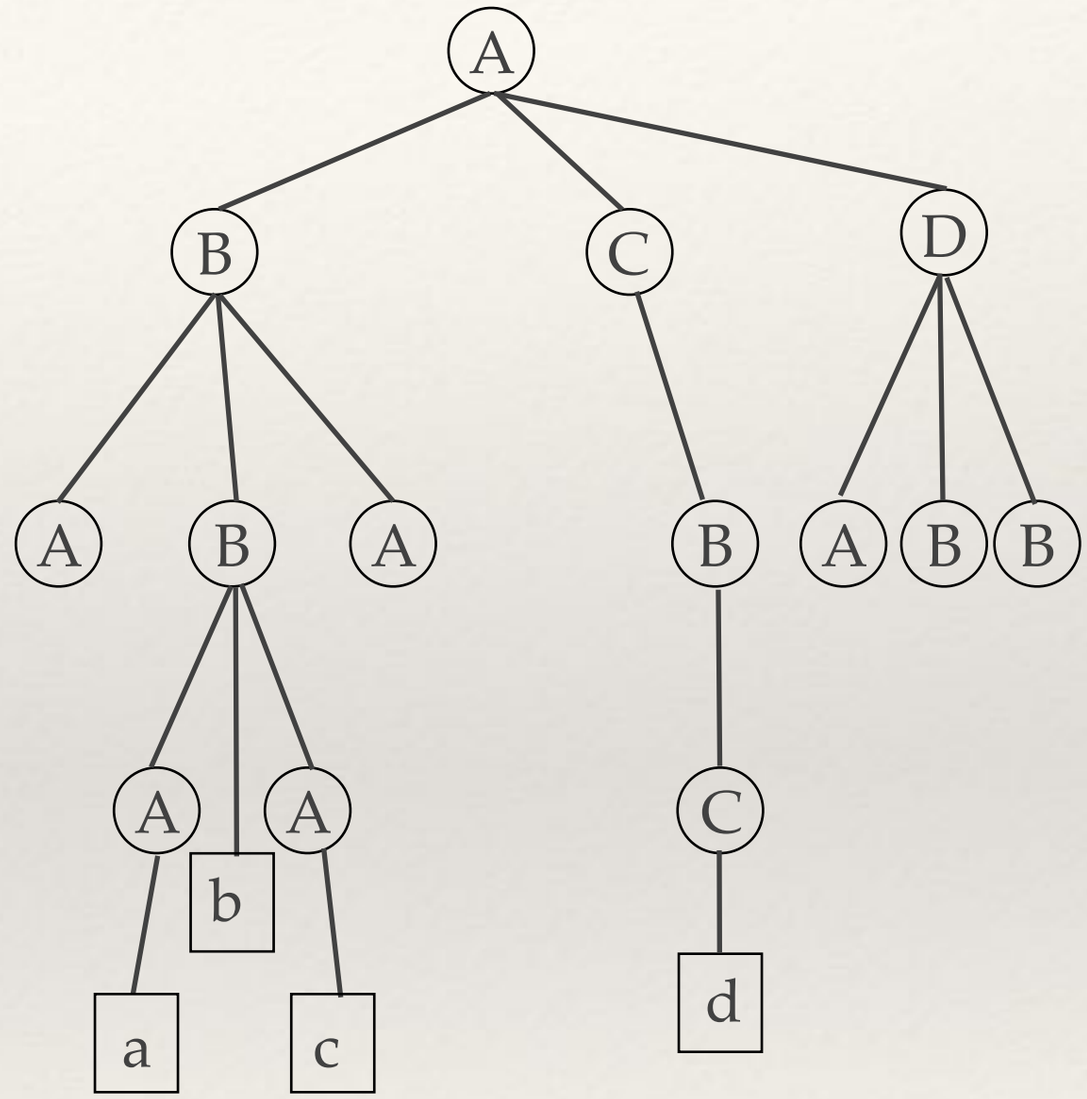


Getting text

Query:

```
/descendant::B/text()
```

returns: [“b”]

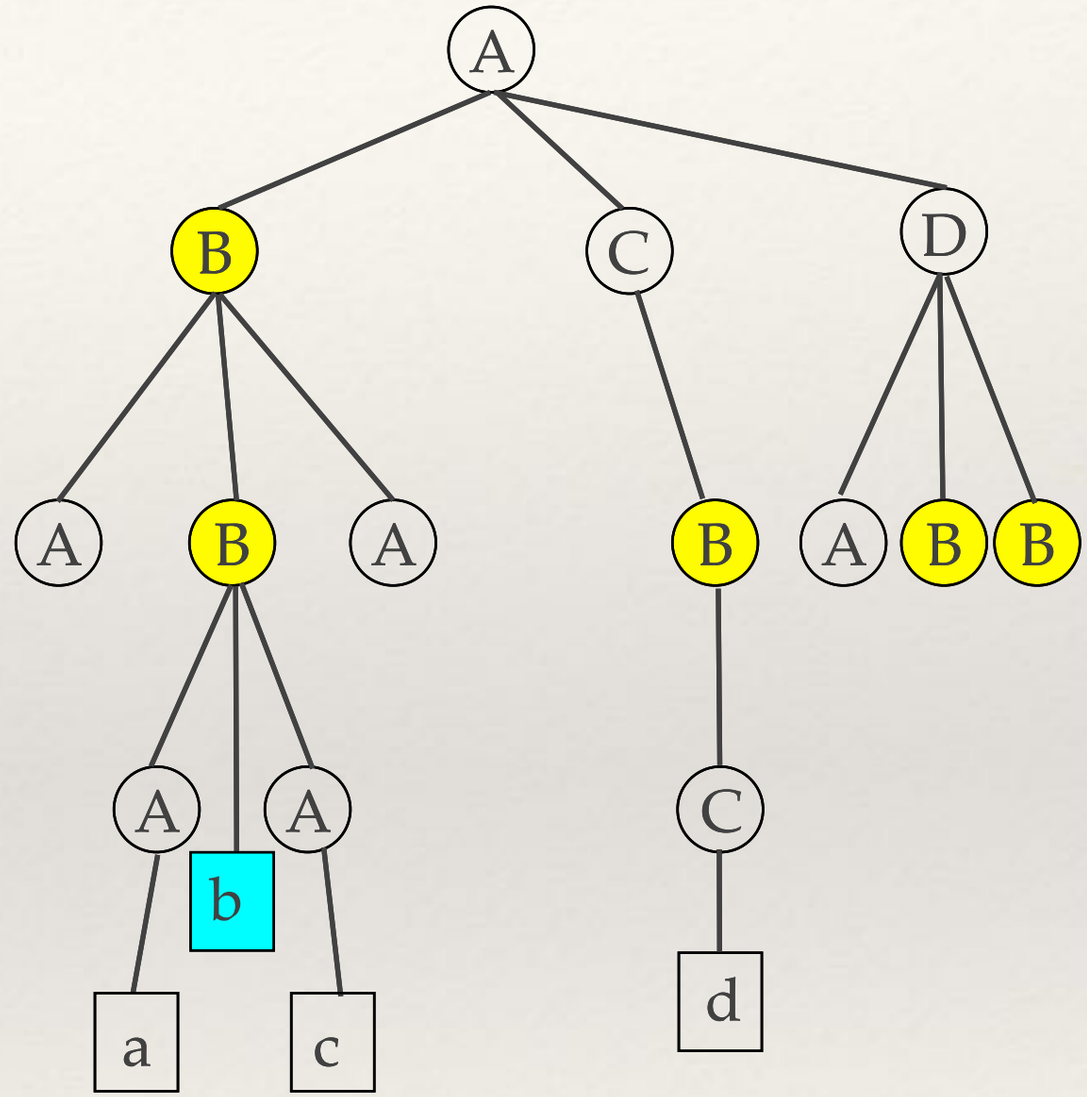


Getting text

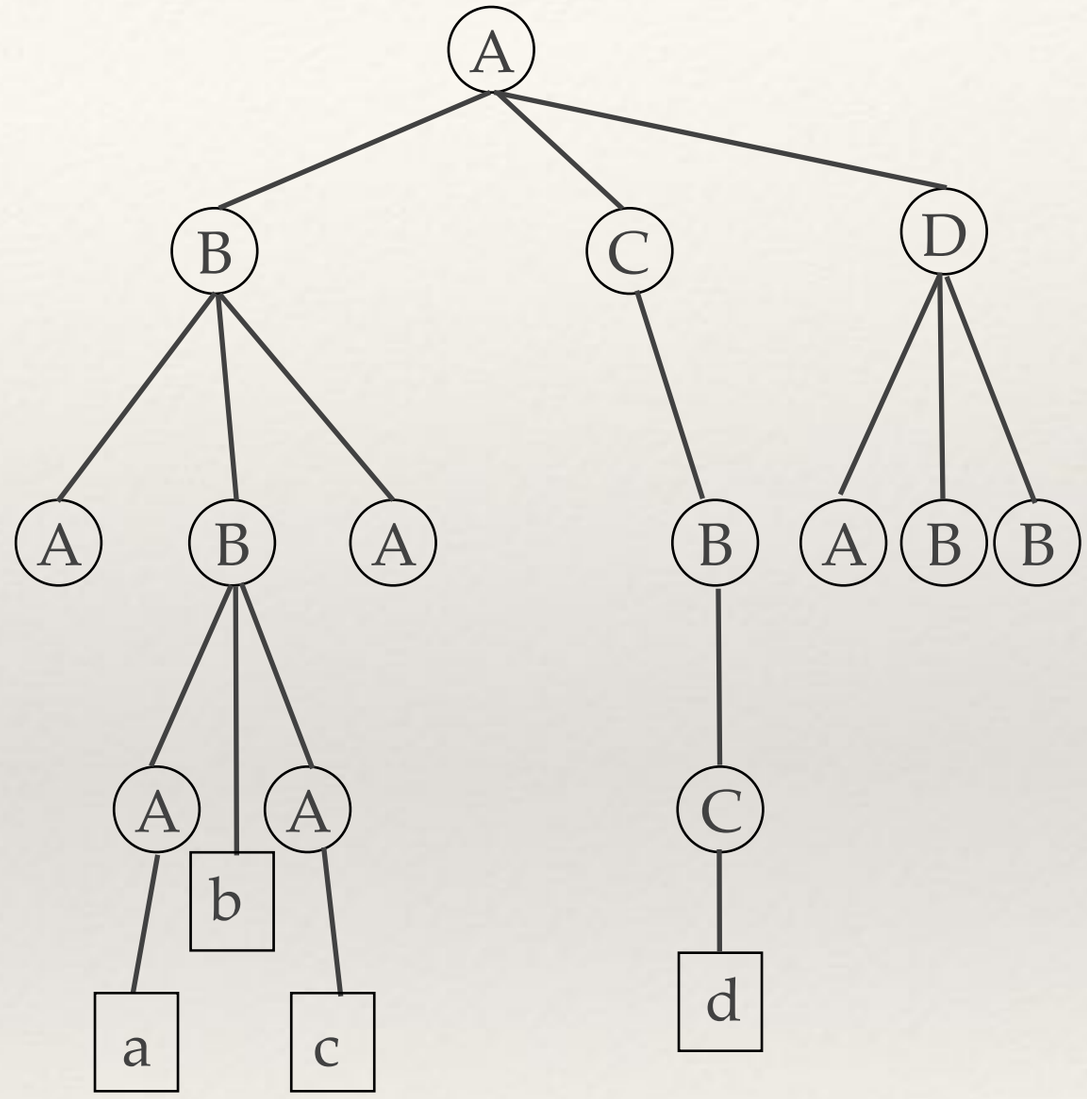
Query:

```
/descendant::B/text()
```

returns: [“b”]



Getting text

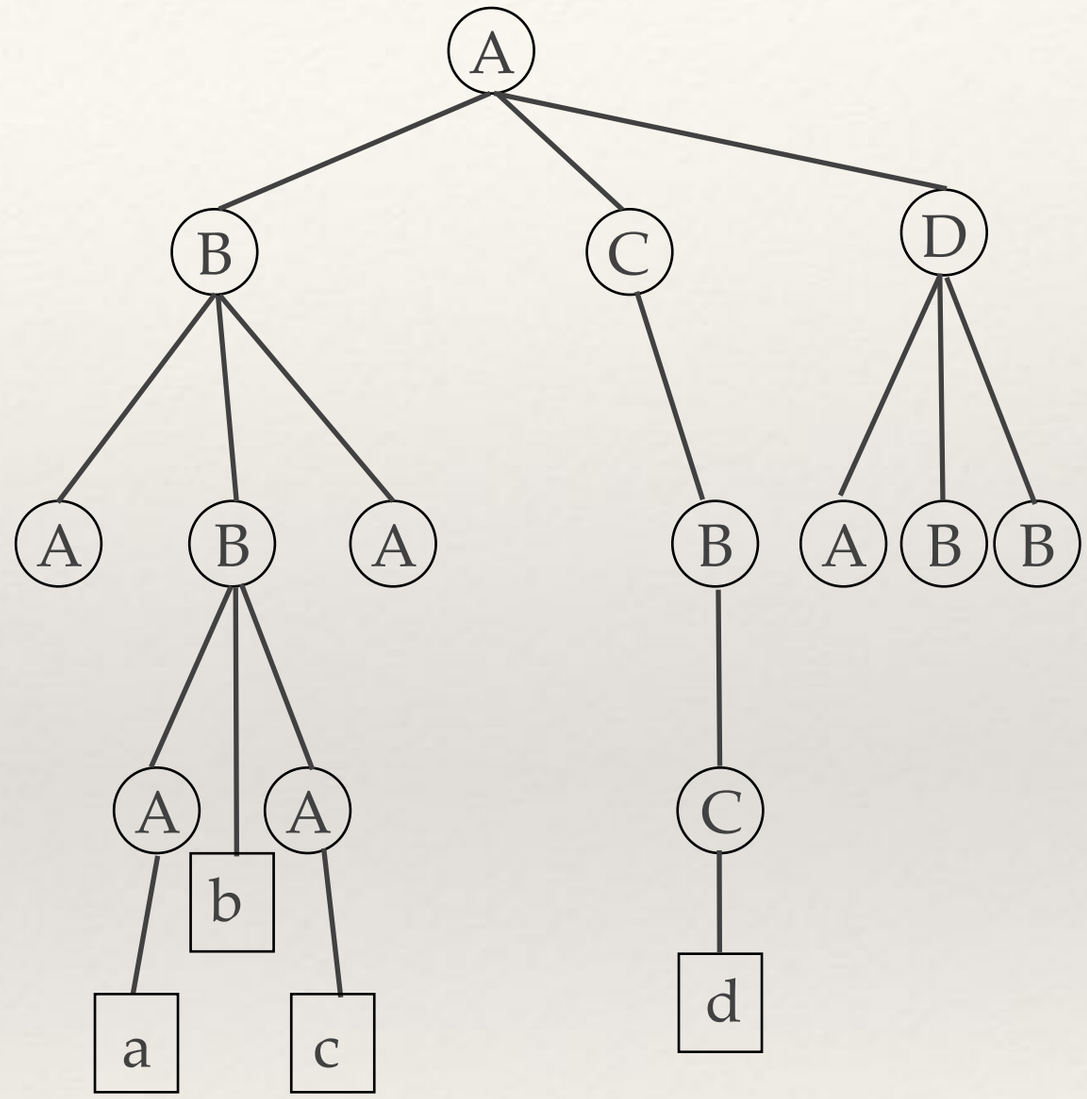


Getting text

Query:

```
//child::*[position()=1]/text()
```

returns: [“a”,”d”]

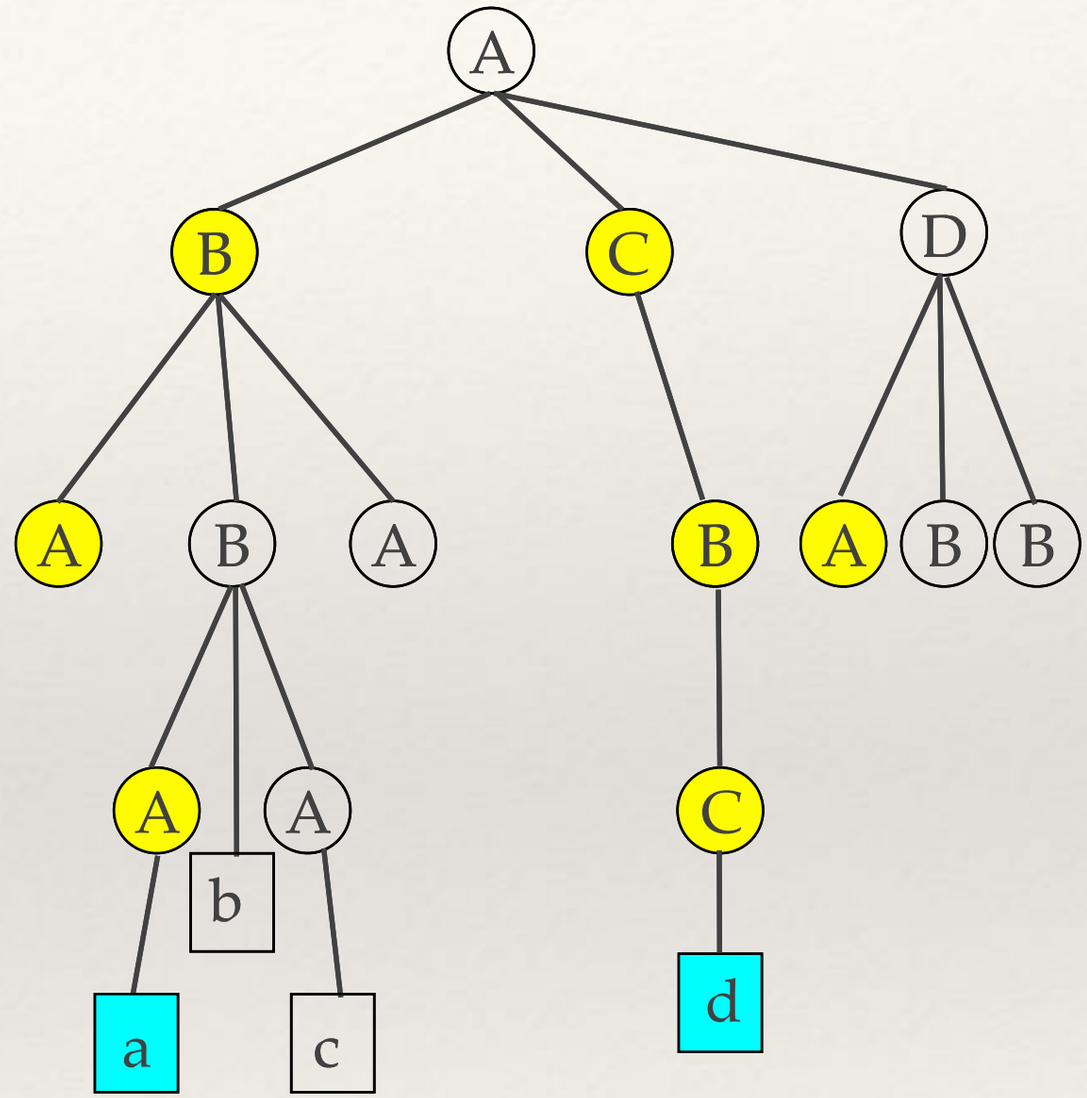


Getting text

Query:

```
//child::*[position()=1]/text()
```

returns: [“a”,”d”]



Online resource

https://www.freeformatter.com/xpath-tester.html

Online resource

https://www.freeformatter.com/xpath-tester.html

```
<A>
     <B>
          <A/>
          <B>
               <A>a</A>
               b
               <A>c</A>
          </B>
          <A/>
     </B>
     <C>
          <B>
               <C>d</C>
          </B>
     </C>
     <D>
          <A/>
          <B/>
          <B/>
     </D>
</A>
```

Web scraping

Web scraping

Web scraping

- Conceptually it amounts to

Web scraping

- Conceptually it amounts to
  - downloading a set of web pages (*crawling*)

Web scraping

- Conceptually it amounts to
  - downloading a set of web pages (*crawling*)
  - extracting information from the downloaded web pages

Web scraping

- Conceptually it amounts to
  - downloading a set of web pages (*crawling*)
  - extracting information from the downloaded web pages
- The two activities influence each other

Web scraping

- Conceptually it amounts to
  - downloading a set of web pages (*crawling*)
  - extracting information from the downloaded web pages
- The two activities influence each other
  - extracting information can prompt to downloading more pages, etc. (like in a web crawler)

Web scraping

Web scraping

- Taxonomy

Web scraping

- Taxonomy
  - breadth of search (how many webpages are targeted?)

Web scraping

- Taxonomy
  - breadth of search (how many webpages are targeted?)
  - depth of search (how complex is the type of information to be gathered?)

Web scraping

- Taxonomy
  - breadth of search (how many webpages are targeted?)
  - depth of search (how complex is the type of information to be gathered?)
  - interaction with human operator (fully automated? partly manual? totally manual?)

Scraping vs. crawling

Scraping vs. crawling

- *Crawling* is usually aimed at bulk download of pages from unknown websites and of unknown content

Scraping vs. crawling

- *Crawling* is usually aimed at bulk download of pages from unknown websites and of unknown content
  - Almost no attempt to information extraction (apart for indexing)

Scraping vs. crawling

- *Crawling* is usually aimed at bulk download of pages from unknown websites and of unknown content
  - Almost no attempt to information extraction (apart for indexing)
- *Scraping* is targeted to smaller sets of pages (e.g., single websites) or to specific information targets (e.g., collecting e-mail addresses [*contact scraping*])

Scraping vs. crawling

- *Crawling* is usually aimed at bulk download of pages from unknown websites and of unknown content
  - Almost no attempt to information extraction (apart for indexing)
- *Scraping* is targeted to smaller sets of pages (e.g., single websites) or to specific information targets (e.g., collecting e-mail addresses [*contact scraping*])
  - In both cases, specific forms of pattern matching are required

Scraping vs. crawling

- *Crawling* is usually aimed at bulk download of pages from unknown websites and of unknown content
  - Almost no attempt to information extraction (apart for indexing)
- *Scraping* is targeted to smaller sets of pages (e.g., single websites) or to specific information targets (e.g., collecting e-mail addresses [*contact scraping*])
  - In both cases, specific forms of pattern matching are required
  - Often, scraping is an activity that depends strongly on the structure of the HTML pages that are being visited

Is scraping legal?

Is scraping legal?

- It may violate the *terms of use* of some websites. Enforceability is less obvious, though…

Is scraping legal?

- It may violate the *terms of use* of some websites. Enforceability is less obvious, though…
- **United States**: unclear… Scrapers can be accused of *trespass to chattel* [“violazione di proprietà”], but it is difficult to prove that the scraping caused a damage to the plaintiff [“querelante”]

Is scraping legal?

- It may violate the *terms of use* of some websites. Enforceability is less obvious, though…
- **United States**: unclear… Scrapers can be accused of *trespass to chattel* [“violazione di proprietà”], but it is difficult to prove that the scraping caused a damage to the plaintiff [“querelante”]
- **Australia**: some form of scraping (e.g. contact scraping) have been declared illegal

Is scraping legal?

- It may violate the *terms of use* of some websites. Enforceability is less obvious, though…
- **United States**: unclear… Scrapers can be accused of *trespass to chattel* [“violazione di proprietà”], but it is difficult to prove that the scraping caused a damage to the plaintiff [“querelante”]
- **Australia**: some form of scraping (e.g. contact scraping) have been declared illegal
- **EU**: some sentences seem to be in favour of crawling, but in a contradictory way

Scraping social networks

Scraping social networks

- Scraping social networks (facebook, twitter etc.) is typically **illegal** and **unadvisable**

Scraping social networks

- Scraping social networks (facebook, twitter etc.) is typically **illegal** and **unadvisable**
- Social networking platforms offer usually specific APIs to access to (some of their) data

Scraping social networks

- Scraping social networks (facebook, twitter etc.) is typically **illegal** and **unadvisable**
- Social networking platforms offer usually specific APIs to access to (some of their) data
- Scraping / crawling should only be used to extract information from websites that do not offer this kind of services