Web Data Models

XPath: Equivalence, Containment

Silviu Maniu

UNIVERSITÉ PARIS SUD

Comprendre le monde, construire l'avenir

université PARIS-SACLAY

XPath Queries

Ways to optimise XPath processing:

- better algorithms (previous lecture)
- query rewriting for more efficient queries
- query containment

Rewriting XPath Queries

This lecture:

- path equivalence
- transforming backward axes into forward axes
- query containment

Preamble: Equality in XPath

XPath 1.0 equality

```
//a[b/d = c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it compare?**

Preamble: Equality in XPath

XPath 1.0 equality

```
//a[b/d = c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it compare?** string value of nodes

Preamble: Equality in XPath

XPath 1.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?**

Preamble: Equality in XPath

XPath 1.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?**

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

Preamble: Equality in XPath

XPath 2.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?**

Preamble: Equality in XPath

XPath 2.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?** identical nodes

Preamble: Equality in XPath

XPath 2.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?** identical nodes

how would it work in XPath 1.0 ?

Preamble: Equality in XPath

XPath 2.0 equality

```
//a[b/d == c/d]
```

```
<a>
  <b>
    <d>t1</d>
    <d>t2</d>
    <d>t3</d>
  </b>
  <b>
    <d>t4</d>
    <d>t5</d>
    <d>t1</d>
  </b>
</a>
```

— **what does it select?** identical nodes

count(p1 | p2) < count(p1) + count(p2)

XPath Equivalence

Fragment of XPath (2.0)

$$
\begin{aligned}
\mathit{path} &::= \mathit{path}\ \texttt{|}\ \mathit{path} \mid \texttt{/}\ \mathit{path} \mid \mathit{path}\ \texttt{/}\ \mathit{path} \mid \mathit{path}\ \texttt{[}\ \mathit{qualif}\ \texttt{]} \mid \mathit{axis}\ \texttt{::}\ \mathit{nodetest} \mid \perp\ .\\
\mathit{qualif} &::= \mathit{qualif}\ \texttt{and}\ \mathit{qualif} \mid \mathit{qualif}\ \texttt{or}\ \mathit{qualif} \mid \texttt{(}\ \mathit{qualif}\ \texttt{)} \mid\\
&\quad\ \ \mathit{path}\ \texttt{=}\ \mathit{path} \mid \mathit{path}\ \texttt{==}\ \mathit{path} \mid \mathit{path}\ .\\
\mathit{axis} &::= \mathit{reverse\_axis} \mid \mathit{forward\_axis}\ .\\
\mathit{reverse\_axis} &::= \texttt{parent} \mid \texttt{ancestor} \mid \texttt{ancestor-or-self} \mid\\
&\quad\ \ \texttt{preceding} \mid \texttt{preceding-sibling}\ .\\
\mathit{forward\_axis} &::= \texttt{self} \mid \texttt{child} \mid \texttt{descendant} \mid \texttt{descendant-or-self} \mid\\
&\quad\ \ \texttt{following} \mid \texttt{following-sibling}\ .\\
\mathit{nodetest} &::= \mathit{tagname} \mid \texttt{*} \mid \texttt{text()} \mid \texttt{node()}\ .
\end{aligned}
$$

S

- path starting with **/** is called absolute path

XPath Equivalence

p1 equivalent to p2 $p_1 \equiv p_2$

- for any document D and any context node N of D, *p1* evaluated on D with context N gives the same result as *p2* evaluated on D with context N

XPath Equivalence

Relative to absolute path

$$p_1 \equiv p_2 \Rightarrow /p_1 \equiv /p_2$$

XPath Equivalence

Adjunct of a path

$$p_1 \equiv p_2 \Rightarrow /p_1/p \equiv /p_2/p$$

$$p_1 \equiv p_2 \Rightarrow p/p_1 \equiv p/p_1$$

$$p_1 \equiv p_2 \Rightarrow p_1[q] \equiv p_2[q]$$

$$p_1 \equiv p_2 \Rightarrow p[p_1] \equiv p[p_2]$$

XPath Equivalence

Qualifier flattening

$$p[p_1/p_2] \equiv p[p_1[p_2]]$$

XPath Equivalence

-or-self axis decompositions

$$\texttt{ancestor-or-self::}n \equiv \texttt{ancestor::}n \mid \texttt{self::}n$$

$$\texttt{descendant-or-self::}n \equiv \texttt{descendant::}n \mid \texttt{self::}n$$

XPath Equivalence

Joins (= or ==)

$$p[p_1 \theta / p_2] \equiv p[p_1[\texttt{self} :: \texttt{node}() \theta / p_2]]$$

XPath Equivalence

Why Path equivalence?

XPath Equivalence

Why Path equivalence?

- big XML documents cannot be kept in memory
- hence streaming algorithms are a better fit for path processing
- but reverse/bacward axes are bad for streaming algorithms (why?)

Removing Backward Axes

Dual of a backward axis

axis	dual
`parent`	
`ancestor`	
`ancestor-or-self`	
`preceding`	
`preceding-sibling`	

Removing Backward Axes

Dual of a backward axis

axis	dual
`parent`	`child`
`ancestor`	`descendant`
`ancestor-or-self`	`descendant-or-self`
`preceding`	`following`
`preceding-sibling`	`following-sibling`

Rewrite Rules

General rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$p[a_m\texttt{::}m\texttt{/}s] \equiv p[\texttt{/descendant::}m[s]\texttt{/}b_m\texttt{::node() == self::node()}] \tag{1}$$

$$\texttt{/}p\texttt{/}a_n\texttt{::}n\texttt{/}a_m\texttt{::}m \equiv \texttt{/descendant::}m[b_m\texttt{::}n \texttt{ == /}p\texttt{/}a_n\texttt{::}n] \tag{2}$$

$$\texttt{/}a_n\texttt{::}n\texttt{/}a_m\texttt{::}m \equiv \texttt{/descendant::}m[b_m\texttt{::}n \texttt{ == /}a_n\texttt{::}n] \tag{2a}$$

Which rewrite rule?

```
/descendant::price/preceding::name
```

Rewrite Rules

Rewrite rule Lemma (1)

- *Let a be one of the axes* `parent`, `ancestor`, `preceding`, `preceding-sibling`, `self`, `following`, *or* `following-sibling`. *Then the following holds:*

$$\texttt{/}a\texttt{::}n \equiv \begin{cases} \texttt{/} & \textit{if } a = \texttt{self} \textit{ and } n = \texttt{node()} \\ \bot & \textit{otherwise} \end{cases}$$

- *Let a be the* `preceding` *or* `ancestor` *axis. Then the following equivalences hold:*

$$\texttt{/child::}m\texttt{/}a\texttt{::}n \equiv \begin{cases} \texttt{/self::node()[child::}m\texttt{]} & \textit{if } a = \texttt{ancestor} \textit{ and } n = \texttt{node()} \\ \bot & \textit{otherwise} \end{cases}$$

$$\texttt{/child::}m\texttt{[}a\texttt{::}n\texttt{]} \equiv \begin{cases} \texttt{/child::}m & \textit{if } a = \texttt{ancestor} \textit{ and } n = \texttt{node()} \\ \bot & \textit{otherwise} \end{cases}$$

Rewrite Rules

Parent rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$
\begin{aligned}
\texttt{descendant::}n\texttt{/parent::}m &\equiv \texttt{descendant-or-self::}m\texttt{[child::}n\texttt{]} && (3)\\
\texttt{child::}n\texttt{/parent::}m &\equiv \texttt{self::}m\texttt{[child::}n\texttt{]} && (4)\\
p\texttt{/self::}n\texttt{/parent::}m &\equiv p\texttt{[self::}n\texttt{]/parent::}m && (5)\\
p\texttt{/following-sibling::}n\texttt{/parent::}m &\equiv p\texttt{[following-sibling::}n\texttt{]/parent::}m && (6)\\
p\texttt{/following::}n\texttt{/parent::}m &\equiv p\texttt{/following::}m\texttt{[child::}n\texttt{]} && (7)\\
&\mid p\texttt{/ancestor-or-self::*[following-sibling::}n\texttt{]}\\
&\quad \texttt{/parent::}m
\end{aligned}
$$

$$
\begin{aligned}
\texttt{descendant::}n\ \texttt{[parent::}m\texttt{]} &\equiv \texttt{descendant-or-self::}m\texttt{/child::}n && (8)\\
\texttt{child::}n\texttt{[parent::}m\texttt{]} &\equiv \texttt{self::}m\texttt{/child::}n && (9)\\
p\texttt{/self::}n\texttt{[parent::}m\texttt{]} &\equiv p\texttt{[parent::}m\texttt{]/self::}n && (10)\\
p\texttt{/following-sibling::}n\texttt{[parent::}m\texttt{]} &\equiv p\texttt{[parent::}m\texttt{]/following-sibling::}n && (11)\\
p\texttt{/following::}n\texttt{[parent::}m\texttt{]} &\equiv p\texttt{/following::}m\texttt{/child::}n && (12)\\
&\mid p\texttt{/ancestor-or-self::*[parent::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n
\end{aligned}
$$

Which rewrite rule?

`/descendant::editor[parent::journal]`

Rewrite Rules

Ancestor rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$
\begin{aligned}
p/\texttt{descendant::}n/\texttt{ancestor::}m &\equiv p[\texttt{descendant::}n]/\texttt{ancestor::}m && (13)\\
&\mid p/\texttt{descendant-or-self::}m[\texttt{descendant::}n] \\
/\texttt{descendant::}n/\texttt{ancestor::}m &\equiv /\texttt{descendant-or-self::}m[\texttt{descendant::}n] && (13a)\\
p/\texttt{child::}n/\texttt{ancestor::}m &\equiv p[\texttt{child::}n]/\texttt{ancestor-or-self::}m && (14)\\
p/\texttt{self::}n/\texttt{ancestor::}m &\equiv p[\texttt{self::}n]/\texttt{ancestor::}m && (15)\\
p/\texttt{following-sibling::}n/\texttt{ancestor::}m &\equiv p[\texttt{following-sibling::}n]/\texttt{ancestor::}m && (16)\\
p/\texttt{following::}n/\texttt{ancestor::}m &\equiv p/\texttt{following::}m[\texttt{descendant::}n] && (17)\\
&\mid p/\texttt{ancestor-or-self::*} \\
&\quad [\texttt{following-sibling::*/descendant-or-self::}n] \\
&\quad /\texttt{ancestor::}m
\end{aligned}
$$

$$
\begin{aligned}
p/\texttt{descendant::}n[\texttt{ancestor::}m] &\equiv p[\texttt{ancestor::}m]/\texttt{descendant::}n && (18)\\
&\mid p/\texttt{descendant-or-self::}m/\texttt{descendant::}n \\
/\texttt{descendant::}n[\texttt{ancestor::}m] &\equiv /\texttt{descendant-or-self::}m/\texttt{descendant::}n && (18a)\\
p/\texttt{child::}n[\texttt{ancestor::}m] &\equiv p[\texttt{ancestor-or-self::}m]/\texttt{child::}n && (19)\\
p/\texttt{self::}n[\texttt{ancestor::}m] &\equiv p[\texttt{ancestor::}m]/\texttt{self::}n && (20)\\
p/\texttt{following-sibling::}n[\texttt{ancestor::}m] &\equiv p[\texttt{ancestor::}m]/\texttt{following-sibling::}n && (21)\\
p/\texttt{following::}n[\texttt{ancestor::}m] &\equiv p/\texttt{following::}m/\texttt{descendant::}n && (22)\\
&\mid p/\texttt{ancestor-or-self::*}[\texttt{ancestor::}m] \\
&\quad /\texttt{following-sibling::*/descendant-or-self::}n
\end{aligned}
$$

Rewrite Rules

Preceding-sibling rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$
\begin{aligned}
\texttt{descendant::}n\texttt{/preceding-sibling::}m &\equiv \texttt{descendant::}m\texttt{[following-sibling::}n\texttt{]} && (23)\\
\texttt{child::}n\texttt{/preceding-sibling::}m &\equiv \texttt{child::}m\texttt{[following-sibling::}n\texttt{]} && (24)\\
p\texttt{/self::}n\texttt{/preceding-sibling::}m &\equiv p\texttt{[self::}n\texttt{]/preceding-sibling::}m && (25)\\
p\texttt{/following-sibling::}n\texttt{/preceding-sibling::}m &\equiv p\texttt{[self::}m\texttt{/following-sibling::}n\texttt{]} && (26)\\
&\mid p\texttt{[following-sibling::}n\texttt{]/preceding-sibling::}m\\
&\mid p\texttt{/following-sibling::}m\texttt{[following-sibling::}n\texttt{]}\\
p\texttt{/following::}n\texttt{/preceding-sibling::}m &\equiv p\texttt{/following::}m\texttt{[following-sibling::}n\texttt{]} && (27)\\
&\mid p\texttt{/ancestor-or-self::*[following-sibling::}n\texttt{]}\\
&\quad \texttt{/preceding-sibling::}m\\
&\mid p\texttt{/ancestor-or-self::}m\texttt{[following-sibling::}n\texttt{]}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{descendant::}n\texttt{[preceding-sibling::}m\texttt{]} &\equiv \texttt{descendant::}m\texttt{/following-sibling::}n && (28)\\
\texttt{child::}n\texttt{[preceding-sibling::}m\texttt{]} &\equiv \texttt{child::}m\texttt{/following-sibling::}n && (29)\\
p\texttt{/self::}n\texttt{[preceding-sibling::}m\texttt{]} &\equiv p\texttt{[self::}n\texttt{]/following-sibling::}m && (30)\\
p\texttt{/following-sibling::}n\texttt{[preceding-sibling::}m\texttt{]} &\equiv p\texttt{[self::}m\texttt{]/following-sibling::}n && (31)\\
&\mid p\texttt{/following-sibling::}m\texttt{/following-sibling::}n\\
&\mid p\texttt{[preceding-sibling::}m\texttt{]/following-sibling::}n\\
p\texttt{/following::}n\texttt{[preceding-sibling::}m\texttt{]} &\equiv p\texttt{/following::}m\texttt{/following-sibling::}n && (32)\\
&\mid p\texttt{/ancestor-or-self::*[preceding-sibling::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n\\
&\mid p\texttt{/ancestor-or-self::/following-sibling::}n
\end{aligned}
$$

Rewrite Rules

Preceding rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$\begin{aligned}
p/\texttt{descendant::}n/\texttt{preceding::}m &\equiv p[\texttt{descendant::}n]/\texttt{preceding::}m && (33)\\
&\quad \mid p/\texttt{child::*}\\
&\quad\ [\texttt{following-sibling::*/descendant-or-self::}n]\\
&\quad\ /\texttt{descendant-or-self::}m\\
/\texttt{descendant::}n/\texttt{preceding::}m &\equiv /\texttt{descendant::}m[\texttt{following::}n] && (33a)\\
p/\texttt{child::}n/\texttt{preceding::}m &\equiv p[\texttt{child::}n]/\texttt{preceding::}m && (34)\\
&\quad \mid p/\texttt{child::*}[\texttt{following-sibling::}n]\\
&\quad\ /\texttt{descendant-or-self::}m\\
p/\texttt{self::}n/\texttt{preceding::}m &\equiv p[\texttt{self::}n]/\texttt{preceding::}m && (35)\\
p/\texttt{following-sibling::}n/\texttt{preceding::}m &\equiv p[\texttt{following-sibling::}n]/\texttt{preceding::}m && (36)\\
&\quad \mid p/\texttt{following-sibling::*}[\texttt{following-sibling::}n]\\
&\quad\ /\texttt{descendant-or-self::}m\\
&\quad \mid p[\texttt{following-sibling::}n]/\texttt{descendant-or-self::}m\\
p/\texttt{following::}n/\texttt{preceding::}m &\equiv p[\texttt{following::}n]/\texttt{preceding::}m && (37)\\
&\quad \mid p/\texttt{following::}m[\texttt{following::}n]\\
&\quad \mid p[\texttt{following::}n]/\texttt{descendant-or-self::}m
\end{aligned}$$

Which rewrite rule?

`/descendant::price/preceding::name`

Rewrite Rules

Preceding rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$
\begin{aligned}
p/\texttt{descendant::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/descendant::}n && (38)\\
&\mid p/\texttt{child::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::*/descendant-or-self::}n\\
\texttt{/descendant::}n\texttt{[preceding::}m\texttt{]} &\equiv \texttt{/descendant::}m\texttt{/following::}n && (38a)\\
p/\texttt{child::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/child::}n && (39)\\
&\mid p/\texttt{child::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n\\
p/\texttt{self::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/self::}n && (40)\\
p/\texttt{following-sibling::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/following-sibling::}n && (41)\\
&\mid p/\texttt{following-sibling::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n\\
&\mid p\texttt{[descendant-or-self::}m\texttt{]/following-sibling::}n\\
p/\texttt{following::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/following::}n && (42)\\
&\mid p/\texttt{following::}m\texttt{/following::}n\\
&\mid p\texttt{[descendant-or-self::}m\texttt{]/following::}n
\end{aligned}
$$

Rewrite Rules

Preceding rules (a_m reverse axis, b_m dual axis, a_n forward axis, n, m node tests)

$$
\begin{aligned}
p/\texttt{descendant::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/descendant::}n && (38)\\
&\mid p/\texttt{child::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::*/descendant-or-self::}n\\
\texttt{/descendant::}n\texttt{[preceding::}m\texttt{]} &\equiv \texttt{/descendant::}m\texttt{/following::}n && (38a)\\
p/\texttt{child::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/child::}n && (39)\\
&\mid p/\texttt{child::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n\\
p/\texttt{self::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/self::}n && (40)\\
p/\texttt{following-sibling::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/following-sibling::}n && (41)\\
&\mid p/\texttt{following-sibling::*[descendant-or-self::}m\texttt{]}\\
&\quad \texttt{/following-sibling::}n\\
&\mid p\texttt{[descendant-or-self::}m\texttt{]/following-sibling::}n\\
p/\texttt{following::}n\texttt{[preceding::}m\texttt{]} &\equiv p\texttt{[preceding::}m\texttt{]/following::}n && (42)\\
&\mid p/\texttt{following::}m\texttt{/following::}n\\
&\mid p\texttt{[descendant-or-self::}m\texttt{]/following::}n
\end{aligned}
$$

Rewrite Rules

Two “rule sets”:

- RuleSet 1: (1),(2),(2a) and Lemma (1)
- RuleSet 2: (3)—(42) and Lemma (1)

Rewrite Theorems

Theorem 1

For an absolute path *p* in which no joins occur, there exists an equivalent path *p'* with no reverse steps. Using RuleSet 1, *p*' has a length and can be computed in linear time in the length of *p*.

Rewrite Theorems

Theorem 2

For an absolute path *p* in which no joins occur, there exists an equivalent path *p'* with no reverse steps. Using RuleSet 2, *p*' has a length and can be computed in exponential time in the length of *p*.

Rewrite Algorithm

rare **- R**everse **A**xis **Re**moval

Let ξ = $\mathrm{RuleSet}_1$ or $\mathrm{RuleSet}_2$.

Auxiliary functions:

match(p): returns the result of a rule application from ξ to the first reverse location step in p.

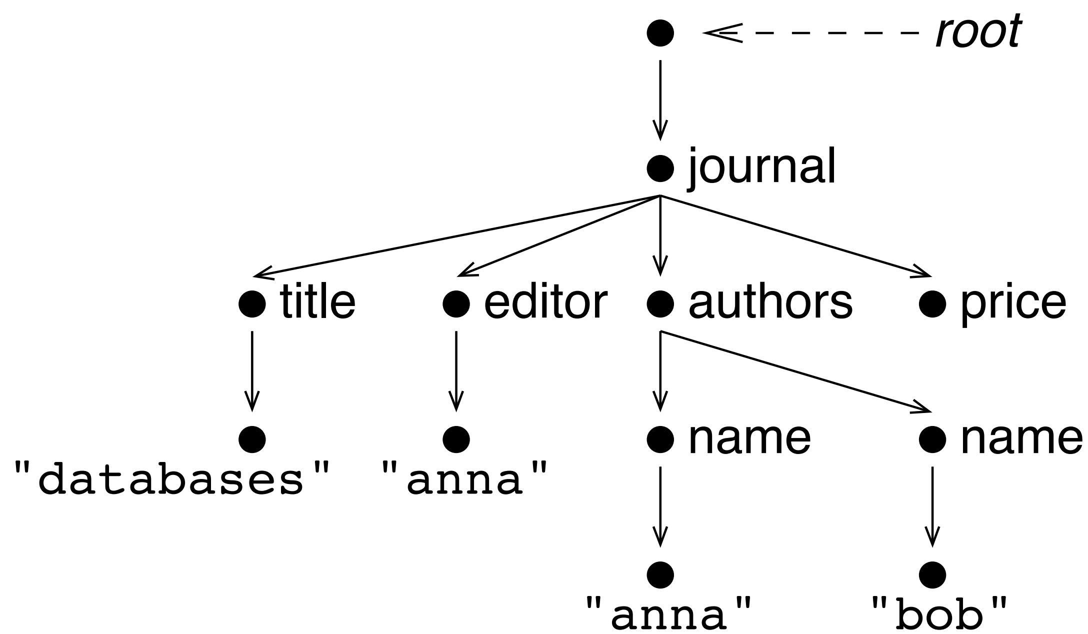
apply-lemmas(p): returns p if Rules (3.1.1-8) are not applicable to p. Otherwise, it returns the result of the repeated application of Rules (3.1.1-8) to p.

union-flattening(p): returns a path equivalent to p with unions at top level only.

***rare*(p)**
Input: p {absolute location path without qualifiers containing RR joins}.

$p \leftarrow$ *apply-lemmas*(p).
$p \leftarrow$ *union-flattening*$(p) = U_1 \mid \ldots \mid U_n \ (n \geq 1)$.
$S \leftarrow$ empty stack.
for $i \leftarrow 1$ to n **do**
  push(U_i, S).
end for
$p' \leftarrow \perp$. {initialization}
while *not*(*empty*(S)) **do**
  $U \leftarrow$ *pop*(S).
  while U contains a reverse step **do**
    $U \leftarrow$ *match*(U).
    $U \leftarrow$ *apply-lemmas*(U).
    $U \leftarrow$ *union-flattening*$(U) = V_1 \mid \ldots \mid V_n \ (n \geq 1)$.
    for $i \leftarrow 2$ to n **do**
      push(V_i, S).
    end for
    $U \leftarrow V_1$.
  end while
  $p' \leftarrow p' \mid U$.
end while
Output: p' {location path without reverse axes equivalent to p}.

Rewrite Example




```
<journal>
  <title>databases</title>
  <editor>anna</editor>
  <authors>
    <name>anna</name>
    <name>bob</name>
  </authors>
  <price />
</journal>
```

$$p[a_m\texttt{::}m/s] \equiv p[\texttt{/descendant::}m[s]/b_m\texttt{::node() == self::node()}] \tag{1}$$

$$/p/a_n\texttt{::}n/a_m\texttt{::}m \equiv \texttt{/descendant::}m[b_m\texttt{::}n \texttt{ == } /p/a_n\texttt{::}n] \tag{2}$$

$$/a_n\texttt{::}n/a_m\texttt{::}m \equiv \texttt{/descendant::}m[b_m\texttt{::}n \texttt{ == } /a_n\texttt{::}n] \tag{2a}$$

RuleSet 1

`/descendant::name/preceding::title[ancestor::journal]`

XPath Containment

Intuitive definition:

Given two paths p, q: are all nodes selected by p also selected by q?

XPath Containment

- if a document matches *p*, and *p* is contained in *q*, the we know the document also matches *q*

- if a document does not match *q*, and *p* is contained in *q*, then we know the document does not match *p*

XPath Containment

Applications:

- decrease online time for publish/subscribe systems
- decrease query time by using materialised intermediate results
- query optimization: ruling out queries with empty results,…

XPath Containment

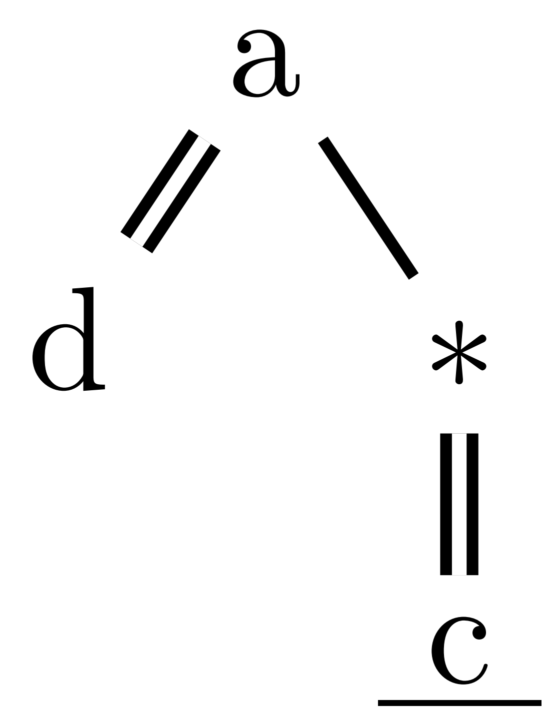
Types of containment

0-containment	$p \subseteq_0 q$	for every tree, if p selects a node then so does *q*
1-containment	$p \subseteq_1 q$	for every tree, all nodes selected by *p* are also selected by *q*
2-containment	$p \subseteq_2 q$	for every tree and every context node **N**, all nodes selected by *p* from **N**, are also selected by *q* from **N**

Pattern Trees

XPath(/,//,*,[])

```
a[.//d]/*//c
```

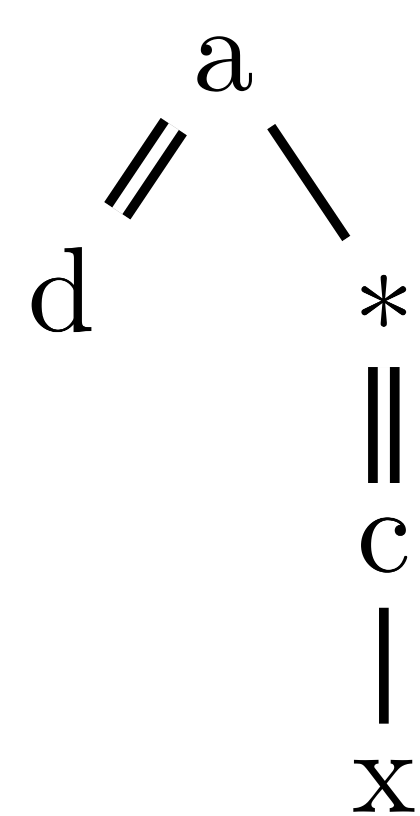


selection pattern tree

Pattern Trees

XPath(/,//,*,[])

`a[.//d]/*//c`



match pattern tree

Containment Check Techniques

1. Canonical Model Technique
2. Homomorphism Technique
3. Automaton Technique
4. Chase Technique

Canonical Model

If there exists a tree that matches p but not q, then a tree exists of size polynomial in the size of p and q.

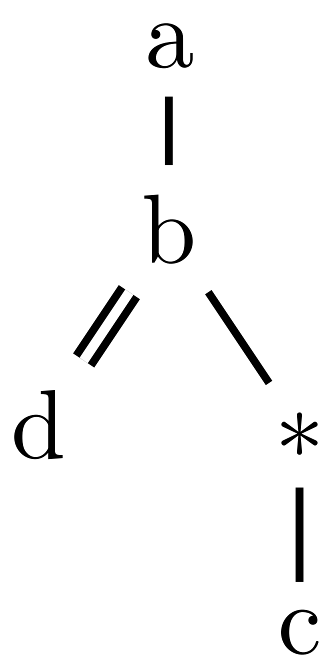
Canonical Model

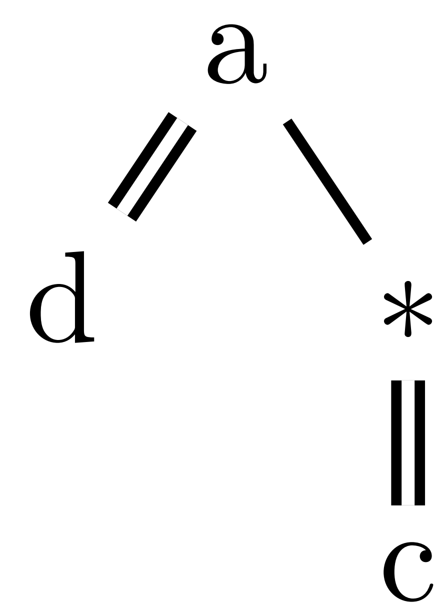
Method:

1. Find all possible enumerations of the query

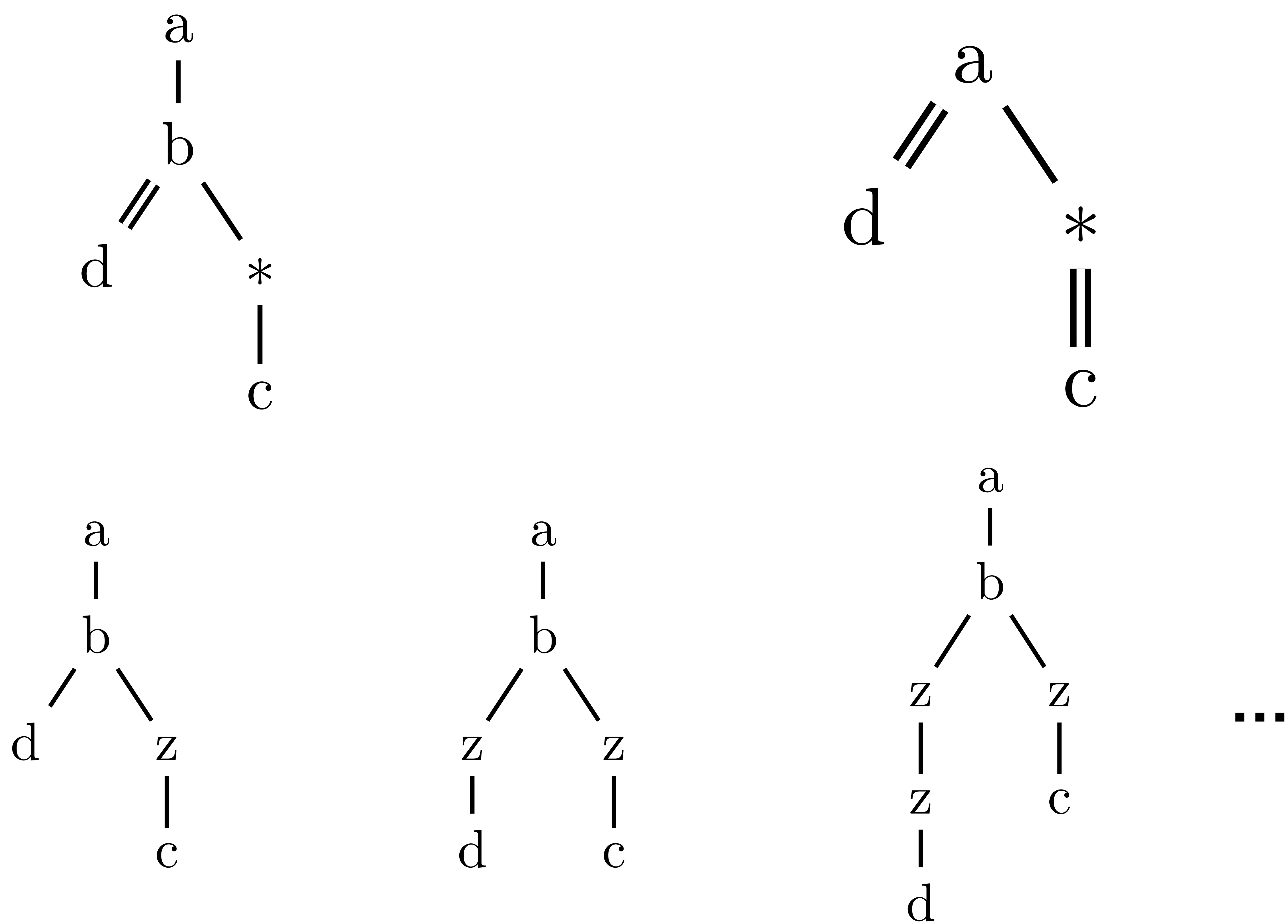
2. Construct a counter example tree, by replacing in *p*, every ***** by a new symbol (say "z"), every **//** by $\{z/, z/z/, z/z/z/, \ldots, z/z/\ldots/z\}$

Canonical Model





Canonical Model



Homomorphism

$\boldsymbol{h}$: map each node of *q*'s pattern tree Q to a node of *p*'s pattern tree P s.t.:

- $root(Q)$ mapped to $root(B)$
- if (u,v) child edge of Q then $(h(u),h(v))$ is child-edge of P
- if (u,v) descendant edge of Q, then $h(v)$ is "below" $h(u)$ in P
- if u is labeled, then $h(u)$ is also labeled the same (except for *****)

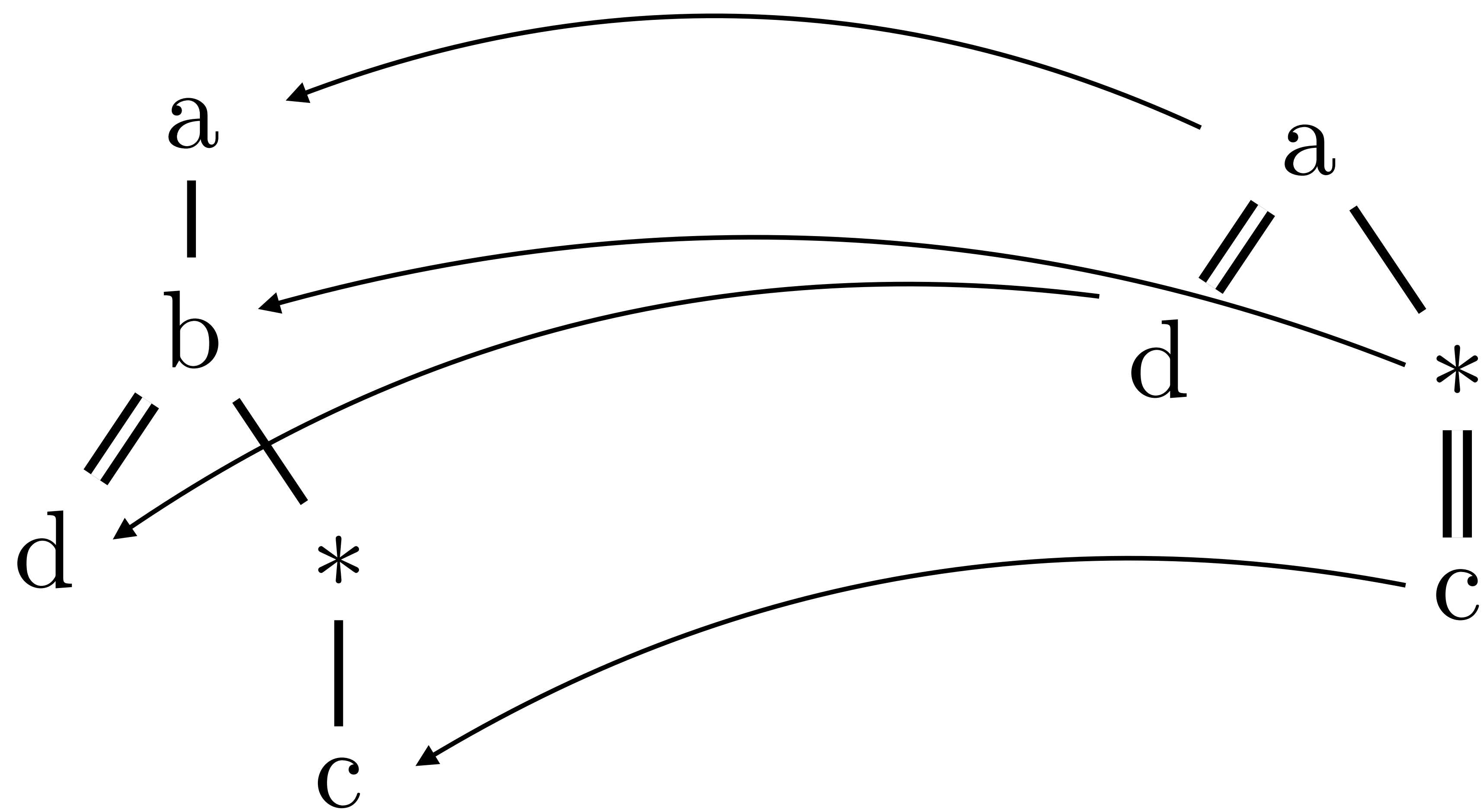
Homomorphism

Theorem

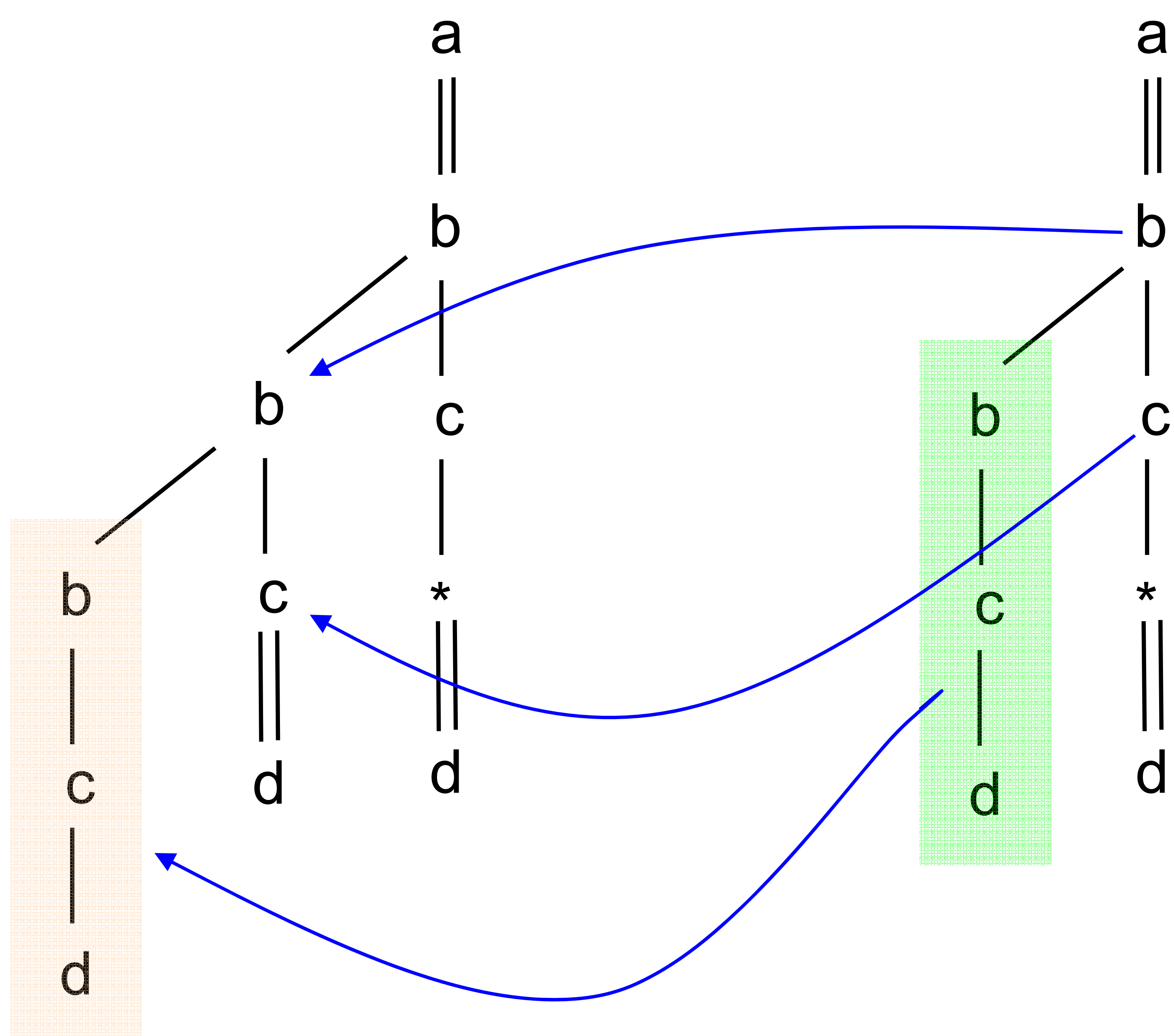
For p, q expressions in $\mathrm{XPath}(/,//,[\,])$, p is 0-contained in q if and only if there is a homomorphism from $\mathbf{Q}$ to $\boldsymbol{P}$.

Homomorphism

Homomorphism



Homomorphism



The theorem is not generally true in XPath with *

The Automata Technique

For every DTD there is a tree automaton which recognises the corresponding document trees.

In the same way, for any *p* in $\mathrm{XPath}(/,//,[\,],*,|)$ there exists a (non-deterministic) automaton which accepts a tree iff *p* matches the tree.

The Automata Technique

Theorem

Containment test of $\mathrm{XPath}(/,//,[\,],*,|)$ in the presence of DTDs can be solved in **EXPTIME**.

The Automata Technique

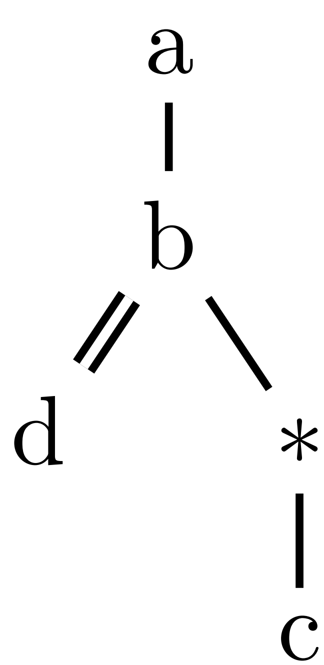
Theorem

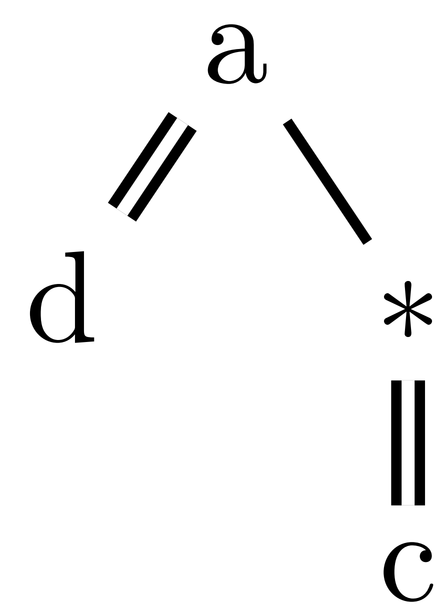
Containment test of $\mathrm{XPath}(/,//)$ in the presence of DTDs can be solved in **PTIME**.

The Chase Technique

The Chase: classic relational DB technique to check query containment in the presence of *integrity contraints*

The Chase Technique

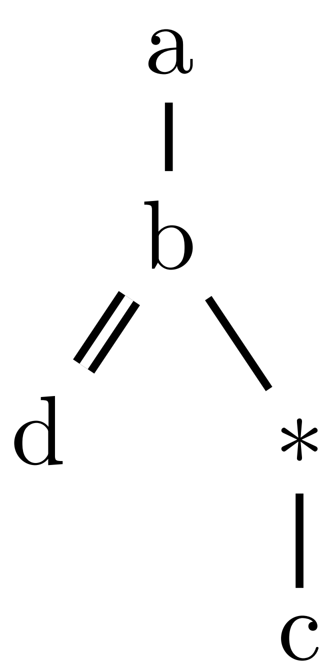


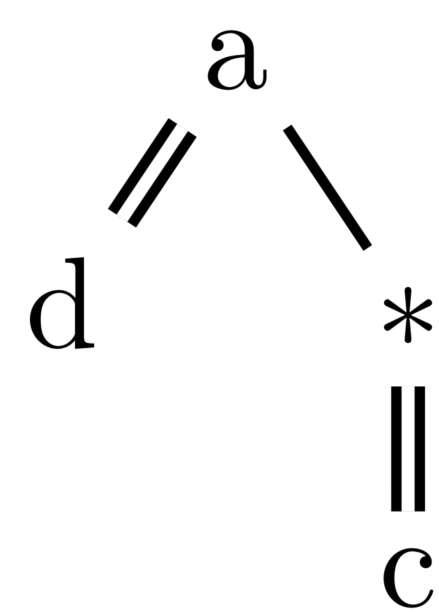


DTD

$$\text{root} \rightarrow a^*$$
$$a \rightarrow b^* \mid c^*$$
$$b \rightarrow d^+ c^+$$
$$c \rightarrow b?c?$$

The Chase Technique





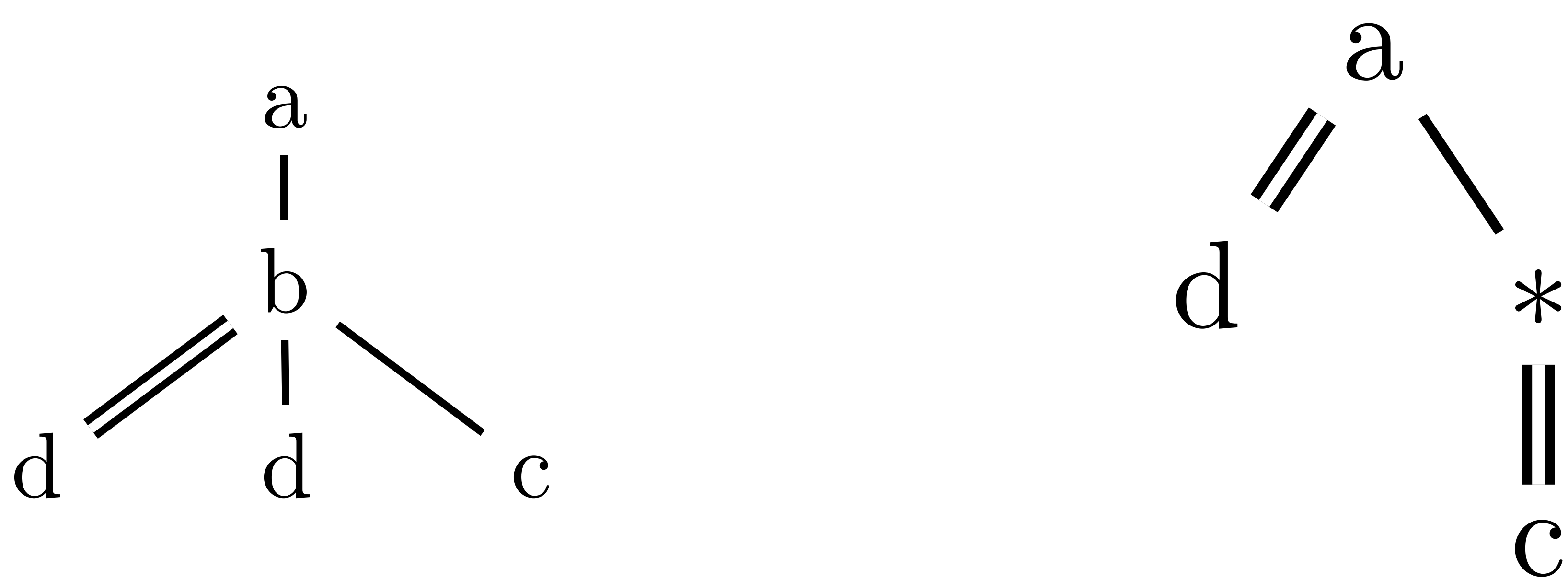
$c_1 : b \rightarrow d$

$c_2 : b \rightarrow c$

DTD

$$\begin{aligned} \text{root} &\longrightarrow a^* \\ a &\longrightarrow b^* \mid c^* \\ b &\longrightarrow d^+ c^+ \\ c &\longrightarrow b?c? \end{aligned}$$

The Chase Technique



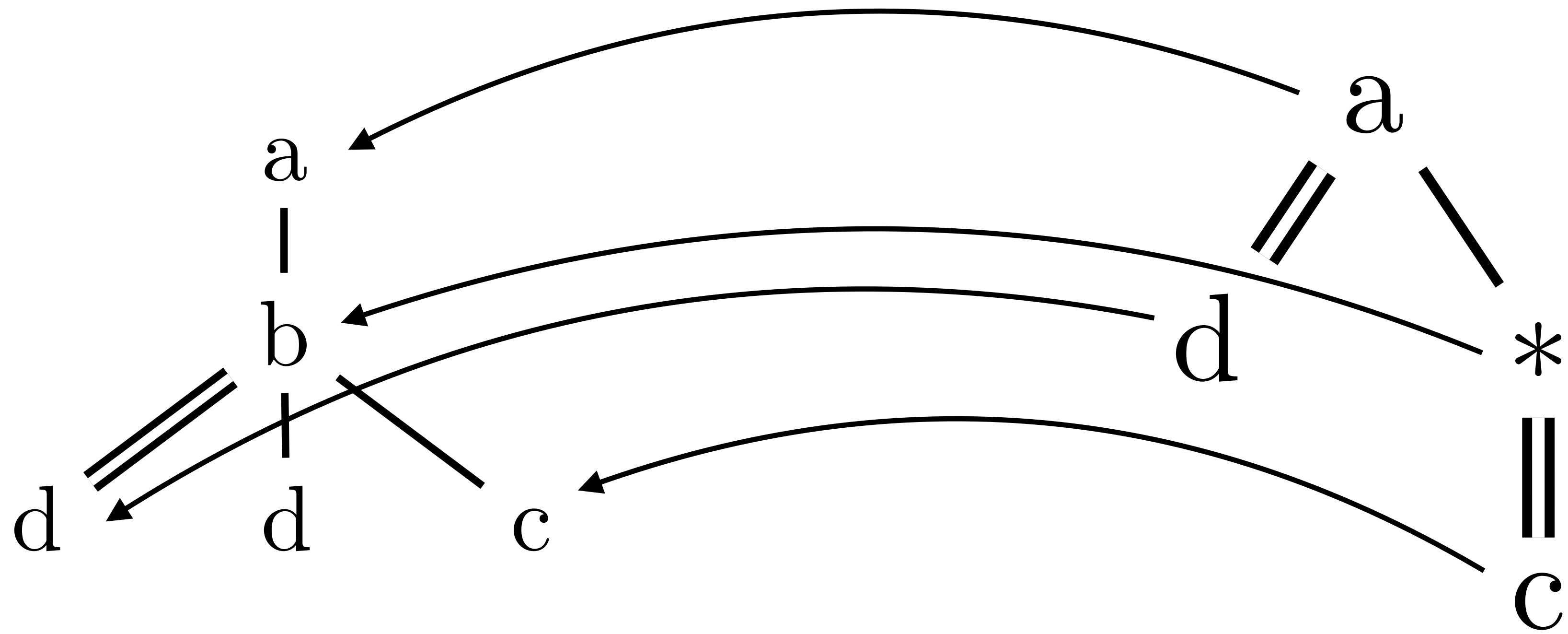
$$c_1 : b \to d$$

$$c_2 : b \to c$$

DTD

$$\mathrm{root} \to a^*$$
$$a \to b^* \mid c^*$$
$$b \to d^+ c^+$$
$$c \to b?c?$$

The Chase Technique



$c_1 : b \to d$

$c_2 : b \to c$

DTD

$$\mathrm{root} \to a^*$$
$$a \to b^* \mid c^*$$
$$b \to d^+ c^+$$
$$c \to b?c?$$

The General Landscape

PTIME	XP(/, //, ∗) [21] XP(/, [ ], ∗) (see [19]) XP(/, //, [ ]) [2], with fixed bounded SXICs [9] XP(/, //) + DTDs [22] XP[/, [ ]] + DTDs [22]
coNP	XP(/, //, [ ], ∗) [19] XP(/, //, [ ], ∗, |), XP(/, |), XP(//, |) [22] XP(/, [ ]) + DTDs [22] XP(//, [ ]) + DTDs [22]
Π_2^p	XP(/, //, [ ], |) + existential variables + path equality + `ancestor-or-self` axis + fixed bounded SXICs [9] XP(/, //, [ ], ∗, |) + existential variables + all backward axes + fixed bounded SXICs [9] XP(/, //, [ ], |) + existential variables with inequality [22]
PSPACE	XP(/, //, [ ], ∗, |) and XP(/, //, |) if the alphabet is finite [22] XP(/, //, [ ], ∗, |) + variables with XPath semantics [22]
EXPTIME	XP(/, //, [ ], |) + existential variables + bounded SXICs [9] XP(/, //, [ ], ∗, |) + DTDs [22] XP(/, //, |) + DTDs [22] XP(/, //, [ ], ∗) + DTDs [22]
Undecidable	XP(/, //, [ ], |) + existential variables + unbounded SXICs [9] XP(/, //, [ ], |) + existential variables + bounded SXICs + DTDs [9] XP(/, //, [ ], ∗, |) + nodeset equality + simple DTDs [22] XP(/, //, [ ], ∗, |) + existential variables with inequality[22]

Useful Reading

- Olteanu, Meuss, Fruche, Bry. "**XPath: Looking Forward**", XMLDM 2002.
- Schwentich. "**XPath Query Containment**", ACM SIGMOD Record 33(1), 2004.
- Miklau, Suciu. "**Containment and Equivalence for a Fragment of XPath**", J. ACM 51(1), 2004.