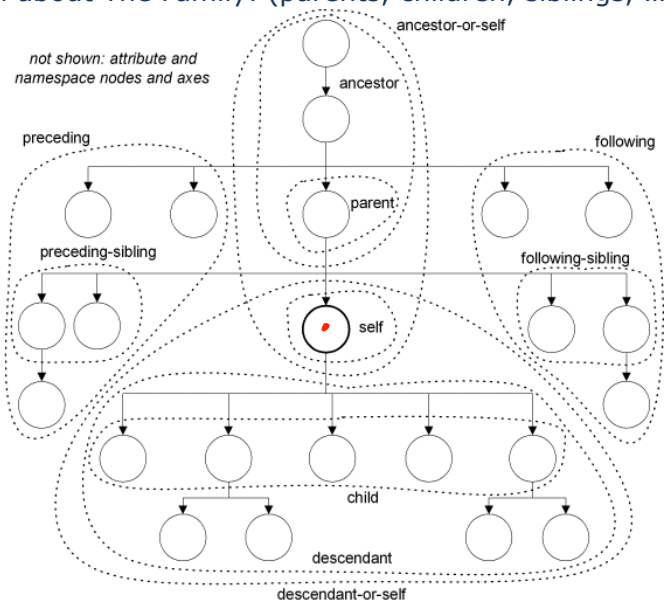
Announcements

- **HW#1**
  - Grades online (end of last week)
  - Pick-up outside of 3051 Kemper Hall
- **Group Project #1**
  - Due this Friday (via hand-in)
- **Individual HW#2**
  - Due Monday (Feb 3rd)
- **Wednesday Discussion**
  - XPath examples

XPath (and XQuery) using Zorba

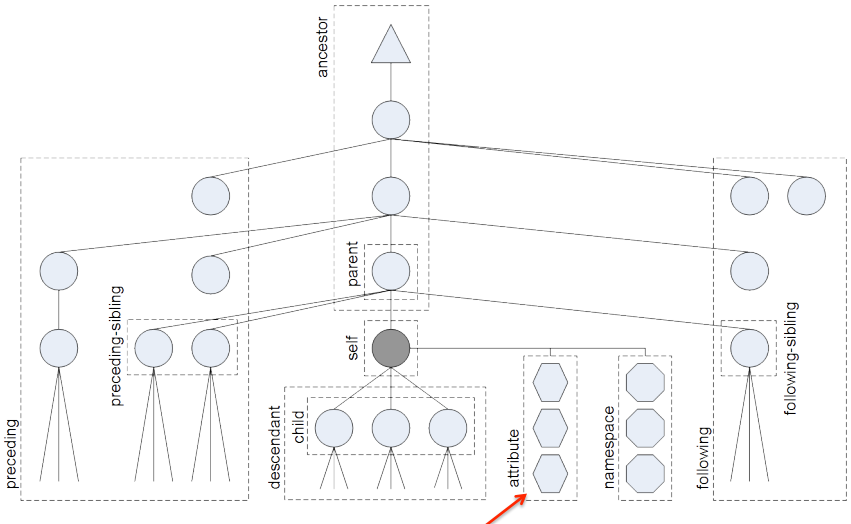
Navigating the XML Tree with XPath:

It's all about The Family! (parents, children, siblings, …)



Navigating the XML Tree with XPath:

It's all about The Family! (parents, children, siblings, …)



… as before, plus **attributes** (and namespaces)

Definition 5.1 (XPath axes)

XPath Axes, related

$$
\begin{aligned}
\text{self} &:= \epsilon \\
\text{child} &:= \text{firstchild.nextsibling}^{*} \\
\text{parent} &:= \text{child}^{-1} \\
\text{descendant} &:= \text{child}^{+} \\
\text{ancestor} &:= \text{descendant}^{-1} \\
\text{descendant-or-self} &:= \text{child}^{*} \\
\text{ancestor-or-self} &:= \text{descendant-or-self}^{-1} \\
\text{following-sibling} &:= \text{nextsibling}^{*} \\
\text{preceding-sibling} &:= \text{following-sibling}^{-1} \\
\text{following} &:= \text{ancestor-or-self.nextsibling}^{+}. \\
&\quad\ \text{descendant-or-self} \\
\text{preceding} &:= \text{following}^{-1}
\end{aligned}
$$

□

family tree/ day

child ↓ par.

(parent. child) = self usibling

[1] Monadic queries over tree-structured data, Gottlob, G. and Koch, C., Logic in Computer Science (LICS), 2002.

Definition 5.1 (XPath axes)

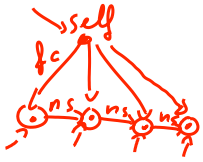
$$
\begin{aligned}
\text{self} &:= \epsilon \\
\text{child} &:= \text{firstchild.nextsibling}^{*} \\
\text{parent} &:= \text{child}^{-1} \\
\text{descendant} &:= \text{child}^{+} \\
\text{ancestor} &:= \text{descendant}^{-1} \\
\text{descendant-or-self} &:= \text{child}^{*} \\
\text{ancestor-or-self} &:= \text{descendant-or-self}^{-1} \\
\text{following-sibling} &:= \text{nextsibling}^{*} \\
\text{preceding-sibling} &:= \text{following-sibling}^{-1} \\
\text{following} &:= \text{ancestor-or-self.nextsibling}^{+}. \\
&\quad\ \text{descendant-or-self} \\
\text{preceding} &:= \text{following}^{-1}
\end{aligned}
$$

□

0 or more times

← flip direction

one or more times



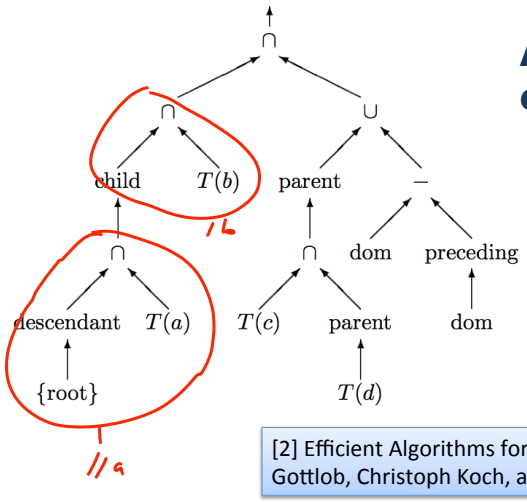
Given, {fc, ns} all other axes can be defined from those

Example 8.3 The Core XPath query

/descendant::a/child::b[child::c/child::d or not(following::*)]

//a /b //a/b[c/d or ...]

is evaluated as specified by the query tree



An XPath query evaluation plan

[2] Efficient Algorithms for Processing XPath Queries, Georg Gottlob, Christoph Koch, and Reinhard Pichler, ACM TODS, 2005

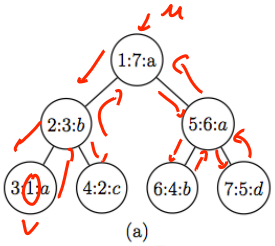
Querying XML in SQL?

- XPath queries can involve **recursion**: (or iteration)
  - e.g. ancestor, descendant, preceding, following axes
- Traditionally: **no** recursion in SQL…
- Ergo:
  - **Cannot** evaluate queries e.g. /a//b in SQL !
  - Right?
  - unless we use WITH RECURSIVE
  - or ... special encodings !

Querying XML in SQL?

- Cannot evaluate queries such as /a//b, right?
- Wrong! For example, use …
  - … XML **extensions** (e.g. in Postgres)
    - Of course we can always add custom functionality, e.g., for XPath
  - … clever encodings
    - e.g., *Interval Encoding* can reduce certain queries, to simple, non-recursive SQL queries (next slide)
    - Caveat: have to generate (and maintain) encoding with separate (outside of SQL) means

XASR: eXtented Access Support Relations



(a)

R	pre	post	parent_pre	lab
	1	7	⊥	a
	2	3	1	b
	3	1	2	a
	4	2	2	c
	5	6	1	a
	6	4	5	b
	7	5	5	d

(b)

Figure 2: Tree (a) and XASR (b).

Example 2.1 The tree shown in Figure 2 (a) can be represented by the XASR of Figure 2 (b). We can define the descendant axis as an SQL view

```
CREATE VIEW descendant AS
SELECT r1.pre, r2.pre FROM R r1, R r2
WHERE r1.pre < r2.pre AND r2.post < r1.post;
```

and the child axis as

```
CREATE VIEW child AS
SELECT parent_pre, pre FROM R
WHERE parent_pre is not NULL;
```

[3] Processing queries on tree-structured data efficiently, Koch, C., PODS 2006.

References

- Lots of basic XPath material out there …
  - Google, e.g., "XPath tutorial"

- Some (quite) advanced resources:

[1] Monadic queries over tree-structured data, Gottlob, G. and Koch, C., Logic in Computer Science (LICS), 2002.

[2] Efficient Algorithms for Processing XPath Queries, Georg Gottlob, Christoph Koch, and Reinhard Pichler, ACM TODS, 2005

[3] Processing queries on tree-structured data efficiently, Koch, C., PODS 2006.

[4] An XML Toolkit for Light-weight XML Stream Processing, Yours Truly (TJ Green) et al., 2003, http://homes.cs.washington.edu/~suciu/XMLTK/