Foundations of Data and Knowledge Bases

XPath

Joachim Niehren

Links: Linking Dynamic Data
Inria Lille

September 7, 2022

Outline

Repetition

Unranked Trees

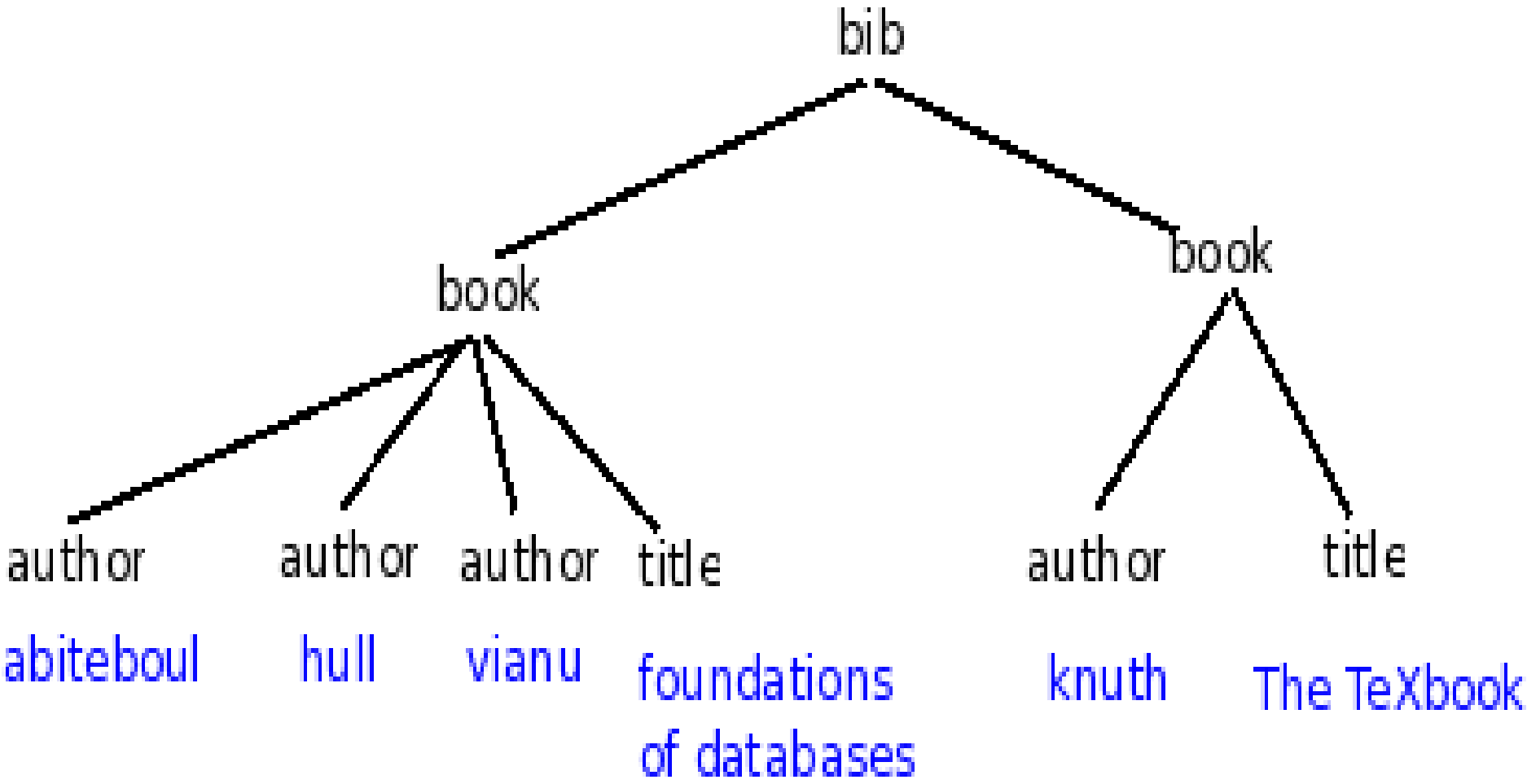
For a finite set Δ of labels (XML tags):

$$t \in T_\Delta ::= a(t_1, \ldots, t_n) \qquad n \geq 0, a \in \Delta$$

Simplifications

- textual content of nodes is not part of the tree (external annotation).
- no attributes

Example


bib
book
book
author
author
author
title
author
title
abiteboul
hull
vianu
foundations
of databases
knuth
The TeXbook

Node Selection Queries

n-ary Queries Q

functions mapping trees to sets of n-tuples of nodes

$$\forall t \in T_\Sigma : \; Q(t) \subseteq nod(t)^n$$

Examples

monadic select all book authors

binary select all pairs of authors and titles in some book

n-ary select n-tuples of nodes in trees

Boolean where $n = 0$. Select empty tuple $()$ in documents that satisfy the schema, and nothing otherwise.

Basic Queries

monadic leaf selects all leaf nodes.

binary child selects all pairs of father-child nodes.

Monadic Datalog for Unranked Trees

Signature

Constants	$\text{Consts} = \{\mathit{root}\}$
Relation symbols	$\text{Rels} = \{\text{next\_sib}(.,.), \text{first\_child}(.,.),$ $\text{last\_child}(.,.), \mathit{leaf}(.)\}$

Syntax (all variable $q \in \text{RelVars}$ have arity 1)

terms	$t ::= x \mid a$	($x \in \text{Vars}$ and $a \in \text{Consts}$)
literals	$l ::= q(t) \mid r(t_1, \ldots, t_{ar(q)})$	($q \in \text{RelVars}$ and $r \in \text{Rels}$)
Horn clauses	$q(t) :- l_1, \ldots, l_n.$	
Programs	P finite set of Horn clauses	

Example

Each second line of HTML tables: $table(tr, \ldots, tr)$

$$q_{even}(x) :- \text{first\_child}(root, x).$$
$$q_{odd}(x) :- \text{next\_sib}(y, x), q_{even}(y).$$
$$q_{even}(x) :- \text{next\_sib}(y, x), q_{odd}(y).$$

Normal Forms

Clause Formats

$q(root).$	test for root
$q(x) :- leaf(x)$	test for leaf
$q(x) :- \text{first\_child}(x, y), q'(y).$	go down left most
$q(x) :- \text{first\_child}(y, x), q'(y).$	go up left most
$q(x) :- \text{next\_sib}(x, y), q'(y).$	go right
$q(x) :- \text{next\_sib}(y, x), q'(y).$	go left
$q(x) :- \text{last\_child}(x, y), q'(y).$	go down right most
$q(x) :- \text{last\_child}(y, x), q'(y).$	go up right most
$q(x) :- q_1(x), q_2(x).$	run in parallel

Proposition (Gottlob& Koch PODS'02)

All monadic Datalog programs for unranked trees can be brought in normal form in linear time.

Lemma

Normal forms programs P can be grounded for a tree t in combined linear time $O(|t| \cdot |P|)$.

Efficient Query Answering

Define Monadic Queries

by Datalog program P and a goal predicate q

$$Q_{P,q}(t) = lfp^t_P(q)$$

Theorem (Gottlob& Koch PODS'02)

Answers of monadic queries $Q_{P,q}(t)$ defined monadic Datalog programs in unranked trees t can computed in combined linear time $O(|t| \cdot |P|)$.

Proof.

Compute normal form of P, make it ground with respect to t and compute the least solution. □

Outline

FO Logic for Unranked Trees

Syntax (where $a \in \Delta$ and $x, y \in \mathrm{Vars}$)

$$\phi ::= lab_a(x) \mid \mathrm{child}^*(x, y) \mid \mathrm{next\_sib}^*(x, y) \mid \neg\phi \mid \exists x\phi \mid \phi_1 \wedge \phi_2$$

Transitive closure is not definable in FO

We have to add child^* to the structures if we want to talk about descendants in FO, and similiarly $\mathrm{next\_sib}^*$ in order to talk about horizontal recursion.

FO Queries

a FO formula ϕ and a sequence of variables $\overline{x} = (x_1, \ldots, x_n)$ define an n-ary query $Q_{\phi(\overline{x})}$ such that for all trees $t \in T_\Delta$:

$$Q_{\phi(\overline{x})}(t) = \{(\alpha(x_1), \ldots, \alpha(x_n)) \mid t, \alpha \models \phi,\ \alpha : \mathrm{Vars} \to nod(t)\}$$

Outline

XPath 1.0

W3C Standard (2005)

- language for defining monadic queries in unranked trees.
- no variables!

Navigational language

- inspired by modal logics, i.e., variable free logics definable in FO.
- define navigation paths from start nodes to end nodes
- monadic queries select end-nodes when starting at the root

Core XPath 1.0 (Gottlob & Koch PODS'01, JACM'03)

- many simplifications
  - removes arithmethics
  - removes functions on data content
  - leaves only the navigational core
- formal semantics

Examples

Select all books in bibliographie

XPath short: $//book$

XPath long: $\text{child}^*::book$

FO logic: $\text{child}^*(root, x) \wedge lab_{book}(x)$

Select all co-authors of Abiteboul in books of bibliographie

XPath short: $//book[author/text() =' abitebool'] /$
$author[not[text() =' abiteboul']]$

XPath long: $\text{child}^*::book[\text{child}::author[text() =' abiteboul']] /$
$\text{child}::author[not[text() =' abiteboul']]$

FO logic: $\exists y_1\ (\text{child}^*(root, y_1) \wedge lab_{book}(y_1)$
$\wedge \exists y_2(\text{child}(y_1, y_2) \wedge lab_{author}(y_2) \wedge text_{' abiteboul'}(y_2))$
$\wedge \text{child}(y_1, x) \wedge lab_{author}(x) \wedge \neg text_{' abiteboul'}(x))$

Core XPath 1.0

Syntax where $a \in \Delta$:

$$
\begin{array}{llll}
\text{filter} & f & ::= & * \mid a \mid not[f] \mid f[p] \\
\text{axis} & \mathrm{r} & ::= & \text{child} \mid \text{next\_sib} \mid \mathrm{r}^{-1} \mid \mathrm{r}^{*} \mid \text{self} \mid \ldots \\
\text{paths} & p & ::= & \mathrm{r}::f \mid p/p' \mid p \cup p' \mid /p
\end{array}
$$

Semantics for unranked tree $t \in T_\Delta$:

$$
\begin{aligned}
eval^t(f) &\subseteq nod(t) \\
eval^t(\mathrm{r}) &\subseteq nod(t)^2 \\
eval^t(p) &\subseteq nod(t)^2
\end{aligned}
$$

Semantics

$$
\begin{aligned}
&eval^t(*) = nod(t)\\
&eval^t(a) = lab^t_a\\
&eval^t(not[f]) = nod(t) \setminus eval^t(f)\\
&eval^t(f[p]) = \{\pi \in eval^t(f) \mid \exists \pi'.\ (\pi, \pi') \in eval^t(p)\}
\end{aligned}
$$

$$
\begin{aligned}
&eval^t(\mathrm{child}) = \mathrm{child}^t\\
&eval^t(\mathrm{next\_sib}) = \mathrm{next\_sib}^t\\
&eval^t(\mathrm{r}^{-1}) = eval^t(\mathrm{r})^{-1}\\
&eval^t(\mathrm{r}^*) = eval^t(\mathrm{r})^*\\
&eval^t(\mathrm{self}) = \{(\pi, \pi) \mid \pi \in nod(t)\}
\end{aligned}
$$

$$
\begin{aligned}
&eval^t(\mathrm{r}::f) = \{(\pi, \pi') \in eval^t(\mathrm{r}) \mid \pi' \in eval^t(f)\}\\
&eval^t(p/p') = eval^t(p) \circ eval^t(p')\\
&eval^t(p \cup p') = eval^t(p) \cup eval^t(p')\\
&eval^t(/p) = \{(root^t, \pi) \mid (root^t, \pi) \in eval^t(p)\}
\end{aligned}
$$

Examples

first authors of books in bibliography
$/\mathit{child}^*::\mathit{book}/\mathrm{child}::\mathit{author}[\mathit{not}[\mathrm{next\_sib}^{-1}::*]]$
titles of book with at least two authors
$/\mathrm{child}^*::\mathit{book}[\mathrm{child}::\mathit{author}[\mathrm{next\_sib}::\mathit{author}]]/\mathit{child} :: \mathit{title}$
all author or editor of books
$/\mathit{child}^*::\mathit{book}/((\mathrm{child}^*::\mathit{author}) \cup (\mathrm{child}^*::\mathit{editor}))$
test wether no book has title and booktitle
$\mathit{not}[/\mathit{child}^*::\mathit{book}[[\mathrm{child}::\mathit{title}]\ [\mathrm{child}::\mathit{booktitle}]]$
test whether all books of bibliography have an author:
$\mathit{not}[\mathrm{child}^*::\mathit{book}[\mathit{not}[\mathrm{child}::\mathit{author}]]]$

Exercises: Quantifiers in Core XPath 1.0

1. Define the following FO monadic query in Core XPath 1.0

$$\mathit{exists\_desc}_a(x) =_{\mathrm{df}} \exists y(\mathrm{child}^+(x,y) \wedge \mathit{lab}_a(y))$$

2. Define the following FO monadic query in Core XPath 1.0

$$\mathit{forall\_desc}_a(x) =_{\mathrm{df}} \forall y(\mathrm{child}^+(x,y) \rightarrow \mathit{lab}_a(y))$$

3. Supppose that the set of tree labels is $\Delta = \{a, b\}$. Define a Core XPath 1.0 filter satisfied by the root of all trees such that:

$$\begin{aligned} \mathit{lab}_a(\mathit{root}) \wedge \forall x.\forall y. \quad & (\mathit{lab}_a(x) \wedge \mathrm{next\_sib}(x,y)) \rightarrow \mathit{lab}_a(y) \\ \wedge \quad & (\mathit{lab}_a(x) \wedge \mathrm{first\_child}(x,y)) \rightarrow \mathit{lab}_b(y) \\ \wedge \quad & (\mathit{lab}_b(x) \wedge \mathrm{next\_sib}(x,y)) \rightarrow \mathit{lab}_b(y) \\ \wedge \quad & (\mathit{lab}_b(x) \wedge \mathrm{first\_child}(x,y)) \rightarrow \mathit{lab}_a(y) \end{aligned}$$

Exercises: Expressiveness of Core XPath 1.0

4. Define the following query in Core XPath 1.0:
   All nodes reachable from the root over a path with labels in a^*b
   Hint: In Core XPath 1.0 you cannot do it forwards.
5. Can you define the following query in Core XPath?
   All nodes that are reachable from the root over a path with labels in $(aa)^*$?
   Please argue why! Can you define the same query in monadic Datalog for trees?

Translation to FO Logic

Proposition

Every expression of XPath 1.0 can be translated in linear time to an FO formula with 2 free variables, that define the same binary query.

$$[\![*]\!]_x = \mathit{true} \qquad [\![\mathrm{child}]\!]_{x,y} = \mathrm{child}(x,y)$$
$$[\![a]\!]_x = \mathit{lab}_a(x) \qquad [\![\mathrm{next\_sib}]\!]_{x,y} = \mathrm{next\_sib}(x,y)$$
$$[\![\mathit{not}[f]]\!]_x = \neg[\![f]\!]_x \qquad [\![\mathrm{r}^{-1}]\!]_{x,y} = \mathrm{r}(y,x)$$
$$[\![f[p]]\!]_x = [\![f]\!]_x \wedge \exists y [\![p]\!]_{x,y} \qquad [\![\mathrm{r}^*]\!]_{x,y} = \mathrm{r}^*(x,y)$$
$$[\![(\mathrm{r}^*)^{-1}]\!]_{x,y} = \mathrm{r}^*(y,x)$$
$$[\![\mathit{self}]\!]_{x,y} = \ x{=}y$$

$$[\![\mathrm{r}::f]\!]_{x,y} = [\![\mathrm{r}]\!]_{x,y} \wedge [\![f]\!]_y \qquad [\![p/p']\!]_{x,y} = \exists z([\![p]\!]_{x,z} \wedge [\![p']\!]_{z,y})$$
$$[\![p \cup p']\!]_{x,y} = [\![p]\!]_{x,y} \vee [\![p']\!]_{x,y} \qquad [\![/p]\!]_{x,y} = \ x{=}\mathit{root} \wedge [\![p]\!]_{x,y}$$

Query Answering for Core XPath 1.0

For start set $\Pi \subseteq nod(t)$ let $eval^t_\Pi(p) = \{\pi' \mid \pi \in \Pi,\ (\pi, \pi') \in eval^t(p)\}$

Theorem (Gottlob & Koch (TODS'05))

For all expressions p of Core XPath 1.0, trees t, and start sets Π one can compute the monadic query $eval^t_\Pi(p)$ in time $O(|t| \cdot |p|)$.

- Query answering for FO formulas is PSpace complete in contrast!
- Translated path expressions $[\![p]\!]_{x,y}$ are simple FO-formulas:
  - has 3 visible variables in all position only (Immerman and Kozen's result from 89 applies)!
  - Without negation, we obtain acyclic conjunctive queries (Yanakaki's result from 81 applies)!
- But why is it in combined linear time $O(|t| \cdot |p|)$?
  (Combined linear time is quadratic in the strict sence.)

Compute Filter Semantics

Proposition

For every filter expression f of Core XPath 1.0 and every tree t, one can compute $eval^t(f)$ in time $O(|t| \cdot |f|)$.

Simple Filters

$$f ::= * \mid a \mid f[f'] \mid [r::f] \mid not[f]$$

Simple Filters are Enough

replace path composition by filter composition, for instance:

$[\mathrm{r}_1::f_1/\mathrm{r}_2::f_2/\mathrm{r}_3::f_3]$ becomes $[\mathrm{r}_1::f_1[\mathrm{r}_2::f_2[\mathrm{r}_3::f_3]]]$

Proof of Proposition

by induction on the structure of simple filters. Computing $eval^t(*)$, $eval^t(a)$, $eval^t(f[f'])$ and $eval^t(not[f])$ is easy. It remains to compute $eval^t([\mathrm{r}::f])$ for all possible axis r.

Compute $eval^t([\mathrm{r}::f])$

Generic program can be in $O(|t|^2 \cdot |f|)$

```
for π in eval^t(f) do
    for π' with (π', π) ∈ r^t do
        collect(π')
```

Interesting case: all descendants of set of nodes $\mathrm{r} = (\mathrm{child}^*)^{-1}$

compute $\Pi = eval^t(f)$, run down all paths of t in parallel, memoize whether node in Π has been seen, and collect all later nodes in this case. This is in $O(|t|)$ plus the time for computing $eval^t(f)$ which is in $O(|t| \cdot |f|)$ by induction hypothesis. Alltogether this is $O(|t| \cdot |[\mathrm{r}::f]|)$.

Be careful

Linear time combined complexity depends on the nature of the axis!

Compute Path Semantics

Set based computation

$$eval^t_\Pi(p/p') = eval^t_{eval^t_\Pi(p)}(p')$$
$$eval^t_\Pi(p \cup p') = eval^t_\Pi(p') \cup eval^t_\Pi(p)$$
$$eval^t_\Pi(\mathrm{r}::f) = \{\pi' \mid \exists \pi \in \Pi : \ (\pi, \pi') \in \mathrm{r}^t\} \cap eval^t(f)$$

Linear time complexity

Consider $eval^t_\Pi(\mathrm{r}::f)$ above. The first set above on the right can be computed in time $O(|t|)$ with the same trick as for filters (and thus depends on the choice of axis), the second set has been computed at before hand in time $O(|t| \cdot |f|)$, see above Proposition.

Exercises [Query Answering Algorithm for Core XPath 1.0]

A node π follows another π' in a trees t, if during a preorder traversal, the first visit of π' is before the first visit of π.

6. Present an algorithm that inputs a tree t and a set $\Pi \subseteq nod(t)$ and computes in time $O(|t|)$ the node set $eval^t_\Pi(\text{following})$.
7. The query answering algorithm from the lecture applies to Core XPath 1.0 with the restricted set of axis above. Is is possible to extend this algorithm, such that it supports that axis following in addition, while preserving combined linear time complexity? Please explain your answer.

Core XPath 1.0

13. Define in CoreXPath 1.0 the relations:
    - *firstchild* relating a node to its first child if there is some
    - *lastchild* relating a node to its last child if there is some
    - *firstleaf* relating a node to its left most leaf descendant if there is some
    - *lastleaf* relating a node to its right most leaf descendant if there is some
14. Define in the first order-logic of unranked trees with relations child^* and $\mathrm{next\_sib}^*$ the relations:
    - *firstchild** the reflexive transitive closure of firstchild
    - *lastchild** the reflexive transitive closure of lastchild

    Can you define the same relations in Core XPath 1.0 ? Or is some useful operator missing there?

Outline

XPath 2.0 (W3C, 2007)

"Few" Variables are Back

- variables to capture nodes as in hybrid logic (Blackburn 95).
- define n-ary queries by navigations paths with n-variables
- not available in XPath 1.0

Recover Missing FO-Operators

- complementation on path level
- universal quantifiers, but not added below since not needed for expressiveness.

Core XPath 2.0 (Ten Cate & Marx, ICDT 2007)

Syntax where $a \in \Delta$ and $x \in \mathit{Vars}$:

$$
\begin{array}{llll}
\text{filter} & f & ::= & * \mid a \mid \mathit{not}[f] \mid f[p] \mid [x] \\
\text{axis} & \mathrm{r} & ::= & \text{child} \mid \text{next\_sib} \mid \mathrm{r}^{-1} \mid \mathrm{r}^{*} \mid \text{self} \mid \ldots \\
\text{paths} & p & ::= & \mathrm{r}{::}f \mid p/p' \mid p \cup p' \mid /p \mid p^{\mathit{compl}}
\end{array}
$$

Semantics for $t \in T_\Delta$ and $\alpha : \mathrm{Vars} \to \mathit{nod}(t)$:

$$
\begin{array}{ll}
\mathit{eval}^{t,\alpha}(f) \subseteq \mathit{nod}(t) & \mathit{eval}^{t,\alpha}([x]) = \{\alpha(x)\} \\
\mathit{eval}^{t,\alpha}(\mathrm{r}) \subseteq \mathit{nod}(t)^2 & \\
\mathit{eval}^{t,\alpha}(p) \subseteq \mathit{nod}(t)^2 & \mathit{eval}^{t,\alpha}(p^{\mathit{compl}}) = \mathit{nod}(t)^2 \setminus \mathit{eval}^{t,\alpha}(p)
\end{array}
$$

Example

Select all author-title-pairs in books of a bibliography:

Core XPath 2.0 $/\text{child}^*::book[\text{child}::author[x]][\text{child}::title[y]]$

FO $\exists z_1, z_2($
$z_1{=}root \wedge \text{child}^*(z_1, z_2) \wedge lab_{book}(z_2) \wedge$
$\text{child}(z_2, x) \wedge lab_{author}(x) \wedge$
$\text{child}(z_2, y) \wedge lab_{title}(y)\ )$

FO-Expressiveness

Theorem [Marx PODS'05]

All FO-queries can be expressed by Core XPath 2.0 queries and vice versa.

Proof.

- quantifier elimination (Schwentick 2000):
  - Ehrenfeucht-Frässe games
  - Shelah's composition method
- these techniques are difficult.

□

Theorem [Filiot&Niehren&Talbot&Tison PODS'07]

There is an FO-complete fragment of Core XPath 2.0 with polynomial time query answering:

- Syntactic restrictions on variables:
  - For all subexpressions p/p': no variable sharing between p and p'
  - No variable below complementation

Example: FO Query Not Expressible in Core XPath 1.0

All nodes having a descendant over a path labeled by a^*b

In FO logic: select all nodes x with b-labeled descendant z such that all nodes y between x and z are labeled by a.

$$\mathrm{child}^*(\mathit{root}, x) \wedge \exists z.(\mathit{lab}_b(z) \wedge \forall y.\ ((\mathrm{child}^*(x, y) \wedge \mathrm{child}^*(y, z)) \rightarrow \mathit{lab}_a(y)))$$

In extensions of Core XPath 1.0 with reflexive transitive closure:

$$/\mathrm{child}^*::*[(\mathrm{child}::a)^*/\mathrm{child}::b]$$

In Core XPath 2.0:

$$/\mathrm{child}^*::*[((\mathrm{child}^*::*) \cap (\mathrm{child}^*::*[\mathit{not}[a]]/\mathrm{child}^*)^{\mathit{compl}})/\mathrm{child}::b]$$

Conditional XPath (Marx 05) with Variables

Syntax where $a \in \Delta$ and $x \in \mathit{Vars}$:

$$
\begin{array}{llll}
\text{filter} & f & ::= & * \mid a \mid \mathit{not}[f] \mid f[p] \mid [x] \\
\text{axis} & \mathrm{r} & ::= & \text{child} \mid \text{next\_sib} \mid \mathrm{r}^{-1} \mid \text{self} \mid \ldots \\
\text{paths} & p & ::= & (\mathrm{r}::f)^* \mid \mathrm{r}::f \mid p/p' \mid p \cup p' \mid /p
\end{array}
$$

Expressivenss

Conditional XPath = F0 = Core XPath 2.0

Efficiency

Without variables, the algorithm of Gottlob and Koch extends to Conditional XPath in combined linear time, but fails for Core XPath 2.0 (where cubic time depending on the data size is needed)

Exercises

Expressiveness of Core XPath 2.0

9. How would you define the FO-logic of words in $\{a, b\}^*$. Can you define the set of words a^*b^* by a closed formula in FO-logic? Can you define the set of all trees with the following DTD in Core XPath 2.0?

$$c \to a^*b^*, \;\; a \to \epsilon, \;\; b \to \epsilon$$

10. Define a query in Core XPath 2.0 that selects all pairs of teacher names and lecture titles in the database from the first lecture.
11. Define the conditional path $(\mathrm{child}[f])^*$ in Core XPath 2.0 for arbitrary Core XPath 2.0 filters f.

Outline

Extensions of Data Model

Data Values

$$
\begin{array}{rcl}
\mathit{value} & ::= & \mathit{string} \mid \mathit{number} \mid \ldots \\
\mathit{string} & ::= & \text{"}\mathit{abiteboul}\text{"} \mid \ldots. \\
\mathit{number} & ::= & 0 \mid 1 \mid 2 \mid 3.4 \mid \ldots \\
\mathit{boolean} & ::= & 0 \mid 1
\end{array}
$$

Data Trees

trees annotated with data values

every node has a content of type string

every node has a string value for all of its attributes

XPath Extension for Data Trees

Functions

access to node content	$text : unit \to string$
values of node attributes	$@a : string \cup number$
counting set	$count : 2^{value} \to number$
arithmetics	$+ : number \times number \to number$
equality	$=: value \times value \to bool$
disequality	$\neq: value \times value \to bool$
number comparison	$\leq, \geq: number \times number \to bool$

Joins, Aggregations, Functions

$$
\begin{array}{rcl}
f & ::= & [p = p'] \mid [p \neq p'] \mid [p \leq p'] \mid [p \geq p'] \mid \ldots \\
p & ::= & text() \mid @a \mid count(p) \mid p + p' \mid \ldots
\end{array}
$$

Examples in Short Notation

count number of authors of a book: $count(author)$
select last names of all teachers of lectures: $//teacher/lname/text()$
count number of teachers of a lecture

$$count(//teacher/lname/text()))$$

test whether there exist two books with the same author

$$[/bib[book[(\mathrm{child}::/author/@id) = (/\mathrm{next\_sib}^*::*/author)/@id]]]$$

books by Springer after 1991:

$$//bib/book[publisher = "Springer" \text{ and } @year \geq 1991]$$