XPath: (P)DL on trees.

Maarten Marx

ReasoningWeb2009

ReasoningWeb2009

Overview

1. Knowledge Representation on the Web
2. Logical research questions for XML
3. Getting familiar with XPath(s)
4. Zoom in
   i. Expressivity
   ii. Complexity
5. Conclusions

KR on the Web

ABS2000 Edge labelled graphs queried by regular path expressions

XML Node labelled sibling ordered trees queried by XPath

RDF triples and non wellfounded sets

- . . . but most web information is of course in the form of . . .

KR on the Web

ABS2000 Edge labelled graphs queried by regular path expressions

XML Node labelled sibling ordered trees queried by XPath

RDF triples and non wellfounded sets

- . . . but most web information is of course in the form of . . . text

KR on the Web

ABS2000 Edge labelled graphs queried by regular path expressions

XML Node labelled sibling ordered trees queried by XPath

RDF triples and non wellfounded sets

- . . . but most web information is of course in the form of . . . text sometimes generated from a relational database.
- This talk: XML.

Graphs and trees

- Edge labelled graphs can very directly encode ER diagrams.
- These can always be represented as trees
  - Sometimes as just trees
  - Cyclic information needs ID's and IDREF's.

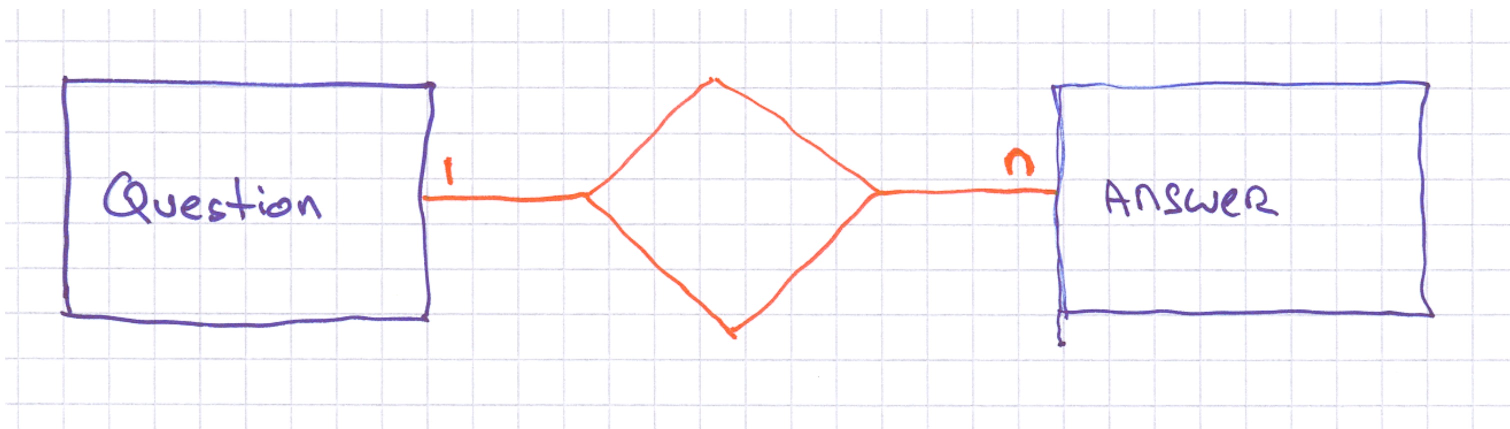
Consequences of the choice of your representation

- query processing costs
- needed expressive power for your
  - ⋆ query language
  - ⋆ constraint language
- robustness for changes in the data-structures

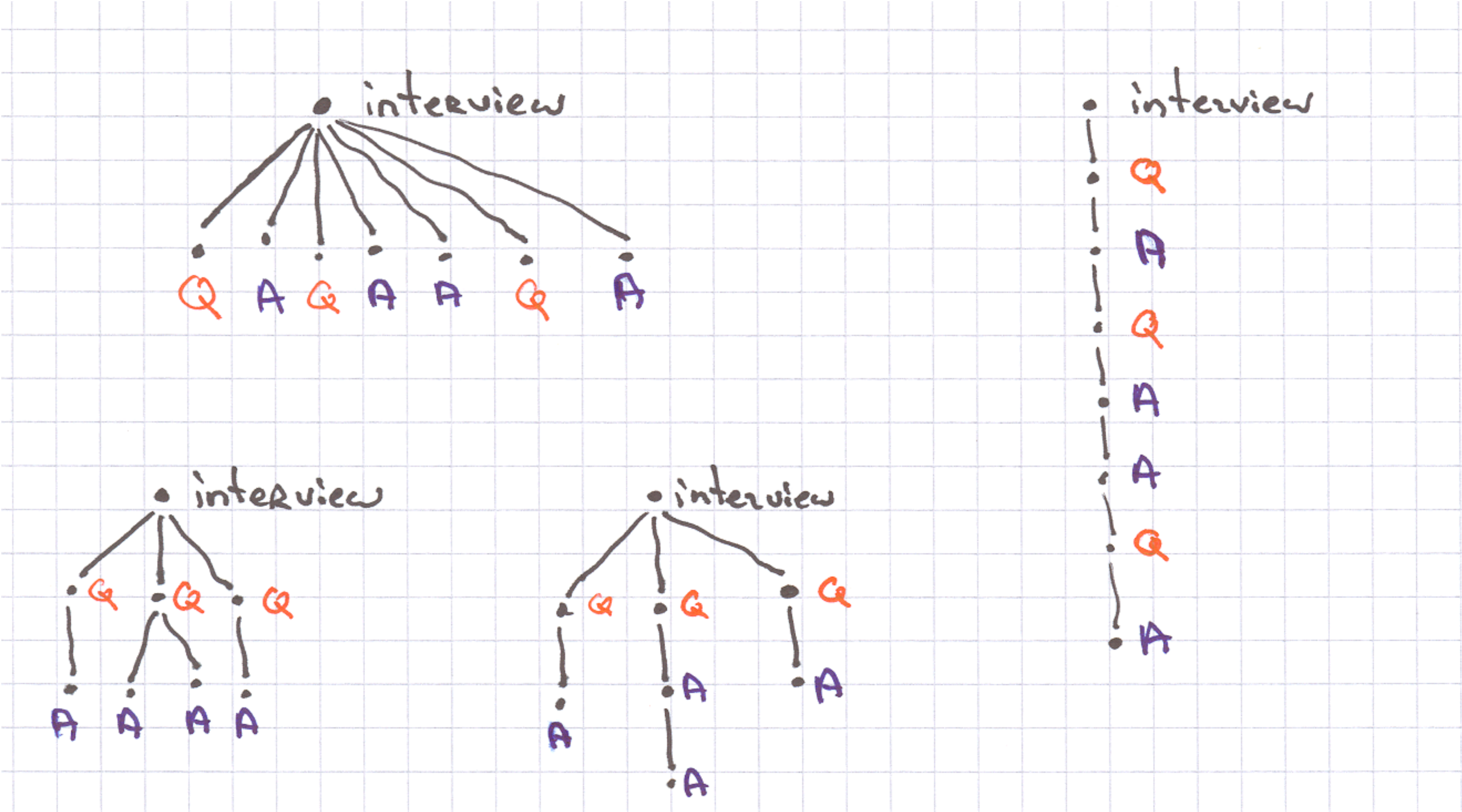
Example: interviews

- Sigmod Record Distinguished DB Profiles
- Simple model:

An interview consists of a list of questions each followed by a list of answers.



exemelify this



In practice

```
wget http://www.sigmod.org/sigmod/record/issues/0409/7.phil-bernstein-final.pdf
|
pdftohtml -xml
|
saxon MakeInterviewTree.xsl
>>
interview.xml
```

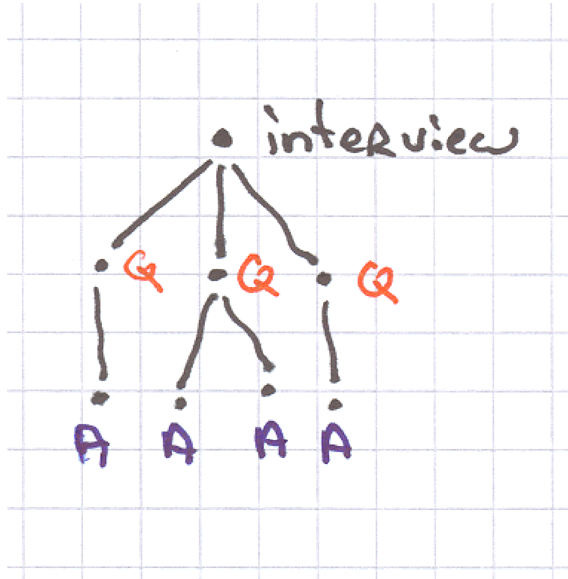
Quiztime

1. How will the output of `pdftohtml` look as a tree?

2. What will be the easiest (and fastest) tree transformation?

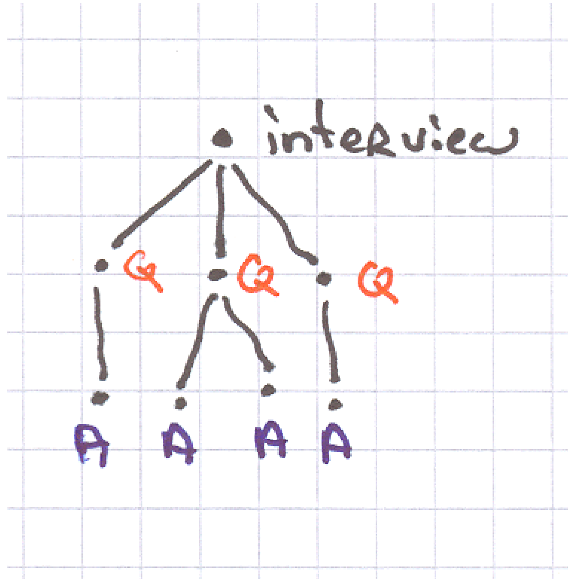
3. Which of the 4 tree models?

In theory: TREE model



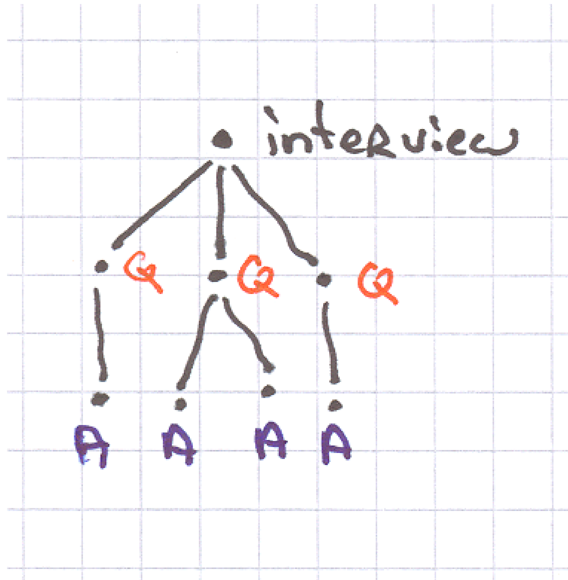
- Query: give me all QA pairs.

In theory: TREE model



- Query: give me all QA pairs.
- In "hybrid DL":
- for \$q such that \$q $\models$ Q, return
  (\$q, { a | a $\models$ A $\sqcap$ $\exists$.parent \$q })

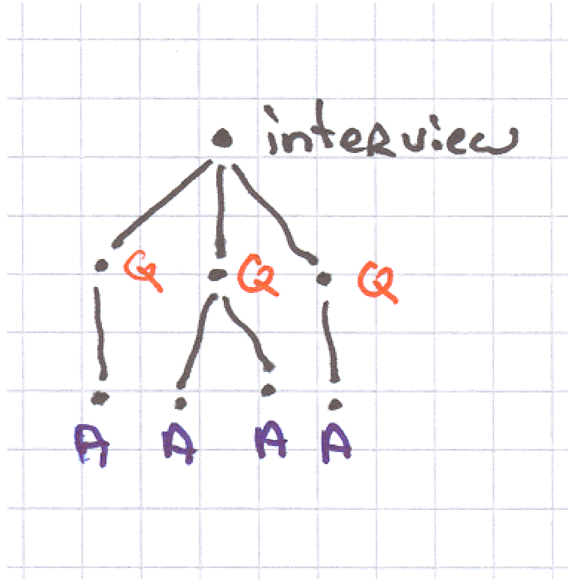
In theory: TREE model



- In XPath 2.0:

- `for $q in //Q return ($q,$q/A)`

In theory: TREE model

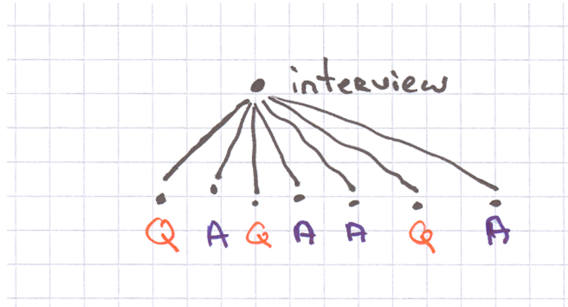


- In XPath 2.0:
- `for $q in //Q return ($q,$q/A)`

XPath and Description Logic

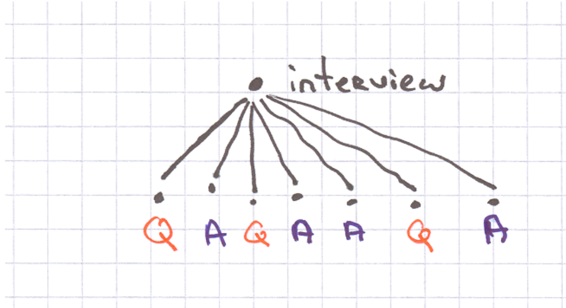
- Specifying nodes from a different perspective
- In Description Logic you describe the node that you want as if you are standing on the wanted node.
- In XPath you describe how to get there, as if you are standing at the root.

Same query on the practical FLAT model



- Query: return all A-nodes answering a give Q node
- Tree model: simple ALC-formula using the tree-order
- Flat tree model:

Same query on the practical FLAT model



- Query: return all A-nodes answering a give Q node
- Tree model: simple ALC-formula using the tree-order
- Flat tree model:
  - ⋆ use the document-order or the sibling-order
  - ⋆ all A nodes after the given Q, but before the next Q
  - ⋆ 3 variables . . .
  - ⋆ not modally expressible . . .
  - ⋆ the wanted A-nodes must satisfy $A \wedge \mathrm{since}(\$q, \neg Q)$

Lesson Learned

- Choice of representation influences what query-language may be needed later-on.

Constraining the models: theory vs practice

- XML constraint languages are based on tree-automata
- languages use regular expressions over node-labels.
- these describe the children of a node read from left to right

Flat model

Constraining the models: theory vs practice

- XML constraint languages are based on tree-automata
- languages use regular expressions over node-labels.
- these describe the children of a node read from left to right

Flat model

```
interview -> (Q,A+)+
```

Tree model

Constraining the models: theory vs practice

- XML constraint languages are based on tree-automata
- languages use regular expressions over node-labels.
- these describe the children of a node read from left to right

Flat model

```
interview -> (Q,A+)+
```

Tree model

```
interview -> Q+.   Q -> A+
```

Data Actual question and answer text is stored in attribute nodes.

Constraining the models: theory vs practice: robustness

- Example: Extend our constraints: every interview ends with a bye-bye question which receives no answer.
- In all models this is expressible as a FO sentence: thus a regular tree language.

New Flat model

Constraining the models: theory vs practice: robustness

- Example: Extend our constraints: every interview ends with a bye-bye question which receives no answer.
- In all models this is expressible as a FO sentence: thus a regular tree language.

New Flat model

Easy: `interview -> (Q,A+)+,Q`

New Tree model

Constraining the models: theory vs practice: robustness

- Example: Extend our constraints: every interview ends with a bye-bye question which receives no answer.
- In all models this is expressible as a FO sentence: thus a regular tree language.

New Flat model

Easy: `interview -> (Q,A+)+,Q`

New Tree model

Hard! Not expressible by a DTD. (Proof later)

Bad!

- Difficult to accept and understand non-expressibility by practitioners
- leads to underspecified documents
- leads to frustration and unsafe coupling

New Tree model

- We need types to express the last answerless question.
- Specialized DTD's = MSO = regular tree languages [Papakonstantinou, Vianu 00]
- `NormalQ` and `EndQ` are types of `Q`
- `interview -> NormalQ+,EndQ`
- `NormalQ -> A+`
- `EndQ -> EMPTY`

New Tree model

- We need types to express the last answerless question.
- Specialized DTD's = MSO = regular tree languages [Papakonstantinou, Vianu 00]
- `NormalQ` and `EndQ` are types of `Q`
- `interview -> NormalQ+,EndQ`
- `NormalQ -> A+`
- `EndQ -> EMPTY`
- This is not expressible in XML Schema!

Relax

- But it is expressible in Relax NG.
- In exactly the way given.
- Relax NG is a Schema Language by Clark and Murata.

KR on the web: wrap up

- Most information on the web is in implicitly structured text.
- Asking complex queries to the web thus means to extract and make this structure explicit.
- This often leads to rather flat ("reading text-ordered") XML.
- KR languages are important to describe, constrain and validate the XML,
- because these XML files are themselves often input to other knowledge-extraction programs (tree-transformations, queries)

Where are we?

1. ~~KR on the Web~~

2. Logical research questions for XML

3. Getting familiar with XPath(s)

4. Zoom in

   i. Expressivity
   ii. Complexity

5. Conclusions

XML-tasks

[Schwentick 04] distinguishes the following four:

- Validation
- Transformation
- Navigation
- Querying

Every task must be described in some (logical) language.

Usual research questions

Given some language L

- What tasks can I express in L? How well can I express them in L?
- Given an L expression and data, what are the computation costs to perform the task?

Usual research questions

Given some language L

- What tasks can I express in L? How well can I express them in L?
- Given an L expression and data, what are the computation costs to perform the task?
- Each task may involve more specialized questions: e.g.
- Typechecking: given input conform $I_1 \in L_1$, given a transformation $T \in L_T$, will the output always be conform $I_2 \in L_2$?
- [Milo, Suciu, Vianu, 00] Decidable for DTD and Core XSLT.

This talk: focus on validation and navigation

Expressive power on trees

- relative to yardsticks as CQ, FO, MSO, tree automata
- semantic characterizations
- succinctness questions
- rewrite systems

This talk: focus on validation and navigation

Expressive power on trees

- relative to yardsticks as CQ, FO, MSO, tree automata
- semantic characterizations
- succinctness questions
- rewrite systems

Complexity
- Model checking: given a tree T and a formula F , does T satsify F?
- Static analysis: containment, equivalence, satisfiability of expressions.

Major techniques and strategies

- Similar research strategy as in DL: understand a language landscape by asking the same question for many different fragments.
- Where are the **borders** of decidability and tractability?
- Develop handy tools to show that something is **not** expressible in some fragment.
- Techniques include
  - Finite models
  - Tree automata, regular tree languages
  - tree decompositions

Where are we?

1. ~~KR on the Web~~
2. ~~Logical research questions for XML~~
3. Getting familiar with XPath(s)
4. Zoom in
   i. Expressivity
   ii. Complexity
5. Conclusions

XPath

- two sorted language, just as (P)DL
  - ⋆ path sort binary relation between nodes
  - ⋆ node sort set of nodes
- interpreted on a special class of models:
  - ⋆ finite, sibling ordered, node-labelled unranked trees
- XPath, like DL, is not a language, more a “style”, a “family”

Operators on node sort are very familiar

- atomic tests
- test for being in the domain of a relation. (just like $\exists R.F$)
- closed under the booleans.
- (sometimes) $n \models R{=}S$ iff $\exists m.\ (n, m) \in R \cap S$.

Operators on node sort are very familiar

- atomic tests
- test for being in the domain of a relation. (just like $\exists R.F$)
- closed under the booleans.
- (sometimes) $n \models R{=}S$ iff $\exists m.\ (n, m) \in R \cap S$.
  - term-definable from $w \models R^{\mathrm{loop}}$ iff $(w, w) \in R$.
  - $R{=}S \equiv (R; S^{-1})^{\mathrm{loop}}$

Primitive relations are tree relations

- down,up,left,right
- their transitive closures: descendant, ancestor, . . .
- often syntactic sugar: following =
  `ancestor*/right+/descendant*`
- stay relation with a test:

Operators on path sort are also very familiar

- Regular operators: union, concatenation, Kleene closure
- Boolean operators: `intersect` and `except`
- Variables and binders: as in hybrid logic.
  - `for $x in PATH1 return PATH2`
  - Meaning: $\downarrow$ `y.PATH1/`$\downarrow$ `x.  @y/PATH2`

Immediate relations to known formalisms

- node and path-formulas of PDL
- almost all operators can be found in some DL-language
- Trees: CTL, tree logics of [Blackburn, de Rijke, Meijer-Viol '96]
- without Kleene *, all languages are inside FO.

Real life complications (1)

- Two syntaxes
- Unix path style:

```
/book//section[./paragraph[contains(.,''XML'')]]
```

- Official style:

```
/child::book/descendant::section[child::paragraph[contains(.,''XML'')]]
```

- Unix style only "up and down". Official style: everything.

Real life complications (2)

XPath has many uses and interpretations.

1. Path formula denotes binary relation

   when used for navigation within other languages

2. path formula denotes set of nodes
   - when used as a stand-alone query language
   - Meaning of `PATH` is range of `PATH`.
   - Natural with `/PATH` (all nodes reachable by `PATH` from the root)

3. Path formula denotes a set of trees
   - XPath used as a constraint language
   - "all trees having a `PATH` from the root"

Example

- Task Express the tree-like interview model in XPath.
- For N a node-formula ("modal formula"), N holds everywhere iff the root starts path

```
.[not //*[not N]].
```

```
Q -> A+
```

Example

- Task Express the tree-like interview model in XPath.
- For `N` a node-formula ("modal formula"), `N` holds everywhere iff the root starts path

`.[not //*[not N]]`.

```
Q -> A+          Q and (not child::A or child::*[not A])
```

Example

- Task Express the tree-like interview model in XPath.
- For `N` a node-formula ("modal formula"), `N` holds everywhere iff the root starts path

`.[not //*[not N]]`.

```
Q -> A+              Q and (not child::A or child::*[not A])

last Q without A
```

Example

- Task Express the tree-like interview model in XPath.
- For N a node-formula ("modal formula"), N holds everywhere iff the root starts path

```
.[not //*[not N]].
```

```
Q -> A+           Q and (not child::A or child::*[not A])
last Q without A  Q and not right::Q and child::A
```

Real life benefits

- Firefox and IE support XPath.
- Fast free XPath evaluators (Saxon, Libxslt)
- Good editors for XPath available
  - ⋆ syntax highlighting
  - ⋆ help with debugging
  - ⋆ evaluation on XML docs

XPath practice

We define two information needs in terms of XPath.

1. a descendant with lots of specific ancestors along the way

2. question-answer pairs

Practice 1

Q return all q descendants of current node

Practice 1

Q return all q descendants of current node

A `descendant::q` or `.//q`

Practice 1

Q return all q descendants of current node

A `descendant::q` or `.//q`

Q return all q descendants reachable trough $\mathtt{p}_1, \ldots, \mathtt{p}_\mathtt{n}$ nodes

Practice 1

Q return all q descendants of current node

A `descendant::q` or `.//q`

Q return all q descendants reachable trough $p_1, \ldots, p_n$ nodes

A1 `.//p₁//q intersect ... intersect .//pₙ//q`

Practice 1

Q return all q descendants of current node

A `descendant::q` or `.//q`

Q return all q descendants reachable trough $p_1, \ldots, p_n$ nodes

A1 `.//p1//q intersect ... intersect .//pn//q`

A2 big union for all permutions ρ of `1, ... ,n` of

$$.//p_{\rho(1)}//p_{\rho(2)}// \cdots //p_{\rho(n)}//q$$

Practice 2: question-answers pairs

- Flat (QA+)+ models
- Find an XPath expression `$x/...` which returns
  - ⋆ when `$x` is bound to a `Q` node
  - ⋆ all following `A` nodes until the next `Q`.

`$x`

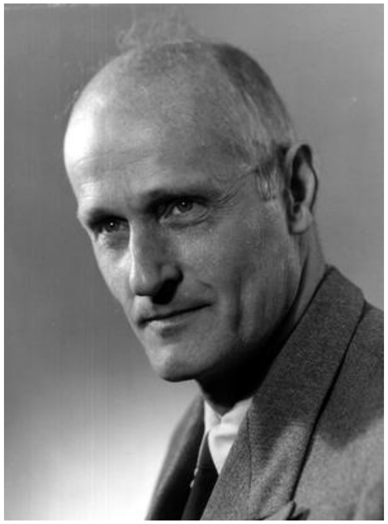
... QAA Q AAAQAQAAAA ...

Kleene style

Kleene style

`$x/(right::A)+`.

- `(.)+` is the transitive closure operator.
- But `(.)+` is not available (and not expressible) in W3C XPath dialects (because that is just FO).



Tarski style

Tarski style

$$\$x/(\,\texttt{following}-\texttt{sibling} :: \texttt{A}\ \ \texttt{except}$$
$$\texttt{following}-\texttt{sibling} :: \texttt{Q}/\texttt{following}-\texttt{sibling} :: \texttt{A})$$

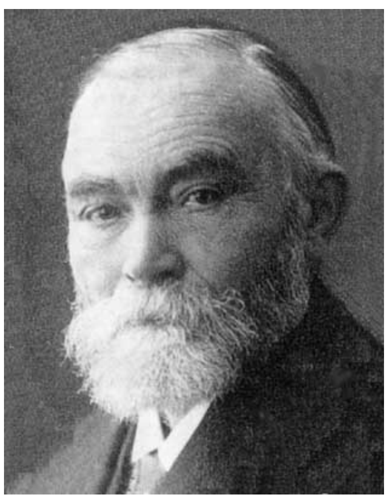
- Expressible in XPath with Booleans on path expressions [Hidders, 2003]

Frege (or first-order) style

Frege (or first-order) style

```
$x/following-sibling::A[  not
  preceding-sibling::Q/preceding-sibling::Q[. is $x]]
```

- Uses variables bound to nodes
- Test `. is $x` is the hybrid logic variable test.



Where are we?

1. ~~KR on the Web~~
2. ~~Logical research questions for XML~~
3. ~~Getting familiar with XPath(s)~~
4. Zoom in
   i. Expressivity
   ii. Complexity
5. Conclusions

Expressivity questions on trees

Rabin's theorem sets a clear upper bound:

MSO = tree automata = regular tree languages = decidable.

Questions we will survey:

- expressivity relative to yardsticks
- succinctness
- semantic characterizations

Signature of the languages:

equality, unary predicates for nodes, child, descendant, right, right+

Four XPath dialects

Four flavours of XPath strictly below MSO [ten Cate, M. 2007 survey]

Core XPath ≈ PDL without *

XPath 2.0 no vars ≈ Boolean modal logic ≈ Core XPath plus booleans on paths

XPath 2.0 ≈ hybrid Boolean modal logic

Regular XPath PDL with the four one-step tree relations.

Characterization of Core XPath

- On unary trees ($=$ the line), this is Prior's temporal logic with **F** and **P**.

- Kamp's theorem '68 not enough to capture FO(x) on the line.

- [Etessami, Vardi and Wilke '97]: expressive power is exactly FO$_2(x)$, with an exponential succinctness gap.
  - "any two nodes that agree on $p_1, \ldots, p_n$ also agree on p_0"
  - linear constraint in FO_2, exponential in TL.

- Generalizes to sibling-ordered trees and Core XPath.

Core XPath plus booleans on paths

- Kamp's thm on unary trees: FO(x)= FO_3(x).
- [M. 2005]: Generalizes to XML-trees and paths: FO(x,y) = FO_3(x,y)
- Tarski's thm: $\text{FO}_3(x, y)$ = Tarski relation algebras.
- on trees: Tarski relation algebras = Core XPath plus booleans on paths

---

- Core XPath plus booleans on paths = FO(x,y) on XML trees.

Regular XPath

- Captures $FO(x, y)$ (because it captures "since and until").
- [ten Cate 06] With additional loop it captures $FO^*(x, y)$.

$$T, x \models R^{\text{loop}} \text{ iff } T, (x, x) \models R.$$

- [ten Cate, Segoufin 08] With additional subtree relativization it captures FO extended with monadic TC.

$$T, x \models \mathbf{W}\phi \text{ iff } T_x, x \models \phi.$$

- [ten Cate, Segoufin 08] Both are strictly less expressive than MSO.

Summary

XPath dialect	Core XPath 1.0	$\subsetneq$	Variable-free Core XPath 2.0	$\equiv$	C
Equivalent FO-dialect	$\exists FO^{\text{mon}\neg}_{\text{tree}}$		FO_{tree}		
	(exponential succinctness gap)		(at least exponential succinctness gap)		(no lin

$\equiv$	Core XPath 2.0	$\subsetneq$	Regular XPath$^{\approx}$
	FO_{tree}		FO^{*}_{tree}
	(no succinctness gap: linear translations)		(non-elementary succinctness gap)

Semantic characterizations

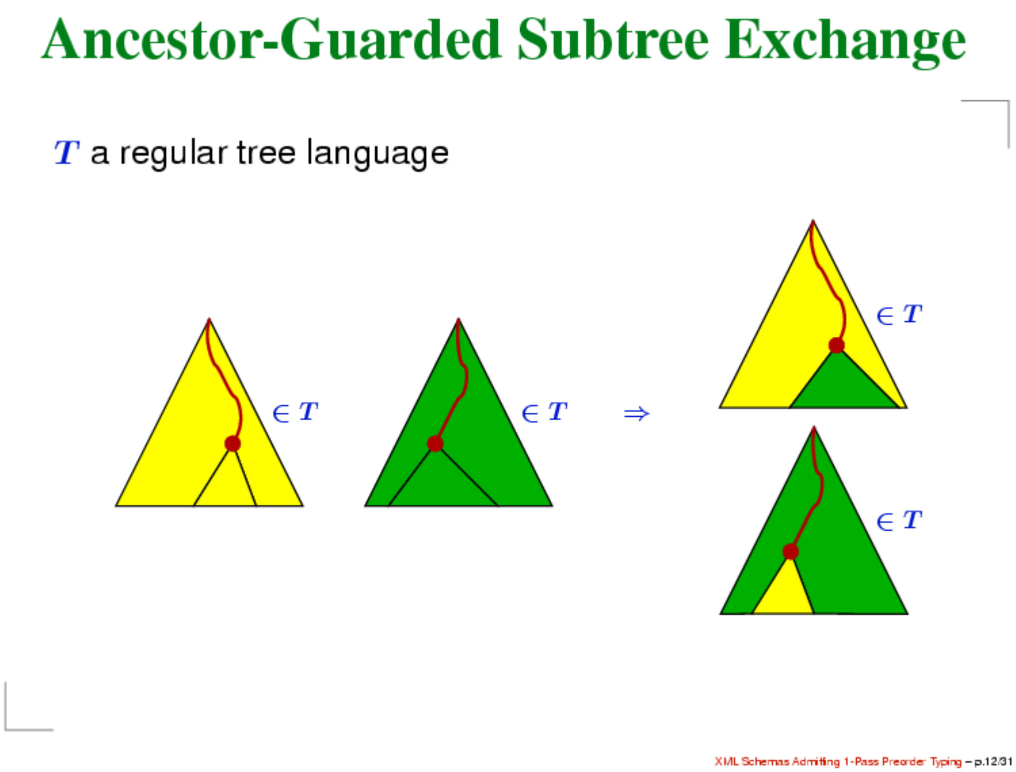
- class of trees C is definable in L iff C is closed under . . .
- Useful for inexpressivity results.
- Real-life languages (W3C standards) often have practical constraints with unexpected theoretical effects
- DTD's: must be deterministic

  (a+b)*a(a+b) is not expressible by a DTD [Brüggemann-Klein Wood 98]

- XML schema's must be single-typed specialized DTD's [Murata, Lee, Mani '01]

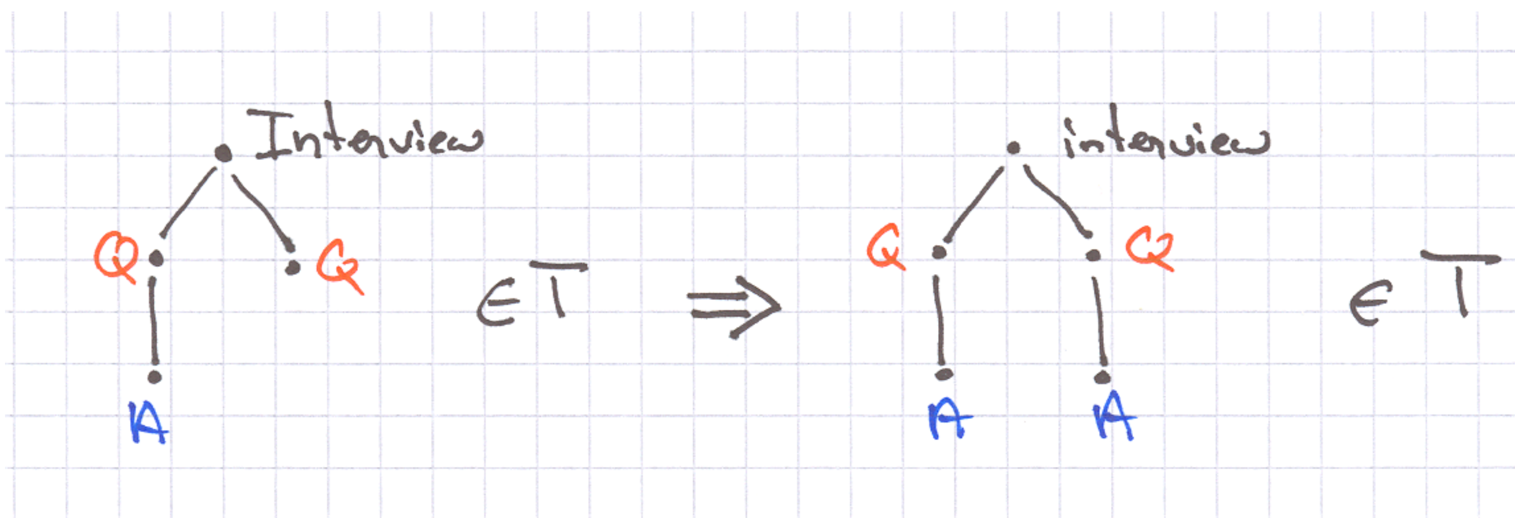
Characterization of single type SDTD

- [Martens, Neven, Schwentick 05] For T a regular tree language, T is definable by a single type SDTD iff T is closed under ancestor-guarded subtree exchange.



(QA+)+Q is not definable on hierarchical models

- Interviews ending in a Q without an A.
- We could not find a DTD specifying this in the hierarchical model.
- Now we can prove it:



Where are we?

1. ~~KR on the Web~~
2. ~~Logical research questions for XML~~
3. ~~Getting familiar with XPath(s)~~
4. Zoom in
   i. ~~Expressivity~~
   ii. Complexity
5. Conclusions

Complexity questions: evaluation

- Model checking. Validation, querying

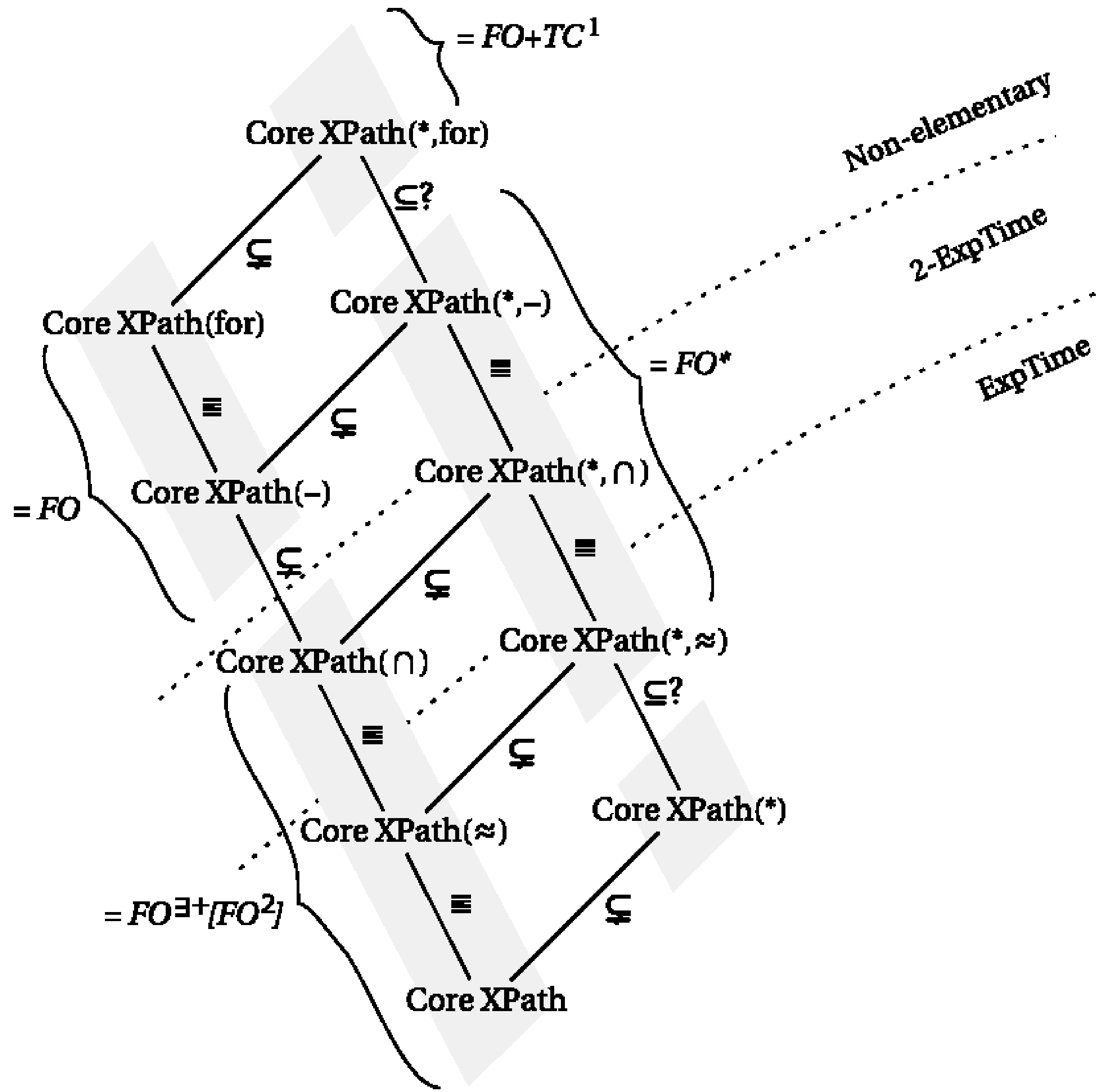
Input Tree, node(s), formula. Output Boolean

- PSPACE complete for FO. PTIME for fixed variable FO.

Fragment	Evaluation complexity	
Core XPath	PTime (linear)	[Gottlob, Koch, Pichl
Core XPath 2 no vars	PTime (quadratic)	(from FO)
Core XPath 2	Pspace	(from FO)
Regular XPath	PTime (linear)	(from PDL)
Regular XPath+	PTime (linear)	[Gottlob Koch 04]
TMNF tests (=MSO)		

Complexity: Static analysis

- Satisfiability, equivalence, . . .
- Decidable for MSO. Non-elementary hard already for FO on unary trees [Rabin; Meyer]
- Complexity overview [ten Cate, Lutz, 2007]
  - ⋆ Satisfiability.
  - ⋆ Lower bound is EXPTIME, already for Core XPath
  - ⋆ Small language extensions may yield large leaps in complexity


= FO+TC1
Core XPath(*,for)
Non-elementary
⊆?
⊊
Core XPath(for)
Core XPath(*,–)
2-ExpTime
≡
= FO*
⊊
ExpTime
= FO
Core XPath(–)
Core XPath(*,∩)
Core XPath(∩)
Core XPath(*,≈)
Core XPath(≈)
Core XPath(*)
= FO∃+[FO2]
Core XPath

Where are we?

1. ~~KR on the Web~~
2. ~~Logical research questions for XML~~
3. ~~Getting familiar with XPath(s)~~
4. Zoom in
   i. ~~Expressivity~~
   ii. ~~Complexity~~
5. Conclusions

XML language research and (P)DL: close relations

- both rooted in KR
- trees as fundamental models
- strong emphasis on working systems
- huge tables with acronyms and complexity classes ;-)

strong Description Logic–XML interplay

- KR aspects
- Data integration and mediation [Halevy, Rome school] (certain answers are hard to compute)
- Design, maintenance, reuse, integration of ontologies is daily headache for XML/web-engineers
- DL's research on modularity of TBoxes [Manchester school] seems useful.

Thank you

© Dubhacan

Thank you

Thank you