Queries, Modalities, Relations, Trees, XPath
Lecture VII
Core XPath and beyond

Tadeusz Litak

Department of Computer Science
University of Leicester

July 2010: draft

Basic Axioms I: Idempotent Semirings

ISAx1	$(A \cup B) \cup C$	$\equiv$	$A \cup (B \cup C)$
ISAx2	$A \cup B$	$\equiv$	$B \cup A$
ISAx3	$A \cup A$	$\equiv$	A
ISAx4	$A/(B/C)$	$\equiv$	$(A/B)/C$
ISAx5	$\cdot/A$	$\equiv$	A
	$A/\cdot$	$\equiv$	A
ISAx6	$A/(B \cup C)$	$\equiv$	$A/B \cup A/C$
	$(A \cup B)/C$	$\equiv$	$A/C \cup B/C$
ISAx7	$\bot$	$\subseteq$	A

Distributive lattices, Kleene algebras, Tarski's relation algebras: they all have idempotent semiring reducts.
Idempotency is the axiom ISAx3.
$\bot$ abbreviates $\cdot[\neg\langle\cdot\rangle]$

Basic Axioms II: Predicate Axioms

PrAx1	$A\,[\neg\langle B\rangle]\,/B$	$\equiv$	$\bot$
PrAx2	$A\,[\phi \vee \psi]$	$\equiv$	$A\,[\phi] \cup A\,[\psi]$
PrAx3	$(A/B)\,[\phi]$	$\equiv$	$A/B\,[\phi]$
PrAx4	$\cdot\,[\langle\cdot\rangle]$	$\equiv$	$\cdot$

In Tarski's relation algebras and XPath 2.0,
predicates can be defined away
Note that PrAx3 would not be valid
if we allowed unrestricted positional predicates

Basic Axioms III: Node Axioms

NdAx1	ϕ	$\equiv$	$\neg(\neg\phi \vee \psi) \vee \neg(\neg\phi \vee \neg\psi)$
NdAx2	$\langle A \cup B \rangle$	$\equiv$	$\langle A \rangle \vee \langle B \rangle$
NdAx3	$\langle A/B \rangle$	$\equiv$	$\langle A\,[\langle B \rangle]\,\rangle$
NdAx4	$\langle \cdot\,[\phi]\,\rangle$	$\equiv$	ϕ

Note how little was needed to ensure booleanity!
(by Huntington's result from the 1930's)
And NdAx2–NdAx4 just mimick PrAx2—PrAx4
(redundancy: price to pay for two-sorted signature)

Axioms in one-sorted signature

Recall all the two-sorted axioms for predicates and expressions:

PrAx1	$A\,[\neg\langle B\rangle]\,/B$	$\equiv$	$\bot$
PrAx2	$A\,[\phi \vee \psi]$	$\equiv$	$A\,[\phi] \cup A\,[\psi]$
PrAx3	$(A/B)\,[\phi]$	$\equiv$	$A/B\,[\phi]$
PrAx4	$\cdot\,[\langle\cdot\rangle]$	$\equiv$	$\cdot$
NdAx1	ϕ	$\equiv$	$\neg(\neg\phi \vee \psi) \vee \neg(\neg\phi \vee \neg\psi)$
NdAx2	$\langle A \cup B\rangle$	$\equiv$	$\langle A\rangle \vee \langle B\rangle$
NdAx3	$\langle A/B\rangle$	$\equiv$	$\langle A\,[\langle B\rangle]\,\rangle$
NdAx4	$\langle\cdot\,[\phi]\,\rangle$	$\equiv$	ϕ

Axioms in one-sorted signature

Here is a one-sorted axiomatization for $\sim$ over idempotent semi-ring axioms found by Hollenberg:

$$
\begin{array}{lcl}
\sim A/A & \equiv & \bot \\
\sim\sim A/A & \equiv & A \\
\sim(A/B)/A & \equiv & (\sim(A/B)/A)/\sim B \\
\sim(A \cup B) & \equiv & \sim A/\sim B \\
\sim A \cup \sim B & \equiv & \sim\sim(\sim A \cup \sim B)
\end{array}
$$

We need to add one more axiom for tests:

$$?p \equiv \sim\sim ?p$$

Now, you may have the feeling that
there was nothing XPath-specific yet

Now, you may have the feeling that
there was nothing XPath-specific yet
But in fact there is a fragment for which
it is all there is:

Now, you may have the feeling that
there was nothing XPath-specific yet
But in fact there is a fragment for which
it is all there is:
Core XPath($\downarrow$), the child-axis-only fragment!

Theorem

The axioms presented so far are complete for all valid equivalences of Core XPath($\downarrow$).

Now, you may have the feeling that
there was nothing XPath-specific yet
But in fact there is a fragment for which
it is all there is:
Core XPath($\downarrow$), the child-axis-only fragment!

Theorem

The axioms presented so far are complete for all valid equivalences of Core XPath($\downarrow$).

In order to find more interesting equivalences,
we have to move to other fragments

Axioms for Linear Axes

The non-transitive case:

LinAx1 $\mathsf{s}\,[\neg\phi] \quad \equiv \quad \cdot\,[\neg\langle \mathsf{s}\,[\phi]\,\rangle]\,/\mathsf{s}$ for $\mathsf{s} \in \{\rightarrow, \leftarrow, \uparrow\}$

This forces functionality of the corresponding axis

Axioms for Transitive Axes

One for node expressions, one for path expressions:

TransAx1	$\langle \mathsf{s}^+[\phi]\rangle$	$\equiv$	$\langle \mathsf{s}^+[\phi \wedge \neg\langle \mathsf{s}^+[\phi]\rangle]\rangle$
TransAx2	s^+	$\equiv$	$\mathsf{s}^+ \cup \mathsf{s}^+/\mathsf{s}^+$

The first one is called the Löb axiom and forces well-foundedness
Don't get modal logicians started on it—
people wrote books about this formula

In particular, all the consequences of TransAx2 for *node expressions*
can be already derived from TransAx1
I can neither prove nor disprove that for *path expressions*
TransAx2 is (ir-)redundant

Finally, Axes which Are Both Transitive and Linear

LinAx2 $\cdot [\langle s^{+} [\phi] \rangle] / s^{+} \quad \equiv \quad s^{+} [\phi] \cup s^{+} [\phi] / s^{+} \cup s^{+} [\langle s^{+} [\phi] \rangle]$
for $s \in \{\rightarrow, \leftarrow, \uparrow\}$

together with transitivity axioms

This forces the corresponding axis is a linear order

Single Axis Completeness Result

Theorem

- *Base axioms are complete for Core XPath($\downarrow$)*
- *Base axioms with LinAx1 are complete for other intransitive single axis fragments*
- *Base axioms with TransAx1 and TransAx2 are complete for Core XPath($\downarrow^{+}$)*
- *Base axioms with TransAx1, TransAx2 and LinAx2 are complete for other transitive single axis fragments*

A Few Words About Proofs

- First, rewrite node expressions to simple node expressions:

$$\text{siNode} ::= \langle\cdot\rangle \mid p \mid \langle \text{a}\,[\text{siNode}]\,\rangle \mid \neg\text{siNode} \mid \text{siNode}\vee\text{siNode}$$

A Few Words About Proofs

- First, rewrite node expressions to simple node expressions:

  $\text{siNode} ::= \langle\cdot\rangle \mid p \mid \langle a\,[\text{siNode}]\,\rangle \mid \neg\text{siNode} \mid \text{siNode} \vee \text{siNode}$

  They are isomorphic variants of modal formulas

A Few Words About Proofs

- First, rewrite node expressions to simple node expressions:

  $$\text{siNode} ::= \langle\cdot\rangle \mid p \mid \langle \text{a}\,[\text{siNode}]\,\rangle \mid \neg\text{siNode} \mid \text{siNode}\vee\text{siNode}$$

  They are isomorphic variants of modal formulas
- Using normal form theorems for modal logic, we provide a completeness proof for node expressions

A Few Words About Proofs cntd.

- Then we rewrite all path expressions as sums of sum-free expressions of the form

$$S = \cdot\,[\beta_1]\,/\mathsf{a}\,[\beta_2]\,/\ldots/\mathsf{a}\,[\beta_\ell]\,,$$

(all β_i are normal forms of

- the same nesting degree in case of transitive axes
- strictly decreasing degree for intransitive axes)

In case of linear axes, we can even guarantee that every formula is witnessed further down the chain

A Few Words About Proofs cntd.

- Then we rewrite all path expressions as sums of sum-free expressions of the form

$$S = \cdot\,[\beta_1]\,/a\,[\beta_2]\,/\ldots/a\,[\beta_\ell]\,,$$

(all β_i are normal forms of

  - the same nesting degree in case of transitive axes
  - strictly decreasing degree for intransitive axes)

In case of linear axes, we can even guarantee that every formula is witnessed further down the chain

- We prove that for every two such expressions either

A Few Words About Proofs cntd.

- Then we rewrite all path expressions as sums of sum-free expressions of the form

$$S = \cdot\, [\beta_1]\, /\mathsf{a}\, [\beta_2]\, / \ldots /\mathsf{a}\, [\beta_\ell]\, ,$$

  (all β_i are normal forms of

  - the same nesting degree in case of transitive axes
  - strictly decreasing degree for intransitive axes)

  In case of linear axes, we can even guarantee that every formula is witnessed further down the chain

- We prove that for every two such expressions either
  - one is a subsequence of the other—provably contained or

A Few Words About Proofs cntd.

- Then we rewrite all path expressions as sums of sum-free expressions of the form

$$S = \cdot\,[\beta_1]\,/a\,[\beta_2]\,/\ldots/a\,[\beta_\ell]\,,$$

(all β_i are normal forms of

  - the same nesting degree in case of transitive axes
  - strictly decreasing degree for intransitive axes)

In case of linear axes, we can even guarantee that every formula is witnessed further down the chain

- We prove that for every two such expressions either
  - one is a subsequence of the other—provably contained or
  - there is a countermodel for containment

Aside: the issue of labels

There is a fact about XML trees we did not take into account (unless we opt to render attribute-value pairs as additional labels)

Aside: the issue of labels

There is a fact about XML trees we did not take into account (unless we opt to render attribute-value pairs as additional labels)

The labels are disjoint!

Aside: the issue of labels

There is a fact about XML trees we did not take into account (unless we opt to render attribute-value pairs as additional labels)

The labels are disjoint!

However, this is easy to fix: add node axiom

$$p \wedge q \equiv \bot$$

for distinct p and q
This axiom itself is not substitution-invariant,
this is why we do not like it

Aside: the issue of labels

There is a fact about XML trees we did not take into account (unless we opt to render attribute-value pairs as additional labels)

The labels are disjoint!

However, this is easy to fix: add node axiom

$$p \wedge q \equiv \bot$$

for distinct p and q
This axiom itself is not substitution-invariant,
this is why we do not like it

But as our proofs used only Birkhoff's rules
they are quite flexible and adding this axiom does not hurt

Starting from the Other End

Instead of beginning with single axes
and then trying to combine two or more

Starting from the Other End

Instead of beginning with single axes
and then trying to combine two or more

LET'S GO FOR THE WHOLE CORE XPATH!

Axiom For Axes Dependencies

$$\text{TreeAx1}\begin{cases} s^{+}/s \cup s \equiv s^{+} \\ s/s^{+} \cup s \equiv s^{+} \end{cases}$$

$$\text{TreeAx2} \quad s[\phi]/s^{\smile} \equiv \cdot[\langle s[\phi]\rangle] \text{ (for s distinct than } \uparrow)$$

$$\text{TreeAx3} \quad \uparrow[\phi]/\downarrow \equiv (\leftarrow^{+} \cup \rightarrow^{+} \cup \cdot)[\langle\uparrow[\phi]\rangle]$$

$$\text{TreeAx4}\begin{cases} \leftarrow^{+} \equiv \leftarrow^{+}[\langle\uparrow\rangle] \\ \rightarrow^{+} \equiv \rightarrow^{+}[\langle\uparrow\rangle] \end{cases}$$

TreeAx1 says: s^{+} is a transitive closure of s
TreeAx2 says non-child axes are functional
and describes their converse
TreeAx3 forces $\uparrow$ is the converse of (non-functional) $\downarrow$
with TreeAx4, it also describes how horizontal and vertical axes interplay

Theorem

The axioms presented so far are complete for Core XPath node expressions

Theorem

The axioms presented so far are complete for
Core XPath node expressions

Proof.

By reduction to simple node expressions
and derivation of all axioms of modal logic of finite trees
by Blackburn, Meyer-Viol, de Rijke □

(boolean axioms)

$$
\begin{array}{lcl}
\langle \mathsf{s}[\neg\langle\cdot\rangle]\rangle & \equiv & \neg\langle\cdot\rangle \\
\langle \mathsf{s}[\phi \vee \psi]\rangle & \equiv & \langle \mathsf{s}[\phi]\rangle \vee \langle \mathsf{s}[\psi]\rangle \\
\phi & \leq & \neg\langle \mathsf{s}[\neg\langle \mathsf{s}^{\smile}[\phi]\rangle]\rangle \\
\langle \mathsf{s}[\neg\phi]\rangle \wedge \langle \mathsf{s}[\phi]\rangle & \equiv & \neg\langle\cdot\rangle(\text{ for s distinct than } \uparrow) \\
\langle \mathsf{s}[\phi]\rangle \vee \langle \mathsf{s}[\langle \mathsf{s}^{+}[\phi]\rangle]\rangle & \equiv & \langle \mathsf{s}^{+}[\phi]\rangle \\
\neg\langle \mathsf{s}[\phi]\rangle \wedge \langle \mathsf{s}^{+}[\phi]\rangle & \leq & \langle \mathsf{s}^{+}[\neg\phi \wedge \langle \mathsf{s}[\phi]\rangle]\rangle \\
\langle \mathsf{s}[\langle\cdot\rangle]\rangle & \leq & \langle \mathsf{s}^{+}[\neg\langle \mathsf{s}[\langle\cdot\rangle]\rangle]\rangle \\
\text{TransAx1 for } \downarrow^{+} \text{ and } \rightarrow^{+} & & \\
\langle \downarrow[\neg\langle\leftarrow\rangle \wedge \neg\langle\rightarrow^{*}[\phi]\rangle]\rangle & \leq & \neg\langle \downarrow[\phi]\rangle \\
\langle \downarrow[\phi]\rangle & \leq & \langle \downarrow[\neg\langle\leftarrow\rangle]\rangle \wedge \langle \downarrow[\neg\langle\rightarrow\rangle]\rangle \\
\neg\langle\uparrow\rangle & \leq & \neg\langle\leftarrow\rangle \wedge \neg\langle\rightarrow\rangle
\end{array}
$$

A Nasty Trick

A Nasty Trick

We can use this to provide
an axiomatization for path expressions ...

A Nasty Trick

We can use this to provide
an axiomatization for path expressions . . .

. . . of a sort—a non-orthodox one! :(

A Nasty Trick

We can use this to provide
an axiomatization for path expressions ...

...of a sort—a non-orthodox one!

Add the separability rule:

(Sep) IF $\langle A\,[p]\,\rangle \equiv \langle B\,[p]\,\rangle$ for p not occurring in A, B

THEN $A \equiv B$.

A Nasty Trick

We can use this to provide
an axiomatization for path expressions . . .

. . . of a sort—a non-orthodox one!
Add the separability rule:

(Sep) IF $\langle A\,[p]\,\rangle \equiv \langle B\,[p]\,\rangle$ for p not occurring in A, B

THEN $A \equiv B$.

Except for spoiling the whole equational story,
it does not sit too well with the labelling axiom . . .

The Nasty Trick Does Its Job

. . . but it's perfect for obtaining complexity results
for query equivalence problem
by using reductions to corresponding modal logics

Complexity Theorem

Theorem

- *Query equivalence of Core XPath($\rightarrow^+, \leftarrow^+$), Core XPath($\uparrow^+$), Core XPath($\mathsf{s}$) (for $\mathsf{s} \in \{\uparrow, \leftarrow, \rightarrow\}$) is coNP-complete.*
- *Query equivalence of Core XPath($\leftarrow^+, \leftarrow, \rightarrow^+, \rightarrow, \uparrow^+, \uparrow$) is PSPACE-complete.*

  Thus, the PSPACE upper bound applies to all its sublanguages.
- *Query equivalence of Core XPath($\downarrow$) and Core XPath($\downarrow^+$) is PSPACE-complete.*

  Thus, all extensions of this fragment are PSPACE-hard.
- *Query equivalence of Core XPath($\downarrow, \downarrow^+$) is EXPTIME-complete.*

  Thus, all extensions of this fragment are EXPTIME-hard.

Proofs

…by reductions to complexity results for modal logics like K, K4, Alt.1 and fragments of tense/temporal logic on linear and branching orders.

The most interesting one is for the second clause—somewhat tricky embedding into a logic of Sistla and Clarke.

What we have seen so far . . .

We have seen:

- equational axiomatizations for path equivalences of all eight single axis fragments of Core XPath

What we have seen so far . . .

We have seen:

- equational axiomatizations for path equivalences of all eight single axis fragments of Core XPath
- equational axiomatizations for node equivalences of full Core XPath 1.0

What we have seen so far . . .

We have seen:

- equational axiomatizations for path equivalences of all eight single axis fragments of Core XPath
- equational axiomatizations for node equivalences of full Core XPath 1.0
- non-orthodox axiomatization for path equivalences of full Core XPath 1.0

What we have seen so far . . .

We have seen:

- equational axiomatizations for path equivalences of all eight single axis fragments of Core XPath
- equational axiomatizations for node equivalences of full Core XPath 1.0
- non-orthodox axiomatization for path equivalences of full Core XPath 1.0
- computational complexity results for path equivalences in most meaningful sublanguages of Core XPath 1.0

What we have not seen so far . . .

- Definability and expressivity results (for finite sibling-ordered trees . . .)
- Results for fragments of XPath stronger than CoreXPath 1.0

From now on, I am going to use Balder Ten Cate's M4M 2007 slides

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- . . . — e.g., in between FO and MSO lies FO(TC)

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- . . . — e.g., in between FO and MSO lies FO(TC)

What kind of queries do we want to characterize

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- ... — e.g., in between FO and MSO lies FO(TC)

What kind of queries do we want to characterize

- *Binary relations* definable by *path expressions*?
- *Node sets* definable by *node expressions*?
- *Properties of trees* definable by *node expressions evaluated at the root*?

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- . . . — e.g., in between FO and MSO lies FO(TC)

What kind of queries do we want to characterize

- *Binary relations* definable by *path expressions*?
- *Node sets* definable by *node expressions*?
- *Properties of trees* definable by *node expressions evaluated at the root*?

Possible types of characterizations:

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- ... — e.g., in between FO and MSO lies FO(TC)

What kind of queries do we want to characterize

- *Binary relations* definable by *path expressions*?
- *Node sets* definable by *node expressions*?
- *Properties of trees* definable by *node expressions evaluated at the root*?

Possible types of characterizations:

- *Syntactic* (e.g. "*L* is equivalent to the two variable ...") versus *semantic* (e.g., "bisimulation invariant fragment ...")

Expressive power

Possible yardsticks for expressive power on trees:

- First-order logic (FO), (cf. Codd completeness of SQL/RA)
- Monadic second-order logic (MSO)
- ... — e.g., in between FO and MSO lies FO(TC)

What kind of queries do we want to characterize

- *Binary relations* definable by *path expressions*?
- *Node sets* definable by *node expressions*?
- *Properties of trees* definable by *node expressions evaluated at the root*?

Possible types of characterizations:

- *Syntactic* (e.g. "*L* is equivalent to the two variable ...") versus *semantic* (e.g., "bisimulation invariant fragment ...")

Decidable characterizations?

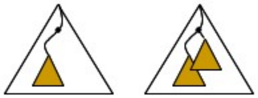
Expressive power (ct'd)

Descendant-only fragment

CoreXPath($\downarrow^*$) node expressions have the same expressive power as MSO formulas $\varphi(x)$ for which

(i) truth of $\varphi(x)$ at a node depends only on the subtree

(ii) $\varphi(x)$ does not distinguish children from descendants, i.e., the following operation preserves truth/falsity at the root:



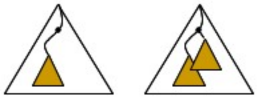
Expressive power (ct'd)

Descendant-only fragment

CoreXPath($\downarrow^*$) node expressions have the same expressive power as MSO formulas $\varphi(x)$ for which

(i) truth of $\varphi(x)$ at a node depends only on the subtree

(ii) $\varphi(x)$ does not distinguish children from descendants, i.e., the following operation preserves truth/falsity at the root:



Easy proof from *De Jongh-Sambin fixed point theorem for GL* and *Janin-Walukiewicz theorem for μ-calculus*, see M4M proceedings paper.

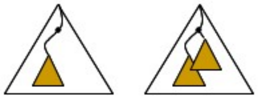
Expressive power (ct'd)

Descendant-only fragment

CoreXPath($\downarrow^*$) node expressions have the same expressive power as MSO formulas $\varphi(x)$ for which

(i) truth of $\varphi(x)$ at a node depends only on the subtree

(ii) $\varphi(x)$ does not distinguish children from descendants, i.e., the following operation preserves truth/falsity at the root:



Easy proof from *De Jongh-Sambin fixed point theorem for GL* and *Janin-Walukiewicz theorem for μ-calculus*, see M4M proceedings paper.

Moreover, the proof is *effective*: it yields a decision procedure.

A Lost Exercise

Exercise

1. Prove that finite sibling-ordered trees are bisimiliar iff they are ismorphic
2. Does this result hold for arbitrary trees?

Expressive power (ct'd)

What about the full *Core XPath* language?

Expressive power (ct'd)

What about the full *Core XPath* language?

No decidable characterization in terms of MSO is known. All we have is:

Expressive power (ct'd)

What about the full *Core XPath* language?

No decidable characterization in terms of MSO is known. All we have is:

Syntactic characterization of Core XPath (Marx-De Rijke 05)

Core XPath node expressions have the same expressive power as formulas $\phi(x)$ in the *two-variable fragment of* $FO[R_{\downarrow}, R_{\downarrow^*}, R_{\rightarrow}, R_{\rightarrow^*}]$.
There is a similar characterization for *path expressions*.

Brief recap

In summary...

- Core XPath is *reasonably expressive* yet *computationally attractive*.

Brief recap

In summary…

- Core XPath is *reasonably expressive* yet *computationally attractive*.
- In the remainder of this talk, we consider two extensions:

Brief recap

In summary…

- Core XPath is *reasonably expressive* yet *computationally attractive*.

- In the remainder of this talk, we consider two extensions:

  1. *Regular XPath*: the extension of Core XPath with full *transitive closure*.

Brief recap

In summary…

- Core XPath is *reasonably expressive* yet *computationally attractive*.

- In the remainder of this talk, we consider two extensions:

  1. *Regular XPath*: the extension of Core XPath with full *transitive closure*.
  2. *Core XPath 2.0*: the navigational core of XPath 2.0, featuring *path intersection and complementation* and more.

Motivation for adding transitive closure

Several people have proposed extending XPath with transitive closure, for various reasons.

Motivation for adding transitive closure

Several people have proposed extending XPath with transitive closure, for various reasons.

- Practical reasons: some applications require the use of transitive closure.

Motivation for adding transitive closure

Several people have proposed extending XPath with transitive closure, for various reasons.

- Practical reasons: some applications require the use of transitive closure.
- Core XPath extended with transitive closure has *full first-order expressive power*, is rich enough to express DTDs, and admits view based query rewriting with recursive views.

Motivation for adding transitive closure

Several people have proposed extending XPath with transitive closure, for various reasons.

- Practical reasons: some applications require the use of transitive closure.
- Core XPath extended with transitive closure has *full first-order expressive power*, is rich enough to express DTDs, and admits view based query rewriting with recursive views.
- From the perspective of PDL, extending XPath with transitive closure seems a very natural thing to do.

Motivation for adding transitive closure

Several people have proposed extending XPath with transitive closure, for various reasons.

- Practical reasons: some applications require the use of transitive closure.
- Core XPath extended with transitive closure has *full first-order expressive power*, is rich enough to express DTDs, and admits view based query rewriting with recursive views.
- From the perspective of PDL, extending XPath with transitive closure seems a very natural thing to do.

The extension of Core XPath with transitive closure is called *Regular XPath*.

Syntax of Regular XPath

- Regular XPath has two types of expressions:

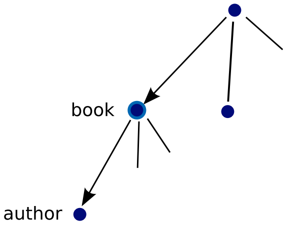
Syntax of Regular XPath

- Regular XPath has two types of expressions:
  - *path expressions*
    $\alpha \;::=\; \uparrow \mid \downarrow \mid \leftarrow \mid \rightarrow \mid \; . \; \mid \alpha/\beta \mid \alpha \cup \beta \mid \alpha^* \mid \alpha[\phi]$

Syntax of Regular XPath

- Regular XPath has two types of expressions:
  - *path expressions*
    $\alpha \ ::= \uparrow \mid \downarrow \mid \leftarrow \mid \rightarrow \mid \ . \ \mid \alpha/\beta \mid \alpha \cup \beta \mid \alpha^* \mid \alpha[\phi]$
  - *node expressions*
    $\phi \ ::= \ p \mid \neg\phi \mid \phi \wedge \psi \mid \langle\alpha\rangle$

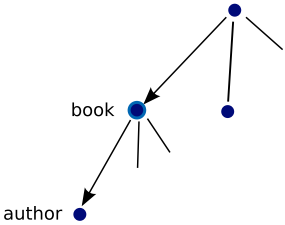
An example



"Go to the next book that has at least two authors."

In Regular XPath:

An example



"Go to the next book that has at least two authors."

In Regular XPath:

$$(\rightarrow [\neg \mathit{twoauthorbook}])^{*} / \rightarrow [\mathit{twoauthorbook}]$$

where *twoauthorbook* stands for
$\mathit{book} \wedge \langle \downarrow [\mathit{author}] / \rightarrow^{+} [\mathit{author}] \rangle$.

Another example

The following can be expressed in Regular XPath:

"The tree has an even number of nodes"

To see this, note that

Another example

The following can be expressed in Regular XPath:

"The tree has an even number of nodes"

To see this, note that

- Let (α *while* ϕ) be shorthand for $(.[\phi]/\alpha)^*$.

Another example

The following can be expressed in Regular XPath:

"The tree has an even number of nodes"

To see this, note that

- Let (α *while* ϕ) be shorthand for $(.[\phi]/\alpha)^*$.
- Let *root* be short for $\neg\langle\uparrow\rangle$.
  Let *leaf* be short for $\neg\langle\downarrow\rangle$.
  Let *first* be short for $\neg\langle\leftarrow\rangle$.
  Let *last* be short for $\neg\langle\rightarrow\rangle$.

Another example

The following can be expressed in Regular XPath:

"The tree has an even number of nodes"

To see this, note that

- Let (α *while* ϕ) be shorthand for $(.[\phi]/\alpha)^*$.
- Let *root* be short for $\neg\langle\uparrow\rangle$.
  Let *leaf* be short for $\neg\langle\downarrow\rangle$.
  Let *first* be short for $\neg\langle\leftarrow\rangle$.
  Let *last* be short for $\neg\langle\rightarrow\rangle$.
- Let *suc* be shorthand for $\downarrow[\textit{first}] \cup .[\textit{leaf}]/(\uparrow \textit{ while last})/\rightarrow$
  (the successor in depth first left-to-right ordering).

Another example

The following can be expressed in Regular XPath:

"The tree has an even number of nodes"

To see this, note that

- Let (α *while* ϕ) be shorthand for $(.[\phi]/\alpha)^*$.
- Let *root* be short for $\neg\langle\uparrow\rangle$.
  Let *leaf* be short for $\neg\langle\downarrow\rangle$.
  Let *first* be short for $\neg\langle\leftarrow\rangle$.
  Let *last* be short for $\neg\langle\rightarrow\rangle$.
- Let *suc* be shorthand for $\downarrow[\mathit{first}] \cup .[\mathit{leaf}]/(\uparrow \textit{ while last})/\rightarrow$ (the successor in depth first left-to-right ordering).
- Then $\Big\langle(\mathit{suc}/\mathit{suc})^*[\mathit{leaf}]/(\uparrow \textit{ while last})[\mathit{root}]\Big\rangle$ is true at the root iff the number of nodes is even.

One more example

- Consider game trees:
  - *leafs* are labeled by *Anne-wins* or *Bob-wins*
  - *inner nodes* are labeled by *Anne's-move* or *Bob's-move*

One more example

- Consider game trees:
  - *leafs* are labeled by *Anne-wins* or *Bob-wins*
  - *inner nodes* are labeled by *Anne's-move* or *Bob's-move*
- Puzzle:
  Show that "*Anne has a winning strategy*" is expressible.

Expressive power of Regular XPath

Expressive power of Regular XPath

- What is the expressive power of *Regular XPath*?
- We know that

$$FO \subsetneq \textit{Regular XPath} \subseteq FO(TC)$$

(The first inclusion follows from results by Marx 2004).

Expressive power of Regular XPath

- What is the expressive power of *Regular XPath*?
- We know that

  $$FO \quad \subsetneq \quad \textit{Regular XPath} \quad \subseteq \quad FO(TC)$$

  (The first inclusion follows from results by Marx 2004).

- A natural conjecture:

  $$\textit{Regular XPath} \quad \equiv \quad FO(TC)$$

  (after all, Regular XPath *has* a transitive closure operator!)

Expressive power of Regular XPath

- What is the expressive power of *Regular XPath*?
- We know that

$$FO \subsetneq \textit{Regular XPath} \subseteq FO(TC)$$

(The first inclusion follows from results by Marx 2004).

- A natural conjecture:

$$\textit{Regular XPath} \equiv FO(TC)$$

(after all, Regular XPath *has* a transitive closure operator!)

- We managed to prove this only if we extend Regular XPath with a "*within*" operator W:

$$T, n \models W\phi \quad \text{iff} \quad T_n, n \models \phi$$

(cf. temporal logics with forgettable past)

Expressive power of Regular XPath (ct'd)

Expressive power of Regular XPath (ct'd)

- *FO*(*TC*) is the extension of first-order logic with a transitive closure operator for binary relations.

Expressive power of Regular XPath (ct'd)

- $FO(TC)$ is the extension of first-order logic with a transitive closure operator for binary relations.

Theorem: (Ten Cate and Segoufin, PODS 2008, JACM 2010)

Regular XPath(W) path expressions define the same binary relations as $FO(TC)$ formulas with two free variables. Similarly for node expressions.

Expressive power of Regular XPath (ct'd)

- $FO(TC)$ is the extension of first-order logic with a transitive closure operator for binary relations.

Theorem: (Ten Cate and Segoufin, PODS 2008, JACM 2010)

Regular XPath(W) path expressions define the same binary relations as $FO(TC)$ formulas with two free variables. Similarly for node expressions.

- **Corollary:** Regular XPath(W) is closed under path intersection and complementation.

No axiomatizations are known yet for Regular XPath and Regular XPath(W).

Axiomatizations and complexity

No axiomatizations are known yet for Regular XPath and Regular XPath(W).

As for complexity,

Axiomatizations and complexity

No axiomatizations are known yet for Regular XPath and Regular XPath(W).

As for complexity,

- Query evaluation can still be performed in *PTime* even for *Regular XPath(W)*.
- Query containment is still *ExpTime-complete* for *Regular XPath* but it is *2ExpTime-complete* for *Regular XPath(W)*

Core XPath 2.0

XPath 2.0 extends XPath 1.0 with many features, including the following new navigational operations:

- *Intersection* and *complementation* of path expressions.
  α `intersect` β and α `except` β

XPath 2.0 extends XPath 1.0 with many features, including the following new navigational operations:

- *Intersection* and *complementation* of path expressions.
  α `intersect` β and α `except` β

  Example: $\downarrow^*$`[p] except `$\downarrow^*$`[q]/`$\downarrow^*$`[p]`

XPath 2.0 extends XPath 1.0 with many features, including the following new navigational operations:

- *Intersection* and *complementation* of path expressions.
  α `intersect` β and α `except` β

  Example: $\downarrow^*$`[p] except `$\downarrow^*$`[q]/`$\downarrow^*$`[p]`

- *Variables* and *`for` loops*
  `for $x in` α `return` β and
  α`[. is $x]`

XPath 2.0 extends XPath 1.0 with many features, including the following new navigational operations:

- *Intersection* and *complementation* of path expressions.
  α `intersect` β and α `except` β

  Example: $\downarrow^*$`[p] except `$\downarrow^*$`[q]/`$\downarrow^*$`[p]`

- *Variables* and `for` *loops*
  `for $x in` α `return` β and
  α`[. is $x]`

  Example:
  `for $x in . return `$\downarrow^*$`[p `$\wedge\neg\left\langle \uparrow^*\texttt{[q]/}\uparrow^*\texttt{[. is \$x]}\right\rangle$`]`

XPath 2.0 extends XPath 1.0 with many features, including the following new navigational operations:

- *Intersection* and *complementation* of path expressions.
  α `intersect` β and α `except` β

  Example: $\downarrow^*$`[p] except` $\downarrow^*$`[q]/`$\downarrow^*$`[p]`

- *Variables* and `for` *loops*
  `for $x in` α `return` β and
  α`[. is $x]`

  Example:
  `for $x in . return` $\downarrow^*$`[p` $\wedge\neg\langle$ $\uparrow^*$`[q]/`$\uparrow^*$`[. is $x]`$\rangle$`]`

- Core XPath 2.0 is the extension of Core XPath with these features.

- The *path intersection and complementation* turn Core XPath 2.0 into a version of Tarski's relation algebra (interpreted on finite ordered trees).

- The *path intersection and complementation* turn Core XPath 2.0 into a version of Tarski's relation algebra (interpreted on finite ordered trees).
- The *variables and* `for`*-loops* make it possible to give a linear translation from first-order logic to Core XPath 2.0:

$$TR(\phi(x,y)) = \texttt{for \$x in ., \$y in } \top \texttt{ return \$y[}TR'(\phi)\texttt{]}$$

$$\begin{aligned}
TR'(x=y) &= \langle \top\texttt{[. is \$x} \wedge \texttt{. is \$y]}\rangle \\
TR'(R_{\downarrow}xy) &= \langle \top\texttt{[. is \$x} \wedge \langle\downarrow\texttt{[. is \$y]}\rangle\texttt{]}\rangle \\
TR'(R_{\downarrow^*}xy) &= \langle \top\texttt{[. is \$x} \wedge \langle\downarrow^*\texttt{[. is \$y]}\rangle\texttt{]}\rangle \\
TR'(R_{\rightarrow}xy) &= \langle \top\texttt{[. is \$x} \wedge \langle\rightarrow\texttt{[. is \$y]}\rangle\texttt{]}\rangle \\
TR'(R_{\rightarrow^*}xy) &= \langle \top\texttt{[. is \$x} \wedge \langle\rightarrow^*\texttt{[. is \$y]}\rangle\texttt{]}\rangle
\end{aligned}$$

$$\begin{aligned}
TR'(\phi \wedge \psi) &= TR'(\phi) \wedge TR'(\psi) \\
TR'(\neg\phi) &= \neg TR'(\phi) \\
TR'(\exists x.\phi) &= \texttt{for \$x in } \top \texttt{ return } TR'(\phi)
\end{aligned}$$

where $\top$ is shorthand for $\uparrow^* / \downarrow^*$ (the universal relation)

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).
- The complexity of the *query equivalence* problem is *non-elementary*, both with and without variables.

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).
- The complexity of the *query equivalence* problem is *non-elementary*, both with and without variables.

  (even adding only *path intersection* to Core XPath makes it 2ExpTime-complete.)

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).
- The complexity of the *query equivalence* problem is *non-elementary*, both with and without variables.

  (even adding only *path intersection* to Core XPath makes it 2ExpTime-complete.)
- Is there anything interesting left to say about XPath 2.0?

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).
- The complexity of the *query equivalence* problem is *non-elementary*, both with and without variables.

  (even adding only *path intersection* to Core XPath makes it 2ExpTime-complete.)
- Is there anything interesting left to say about XPath 2.0?
- Sure! For example, *axiomatization*.

Expressivity and complexity

- Core XPath 2.0 has the *same expressive power as first-order logic*, both with and without variables (in the case with variables there is a linear translation).
- The complexity of the *query equivalence* problem is *non-elementary*, both with and without variables.

  (even adding only *path intersection* to Core XPath makes it 2ExpTime-complete.)
- Is there anything interesting left to say about XPath 2.0?
- Sure! For example, *axiomatization*.
- We have two *complete axiomatizations* of path equivalence in Core XPath 2.0: one with and one without variables.

The case without variables

- Recall that, without variables, Core XPath 2.0 is essentially a version of *Relation Algebra* interpreted on *finite sibling ordered trees*.

The case without variables

- Recall that, without variables, Core XPath 2.0 is essentially a version of *Relation Algebra* interpreted on *finite sibling ordered trees*.
- One apparent problem: Relation Algebra has no node tests.
  However, these can easily be translated away:

$$\begin{array}{llcl}
\textit{Pred}1. & \alpha[\phi \wedge \psi] & \equiv & \alpha[\phi][\psi] \\
\textit{Pred}2. & \alpha[\phi \vee \psi] & \equiv & \alpha[\phi] \cup \alpha[\psi] \\
\textit{Pred}3. & \alpha[\neg\phi] & \equiv & \alpha - \alpha[\phi] \\
\textit{Pred}4. & \alpha[\langle\beta\rangle] & \equiv & \alpha/((\beta/\top) \cap .)
\end{array}$$

The case without variables

- Recall that, without variables, Core XPath 2.0 is essentially a version of *Relation Algebra* interpreted on *finite sibling ordered trees*.
- One apparent problem: Relation Algebra has no node tests.
  However, these can easily be translated away:

$$\begin{array}{llcl} \textit{Pred}1. & \alpha[\phi \wedge \psi] & \equiv & \alpha[\phi][\psi] \\ \textit{Pred}2. & \alpha[\phi \vee \psi] & \equiv & \alpha[\phi] \cup \alpha[\psi] \\ \textit{Pred}3. & \alpha[\neg\phi] & \equiv & \alpha - \alpha[\phi] \\ \textit{Pred}4. & \alpha[\langle\beta\rangle] & \equiv & \alpha/((\beta/\top) \cap .) \end{array}$$

- Besides these axioms, our axiomatization for variable free Core XPath 2.0 contains *two groups of axioms*:
  - General axioms of Boolean Algebra and Relation Algebra
  - Axioms describing (first-order) properties of trees.

Axioms of Boolean algebra

- BA1. $\alpha \cup (\beta \cup \gamma) \equiv (\alpha \cup \beta) \cup \gamma$
- BA2. $\alpha \cap (\beta \cap \gamma) \equiv (\alpha \cap \beta) \cap \gamma$
- BA3. $\alpha \cup \beta \equiv \beta \cup \alpha$
- BA4. $\alpha \cap \beta \equiv \beta \cap \alpha$
- BA5. $\alpha \cup (\beta \cap \gamma) \equiv (\alpha \cup \beta) \cap (\alpha \cup \gamma)$
- BA6. $\alpha \cap (\beta \cup \gamma) \equiv (\alpha \cap \beta) \cup (\alpha \cap \gamma)$
- BA7. $\alpha \cup (\alpha \cap \beta) \equiv \alpha$
- BA8. $\alpha \cap (\alpha \cup \beta) \equiv \alpha$
- BA9. $\alpha \cup (\top - \alpha) \equiv \top$
- BA10. $\alpha \cap (\top - \alpha) \equiv \bot$
- BA11. $\alpha \cap (\top - \beta) \equiv \alpha - \beta$

The axioms of Relation algebra

- RA1. $\alpha/(\beta/\gamma) \equiv (\alpha/\beta)/\gamma$
- RA2. $\alpha/. \equiv \alpha$
- RA3. $(\alpha \cup \beta)/\gamma \equiv \alpha/\gamma \cup \beta/\gamma$
- RA4. $(\alpha \cup \beta)^{\smile} \equiv \alpha^{\smile} \cup \beta^{\smile}$
- RA5. $(\alpha/\beta)^{\smile} \equiv \beta^{\smile}/\alpha^{\smile}$
- RA6. $(\alpha^{\smile})^{\smile} \equiv \alpha$
- RA7. $(\alpha/(\top - (\alpha^{\smile}/\beta)) \subseteq \top \ \texttt{except} \ \beta$

- To completely axiomatize relation algebra, normally, one needs to add also *Venema's Rule*:

  If X is a relation variable not occurring in α *and* $X - (((\top - .)/X/\top) \cup (\top/X/(\top - .))) \subseteq \alpha$ *then* $\alpha \equiv \top$*.*

  Fortunately, this rule turns out to be derivable in our case.

The axioms for finite sibling ordered trees

*Tr*1.	$\downarrow^+ / \downarrow^+$	$\subseteq$	$\downarrow^+$
*Tr*2.	$\downarrow^+ \cap \uparrow^+$	$\equiv$	$\bot$
*Tr*3*a*.	$\downarrow^+$	$\equiv$	$\downarrow \cup (\downarrow / \downarrow^+)$
*Tr*3*b*.	$\downarrow$	$\equiv$	$\downarrow^+ - (\downarrow^+ / \downarrow^+)$
*Tr*4.	$.[\langle\uparrow\rangle]$	$\equiv$	$.[\uparrow^+ [\neg\langle\uparrow\rangle]]$
*Tr*5.	$\downarrow^+ / \uparrow^+$	$\equiv$	$\downarrow^+ \cup .[\langle\downarrow\rangle] \cup .[\langle\downarrow\rangle] / \uparrow^+$
*Tr*6.	$\rightarrow^+ / \rightarrow^+$	$\subseteq$	$\rightarrow^+$
*Tr*7.	$\rightarrow^+ \cap \leftarrow^+$	$\equiv$	$\bot$
*Tr*8*a*.	$\rightarrow^+$	$\equiv$	$\rightarrow \cup (\rightarrow / \rightarrow^+)$
*Tr*8*b*.	$\rightarrow$	$\equiv$	$\rightarrow^+ - (\rightarrow^+ / \rightarrow^+)$
*Tr*9.	$.[\leftarrow]$	$\equiv$	$.[\langle\leftarrow^+ [\neg\langle\leftarrow\rangle]\rangle]$
*Tr*10.	$\rightarrow^+ \cup \leftarrow^+$	$\equiv$	$(\uparrow / \downarrow) - .$
*Tr*11.	$. \cup \uparrow^+ \cup \downarrow^+ \cup$ $(\uparrow^* / \rightarrow^+ / \downarrow^*) \cup (\uparrow^* / \leftarrow^+ / \downarrow^*)$	$\equiv$	$\top$
Ind.	$\top[\langle\alpha\rangle]$	$\equiv$	$\top[\langle\alpha - (\alpha / <<)\rangle]$

Rounding up

Three languages:

Rounding up

Three languages:

- **Core XPath**: the navigational core of XPath 1.0
  Expressivity: FO^2
  Query evaluation: PTime
  Query equivalence: ExpTime-complete

Rounding up

Three languages:

- **Core XPath**: the navigational core of XPath 1.0
  Expressivity: FO^2
  Query evaluation: PTime
  Query equivalence: ExpTime-complete

- **Regular XPath(W)**: the extension with $*$ and W.
  Expressivity: same as $FO(TC)$
  Query evaluation: PTime
  Query equivalence: 2ExpTime-complete.

Rounding up

Three languages:

- **Core XPath**: the navigational core of XPath 1.0
  Expressivity: FO^2
  Query evaluation: PTime
  Query equivalence: ExpTime-complete

- **Regular XPath(W)**: the extension with $*$ and W.
  Expressivity: same as $FO(TC)$
  Query evaluation: PTime
  Query equivalence: 2ExpTime-complete.

- **Core XPath 2.0**: the navigational core of XPath 2.0
  Expressivity: same as FO.
  Query evaluation: PSpace-complete
  Query equivalence: non-elementary hard.

Some references

Expressive power:

Marx and De Rijke. *Semantic characterizations of navigational XPath*. SIGMOD Record 34(2), 2005

Ten Cate, Fontaine and Litak. *Some modal aspects of XPath*. M4M'07. Journal version for a special issue of JANCL 2010 in preparation

Ten Cate and Segoufin. *XPath, transitive closure logic, and nested tree walking automata*. Journal of the ACM, 2010. Extended abstract appeared in PODS 2008.

Axiomatization:

Ten Cate, Fontaine and Litak. *Some modal aspects of XPath*. M4M'07. Journal version for a special issue of JANCL 2010 in preparation

Ten Cate, Litak and Marx. Complete axiomatizations of XPath fragments. JAL 2010. Extended abstract presented at LiD 2008.

Ten Cate and Marx. *Axiomatizing the logical core of XPath 2.0*. ICDT'07.

Complexity:

Gottlob, Koch and Pichler. *Efficient algorithms for processing XPath queries*. TODS 30(2), 2005

Ten Cate and Lutz. *Query containment in very expressive XPath dialects*. PODS'07.