Queries on Trees

Jérôme Champavère Emmanuel Filiot Olivier Gauwin
Édouard Gilbert Sławek Staworko

INRIA Lille, Mostrare

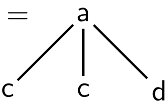
2008

Framework

n-ary queries on unranked labeled finite ordered trees

Trees

t =



finite alphabet: $\Sigma = \{a, b, c, d, e\}$

t is the structure $(\mathtt{D}, \mathtt{ch}_*, \mathtt{ns}_*, \mathtt{label})$ with:

- $\mathtt{D} = \{\epsilon,\ 1,\ 2,\ 3\}$: prefix-closed finite subset of $\mathbb{N}$
- $\mathtt{ch}_*$ = reflexive-transitive closure of $\mathtt{ch}$, defined by:

$$\mathtt{ch}(\pi_1, \pi_2) \ \Leftrightarrow\ \pi_2 = \pi_1 \cdot i \text{ for some } i \in \mathbb{N}$$

- $\mathtt{ns}_*$ = reflexive-transitive closure of $\mathtt{ns}$, defined by:

$$\mathtt{ns}(\pi_1, \pi_2) \ \Leftrightarrow\ \pi_1 = \pi \cdot i \text{ and } \pi_2 = \pi \cdot (i+1) \text{ for some } \pi, i \in \mathbb{N}^* \times \mathbb{N}$$

- $\mathtt{label} : \mathtt{D} \to \Sigma$. Can also be seen as a partition $(\mathtt{label}_a)_{a \in \Sigma}$ of $\mathtt{D}$.

$$\mathtt{label}(1 \cdot 3) = d \qquad \mathtt{label}_d(1 \cdot 3)$$

Queries

n-ary queries

- $q(t) \subseteq \mathtt{D}_t^n$

n=0: Boolean queries

- $q(t) = \emptyset$ or $q(t) = \{()\}$
- q defines $L_q = \{t \mid q(t) = \{()\}\}$

Questions

- expressivity
- complexity of:
  - model-checking: $x \in q(t)$
  - satisfiability: $\exists t, q(t) \neq \emptyset$

Existing material

Surveys

- Logics over unranked trees: an overview [Lib06]
- Automata, logic, and XML [Nev02b, Nev02a]
- Automata for XML – a survey [Sch07]
- Effective Characterizations of Tree Logics [Boj08a]
- Tree-walking automata [Boj08b]

Books

- Finite Model Theory [EF99, Lib04]
- Foundations of Databases [AHV95]

Outline

Part I

Classical Logics, Automata

Outline

1. Classical logics (FO, MSO)

2. Queries by Tree Automata
   - Tree-walking automata
   - Schema Languages & Tree Automata

FO

Well-formed formulas based on:

- predicates from the structure: $\mathtt{ch}_*, \mathtt{ns}_*, (\mathtt{label}_a)_{a \in \Sigma}$
- Boolean connectives: $\wedge, \neg$
- FO variables: $x, y \ldots$
- quantifiers on FO variables: $\exists x$

Querying using FO

We use free variables:

$$q(x) = \exists y.\exists z.(\mathtt{ch}_*(x,y) \wedge \mathtt{ch}_*(y,z) \wedge \mathtt{label}_a(z))$$

This way we can define queries of any arity.

FO: Available predicates

Why $\mathtt{ch}_*$ and $\mathtt{ns}_*$?

- because `ch` and `ns` are definable from $\mathtt{ch}_*$ and $\mathtt{ns}_*$ in FO...
- ... but the converse is false

So in the following, we suppose that `ch` and `ns` are also available.

Also definable in FO:

- unary predicates: `root` , `leaf` , `lc` (lastchild)
- binary predicates: `fc` (firstchild)

FO: Complexity

Model-checking

- PSPACE-complete (combined complexity). [Sto74, Var82]
- Remark: PSPACE-hardness is even true for the quantified propositional logic [GO99].

Satisfiability

- non-elementary on trees

FO: Restrictions on the number of variables

$$\mathsf{FO}^k = \text{FO formulas using only } k \text{ variables}$$

Variables might be reused

$$q(x) = \exists y.\exists z.(\mathtt{ch}_*(x,y) \wedge \mathtt{ch}_*(y,z) \wedge \mathtt{label}_a(z)) \notin \mathsf{FO}^2$$

but is equivalent to

$$q'(x) = \exists y.(\mathtt{ch}_*(x,y) \wedge \exists x.(\mathtt{ch}_*(y,x) \wedge \mathtt{label}_a(x))) \in \mathsf{FO}^2$$

Theorem ([Imm82, Var95, GO99])

The model-checking problem for FO^k (with $k \geq 2$) is P-complete on any structure.

FO: Restrictions on the number of variables

$$\mathsf{FO}^2 = \text{FO formulas using only 2 variables}$$

In FO^2, one cannot define `ch` and `ns` from $\mathtt{ch}_*$ and $\mathtt{ns}_*$ anymore. So `ch` and `ns` are added to the signature.

Complexity

Model-checking in FO^2 can be done in $O(|t|^2.|q|)$ [Imm82].

Expressivity

- FO is strictly more expressive than FO^2.
- example of Boolean query: trees where the leaf language is $(ab)^*$.

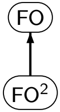
Links between FO^2 and XPath will be shown in Part 3.

Expressivity

$A \longrightarrow B$ $\quad A \subsetneq B$

$A \dashrightarrow B$ $\quad A \subseteq B$

$A \longrightarrow B$ $\quad A \not\subseteq B$



FO: Restrictions on the number of variables

Data values

- predicate $\sim$:
  $x \sim y$ if x and y are two attribute nodes that have the same value
- in XPath semantics: add tests of the form
  $/bib//book/@type = //collection/@style$

Decidability

- FO^2 [$\sim$,$\mathtt{ch}$,$\mathtt{ns}$] is decidable [BDM+06].
- FO^2 [$\sim$,$\mathtt{ch}$,$\mathtt{ns}$,$\mathtt{ch}_*$,$\mathtt{ns}_*$]: open question
- FO^3[$\sim$,$\mathtt{ch}$,$\mathtt{ns}$] is undecidable (even on strings) [BMS+06].

FO: Restrictions on the number of variables

$$\mathrm{FO}_n^k = \text{FO formulas using } (k \text{ bound variables}) + (n \text{ free variables})$$

We assume here that the n free variables are never quantified.

Some results on trees

- $\mathrm{FO}_2 = \mathrm{FO}_2^3$ [Mar05a] (his result is stronger)
- $\mathrm{FO}_3 = \mathrm{FO}_3^3$ [Mar05b]
- $\mathrm{FO}_n = \mathrm{FO}_n^3$
  - translate into a FO_0 formula on alphabet $\Sigma \times \mathbb{B}^n$,
  - $\mathrm{FO}_0 = \mathrm{FO}_0^3$ (consequence of [Mar05b], Th. 3)
  - backward translation: $\mathtt{label}_{(f,\vec{b})}(x)$ becomes $\mathtt{label}_f(x) \bigwedge_{\vec{b}_i=1} x = x_i$

MSO

$$\text{MSO} = \text{FO} + \text{quantification over monadic predicates}$$

"monadic predicates"	also seen as	"sets"
$X(x)$		$x \in X$

$\phi_{odd}(x, y)$

$$= \text{“}y \text{ is a descendant of } x \text{ and the path between them is of odd length”}$$

$$\begin{aligned} = \exists X.\exists Y.\quad & (\forall z.(X(z) \Leftrightarrow \neg Y(z))) \wedge \\ & (\forall z.(X(z) \vee Y(z) \Rightarrow \mathtt{ch}_*(x, z) \wedge \mathtt{ch}_*(z, y))) \wedge \\ & (X(x) \wedge Y(y)) \wedge \\ & (\forall z.\forall v.(\mathtt{ch}_*(x, z) \wedge \mathtt{ch}(z, v) \wedge \mathtt{ch}_*(v, y) \Rightarrow \\ & \qquad (X(z) \Rightarrow Y(v) \wedge Y(z) \Rightarrow X(v)))) \end{aligned}$$

Expressivity

MSO is strictly more expressive than FO (see ϕ_{odd}).

Expressivity

$A \longrightarrow B \qquad A \subsetneq B$

$A \dashrightarrow B \qquad A \subseteq B$

$A \textcolor{red}{\longrightarrow} B \qquad A \not\subseteq B$

$$\begin{tikzcd} \mathrm{MSO} \\ \mathrm{FO} \arrow[u] \\ \mathrm{FO}^2 \arrow[u] \end{tikzcd}$$

MSO: Complexity

Model-checking

- combined complexity: PSPACE_c [Sto74, Var82]
- data complexity: linear (by translation to automaton)

Satisfiability

- non-elementary on trees

Deciding membership to FO

Theorem ([BS05])

Given a regular tree language L, one can decide if L is definable in $FO_{ch,(label_a)_{a\in\Sigma}}$.

Open decision problem

Given a regular tree language L, is it possible to decide if L is definable in FO?

In other words, FO-definability is known to be decidable for unordered trees, but unknown for ordered trees.

Automata for FO

For a definition of automata recognizing exactly FO-definable languages, see [Boj04, Chapter 2].

Outline

1. Classical logics (FO, MSO)

2. Queries by Tree Automata
   - Tree-walking automata
   - Schema Languages & Tree Automata

Tree Automata for Queries

- Branching & Stepwise Tree automata
- Query automata
- Tree-walking automata (TWA)
- Schema languages

Branching & Stepwise Tree Automata I

- Automata over $\Sigma \times \{0, 1\}^n$
  - Canonical languages
  - Same expressive power as MSO
- Automata with selecting states
  - Boolean values into the states
  - Existential run-based queries [NPTT05]
  - Selecting tree automata [FGK03]
- Stepwise tree automata [CNT04]

Branching & Stepwise Tree Automata II

- Decision problems

Membership	PTIME
Non-emptiness	PTIME

- From MSO to tree automata: non-elementary size
  - Upper bound [TW68]
  - Lower bound [FG02]

Query Automata [NS99, NS02]

- Two-way deterministic tree automata [Mor94] over (un)ranked trees extended with a selection function
- Equivalent to MSO
- Decision problems

Non-emptiness	EXPTIME
Containment	EXPTIME
Equivalence	EXPTIME

Outline

1. Classical logics (FO, MSO)

2. Queries by Tree Automata
   - Tree-walking automata
   - Schema Languages & Tree Automata

Context

- Most work is done on ranked trees

Context

- Most work is done on ranked trees
- Still some definitions on unranked cases, but few results

Context

- Most work is done on ranked trees
- Still some definitions on unranked cases, but few results
- Thus we will work on trees of rank 2

Context

- Most work is done on ranked trees
- Still some definitions on unranked cases, but few results
- Thus we will work on trees of rank 2
- Structure: $\mathtt{label}, \mathtt{ch}_1, \mathtt{ch}_2$

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - a tree is accepted whenever the “accept” action is used

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - a tree is accepted whenever the "accept" action is used

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - a tree is accepted whenever the "accept" action is used

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - actions: accept, reject, move to parent with state q, move to left child with state q', ...
  - a tree is accepted whenever the "accept" action is used

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - actions: accept, reject, move to parent with state q, move to left child with state q', ...
  - a tree is accepted whenever the "accept" action is used

Tree-walking automata
on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - actions: accept, reject, move to parent with state q, move to left child with state q', ...
  - a tree is accepted whenever the "accept" action is used a tree can be rejected by looping, the "reject" action is not necessary
- Expressiveness:

Tree-walking automata

on ranked trees

- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - actions: accept, reject, move to parent with state q, move to left child with state q', ...
  - a tree is accepted whenever the "accept" action is used a tree can be rejected by looping, the "reject" action is not necessary
- Expressiveness:
  - any tree-walking automaton can be represented as a branching automaton, but with exponential blowup

Tree-walking automata

on ranked trees

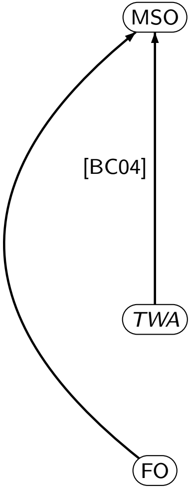
- A tree-walking automaton (TWA)[AU71]:
  - the automaton is located in some node (at first, the root) of the tree and in a given state
  - if some conditions are verified (label, being a leaf, being the left child of one's parent), decide of an action
  - actions: accept, reject, move to parent with state q, move to left child with state q', ...
  - a tree is accepted whenever the "accept" action is used a tree can be rejected by looping, the "reject" action is not necessary
- Expressiveness:
  - any tree-walking automaton can be represented as a branching automaton, but with exponential blowup
  - but the opposite is false: TWA are not as expressive as MSO [BC05]

Expressiveness (ranked case)

$A \longrightarrow B$ $A \subsetneq B$

$A \dashrightarrow B$ $A \subseteq B$

$A \longrightarrow B$ $A \not\subseteq B$



Deterministic TWA and their expressiveness

- Formulae recognised by deterministic TWA are stable by negation
- Formulae recognised by non-deterministic ones are not

Deterministic TWA and their expressiveness

- Formulae recognised by deterministic TWA are stable by negation
- Formulae recognised by non-deterministic ones are not

Deterministic TWA and their expressiveness

- Formulae recognised by deterministic TWA are stable by negation
- Formulae recognised by non-deterministic ones are not $\Rightarrow$ deterministic TWA are strictly less expressive than non-deterministic ones [MSS06]
- FO $\subseteq$ TWA $\subsetneq$ MSO [BC04]

Deterministic TWA and their expressiveness

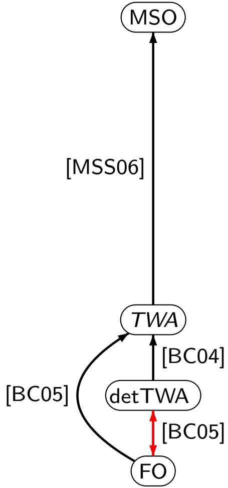
- Formulae recognised by deterministic TWA are stable by negation
- Formulae recognised by non-deterministic ones are not $\Rightarrow$ deterministic TWA are strictly less expressive than non-deterministic ones [MSS06]
- FO $\subseteq$ TWA $\subsetneq$ MSO [BC04]
- FO $\not\subseteq$ DTWA $\not\subseteq$ FO [BC05]

Expressiveness (ranked case)

$A \longrightarrow B$ $A \subsetneq B$

$A \dashrightarrow B$ $A \subseteq B$

$A \longrightarrow B$ $A \not\subseteq B$



Pebble tree-walking automata and stack discipline
[EH99]

- Add a finite number of pebble marked $\{1, \ldots, n\}$ to the automaton

Pebble tree-walking automata and stack discipline
[EH99]

- Add a finite number of pebble marked $\{1, \ldots, n\}$ to the automaton
- New tests: is there a pebble on current node?

Pebble tree-walking automata and stack discipline
[EH99]

- Add a finite number of pebble marked $\{1, \ldots, n\}$ to the automaton
- New tests: is there a pebble on current node?
- New actions: add a pebble to current position, remove a pebble from the current (or any) state

Pebble tree-walking automata and stack discipline
[EH99]

- Add a finite number of pebble marked $\{1, \ldots, n\}$ to the automaton
- New tests: is there a pebble on current node?
- New actions: add a pebble to current position, remove a pebble from the current (or any) state
- Stack discipline: if pebble 1 to i already can only add pebble $i + 1$ or remove pebble i

Expressiveness of pebble TWA

Expressiveness increases with number of pebble [BSSS06]

$\forall n \in \mathbb{N} \quad \mathrm{PTWA}_n \subsetneq \mathrm{PTWA}_{n+1}$

detPTWA $\subseteq$ PTWA[EH99]

it is not known if detPTWA $=$ PTWA
but there is no c s.t. $\mathrm{PTWA}_k \subseteq \mathrm{detPTWA}_{ck}$

Expressiveness without stack discipline

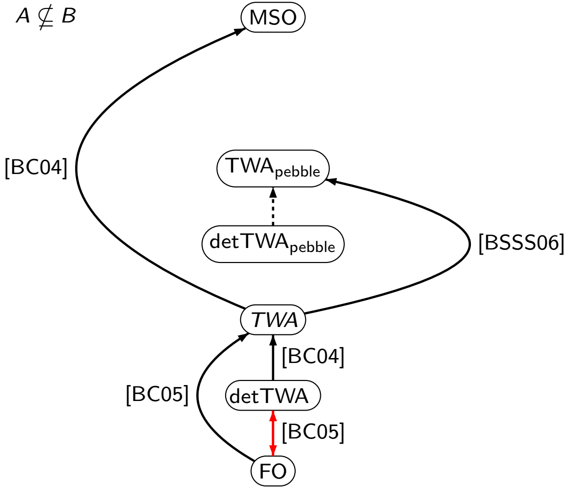
$\mathrm{MSO} \not\subseteq \mathrm{TWA}_{\text{no stack}}$
$\mathrm{TWA}_{\text{no stack}}$ emptiness is undecidable

Expressiveness (ranked case)

$A \longrightarrow B \quad A \subsetneq B$

$A \dashrightarrow B \quad A \subseteq B$

$A \longrightarrow B \quad A \not\subseteq B$



Unbounded pebble TWA

- We now allow an unbounded number of pebble (with stack discipline)

Expressiveness

Unbounded pebble TWA emptiness is undecidable
Invisible pebble TWA = MSO

Unbounded pebble TWA

- We now allow an unbounded number of pebble (with stack discipline)
- We can consider invisible pebble: only the top pebble presence can be tested[EHS07]

Expressiveness

Unbounded pebble TWA emptiness is undecidable
Invisible pebble TWA = MSO

Alternating tree-walking automata

- Two players $\forall, \exists$

Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$

Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$
- If $q \in Q_\forall$, then $\forall$ plays the next move in a given set of rules, otherwise, $\exists$ does

Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$
- If $q \in Q_\forall$, then $\forall$ plays the next move in a given set of rules, otherwise, $\exists$ does
- A tree is accepted if $\exists$ wins, rejected if $\forall$ does

Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$
- If $q \in Q_\forall$, then $\forall$ plays the next move in a given set of rules, otherwise, $\exists$ does
- A tree is accepted if $\exists$ wins, rejected if $\forall$ does
- $\exists$ wins if an accept rule is played by someone or if $\forall$ has no possible move, otherwise $\forall$ wins

Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$
- If $q \in Q_\forall$, then $\forall$ plays the next move in a given set of rules, otherwise, $\exists$ does
- A tree is accepted if $\exists$ wins, rejected if $\forall$ does
- $\exists$ wins if an accept rule is played by someone or if $\forall$ has no possible move, otherwise $\forall$ wins

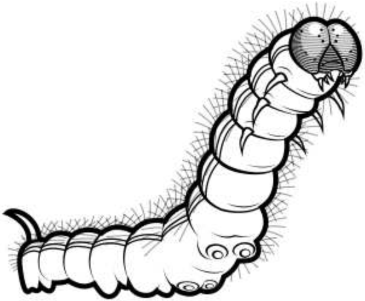
Alternating tree-walking automata

- Two players $\forall, \exists$
- Each state belongs to a player: $Q = Q_\forall \uplus Q_\exists$
- If $q \in Q_\forall$, then $\forall$ plays the next move in a given set of rules, otherwise, $\exists$ does
- A tree is accepted if $\exists$ wins, rejected if $\forall$ does
- $\exists$ wins if an accept rule is played by someone or if $\forall$ has no possible move, otherwise $\forall$ wins

Expressiveness

alternating TWA $=$ MSO

Caterpillar expressions [BKW00]



Caterpillar expressions [BKW00]

- Caterpillar expressions describe runs of tree-walking automata

Caterpillar expressions [BKW00]

- Caterpillar expressions describe runs of tree-walking automata
- Caterpillar alphabet on Σ
  - Commands letter `goleft`, `goright` and `goparent`

Caterpillar expressions [BKW00]

- Caterpillar expressions describe runs of tree-walking automata
- Caterpillar alphabet on Σ
  - Commands letter `goleft`, `goright` and `goparent`
  - Tests letter `leaf`, `isleft`, `isright` and labels $a \in \Sigma$

Caterpillar expressions [BKW00]

- Caterpillar expressions describe runs of tree-walking automata
- Caterpillar alphabet on Σ
  - Commands letter `goleft`, `goright` and `goparent`
  - Tests letter `leaf`, `isleft`, `isright` and labels $a \in \Sigma$
- Caterpillar words describe paths in TWA: `isleft a goleft b` describes paths going from a left child labeled a to its left child labeled b

Caterpillar expressions [BKW00]

- Caterpillar expressions describe runs of tree-walking automata
- Caterpillar alphabet on Σ
  - Commands letter `goleft`, `goright` and `goparent`
  - Tests letter `leaf`, `isleft`, `isright` and labels $a \in \Sigma$
- Caterpillar words describe paths in TWA: `isleft a goleft b` describes paths going from a left child labeled a to its left child labeled b
- Caterpillar expressions: regular expressions on caterpillar alphabet

Cutting caterpillar expressions

- New letters:
  - test $\langle c \rangle$ (nest) where c is a caterpillar expression: true if c applied to current node selects at least one path

Cutting caterpillar expressions

- New letters:
  - test $\langle c \rangle$ (nest) where c is a caterpillar expression: true if c applied to current node selects at least one path
  - command `cut` transform the whole tree into the subtree of the current node — local transform, does not apply outside nests

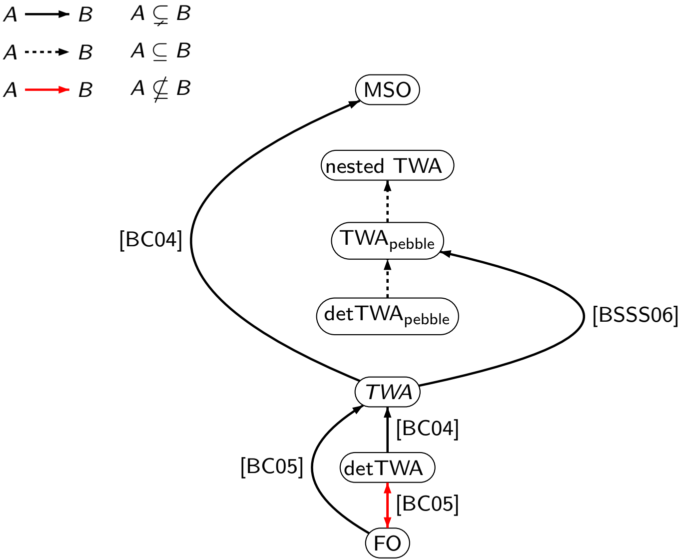
Cutting caterpillar expressions

- New letters:
  - test $\langle c \rangle$ (nest) where c is a caterpillar expression: true if c applied to current node selects at least one path
  - command cut transform the whole tree into the subtree of the current node — local transform, does not apply outside nests
- Expressiveness: if nesting is forbidden under scope of negation, posCAT = PTWA

Cutting caterpillar expressions

- New letters:
  - test $\langle c \rangle$ (nest) where c is a caterpillar expression: true if c applied to current node selects at least one path
  - command `cut` transform the whole tree into the subtree of the current node — local transform, does not apply outside nests
- Expressiveness: if nesting is forbidden under scope of negation, posCAT $=$ PTWA
- Expressiveness: if nesting is allowed under the scope of a negation, as expressive as *nested TWA* (not defined here)

Expressiveness (ranked case)



FO: *Extensions*

Notation: $\bar{z} = (z_1, \ldots, z_n)$

Adding Transitive Closure: TC^n

$$TC^n[\varphi(\bar{x}, \bar{y})](\bar{u}, \bar{v})$$

iff

$$\exists k,\ \exists (\bar{w}_i)_{i \in [1..k]},\ \varphi(\bar{u}, \bar{w}_1) \wedge \varphi(\bar{w}_1, \bar{w}_2) \wedge \ldots \wedge \varphi(\bar{w}_k, \bar{v})$$

By TC^n, we mean "parameter-free" transitive closure, i.e., $\bar{x}$ and $\bar{y}$ are exactly the free variable of φ.
We write TC^n_p for the non-parameter-free transitive closure (i.e., φ can have extra free variables).

FO: *Extensions*

$$\text{FO} + TC^1_p = \text{nested TWA} \quad [\text{tCS08}]$$

FO + TC^1 is often written FO*, and FO + TC^1_p is written FO(MTC).
FO + $TC^1 \subseteq$ FO + TC^1_p: it is unknown whether it is strict.

FO + $TC^1 \subseteq$ MSO

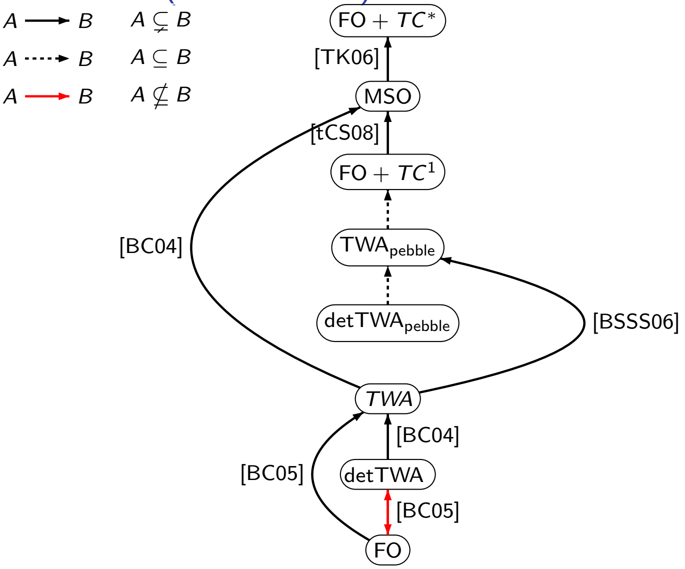
because $TC^1[\varphi(x,y)](u,v) \Leftrightarrow$
$\forall X.\ (u \in X \wedge \forall(x,y).\ (x \in X \wedge \varphi(x,y) \Rightarrow y \in X) \Rightarrow v \in X)$

FO $\subsetneq$ FO + $TC^1 \subsetneq$ MSO

- Transitive closure is not expressible in FO [Fag75].
- Adding TC^1 to FO is not enough to reach MSO [tCS08].

For properties of FO + TC^1 see [Kep06].

Expressiveness (ranked case)



FO: *Extensions*

$$\mathrm{FO} + TC^2$$

$\mathrm{FO} + TC^2 \not\subseteq \mathrm{MSO}$

(cf next slide)

$\mathrm{MSO} \subseteq \mathrm{FO} + TC^2$?

This is an open question. It could be the case that $\mathrm{MSO} \not\subseteq \mathrm{FO} + TC^k$, for all k.

FO: *Extensions*

$$\mathsf{FO} + TC^2 \not\subseteq \mathsf{MSO}$$

For instance $L = \{f(X,X) \mid X \in T_\Sigma\}$ is defined by:

$$\begin{aligned}\varphi = \quad & \mathtt{label}_f(\epsilon) \wedge \\ & \exists u_1.\exists v_1.\mathtt{fc}(\epsilon, u_1) \;\wedge\; \mathtt{ns}(u_1, v_1) \;\wedge\; \mathit{samelabel}(u_1, v_1) \wedge \\ & \neg(\exists w.\mathtt{ns}(v_1, w)) \wedge \\ & \forall u_2.\ \mathtt{ch}_*(u_1, u_2) \Rightarrow \exists v_2.\ TC^2[\psi(\bar{x}, \bar{y})](u_1, u_2, v_1, v_2)\end{aligned}$$

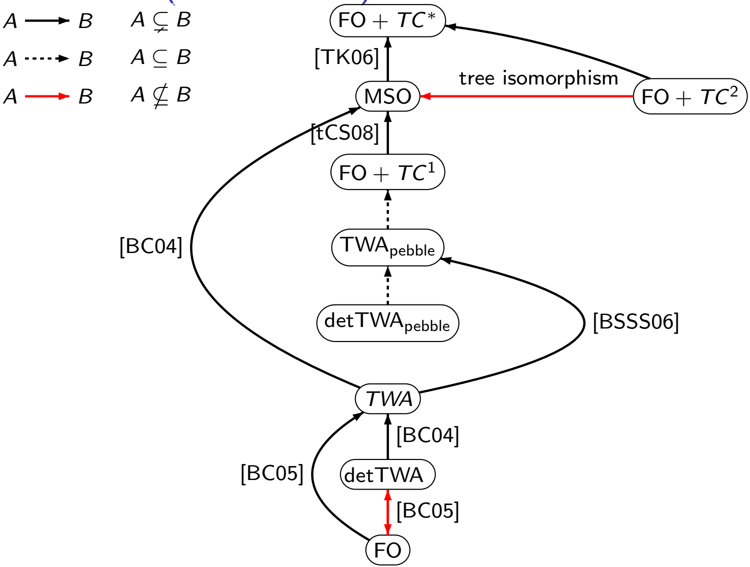
where ψ encodes a step isomorphism:

$$\begin{aligned}\psi(\bar{x}, \bar{y}) = \quad & \mathit{samelabel}(x_2, y_2) \quad \wedge \\ & (\mathtt{fc}(x_1, x_2) \wedge \mathtt{fc}(y_1, y_2)) \quad \vee \quad (\mathtt{ns}(x_1, x_2) \wedge \mathtt{ns}(y_1, y_2))\end{aligned}$$

with:

$$\mathit{samelabel}(x, y) = \bigvee_{a \in \Sigma} \mathtt{label}_a(x) \wedge \mathtt{label}_a(y)$$

Expressiveness (ranked case)



FO: *Extensions*

$$\text{FO} + detTC^1 = \text{detTWA}_{\text{pebble}} \quad \text{[EH06]}$$

Deterministic Transitive Closure of φ = TC on the functional part of φ

$\text{FO} + detTC^1 \subseteq \text{FO} + TC^1$

because
$detTC^1[\varphi(x,y)](u,v) \Leftrightarrow TC^1[\varphi(x,y) \wedge \forall z.\varphi(x,z) \Rightarrow z = y](u,v)$

$\text{FO} + detTC^1 \subsetneq \text{FO} + TC^1$?

Open question (see [Kep06]).

For some properties of FO + $detTC^1$ (linear order, even...) see [Kep06, EI95].

FO: *Extensions*

$$\text{FO} + posTC^1 = \text{TWA}_{\text{pebble}} \quad \text{[EH06]}$$

formulas of $\text{FO} + TC^1$ with TC^1 operators under an even number of negations

$\text{FO} + detTC^1 \quad \subseteq \quad \text{FO} + posTC^1 \quad \subseteq \quad \text{FO} + TC^1$

- inclusions due to TWA characterisations
- whether these 2 inclusions are strict is still open

$\text{FO} + TC^1 \subsetneq \text{MSO}$

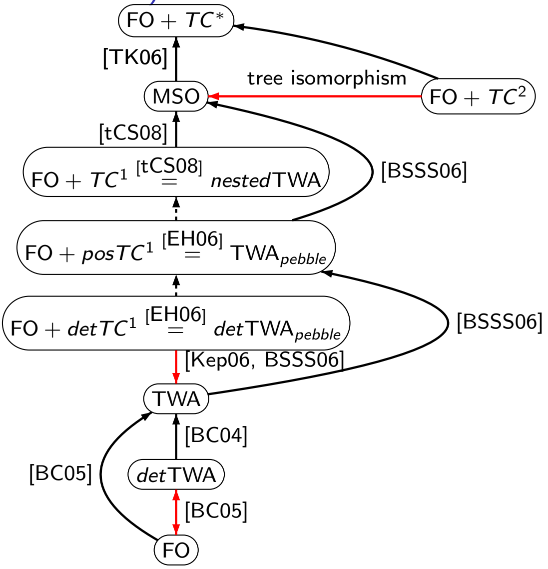
- separation language based on the branching structure [tCS08]

Expressiveness (ranked case)

$A \longrightarrow B$ $\quad A \subsetneq B$

$A \dashrightarrow B$ $\quad A \subseteq B$

$A \longrightarrow B$ $\quad A \not\subseteq B$



Outline

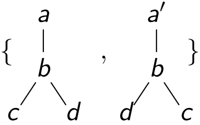
1. Classical logics (FO, MSO)
2. Queries by Tree Automata
   - Tree-walking automata
   - Schema Languages & Tree Automata

XML Schema Languages

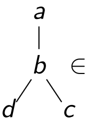
- Describe a set of XML documents
- Theoretical framework: no data, only structure
- Closer to tree grammars [MLM01] than to tree automata
- Tree automata: reference model for the expressiveness

Document Type Definitions (DTDs)
"Standard" DTDs

- Local tree languages

$L = \{$  $\}$

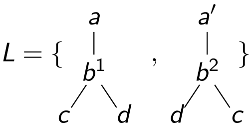
$$\begin{array}{lcl} a(q_b) & \longrightarrow & q_a \\ a'(q_b) & \longrightarrow & q_{a'} \\ b(q_c q_d) & \longrightarrow & q_b \\ b(q_d q_c) & \longrightarrow & q_b \\ c(\epsilon) & \longrightarrow & q_c \\ d(\epsilon) & \longrightarrow & q_d \end{array}$$

$\Rightarrow$  $\in L$!

  - Restriction: no competiting states
- Deterministic content models
  - One-unambiguous regular expressions [BKW98]
  - $ab + ac$: which a to match depends on the next symbol
- Polynomial complexity for other usual decision problems (membership, emptiness, containment), except intersection [MNS04]
- Lack of expressivity

Extended DTDs (EDTDs) [MNSB06, Sch07] I

- Alphabet extended with types (each type is associated to a unique symbol)



$$
\begin{array}{lcl}
a(q_{b^1}) & \rightarrow & q_a \\
a'(q_{b^2}) & \rightarrow & q_{a'} \\
b^1(q_c q_d) & \rightarrow & q_{b^1} \\
b^2(q_d q_c) & \rightarrow & q_{b^2} \\
c(\epsilon) & \rightarrow & q_c \\
d(\epsilon) & \rightarrow & q_d
\end{array}
$$

- Typing problem:
  - Valid assignment of types to the elements w.r.t. EDTD
  - (Consistent) combination of unary queries
- As expressive as (parallel) unranked tree automata of [BKWM01], thus equivalent to regular tree languages
  - Examples of such schema languages: Relax NG [CM01], XDuce [HP03]
  - Restricted EDTDs: single-type, restrained-competition

Extended DTDs (EDTDs) [MNSB06, Sch07] II

- Single-type EDTDs

$$\begin{array}{lcl} a(q_{b^1} + q_{b^2}) & \rightarrow & q_a \\ b^1(\epsilon) & \rightarrow & q_{b^1} \\ b^2(\epsilon) & \rightarrow & q_{b^2} \end{array} \qquad \begin{array}{lcl} a^1(q_{b^1}) & \rightarrow & q_{a^1} \\ a^2(q_{b^2}) & \rightarrow & q_{a^2} \\ b^1(\epsilon) & \rightarrow & q_{b^1} \\ b^2(\epsilon) & \rightarrow & q_{b^2} \end{array}$$

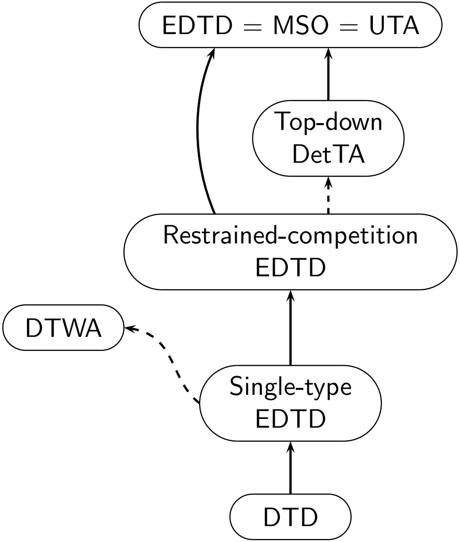
  - Element Declaration Consistent constraint (W3C XML Schemas)
  - Unique top-down typing
  - Validation with deterministic tree-walking automata

- Restrained-competition EDTDs

$$\begin{array}{lcl} a(q_{b^1} \cdot q_{b^2}) & \rightarrow & q_a \\ b^1(\epsilon) & \rightarrow & q_{b^1} \\ b^2(\epsilon) & \rightarrow & q_{b^2} \end{array}$$

  - Unique top-down left-to-right typing
  - Validation with deterministic top-down tree automata

Expressiveness of Schemas



(Homogeneous) regular tree languages

Path-closed tree languages

Local tree languages

Part II

Conjunctive Queries, Monadic Datalog

Outline

Conjunctive Queries

... seen as FO formulas

$\exists \overline{x}.\ \phi(\overline{x}, \overline{y})$ where ϕ is a conjunction of atomic predicates.
For instance:

$$\exists x \exists y \exists w\ R_1(x) \wedge R_2(x, y) \wedge R_3(x, w, z)$$

... seen as rules

$$answer(z) \leftarrow R_1(x), R_2(x, y), R_3(x, w, z)$$

... seen as terms of the Projection/Join algebra

$$\pi_Z(\ R_1(X) \bowtie R_2(X, Y) \bowtie R_3(X, W, Z)\ )$$

These 3 formalisms are equivalent (see [AHV95]).

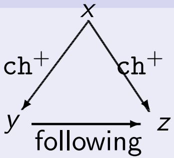
Conjunctive Queries over Trees

XPath axis $\mathcal{X}$: $\mathtt{ch}, \mathtt{ch}_*, \mathtt{ch}^+, \mathtt{ns}, \mathtt{ns}_*, \mathtt{ns}^+$, following and their inverse

$$\text{following} \;=\; (\mathtt{ch}_*)^{-1} \circ \mathtt{ns}^+ \circ \mathtt{ch}_*$$

Example

$\exists x \exists y \; \mathtt{ch}^+(x, y) \wedge \mathtt{ch}^+(x, z) \wedge \text{following}(x, z)$



Boolean Queries

Theorem ([GKS04])

Evaluation of Boolean CQ over $\mathcal{X}$ is NP-complete, even on a fixed tree.

Tractable fragments

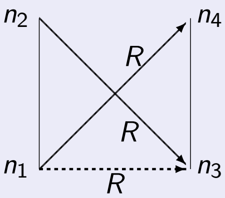
- X underbar property
- Acyclic conjunctive queries
- Twigs

$\underline{X}$ property

- R: a binary relation on the domain $\mathtt{D}_t$ of a tree t
- a total order $<$ on $\mathtt{D}_t$

Definition

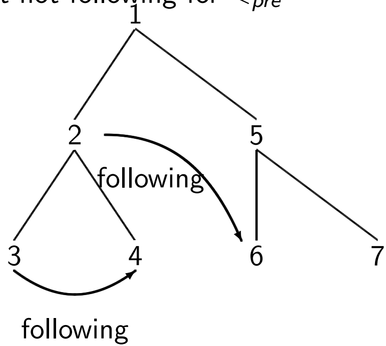
The relation R satisfies the $\underline{X}$ property wrt $<$ if $\forall n_1, n_2, n_3, n_4$ st $n_1 < n_2$ and $n_3 < n_4$:

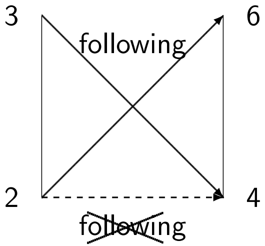


A set of relations $R_1, \ldots, R_n$ satisfies $\underline{X}$ wrt $<$ if every R_i does.

X property: Example

- $\{\mathtt{ch}^+, \mathtt{ch}^*\}$ for the preorder $<_{pre}$ $(ch^+(x,y) \Rightarrow x <_{pre} y)$
- $\{\mathtt{ch}, \mathtt{ns}, \mathtt{ns}^+, \mathtt{ns}^*\}$ for $<_{bflr}$
- but not following for $<_{pre}$





Dichotomy Result

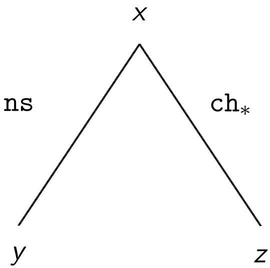
Theorem (Gottlob, Koch, Schulz, 2004)

For all $F \subseteq \mathcal{X}$, $CQ[F]$ Boolean queries can be evaluated in PTIME iff there is a total order $<$ such that F satisfies the $\underline{X}$ property wrt $<$.

Question: generelization to *n*-ary queries? Which complexity measure?
$\longrightarrow$ polynomial in the number of answers.

Acyclic Conjunctive Queries (ACQ)

Acyclic: the query graph is acyclic

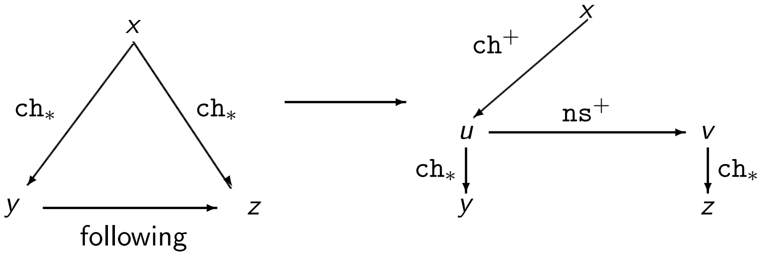


$$\exists x \exists y \exists z, \mathtt{ns}(x, y) \wedge \mathtt{ch}_*(y, z)$$

Expressiveness

- [GKS04]

$$CQ[\mathcal{X}] \underset{\text{exponential}}{\subsetneq} \bigcup ACQ[\mathcal{X}] \subseteq FO[\mathcal{X}]$$



- [Mar05b], over unranked trees,

$$\bigcup ACQ[FO_2] = FO_{nary}$$

ACQ Evaluation

- Yannakakis algorithm: $O(|q|.|db|.|q(db)|)$

$$\exists x \; R(x, y) \wedge R'(x, z)$$

- on trees t with predicates $\mathcal{X}$: $O(|q|.|t|^2.|q(t)|)$

$$\exists x \; \mathtt{ch}_*(x, y) \wedge \mathtt{ns}_*(x, z)$$

Outline

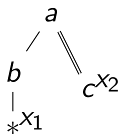
3. Conjunctive Queries over Trees
   - Definition, results and acyclic fragment
   - Twigs and Tree Patterns

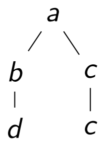
4. Monadic Datalog

Twigs: Testing containment [MS02]

Tree pattern

- (unordered and unranked) tree labeled with elements from $\Sigma \cup \{*\}$
- *child* and *descendant* edges
- n distinguished querying nodes (n-ary query)
- unary tree patterns ($n = 1$) equivalent to $XPath(*, [], //, /)$





$$Ans(p, t) = \{(1 \cdot 1, 2), (1 \cdot 1, 2 \cdot 1)\}$$

Containment (problem statement)

$$p_1 \subseteq p_2 \text{ if and only if } Ans(p_1, t) \subseteq Ans(p_2, t) \text{ for every } t \in T_\Sigma$$

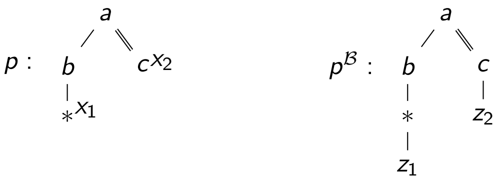
Booleanize your twigs

Boolean tree patterns

Tree patterns p with no querying nodes ($n = 0$)

$$Mod(p) = \{t \in T_\Sigma | t \text{ satisfies } p\}$$

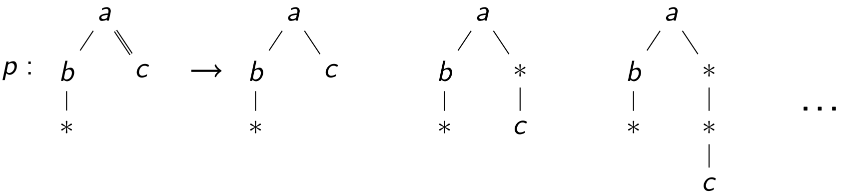
Then, $p_1 \subseteq p_2$ if and only if $Mod(p_1) \subseteq Mod(p_2)$



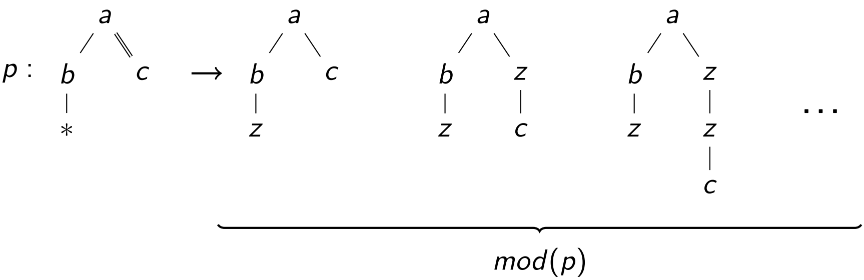
Proposition

For any two n-ary tree patterns p_1 and p_2: $p_1 \subseteq p_2 \Leftrightarrow p_1^{\mathcal{B}} \subseteq p_2^{\mathcal{B}}$

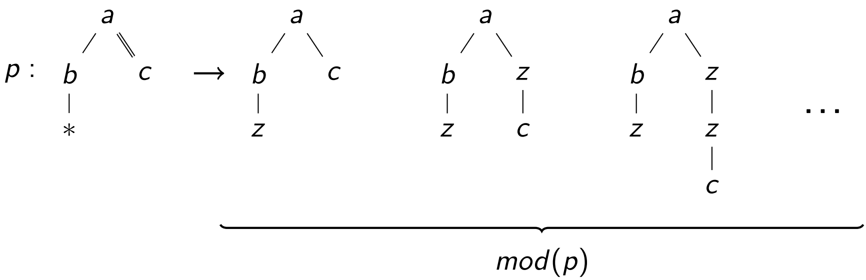
Canonical models of Boolean twigs



Canonical models of Boolean twigs



Canonical models of Boolean twigs

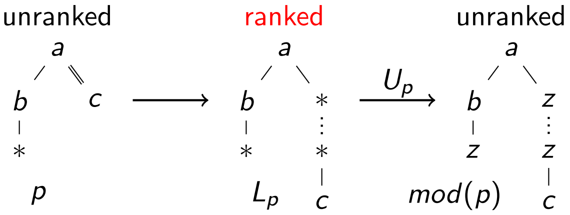


Proposition

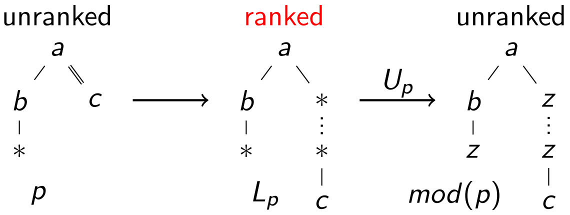
For any Boolean tree patterns p_1 and p_2:

$$p_1 \subseteq p_2 \Leftrightarrow mod(p_1) \subseteq Mod(p_2).$$

Testing containment of Boolean twigs: Outline



Testing containment of Boolean twigs: Outline



Main idea

$$p_1 \subseteq p_2 \Leftrightarrow mod(p_1) \subseteq Mod(p_2) \Leftrightarrow U_{p_1}(L_{p_1}) \subseteq Mod(p_2)$$
$$\Leftrightarrow L_{p_1} \subseteq U_{p_1}^{-1}(Mod(p_2)) \Leftrightarrow A_{p_1} \subseteq A_{p_2},$$

where:

- A_{p_1}: DFTA defining L_{p_1}
- A_{p_2}: AFTA defining $U_{p_1}^{-1}(Mod(p_2))$ complexity: $O(|p_1|2^{|p_2|})$

Testing containment: Conclusions

Positive results

$p_1 \subseteq p_2$ can be decided in time $O(|p_1||p_2|w^d)$, where:

- d is the number of $//$-edges in p_1
- w is the maximal length of $*/*/\ldots/*$ in p_2

Negative results

Deciding containment is coNP-complete. The result holds even if we:

- bound the number of occurrences of $*$
- bound the degree of the nodes of tree patterns

Efficient evaluation of tree patterns

`TwigStack` [BKS02]

- Interval representation used with a variant of B-tree index
- Two phase approach:
  1. Find and stack (partial) solutions to leaf-to-root paths
  2. Join partial solutions
- Linear in the size of the input and output
- I/O and CPU optimal if only //-edges used

`Twig`^{2}`Stack` [CLT$^+$06]

- Generalized tree pattern queries
- One phase bottom-up approach
- May stack elements that are not solutions
- In the worst case the whole document may be stored in main memory
- `HollisticTwigStack` [JLH$^+$07] addresses this shortcoming

Outline

Overview

- Few words on datalog
- Least fixed point
- Monadic datalog over trees

Datalog in (Very) Few Words

- Language used in deductive databases
- Extends conjunctive queries with recursion
- Example: transitive closure of a graph

$$
\begin{array}{l}
TC(x,y) :\!- \ Edge(x,y). \\
TC(x,y) :\!- \ Edge(x,z), TC(z,y).
\end{array}
$$

Model theoretic point of view:

$$
\begin{array}{l}
\forall x, y (Edge(x,y) \rightarrow TC(x,y)) \\
\forall x, y, z ((Edge(x,z) \wedge TC(z,y)) \rightarrow TC(x,y))
\end{array}
$$

- Remark: no function symbols (finite models), no negation

See chapter 12 of [AHV95] for more details

Least Fixed Point I

- P is a *fixed point* of operator F if $F(P) = P$
- The *least fixed point* $\mathit{lfp}(F)$ is the least element of the set of fixed points of F w.r.t. inclusion
- Every monotone operator F (i.e., $P \subseteq Q \Rightarrow F(P) \subseteq F(Q)$) has a least fixed point (Knaster-Tarski, cited by [Lib04]):

$$\mathit{lfp}(F) = \bigcap \{P | F(P) = P\}$$

- Computing the least fixed point (standard closure):

$$\begin{array}{lcl} P^0 & = & \emptyset \\ P^{i+1} & = & F(P^i) \\ \mathit{lfp}(F) & = & P^\infty = \bigcup_{i=0}^{\infty} P^i \end{array}$$

Stabilizes after n steps on finite structures, i.e., $P^\infty = P^n$

Least Fixed Point II

- Datalog *immediate consequence operator* $T_{\mathcal{P}}$(from [GK04]):

$$T_{\mathcal{P}}(Q) := Q \cup \{ f \quad | \quad \begin{array}{l} \exists \phi, \exists h :\!- \ b_1, \dots, b_n \in \mathcal{P} \\ \phi(h) = f \\ \phi(b_1), \dots, \phi(b_n) \in Q \end{array} \quad \}$$

- Example: program $\mathcal{P} = \left\{ \begin{array}{l} TC(x,y) :\!- \ Edge(x,y). \\ TC(x,y) :\!- \ Edge(x,z), TC(z,y). \end{array} \right\}$ and database $Q = \{Edge(1,2), Edge(2,3), Edge(3,1)\}$

$$\begin{array}{lcl} T^0_{\mathcal{P}} & = & Q = \{Edge(1,2), Edge(2,3), Edge(3,1)\} \\ T^1_{\mathcal{P}} & = & T^0_{\mathcal{P}} \cup \{TC(1,2), TC(2,3), TC(3,1)\} \\ T^2_{\mathcal{P}} & = & T^1_{\mathcal{P}} \cup \{TC(1,3), TC(2,1), TC(3,2)\} \\ T^3_{\mathcal{P}} & = & T^2_{\mathcal{P}} \end{array}$$

Finally, $lfp(T_{\mathcal{P}}) \overset{notation}{=} T^{\omega}_{\mathcal{P}} = T^3_{\mathcal{P}} = T^2_{\mathcal{P}} =$
$\{Edge(1,2), Edge(2,3), Edge(3,1), TC(1,2), TC(2,3), TC(3,1), \dots\}$

Monadic Datalog over Trees

- Datalog with unary head predicates
- Built-in predicates (for binary trees): $\mathtt{root}$, $\mathtt{leaf}$, $(\mathtt{label}_a)_{a\in\Sigma}$, $\mathtt{ch}_1$, $\mathtt{ch}_2$
- Example of query: select all nodes labeled by a at even height

$$
\begin{array}{lll}
Q_0(x) & :- \ \mathtt{root}(x). & \\
Q_{(i+1) mod 2}(x) & :- \ Q_i(y), \mathtt{ch}_k(y,x). & \quad \text{(for } k \in \{1,2\}\text{)} \\
Ans(x) & :- \ Q_0(x), \mathtt{label}_a(x). &
\end{array}
$$

The *query predicate* is *Ans*

Monadic Datalog over Trees: Complexity

Model Checking

Over ranked as well as unranked trees, monadic datalog has $O(|\mathcal{P}| * |dom|)$ combined complexity (theo. 4.2 of [GK04])

Proved by rewriting of $\mathcal{P}$ such that it is ground.

Satisfiability

Monadic datalog (over arbitrary finite structures) is NP-complete w.r.t. combined complexity (prop. 3.4 of [GK04])

- Membership: guess a proof tree
- Hardness: boolean conjuctive queries

For trees, satisfiability can be reduced to the emptiness problem for context-free languages [?]. What about the complexity?

Monadic Datalog over Trees: Expressiveness

Equivalence with MSO

A tree language is definable in monadic datalog exactly if it is definable in MSO (coro. 4.7 of [GK04])

Sketch of proof (for monadic queries):

- $\Rightarrow$ Encode the query defined by a monadic datalog program into an MSO formula (prop. 3.3 of [GK04])
- $\Leftarrow$ More intricate, different ways of prooving it:
  1. Using $\equiv_k^{MSO}$-types (theo. 4.4 of [GK04])
  2. Simulating query automata of Neven & Schwentick [NS02] (Section 4.3 of [GK04])
  3. Encoding tree automata with selecting states? (next slides)

Encoding a Tree Automaton A into a Monadic Datalog Program $\mathcal{P}$ I

$R_q(x)$ in *lfp* of $\mathcal{P}$ if a run of A can evaluate node x in state q:

$$\frac{a \longrightarrow q \in \text{rules}(A)}{R_q(x) :\!-\ \texttt{leaf}(x), \texttt{label}_a(x).}$$

$$\frac{f(q_1, q_2) \longrightarrow q \in \text{rules}(A)}{R_q(x) :\!-\ R_{q_1}(y), R_{q_2}(z), \texttt{ch}_1(x,y), \texttt{ch}_2(x,z), \texttt{label}_f(x).}$$

Encoding a Tree Automaton A into a Monadic Datalog Program $\mathcal{P}$ II

$L2F_q(x)$, *aka* $LeadsToFinal_q(x)$, in *lfp* of $\mathcal{P}$ if state q is used in a succesful run of A:

$$\frac{q \in \mathrm{final}(A)}{L2F_q(x) :\!-\ \mathtt{root}(x).}$$

$$\frac{f(q_1, q_2) \to q \in \mathrm{rules}(A)}{\begin{array}{c} L2F_{q_1}(y) :\!-\ L2F_q(x), \mathtt{ch}_1(x,y), \mathtt{ch}_2(x,z), \mathtt{label}_f(x), R_{q_2}(z). \\ L2F_{q_2}(z) :\!-\ L2F_q(x), \mathtt{ch}_1(x,y), \mathtt{ch}_2(x,z), \mathtt{label}_f(x), R_{q_1}(y). \end{array}}$$

Encoding a Tree Automaton A into a Monadic Datalog Program $\mathcal{P}$ III

$Ans(x)$ in *lfp* of $\mathcal{P}$ if x is selected by automaton A, i.e., x is evaluated in state $q \in S$, where $S \subseteq \text{states}(A)$ is the set of selecting states:

$$\frac{q \in S}{Ans(x) \mathrel{:-} R_q(x), L2F_q(x).}$$

Proposition: *Monadic datalog program $\mathcal{P}$ with Ans as query predicate simulates tree automaton with selecting states A*

Part III

μ-calculus, Modal Logics (Temporal Logics, XPath...)

Outline

Structure and formulae

- The structure used here is the one used by Barceló and Libkin. Most of the results are taken from [BL05a, ABL07].
- Tree t with two relations (or more) on position: child $\prec_{\mathtt{ch}}$ and next sibling $\prec_{\mathtt{ns}}$
- Formulae of $L_\mu[\prec]$:
  - constants a
  - second order variables X
  - $\top, \bot, \neg\phi, \phi \vee \phi'$
  - $\diamond(\prec)\phi$
  - $\mu X.\phi$ where X can only appears positively in ϕ

Interpretation

Given a tree t, nodes $s, s' \in \text{Domain}(t)$ and a valuation $v : \mathcal{X} \to \mathcal{P}(\text{Domain}(t))$

- logic operators are interpreted as usual
- $(t, v, s) \models a$ iff $t(s) = a$
- $(t, v, s) \models X$ iff $s \in v(X)$
- $(t, v, s) \models \diamond(\prec)\phi$ iff $(t, v, s') \models \diamond(\prec)\phi$ for some s' such that $s \prec s'$
- $(t, v, s) \models \mu X.\phi$ iff $s \in S$ where S is the least fix point of F_ϕ, defined by $F_\phi(P) = \{s' \mid (t, v[P/X], s') \models \phi\}$

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\bot, F_Y$ are constant

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\perp, F_Y$ are constant
  - $F_X(P) = P$ is increasing

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\bot, F_Y$ are constant
  - $F_X(P) = P$ is increasing
  - if F_ϕ and F'_ϕ both are increasing (resp. decreasing), then $F_{\phi\vee\phi'}(P) = F_\phi(P) \cup F_{\phi'}(P)$ is increasing (resp. decreasing)

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\bot, F_Y$ are constant
  - $F_X(P) = P$ is increasing
  - if F_ϕ and F'_ϕ both are increasing (resp. decreasing), then $F_{\phi \vee \phi'}(P) = F_\phi(P) \cup F_{\phi'}(P)$ is increasing (resp. decreasing)
  - if F_ϕ is increasing (resp. decreasing) then $F_{\diamond(\prec)\phi}$ is increasing (resp. decreasing)

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\bot, F_Y$ are constant
  - $F_X(P) = P$ is increasing
  - if F_ϕ and F'_ϕ both are increasing (resp. decreasing), then $F_{\phi\vee\phi'}(P) = F_\phi(P) \cup F_{\phi'}(P)$ is increasing (resp. decreasing)
  - if F_ϕ is increasing (resp. decreasing) then $F_{\diamond(\prec)\phi}$ is increasing (resp. decreasing)
  - if F_ϕ is increasing (resp. decreasing), then $F_{\neg\phi}(P) = \overline{F_\phi(P)}$ is decreasing (resp. increasing)

Interpretation

- $(t, v, s) \models \mu X.\phi(X)$ iff $s \in S$ where S is the least fix point of F
- Problem: is there a least fix point?
- The function $P \mapsto \{s' \mid (t, v[P/X], s') \models \phi\}$ is monotonically increasing because X can only appear positively in $\mu X.\phi$
  - $F_a, F_\top, F_\bot, F_Y$ are constant
  - $F_X(P) = P$ is increasing
  - if F_ϕ and F'_ϕ both are increasing (resp. decreasing), then $F_{\phi\vee\phi'}(P) = F_\phi(P) \cup F_{\phi'}(P)$ is increasing (resp. decreasing)
  - if F_ϕ is increasing (resp. decreasing) then $F_{\diamond(\prec)\phi}$ is increasing (resp. decreasing)
  - if F_ϕ is increasing (resp. decreasing), then $F_{\neg\phi}(P) = \overline{F_\phi(P)}$ is decreasing (resp. increasing)
  - if F_ϕ is increasing then $F_{\mu X.\phi}(P)$ is increasing

Unary and boolean queries

A formula ϕ from L_μ can be used as a unary query which selects in t the nodes s such that

$$(t, ., s) \models \phi$$

Unary and boolean queries

A formula ϕ from L_μ can be used as a unary query which selects in t the nodes s such that

$$(t, ., s) \models \phi$$

A formula ϕ from L_μ can be used as a boolean query which accepts a tree t iff

$$(t, ., \varepsilon) \models \phi$$

Example

Selects nodes which are ancestors of a node labelled by a:

$$\mu X.(a \vee \diamond(\prec_{\mathtt{ch}})X).$$

Expressiveness
of boolean queries

- $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}]$ cannot express first child...
  - $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$
  - $L_\mu^{\mathsf{full}}[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}]$: one can use $\diamond(\prec^-)\phi$, where $s \prec^- s'$ iff $s' \prec s$
- $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] = L_\mu^{\mathsf{full}}[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}] = MSO$

$L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] \subseteq \mathsf{MSO}$

One can rewrite any $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ formula into a MSO query as follow:

- $\langle a \rangle(x) = \mathtt{label}_a(x)$
- $\langle \mu X.\phi \rangle(z) = \exists X \quad (z \in X \wedge \forall x \in X \Rightarrow \langle \phi \rangle(x) \wedge (\forall Y(\forall y \in Y \Rightarrow \langle \phi \rangle(y)) \Rightarrow X \subseteq Y))$

$L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] \subseteq$ MSO

One can rewrite any $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ formula into a MSO query as follow:

- $\langle a \rangle(x) = \mathtt{label}_a(x)$
- $\langle X \rangle(x) = x \in X$

- $\langle \mu X.\phi \rangle(z) = \exists X \quad (z \in X \wedge \forall x \in X \Rightarrow \langle \phi \rangle(x) \wedge (\forall Y (\forall y \in Y \Rightarrow \langle \phi \rangle(y)) \Rightarrow X \subseteq Y))$

$L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] \subseteq$ MSO

One can rewrite any $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ formula into a MSO query as follow:

- $\langle a \rangle(x) = \mathtt{label}_a(x)$
- $\langle X \rangle(x) = x \in X$
- $\langle \diamond(\prec_{\mathtt{ch}})\phi \rangle(x) = \exists y \mid ch(y, x) \wedge \langle \phi \rangle(y)$, ...
- $\langle \mu X.\phi \rangle(z) = \exists X \quad (z \in X \wedge \forall x \in X \Rightarrow \langle \phi \rangle(x) \wedge (\forall Y(\forall y \in Y \Rightarrow \langle \phi \rangle(y)) \Rightarrow X \subseteq Y))$

$L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] \subseteq$ MSO

One can rewrite any $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ formula into a MSO query as follow:

- $\langle a \rangle(x) = \mathtt{label}_a(x)$
- $\langle X \rangle(x) = x \in X$
- $\langle \diamond(\prec_{\mathtt{ch}})\phi \rangle(x) = \exists y \mid ch(y, x) \wedge \langle \phi \rangle(y)$, ...
- $\langle \mu X.\phi \rangle(z) = \exists X \quad (z \in X \wedge \forall x \in X \Rightarrow \langle \phi \rangle(x) \wedge (\forall Y(\forall y \in Y \Rightarrow \langle \phi \rangle(y)) \Rightarrow X \subseteq Y))$

$L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}] \subseteq$ MSO

One can rewrite any $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ formula into a MSO query as follow:

- $\langle a\rangle(x) = \mathtt{label}_a(x)$
- $\langle X\rangle(x) = x \in X$
- $\langle\diamond(\prec_{\mathtt{ch}})\phi\rangle(x) = \exists y \mid ch(y,x) \wedge \langle\phi\rangle(y)$, ...
- $\langle\mu X.\phi\rangle(z) = \exists X \quad (z \in X \wedge \forall x \in X \Rightarrow \langle\phi\rangle(x) \wedge (\forall Y(\forall y \in Y \Rightarrow \langle\phi\rangle(y)) \Rightarrow X \subseteq Y))$

Finally, the whole query will be $\exists x \quad \mathtt{root}(x) \wedge \langle\phi\rangle(x)$

MSO $\subseteq L_\mu[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}, \prec_{\texttt{fc}}]$

Given a MSO query, let $\mathcal{A}$ be an equivalent deterministic automaton. We can encode $\mathcal{A}$ with a $L_\mu[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}, \prec_{\texttt{fc}}]$ formula.

Example

On ranked trees, $\prec_{\texttt{ch1}}, \prec_{\texttt{ch2}}$,
Automaton $Q = \{q_a, q_b\}$, $Q_F = \{q_a\}$
$a \rightarrow q_a, b \rightarrow q_b, f(q_a, q_b) \rightarrow q_a, f(q_b, q_a) \rightarrow q_b$

$$\mu X_a.a \vee f \wedge \diamond(\prec_{\texttt{ch1}})X_a \wedge \diamond(\prec_{\texttt{ch2}})(\mu X_b.b \vee f \wedge \diamond(\prec_{\texttt{ch1}})X_a \wedge \diamond(\prec_{\texttt{ch2}}))$$

Expressiveness
of unary queries

- $L_\mu[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}, \prec_{\texttt{fc}}]$ cannot express `root`...
- we need to use $L_\mu^{\mathsf{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}]$
- $L_\mu^{\mathsf{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}] = MSO$

Expressiveness
of unary queries

- $L_\mu[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}, \prec_{\texttt{fc}}]$ cannot express `root`...
- we need to use $L_\mu^{\mathsf{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}]$
- $L_\mu^{\mathsf{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{ns}}] = \mathit{MSO}$

Expressiveness

of unary queries

- $L_\mu[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}, \prec_{\mathtt{fc}}]$ cannot express `root`...
- we need to use $L_\mu^{\mathsf{full}}[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}]$
- $L_\mu^{\mathsf{full}}[\prec_{\mathtt{ch}}, \prec_{\mathtt{ns}}] = \mathit{MSO}$

Proofs: similar to Boolean queries, but with query automata instead

Complexities

- Because the structure of trees are acyclic, model checking of $L_\mu^{\text{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{sb}}]$ can be computed in $O(|\phi|^2\,|t|)$. Can be reduced for a subclass of L_μ (as expressive as MSO) to $O(|\phi|\,|t|)$
- Satisfiability of $L_\mu^{\text{full}}[\prec_{\texttt{ch}}, \prec_{\texttt{sb}}]$ is EXPTIME (slightly better bounds in the case of tree than in the general case)

Outline

First-order modal logics
on Unranked Trees

Strong links between:

- XPath
- Modal Logics (temporal, propositional...)
- FO

First-order modal logics

on Unranked Trees

Strong links between:

- XPath
- Modal Logics (temporal, propositional...)
- FO

$\longrightarrow$ remember the first slides about the model and FO

First-order modal logics
on Unranked Trees

Strong links between:

- XPath
- Modal Logics (temporal, propositional...)
- FO

$\longrightarrow$ remember the first slides about the model and FO
$\longrightarrow$ we won't talk about $\mathcal{L}$-definability (*i.e.*, given an automaton, is it equivalent to a formula of the logic $\mathcal{L}$?). See [Boj08a] for a survey.

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes"

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes" ✓ (always false)

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes" ✓ (always false)
- "select trees with even number of *a*-nodes"

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes" ✓ (always false)
- "select trees with even number of *a*-nodes" ×

Binary vs Unranked Trees

FO-definable queries on binary trees?

- “select trees with even number of nodes” ✓ (always false)
- “select trees with even number of *a*-nodes” ✗
- “select trees that have a leaf of even depth”

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes" ✓ (always false)
- "select trees with even number of *a*-nodes" ✗
- "select trees that have a leaf of even depth" ✓ (zigzag technic)

Binary vs Unranked Trees

FO-definable queries on binary trees?

- "select trees with even number of nodes" ✓ (always false)
- "select trees with even number of a-nodes" ✗
- "select trees that have a leaf of even depth" ✓ (zigzag technic)

not clear whether the last query is FO-definable on unranked trees.

XPath 1.0: a W3C recommendation (since 1999)

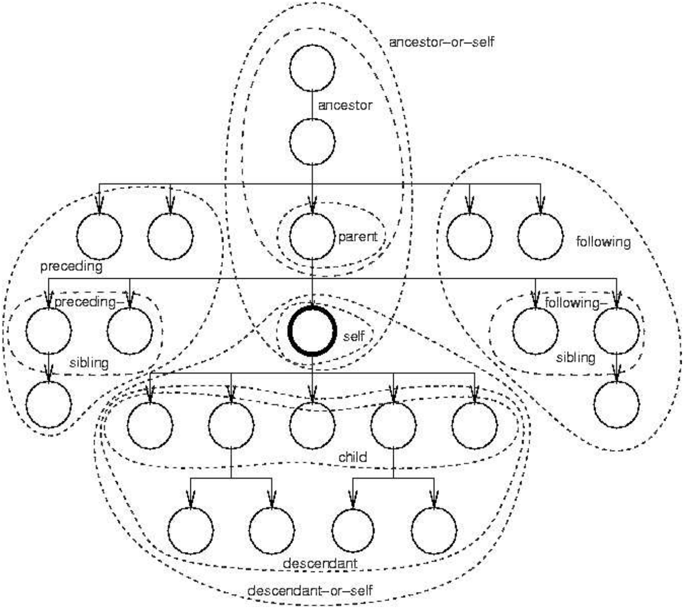
Example:

$/\texttt{descendant} :: a[\texttt{position}() > \texttt{last}() * 0.5 \ \texttt{or} \ \texttt{self} :: * = 100]$

Features:

- select nodes (monadic queries)
- navigation through axis (child... following, preceding)
- node test and filters: /ax1::ntst1[f1][f2[f3]]/...
- context-sensitive functions (position, last...)
- element types (element, attribute, instruction, comments)
- arithmetic operators $(+, - \ldots)$
- data operators/comparators (string-length...)
- aggregators (count, sum...)
- identifiers functions...
- type conversion functions...

XPath axes [Shi08]



XPath 1.0

- first implementations: exponential time in the size of the query
- PTIME combined complexity obtained in [GKP02, GKP03a]: $O(|D|^2.|Q|^4)$ in time, $O(|D|^2.|Q|^2)$ in space.

XPath 1.0

- first implementations: exponential time in the size of the query
- PTIME combined complexity obtained in [GKP02, GKP03a]: $O(|D|^2.|Q|^4)$ in time, $O(|D|^2.|Q|^2)$ in space.

Questions:

- linear time fragment?
- expressiveness? links to other logics?

CoreXPath

The navigational core of XPath

- defined by Gottlob, Koch and Pichler [GKP02, GKP03a]
- restriction to navigation through axis, filters, and nodetests

$$
\begin{array}{rcl}
\mathit{locpath} & ::= & \mathit{axis} :: \mathit{ntst} \mid \mathit{axis} :: \mathit{ntst}[\mathit{fexpr}] \mid /\mathit{locpath} \mid \mathit{locpath}/\mathit{locpath} \\
\mathit{fexpr} & ::= & \mathit{locpath} \mid \mathtt{not}\ \mathit{fexpr} \mid \mathit{fexpr}\ \mathtt{and}\ \mathit{fexpr} \mid \mathit{fexpr}\ \mathtt{or}\ \mathit{fexpr} \\
\mathit{axis} & ::= & \mathtt{self} \mid \mathtt{ch} \mid \mathtt{ch}_+ \mid \mathtt{ch}_* \mid \mathtt{ch}^{-1} \mid \mathtt{ch}_+^{-1} \mid \mathtt{ch}_*^{-1} \mid \mathtt{ns}_+ \mid \mathtt{ns}_+^{-1} \\
\mathit{ntst} & ::= & a,\ a \in \Sigma \mid *
\end{array}
$$

document order axis `following` and `preceding` are syntactic sugar:

- $\mathtt{following} :: \mathit{ntst}[\mathit{fexpr}] \equiv \mathtt{ch}_*^{-1} :: */\mathtt{ns}_+ :: */\mathtt{ch}_* :: \mathit{ntst}[\mathit{fexpr}]$
- $\mathtt{preceding} :: \mathit{ntst}[\mathit{fexpr}] \equiv \mathtt{ch}_*^{-1} :: */\mathtt{ns}_+^{-1} :: */\mathtt{ch}_* :: \mathit{ntst}[\mathit{fexpr}]$

CoreXPath complexity [GKP03b]

- query evaluation becomes linear: $O(|D|.|Q|)$
- it is P-hard wrt. combined complexity...
- ... even when t is limited to depth 3 and only axes $\mathtt{ch}$, $\mathtt{ch}^{-1}$, $\mathtt{ch}_*$ are allowed
- Positive-CoreXPath is LOGCFL-complete
- satisfiability is EXPTIME-complete

CoreXPath expressiveness

CoreXPath $\subseteq$ FO

CoreXPath	$/\mathtt{ch}_+ :: a$	$[\mathtt{ch} :: b]$	$/\mathtt{ch} :: c$
variables	y	z	x
$\phi(x) =$	$\exists y.\ \mathtt{label}_a(y)$	$\wedge\ \exists z.\ \mathtt{label}_c(z)$ $\wedge\ \mathtt{ch}(y,z)$	$\wedge\ \mathtt{label}_c(x)$ $\wedge\ \mathtt{ch}(y,x)$

CoreXPath expressiveness

CoreXPath $\subseteq$ FO

CoreXPath	$/\mathtt{ch}_+ :: a$	$[\mathtt{ch} :: b]$	$/\mathtt{ch} :: c$
variables	y	z	x
$\phi(x) =$	$\exists y.\ \mathtt{label}_a(y)$	$\wedge\ \exists z.\ \mathtt{label}_c(z)$	$\wedge\ \mathtt{label}_c(x)$
		$\wedge\ \mathtt{ch}(y, z)$	$\wedge\ \mathtt{ch}(y, x)$

FO $\not\subseteq$ CoreXPath

- example: select root if the leaf language is $(ab)^*$.
- in fact, CoreXPath $= \mathrm{FO}^2_1$ [Mar05b]

Expressiveness

$A \longrightarrow B$	$A \subsetneq B$
$A \dashrightarrow B$	$A \subseteq B$
$A \longrightarrow B$	$A \not\subseteq B$

$$
\begin{tikzcd}
\boxed{\mathrm{FO}_1} \\
\boxed{\mathrm{FO}_1^2 = \mathrm{CoreXPath}} \arrow[u]
\end{tikzcd}
$$

CondXPath [Mar04]
Conditional XPath

CondXPath =

CoreXPath

+ axis: $\mathtt{ns}, \mathtt{ns}_*, \mathtt{ns}^{-1}, \mathtt{ns}_*^{-1}$

+ *until* operator: $(\mathit{axis} :: \mathit{ntst}[\mathit{fexpr}])^+$ with $\mathit{axis} \in \{\mathtt{ch}, \mathtt{ch}^{-1}, \mathtt{ns}, \mathtt{ns}^{-1}\}$

CondXPath has the same complexity as CoreXPath (for both query evaluation and satisiability).

CondXPath [Mar04]
Conditional XPath

CondXPath $=$

CoreXPath

$+$ axis: $\mathtt{ns}, \mathtt{ns}_*, \mathtt{ns}^{-1}, \mathtt{ns}_*^{-1}$

$+$ *until* operator: $(\mathit{axis} :: \mathit{ntst}[\mathit{fexpr}])^+$ with $\mathit{axis} \in \{\mathtt{ch}, \mathtt{ch}^{-1}, \mathtt{ns}, \mathtt{ns}^{-1}\}$

CondXPath has the same complexity as CoreXPath (for both query evaluation and satisiability).

CondXPath $\subseteq$ FO

For instance $(\mathtt{ch} :: a[\mathtt{ns}_* :: b])^+$ translates to the FO formula:

$$\phi(x, y) =$$
$$\exists z.\ \mathtt{ns}_*(y, z) \wedge \mathtt{label}_b(z) \wedge$$
$$\neg(\exists s.\ \mathtt{ch}_*(x, s) \wedge \mathtt{ch}_*(s, y) \wedge (\neg\mathtt{label}_a(s) \vee \neg\exists s'.\ \mathtt{ns}_*(s, s') \wedge \mathtt{label}_b(s')))$$

CondXPath [Mar04]
Expressiveness

How to prove that FO $\subseteq$ CondXPath?

CondXPath [Mar04]
Expressiveness

How to prove that FO $\subseteq$ CondXPath?
Marx uses an intermediate logic: X_{until}.

$$\begin{tikzcd}
\text{CondXPath} \arrow[rr, bend left, "\text{previous slide}"] & & \text{FO} \arrow[dl, "2"] \\
& X_{\text{until}} \arrow[ul, "1"] &
\end{tikzcd}$$

$\mathsf{X}_{\text{until}}$

Syntax

$$\varphi ::\equiv a \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \theta(\varphi, \varphi') \quad (a \in \Sigma,\ \theta \in \{\Downarrow, \Leftarrow, \Rightarrow, \Uparrow\})$$

Arrows are interpreted as transitive closures of corresponding axis.

Semantics

$(t, \pi) \models a$	*iff*	$\mathtt{label}_a^t(\pi)$
$(t, \pi) \models \neg\varphi$	*iff*	$(t, \pi) \not\models \varphi$
$(t, \pi) \models \varphi \wedge \varphi'$	*iff*	$(t, \pi) \models \varphi$ and $(t, \pi) \models \varphi'$
$(t, \pi) \models \theta(\varphi, \varphi')$	*iff*	there exists π' s.t. $\theta^+(\pi, \pi')$ and $(t, \pi') \models \varphi$ and for all π'' s.t. $\pi\theta^+\pi''\theta^+\pi', (t, \pi'') \models \varphi'$

1. From X_{until} to CondXPath

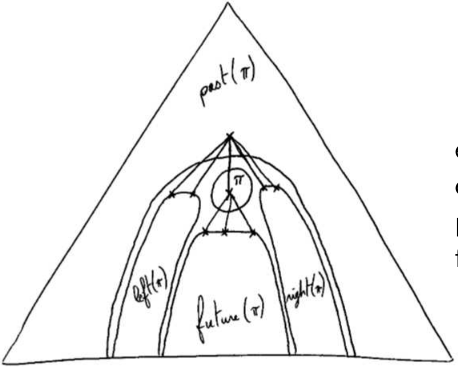
$$
\begin{array}{rcl}
X_{\text{until}} & \rightarrow & \text{CondXPath} \\
\hline
r(a) & = & \texttt{self} :: a \\
r(\neg\varphi) & = & \texttt{not}\ r(\varphi) \\
r(\varphi \wedge \varphi') & = & r(\varphi)\ \texttt{and}\ r(\varphi') \\
r(\theta(\varphi, \varphi')) & = & \theta :: *[r(\varphi)]\ \texttt{or}\ (\theta :: *[r(\varphi')])^{+}/\theta :: *[r(\varphi)]
\end{array}
$$

2. From FO to X_{until}

Separation technic

Theorem ([GHR94], adapted in [Mar04])

If every X_{until} formula is separable over trees, then X_{until} is FO-expressive.



"φ separable" means: equivalent to a Boolean combination of pure past/present/future/left/right formula

2. From FO to X_{until}

Separation technic

Theorem ([Mar04])

Each X_{until} formula is separable.

Query rewriting... with blowup.

Alternative proof

Theorem ([Mar05b])

- *any expansion of CoreXPath which is closed under complementation is FO-expressive*
- *CondXPath is closed under complementation*

Expressiveness

$A \longrightarrow B \quad A \subsetneq B$

$A \dashrightarrow B \quad A \subseteq B$

$A \longrightarrow B \quad A \not\subseteq B$

$$\begin{tikzcd} \mathrm{FO}_1 = \mathrm{CondXPath} \\ \mathrm{FO}_1^2 = \mathrm{CoreXPath} \arrow[u] \end{tikzcd}$$

RegularXPath$^{\approx}$ [tC06]

RegularXPath =

CoreXPath

$+$ axis: $\mathtt{ns}, \mathtt{ns}_*, \mathtt{ns}^{-1}, \mathtt{ns}_*^{-1}$

$+$ *transitive closure*: (RegularXPath expression)*

RegularXPath$^{\approx}$ =

RegularXPath

$+$ *loop* predicate: $[loop(\varphi)]_t = \{\pi \in \mathtt{D}_t \mid (\pi, \pi) \in [\varphi]_t\}$

Both have PTIME combined complexity for query evaluation.

RegularXPath$^{\approx}$ [tC06]

Expressiveness

Theorem

RegularXPath$^{\approx}$ and $FO + TC^1$ have the same expressive power.

In a preceding section, we saw that FO $+ TC^1$ is strictly less expressive than MSO [tCS08].

Corollary

The class of binary relations definable in RegularXPath$^{\approx}$ is closed under intersection and complementation.

It is only conjectured that adding *loop* increases expressivity, *i.e.*, that RegularXPath $\subsetneq$ RegularXPath$^{\approx}$.

Expressiveness

$A \longrightarrow B \qquad A \subsetneq B$

$A \dashrightarrow B \qquad A \subseteq B$

$A \longrightarrow B \qquad A \not\subseteq B$

$$
\begin{tikzcd}
\text{MSO} \\
\text{FO} + TC^1 = \text{RegularXPath}^{\approx} \arrow[u, "\text{[tCS08]}"'] \\
\text{FO} = \text{CondXPath} \arrow[u, "\text{[BC05, BSSS06]}"'] \\
\text{FO}^2 = \text{CoreXPath} \arrow[u]
\end{tikzcd}
$$

RegularXPath variants

- μRegularXPath adds a fixed-point operator [tC06] $\rightarrow$ MSO
- RegularXPath(W) adds a "subtree relativisation operator" [tCS08]

Beware: RegularXPath(W) $= \mathrm{FO} + TC^1_p$, whereas RegularXPath$^{\approx} = \mathrm{FO} + TC^1$. Remind that it is not known whether the inclusion $\mathrm{FO} + TC^1 \subseteq \mathrm{FO} + TC^1_p$ is strict.

XPath 2.0

XPath 2.0

adds the following features to XPath:

- *for* loops: `for $i in` R `return` S
- Boolean intersection (`intersect`) and complementation (`except`) on path expressions
- variables: n-ary queries
- node comparison tests (`is`)

CoreXPath 2 [tCM07]

CoreXPath 2

- *for* loops are interpreted as sets of nodes, not sequences
- no positional/aggregate: `position()`, `last()`, `count()`
- no value comparison operators

CoreXPath 2 [tCM07]

CoreXPath 2

- *for* loops are interpreted as sets of nodes, not sequences
- no positional/aggregate: `position()`, `last()`, `count()`
- no value comparison operators

- adding the last 2 features leads to undecidability.
- equivalence of CoreXPath 2 queries is decidable.
- of course, CoreXPath 2 is FO-expressive (adding `except` to CoreXPath is already sufficient).
- CoreXPath 2 $\leftrightarrow$ FO translations in linear time

Outline

Preliminaries: Linear Temporal Logic (LTL)

Syntax

$$\varphi ::\equiv a \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\,\varphi \mid \mathbf{X}^{-}\,\varphi \mid \varphi\,\mathbf{U}\,\psi \mid \varphi\,\mathbf{S}\,\psi$$

Semantics

Structure: $s = s_0 s_1 \cdots s_n$ a string over Σ

Interpretation: $(s, i) \models \varphi$ (φ is satisfied in s at position i)

label $(s, i) \models a$ **iff** $s_i = a$ **(i.e.** $label_s(i) = a$**)**

next $(s, i) \models \mathbf{X}\,\varphi$ **iff** $(s, i+1) \models \varphi$

prev $(s, i) \models \mathbf{X}^{-}\,\varphi$ **iff** $(s, i-1) \models \varphi$

until $(s, i) \models \varphi\,\mathbf{U}\,\psi$ **iff** $\exists j \geq i.\ (s, j) \models \psi \wedge \forall k \in \{i, \ldots, j-1\}.\ (s, k) \models \varphi$

since $(s, i) \models \varphi\,\mathbf{S}\,\psi$ **iff** $\exists j \leq i.\ (s, j) \models \psi \wedge \forall k \in \{j+1, \ldots, i\}.\ (s, k) \models \varphi$

Querying and expressivity

LTL Boolean queries

$$QA(\varphi, s) = \textbf{true} \Leftrightarrow (s, 0) \models \varphi$$

LTL unary queries

$$QA(\varphi, s) = \{i \in \{0, \ldots, |s|\} : (s, i) \models \varphi\}$$

Kamp's Theorem.

Over strings, LTL = FO

Tree Temporal Logic TL^{tree}

Syntax

$\varphi ::\equiv a \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}_\theta\, \varphi \mid \mathbf{X}_\theta^-\, \varphi \mid \varphi\, \mathbf{U}_\theta\, \psi \mid \varphi\, \mathbf{S}_\theta\, \psi \quad (\theta \in \{\downarrow, \leftarrow\})$

Semantics

$(t, \pi) \models \varphi$ reads "φ is satisfied in t at node π"
$(t, \pi) \models a$ iff $\mathtt{label}_t(\pi) = a$
$(t, \pi) \models \mathbf{X}_\downarrow \varphi$ iff $\exists \pi'$ such that $\pi \downarrow \pi'$ and $(t, \pi') \models \varphi$.
etc.

Theorem [Mar05a]

Over unranked ordered trees, $\mathrm{TL}^{tree} = \mathrm{FO}$ (Boolean and unary queries)

Computational tree logic CTL^*_{past}

Syntax

Node formulas: $\Phi ::\equiv a \mid \neg\Phi \mid \Phi \vee \Psi \mid \mathbf{E}_{\downarrow}\,\varphi \mid \mathbf{E}_{\rightarrow}\,\varphi$

Path formulas: $\varphi = \Phi? \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\,\varphi \mid \mathbf{X}^{-}\,\varphi \mid \varphi\,\mathbf{U}\,\psi \mid \varphi\,\mathbf{S}\,\psi$

Semantics

$(t, \pi) \models \Phi$ reads "Φ is satisfied in t at node π"
$(t, \pi) \models \mathbf{E}_{\downarrow}\varphi$ iff $\exists \pi_1 \downarrow \cdots \downarrow \pi_{i-1} \downarrow \pi \downarrow \pi_{i+1} \downarrow \cdots \downarrow \pi_k$ such that
$(\pi_1 \cdots \pi_k, i) \models_p \varphi$ where:
$(\pi_1 \cdots \pi_k, i) \models_p \Phi?$ iff $(t, \pi_i) \models \Phi$ **etc.**

Theorem [BL05b]

Over unranked ordered trees, $CTL^*_{past} = FO$ (Boolean and unary queries)

Propositional Dynamic Logic for trees PDL_{tree} [ABD+05]

Syntax

Path formulas:

$$\sigma ::\equiv \leftarrow \mid \rightarrow \mid \downarrow \mid \uparrow \mid \sigma/\sigma' \mid \sigma \cup \sigma' \mid \sigma^* \mid \varphi?$$

Propositions:

$$\varphi ::\equiv a \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}_\sigma\, \varphi \mid$$

Semantics

σ defines a binary relation $[\![\sigma]\!]_t$ on nodes of t

$(t, \pi) \models \mathbf{X}_\sigma\, \varphi$ iff $\exists \pi'$ such that $\pi\, [\![\sigma]\!]_t\, \pi'$ and $(t, \pi') \models \varphi$

Expressivity of PDL_{tree} [ABD+05]

Theorem

PDL_{tree} is equivalent to Regular XPath.

Theorem

PDL_{tree} restricted to

$$\sigma ::\equiv \leftarrow \mid \rightarrow \mid \downarrow \mid \uparrow \mid \sigma^* \mid \sigma/\varphi?$$

is equivalent to Conditional XPath which is equivalent to FO.

Theorem

PDL_{tree} restricted to

$$\sigma ::\equiv \leftarrow \mid \rightarrow \mid \downarrow \mid \uparrow \mid \sigma^*$$

is equivalent to Core XPath which is equivalent to FO_2.

References

[ABD+05] Loredana Afanasiev, Patrick Blackburn, Ioanna Dimitriou, Bertrand Gaiffe, Evan Goris, Maarten Marx, and Maarten de Rijke.
PDL for ordered trees.
Journal of Applied Non-Classical Logics, 15(2):115–135, 2005.

[ABL07] Marcelo Arenas, Pablo Barceló, and Leonid Libkin.
Combining temporal logics for querying XML documents.
In Springer-Verlag, editor, *Proceedings of International Conference on Database Theory*, volume 4353 of *Lecture Notes in Computer Science*, pages 359–374, 2007.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu.
Foundations of Databases.
1995.

[AU71] A. V. Aho and J. D. Ullmann.
Translations on a context-free grammar.
Information and Control, 19:439–475, 1971.

[BC04] Mikołaj Bojańczyk and Thomas Colcombet.

Tree-walking automata cannot be determinized.
In *31st International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 246–256. Springer Verlag, 2004.

[BC05] Mikołaj Bojańczyk and Thomas Colcombet.
Tree-walking automata do not recognize all regular languages.
In *37th Annual ACM Symposium on Theory of Computing*, pages 234–243, New York, NY, USA, 2005. ACM-Press.

[BDM+06] Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin.
Two-variable logic on data trees and XML reasoning.
In *Twenty-fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 10–19, 2006.

[BKS02] Nicolas Bruno, Nick Koudas, and Divesh Srivastava.
Holistic twig joins: optimal xml pattern matching.
In *SIGMOD Conference*, pages 310–321, 2002.

[BKW98] Anne Brüggemann-Klein and Derick Wood.

One-unambiguous regular languages.
Information and Computation, 142(2):182–206, May 1998.

[BKW00] Anne Brüggemann-Klein and Derick Wood.
Caterpillars: A context specification technique.
Markup Languages, 2(1):81–106, 2000.

[BKWM01] Anne Brüggemann-Klein, Derick Wood, and Makoto Murata.
Regular tree and regular hedge languages over unranked alphabets: Version 1, April 07 2001.

[BL05a] Pablo Barceló and Leonid Libkin.
Temporal logics over unranked trees.
In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 31–40, 2005.

[BL05b] Pablo Barcelo and Leonid Libkin.
Temporal logics over unranked trees.
In *20th Annual IEEE Symposium on Logic in Computer Science*, pages 31–40. IEEE Comp. Soc. Press, 2005.

[BMS+06] Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David.
Two-variable logic on words with data.
In *21st Annual IEEE Symposium on Logic in Computer Science*, pages 7–16. IEEE Comp. Soc. Press, 2006.

[Boj04] Mikołaj Bojańczyk.
Decidable Properties of Tree Languages.
PhD thesis, Warsaw University, 2004.

[Boj08a] Mikołaj Bojańczyk.
Effective characterizations of tree logics, 2008.
PODS'08 Keynote.

[Boj08b] Mikołaj Bojańczyk.
Tree-walking automata.
Tutorial at LATA'08, 2008.

[BS05] Michael Benedikt and Luc Segoufin.
Regular tree languages definable in FO and FOmod.

In *22nd International Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 327–339. Springer Verlag, 2005.

[BSSS06] Mikołaj Bojańczyk, Mathias Samuelides, Thomas Schwentick, and Luc Segoufin.
Expressive power of pebbles automata.
In *International Colloquium on Automata Languages and Programming (ICALP'06)*, Lecture Notes in Computer Science, pages 157–168. Springer Verlag, 2006.

[CLT+06] Songting Chen, Hua-Gang Li, Jun'ichi Tatemura, Wang-Pin Hsiung, Divyakant Agrawal, and K. Selçuk Candan.
Twig^{2}stack: Bottom-up processing of generalized-tree-pattern queries over xml documents.
In *VLDB*, pages 283–294, 2006.

[CM01] James Clark and Murata Makoto.
Relax ng specification, 2001.

[CNT04] Julien Carme, Joachim Niehren, and Marc Tommasi.

Querying unranked trees with stepwise tree automata.
In *19th International Conference on Rewriting Techniques and Applications*, volume 3091 of *Lecture Notes in Computer Science*, pages 105–118. Springer Verlag, 2004.

[EF99] H. Ebbinghaus and J. Flum.
Finite Model Theory.
Springer Verlag, Berlin, 1999.

[EH99] Joost Engelfriet and Hendrik Jan Hoogeboom.
Tree-walking pebble automata.
In Juhani Karhumäki, Hermann A. Maurer, Gheorghe Paun, and Grzegorz Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83, London, UK, 1999. Springer-Verlag.

[EH06] Joost Engelfriet and Hendrik Jan Hogeboom.
Nested pebbles and transitive closure.

In *23rd Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 477–488. Springer Verlag, 2006.

[EHS07] Joost Engelfriet, Hendrik Jan Hoogeboom, and Bart Samwel.
Xml transformation by tree-walking transducers with invisible pebbles.
In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 63–72, New York, NY, USA, 2007. ACM.

[EI95] Kousha Etessami and Neil Immerman.
Reachability and the power of local ordering.
Theoretical Computer Science, 148(2):227–260, 1995.

[Fag75] Ronald Fagin.
Monadic generalized spectra.
Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 21:89–96, 1975.

[FG02] Markus Frick and Martin Grohe.
The complexity of first-order and monadic second-order logic revisited.
In *LICS '02: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, pages 215–224, Washington, DC, USA, 2002.

[FGK03] Markus Frick, Martin Grohe, and Christoph Koch.
Query evaluation on compressed trees.
In *18th IEEE Symposium on Logic in Computer Science*, pages 188–197, 2003.

[GHR94] D.M. Gabbay, I. Hodkinson, and M. Reynolds.
Temporal Logic (Volume 1: Mathematical Foundations and Computational Aspects).
Oxford Science Publications, 1994.

[GK04] Georg Gottlob and Christoph Koch.
Monadic datalog and the expressive power of languages for web information extraction.

Journal of the ACM, 51(1):74–113, 2004.

[GKP02] Georg Gottlob, Christoph Koch, and Reinhard Pichler.
Efficient algorithms for processing xpath queries.
In *28th International Conference on Very Large Data Bases*, pages 95–106, Hong Kong, 2002.

[GKP03a] G. Gottlob, C. Koch, and R. Pichler.
Xpath query evaluation: Improving time and space efficiency.
In *In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE 03)*, 2003.

[GKP03b] Georg Gottlob, Christoph Koch, and Reinhard Pichler.
The complexity of xpath query evaluation.
In *22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 179–190, 2003.

[GKS04] Georg Gottlob, Christoph Koch, and Klaus U. Schulz.
Conjunctive queries over trees.

In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 189–200, New York, NY, USA, 2004. ACM-Press.

[GO99] Erich Grädel and Martin Otto.
On logics with two variables.
Theoretical Computer Science, 224:73–113, 1999.

[HP03] Haruo Hosoya and Benjamin Pierce.
Regular expression pattern matching for XML.
Journal of Functional Programming, 6(13):961–1004, 2003.

[Imm82] Neil Immerman.
Upper and lower bounds for first order expressibility.
Journal of Computer and System Science, 25:76–98, 1982.

[JLH+07] Zhewei Jiang, Cheng Luo, Wen-Chi Hou, Qiang Zhu, and Dunren Che.
Efficient processing of xml twig pattern: A novel one-phase holistic solution.
In *DEXA*, pages 87–97, 2007.

[Kep06] Stephan Kepser.
Properties of binary transitive closure logics over trees.
In *11th conference on Formal Grammar*, 2006.

[Lib04] Leonid Libkin.
Elements of Finite Model Theory.
Springer Verlag, 2004.

[Lib06] Leonid Libkin.
Logics over unranked trees: an overview.
Logical Methods in Computer Science, 3(2):1–31, 2006.

[Mar04] Maarten Marx.
Conditional XPath, the first order complete XPath dialect.
In *ACP SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 13–22. ACM-Press, 2004.

[Mar05a] Maarten Marx.
Conditional XPath.
ACM Transactions on Database Systems, 30(4):929–959, 2005.

[Mar05b] Maarten Marx.
First order paths in ordered trees.
In *International Conference on Database Theory*, pages 114–128, 2005.

[MLM01] M. Murata, D. Lee, and M. Mani.
Taxonomy of XML schema languages using formal language theory.
In *Extreme Markup Languages*, Montreal, Canada, 2001.

[MNS04] Wim Martens, Frank Neven, and Thomas Schwentick.
Complexity of decision problems for simple regular expressions.
In *Mathematical Foundations of Computer Science 2004, 29th International Symposium*, pages 889–900, 2004.

[MNSB06] Wim Martens, Frank Neven, Thomas Schwentick, and Geert Jan Bex.
Expressiveness and complexity of XML schema.

ACM Transactions of Database Systems, 31(3):770–813, 2006.

[Mor94] Etsuro Moriya.
On two-way tree automata.
Inf. Process. Lett., 50(3):117–121, 1994.

[MS02] Gerome Miklau and Dan Suciu.
Containment and equivalence for an xpath fragment.
In *PODS*, pages 65–76, 2002.

[MSS06] Anca Muscholl, Mathias Samuelides, and Luc Segoufin.
Complementing deterministic tree-walking automata.
Information Processing Letters, 99(1):33–39, 2006.

[Nev02a] Frank Neven.
Automata, logic, and XML.
In *Computer Science Logic*, Lecture Notes in Computer Science, pages 2–26. Springer Verlag, 2002.

[Nev02b] Frank Neven.
Automata theory for XML researchers.

SIGMOD Rec., 31(3):39–46, 2002.

[NPTT05] Joachim Niehren, Laurent Planque, Jean-Marc Talbot, and Sophie Tison.
N-ary queries by tree automata.
In *10th International Symposium on Database Programming Languages*, volume 3774 of *Lecture Notes in Computer Science*, pages 217–231. Springer Verlag, September 2005.

[NS99] Frank Neven and Thomas Schwentick.
Query automata.
In *Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems*, pages 205–214, 1999.

[NS02] Frank Neven and Thomas Schwentick.
Query automata over finite trees.
Theoretical Computer Science, 275(1-2):633–674, 2002.

[Sch07] Thomas Schwentick.
Automata for XML—a survey.

Journal of Computer and System Science, 73(3):289–315, 2007.

[Shi08] John W. Shipman.
XSLT Reference.
2008.

[Sto74] L. J. Stockmeyer.
The Complexity of Decision Problems in Automata Theory.
PhD thesis, Department of Electrical Engineering, MIT, 1974.

[tC06] Balder ten Cate.
The expressiveness of XPath with transitive closure.
In *25st ACM SIGMOD-SIGACT Symposium on Principles of Database Systems*. ACM-Press, 2006.

[tCM07] Balder ten Cate and Maarten Marx.
Axiomatizing the logical core of XPath 2.0.
In *International Conference on Database Theory*, 2007.

[tCS08] Balder ten Cate and Luc Segoufin.

XPath, transitive closure logic, and nested tree walking automata.
In *27th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2008.

[TK06] Hans-Jörg Tiede and Stephan Kepser.
Monadic second-order logic and transitive closure logics over trees.
In *13th Workshop on Logic, Language, Information and Computation*, volume 165 of *Electronical notes in theoretical computer science*, pages 189–199. Elsevier, 2006.

[TW68] J. W. Thatcher and J. B. Wright.
Generalized finite automata with an application to a decision problem of second-order logic.
Mathematical System Theory, 2:57–82, 1968.

[Var82] Moshe Y. Vardi.
The complexity of relational query languages.
In *14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[Var95] Moshe Y. Vardi.
On the complexity of bounded-variable queries.
In *Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 266–276, 1995.