Logics for XML

Pierre Genevès

Institut National Polytechnique de Grenoble
Institut National de Recherche en Informatique et Automatique

Ph.D Defense – December 4th 2006

Outline

XML and Schemas

Extensible Markup Language (XML)

- A markup langage for representing tree structures
- Representation is independent from processing

Schemas

- Each application defines constraints on documents using a schema
- Several formalisms exist for defining schemas (e.g., DTD, XML Schema, Relax NG)

XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example

XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #1

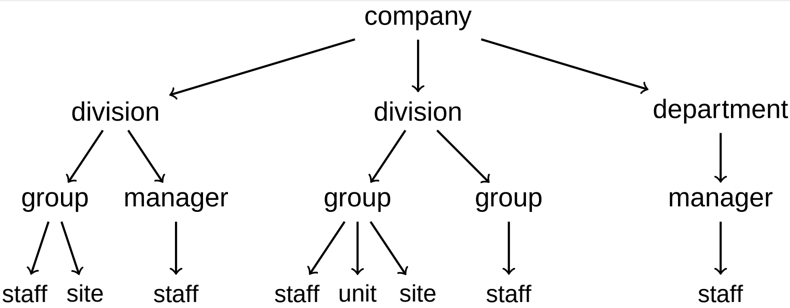
```
child::division/child::group
```

XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #1

```
child::division/child::group
```

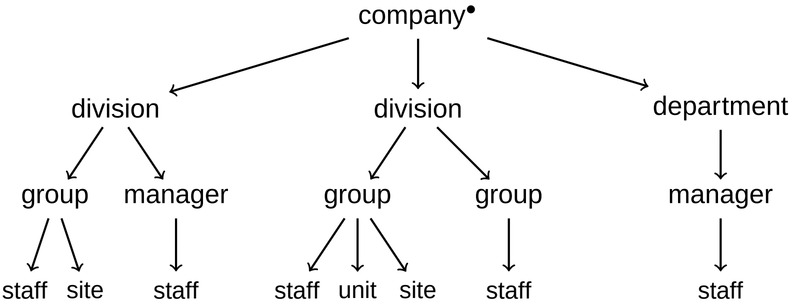


XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #1

```
child::division/child::group
```

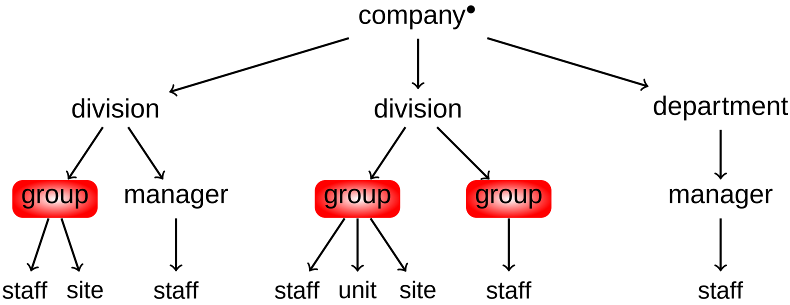


XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #1

`child::division/child::group`

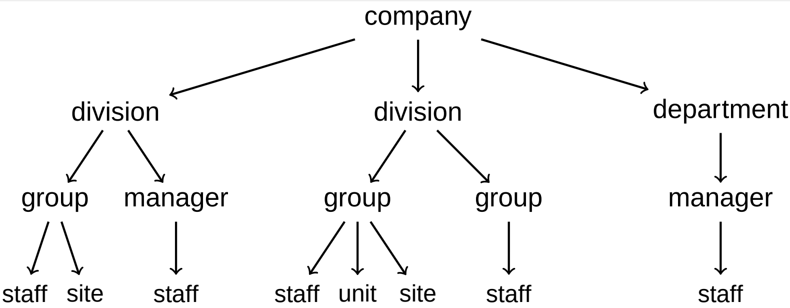


XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #2

```
parent::company/descendant::staff[not parent::manager]
```

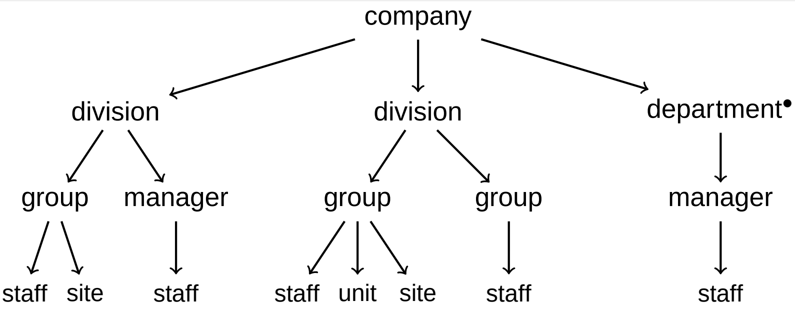


XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #2

```
parent::company/descendant::staff[not parent::manager]
```

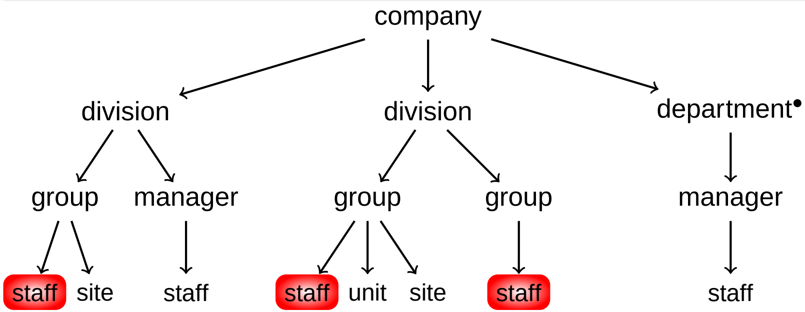


XPath: The Standard Query Language

- For navigating and extracting information from XML trees
- Evaluating an XPath query from a given context node returns a set of matching nodes

XPath Query Example #2

```
parent::company/descendant::staff[not parent::manager]
```



Motivation: Safe and Efficient XML Processing

- XPath plays a central role in key standards (e.g. XSLT, XQuery...)
- Static analysis of XPath has become crucial

Basic Static Analysis Tasks

1. XPath typing
2. XPath query comparisons
   - query containment, emptiness, overlap, equivalence

Main Applications

- Static analysis of host languages (e.g., type-checking of XSLT, XQuery), error-detection, optimization
- Checking integrity constraints in XML databases, XML security

- Objective: effectively analyzing XPath queries with schemas

Challenges

- Query comparisons and typing are undecidable for the complete XPath language

Open Questions

- What are the largest XPath fragments with decidable static analysis?
- Which fragments can be effectively decided in practice?
- Is there a generic algorithm able to solve all related XPath decision problems?

Difficulties

- Considered XPath operators and their combination (e.g., reverse axes, recursion)
- Checking properties on a possibly infinite set of XML documents
- Very high computational complexity

Challenges

- Query comparisons and typing are undecidable for the complete XPath language

Open Questions

- What are the largest XPath fragments with decidable static analysis?
- Which fragments can be effectively decided in practice?
- Is there a generic algorithm able to solve all related XPath decision problems?

Difficulties

- Considered XPath operators and their combination (e.g., reverse axes, recursion)
- Checking properties on a possibly infinite set of XML documents
- Very high computational complexity

Challenges

- Query comparisons and typing are undecidable for the complete XPath language

Open Questions

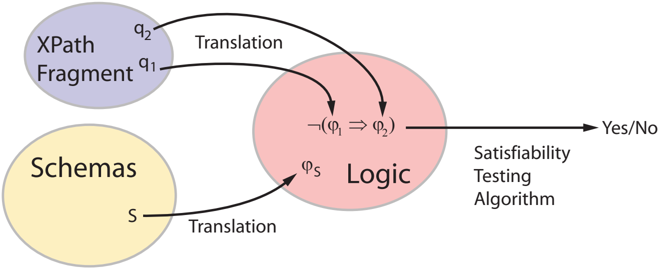
- What are the largest XPath fragments with decidable static analysis?
- Which fragments can be effectively decided in practice?
- Is there a generic algorithm able to solve all related XPath decision problems?

Difficulties

- Considered XPath operators and their combination (e.g., reverse axes, recursion)
- Checking properties on a possibly infinite set of XML documents
- Very high computational complexity

The Logical Approach: Overview

- Find an appropriate logic for reasoning on XML trees
- Formulate the problem into the logic and test satisfiability

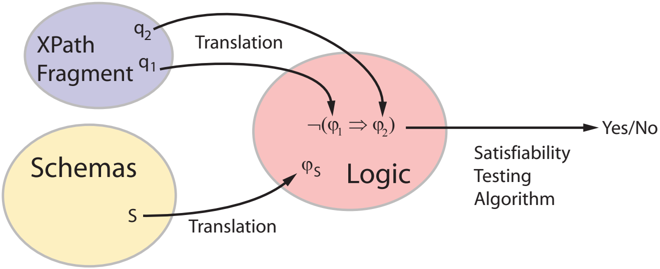


Critical Aspects

1. The logic must be expressive enough
2. The algorithm must be effective in practice for XML translations

The Logical Approach: Overview

- Find an appropriate logic for reasoning on XML trees
- Formulate the problem into the logic and test satisfiability

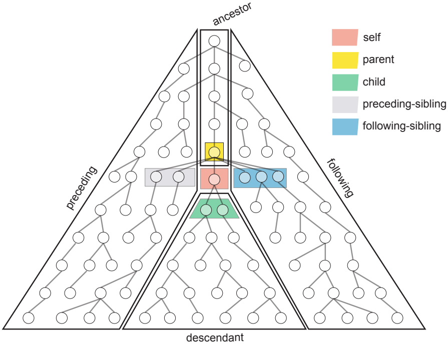


Critical Aspects

1. The logic must be expressive enough
2. The algorithm must be effective in practice for XML translations

A Large XPath Fragment

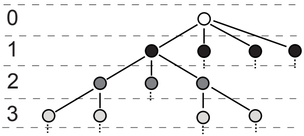
$$
\begin{array}{llcl}
\mathcal{L}_{\mathrm{XPath}} \ni e & ::= & & /p \\
 & & \mid & p \\
 & & \mid & e_1 \mid e_2 \\
 & & \mid & e_1 \cap e_2 \\
\mathit{Path}\ \ p & ::= & & p_1 / p_2 \\
 & & \mid & p[q] \\
 & & \mid & \mathrm{a}::\sigma \\
 & & \mid & \mathrm{a}::* \\
\mathit{Qualif}\ \ q & ::= & & q_1 \text{ and } q_2 \\
 & & \mid & q_1 \text{ or } q_2 \\
 & & \mid & \text{not } q \\
 & & \mid & p \\
 & & \mid & \cancel{\text{count}(p) = n} \\
 & & \mid & \cancel{p_1 = p_2} \\
\mathit{Axis}\ \ a & ::= & & \text{child} \\
 & & \mid & \text{self} \\
 & & \mid & \text{parent} \\
 & & \mid & \text{descendant} \\
 & & \mid & \text{descendant-or-self} \\
 & & \mid & \text{ancestor} \\
 & & \mid & \text{ancestor-or-self} \\
 & & \mid & \text{following-sibling} \\
 & & \mid & \text{preceding-sibling} \\
 & & \mid & \text{following} \\
 & & \mid & \text{preceding}
\end{array}
$$

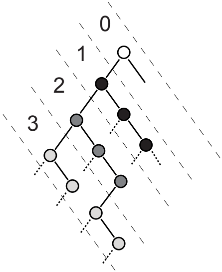


- Multi-directional tree navigation
- Node selection and path existence
- Almost full XPath

Models for XML Documents

- Finite ordered unranked trees, one label per node
- Bijective encoding of unranked trees as binary trees





- XML documents seen as finite ordered binary trees
  - Without loss of generality
  - XPath navigation must be expressed in binary style

Candidate Logics for XML

1. First-Order Logic (FO) and variants (over trees)
   - ✓ close to $\mathcal{L}_{\text{XPath}}$ expressive power
   - ✗ do not fully capture schemas
2. Monadic Second-Order Logic (WS2S)
   - ✓ extends FO with quantification over sets of nodes
   - ✓ captures $\mathcal{L}_{\text{XPath}}$ and finite tree automata
   - ✗ complexity for satisfiability: hyperexponential
   - ✗ blow-ups observable for XPath containment
3. Alternation-free fragment of the μ-calculus (AFMC)
   - ✓ supports schemas and XPath (when extended with *converse*)
   - ✓ complexity for satisfiability: $2^{O(n \cdot \log(n))}$ (with converse)
   - ✗ the solver a priori explores Kripke structures
   - ✗ formulas are more general than needed for XML
   - ✗ low performance in practice (does not scale to large instances)

Candidate Logics for XML

1. First-Order Logic (FO) and variants (over trees)
   - ✓ close to $\mathcal{L}_{\text{XPath}}$ expressive power
   - ✗ do not fully capture schemas
2. Monadic Second-Order Logic (WS2S)
   - ✓ extends FO with quantification over sets of nodes
   - ✓ captures $\mathcal{L}_{\text{XPath}}$ and finite tree automata
   - ✗ complexity for satisfiability: hyperexponential
   - ✗ blow-ups observable for XPath containment
3. Alternation-free fragment of the μ-calculus (AFMC)
   - ✓ supports schemas and XPath (when extended with *converse*)
   - ✓ complexity for satisfiability: $2^{O(n \cdot \log(n))}$ (with converse)
   - ✗ the solver a priori explores Kripke structures
   - ✗ formulas are more general than needed for XML
   - ✗ low performance in practice (does not scale to large instances)

Candidate Logics for XML

1. First-Order Logic (FO) and variants (over trees)
   - ✓ close to $\mathcal{L}_{\text{XPath}}$ expressive power
   - × do not fully capture schemas
2. Monadic Second-Order Logic (WS2S)
   - ✓ extends FO with quantification over sets of nodes
   - ✓ captures $\mathcal{L}_{\text{XPath}}$ and finite tree automata
   - × complexity for satisfiability: hyperexponential
   - × blow-ups observable for XPath containment
3. Alternation-free fragment of the μ-calculus (AFMC)
   - ✓ supports schemas and XPath (when extended with *converse*)
   - ✓ complexity for satisfiability: $2^{O(n \cdot \log(n))}$ (with converse)
   - × the solver a priori explores Kripke structures
   - × formulas are more general than needed for XML
   - × low performance in practice (does not scale to large instances)

Contribution Idea

- design a specific logic whose models are finite trees
- design the algorithm for satisfiability-testing

Remark #1

- Finite tree models
  - avoid exploring useless models
  - allow a bottom-up algorithm for satisfiability-testing

Remark #2

- Only finite recursion is of interest for XPath and Schemas

Contribution Idea

- design a specific logic whose models are finite trees
- design the algorithm for satisfiability-testing

Remark #1

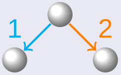
- Finite tree models
  - avoid exploring useless models
  - allow a bottom-up algorithm for satisfiability-testing

Remark #2

- Only finite recursion is of interest for XPath and Schemas

Formulas of the $\mathcal{L}_\mu$ Logic

- Programs $\alpha \in \{1, 2, \overline{1}, \overline{2}\}$ for navigating binary trees ($\overline{\overline{\alpha}} = \alpha$)



$$
\begin{array}{rclll}
\mathcal{L}_\mu \ni \varphi, \psi & ::= & & & \text{formula} \\
 & & \top & & \quad \text{true} \\
 & | & \sigma \quad | \quad \neg\sigma & & \quad \text{atomic prop (negated)} \\
 & | & \gamma^{\bullet} \quad | \quad \neg\gamma^{\bullet} & & \quad \text{context (negated)} \\
 & | & \varphi \vee \psi & & \quad \text{disjunction} \\
 & | & \varphi \wedge \psi & & \quad \text{conjunction} \\
 & | & \langle\alpha\rangle\,\varphi \quad | \quad \neg\,\langle\alpha\rangle\,\top & & \quad \text{existential (negated)} \\
 & | & X & & \quad \text{variable} \\
 & | & \mu X.\varphi & & \quad \text{unary fixpoint} \\
 & | & \mu\overline{X_i.\varphi_i} \text{ in } \psi & & \quad n\text{-ary fixpoint}
\end{array}
$$

- Closed formulas

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

- $\mu Z.\varphi$: finite recursion
- $\{\overline{1}, \overline{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \rightarrow \mathcal{L}_\mu$
- χ is the latest navigation step

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$:

- $\mu Z.\varphi$: finite recursion
- $\{\overline{1}, \overline{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \top^s$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$: *a*

- $\mu Z.\varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \rightarrow \mathcal{L}_\mu$
- χ is the latest navigation step

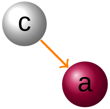
Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$: *a*

- $\mu Z.\varphi$: finite recursion
- $\{\overline{1}, \overline{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \rightarrow \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi =$

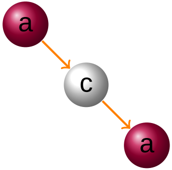


Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$: *a*



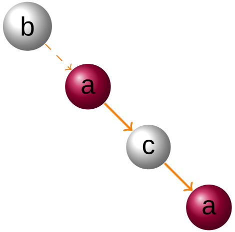
- $\mu Z.\varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma^*$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$: *a*



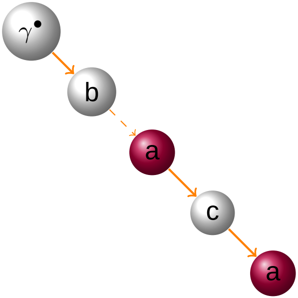
- $\mu Z.\varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi =$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*

in $\mathcal{L}_\mu$: $a \wedge (\mu Z . \langle \bar{2} \rangle \gamma^\bullet \vee \langle \bar{2} \rangle Z)$



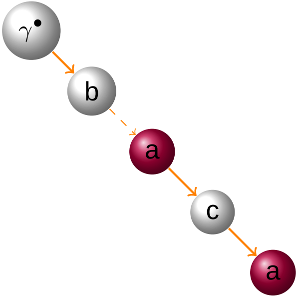
- $\mu Z . \varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma^\bullet$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::*a*/preceding-sibling::*b*

in $\mathcal{L}_\mu$: $a \wedge (\mu Z . \langle \overline{2} \rangle \gamma^{\bullet} \vee \langle \overline{2} \rangle Z)$



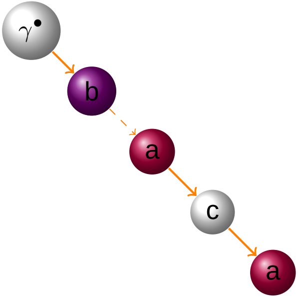
- $\mu Z . \varphi$: finite recursion
- $\{\overline{1}, \overline{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \rightarrow \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::a/preceding-sibling::b

in $\mathcal{L}_\mu$: $b \wedge [\mu Y . \langle 2 \rangle \, (a \wedge (\mu Z . \langle \bar{2} \rangle \, \gamma^\bullet \vee \langle \bar{2} \rangle \, Z)) \vee \langle 2 \rangle \, Y]$



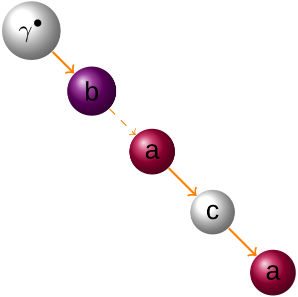
- $\mu Z . \varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \rightarrow \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma^\bullet$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::a/preceding-sibling::b

in $\mathcal{L}_\mu$: $b \wedge [\mu Y . \langle 2 \rangle (a \wedge (\mu Z . \langle \bar{2} \rangle \gamma^\bullet \vee \langle \bar{2} \rangle Z)) \vee \langle 2 \rangle Y]$



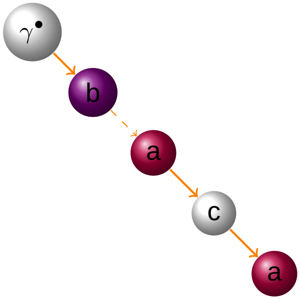
- $\mu Z . \varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma^\bullet$

Semantics of $\mathcal{L}_\mu$

- The set of models of a formula φ is the set of finite binary trees for which φ is satisfied on some node

Translating following-sibling::a/preceding-sibling::b

in $\mathcal{L}_\mu$: $b \wedge [\mu Y.\langle 2\rangle\, (a \wedge (\mu Z.\langle \bar{2}\rangle\, \gamma^\bullet \vee \langle \bar{2}\rangle\, Z)) \vee \langle 2\rangle\, Y]$



- $\mu Z.\varphi$: finite recursion
- $\{\bar{1}, \bar{2}\}$ required for forward axes!
- $\{1, 2\}$ required for reverse axes!
- Converse programs are crucial
- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
- χ is the latest navigation step
  - Initially: $\chi = \gamma^\bullet$

XPath and Closure under Negation of $\mathcal{L}_\mu$

A Very Important Property

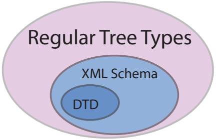
- $\mathcal{L}_{\text{XPath}}$ translations are never of the form: $\mu X . \langle \alpha \rangle X \vee \langle \overline{\alpha} \rangle X$
- $\mathcal{L}_{\text{XPath}}$ translations are always cycle-free
  - no occurrence of both a path and its converse between a fixpoint binder and its variable

- Restricting $\mathcal{L}_\mu$ to cycle-free formulas ensures closure under negation of recursion
  - The negation of finite recursion remains finite recursion
  - $\neg\varphi$ is expressible in $\mathcal{L}_\mu$ for all $\varphi \in \mathcal{L}_\mu$
    - Computable using De Morgan's laws, e.g. $\neg \langle \alpha \rangle \varphi = \neg \langle \alpha \rangle \top \vee \langle \alpha \rangle \neg\varphi$, and $\neg\mu X . \varphi = \mu X . \neg\varphi \left\{ {}^{\neg X}/_{X} \right\}$
- Crucial for implication (e.g., XPath containment)

Translating Schemas into $\mathcal{L}_\mu$

Models for Schemas

- Schema languages correspond to subclasses of *regular tree types* [Murata et al., 2005]



Translating Regular Tree Types into $\mathcal{L}_\mu$

- The binary encoding of trees also applies to tree types
- Binary tree type expressions model schemas without loss of generality
- They can be translated into the logic ($t_{\text{schema}}(\cdot) : \mathcal{L}_{\text{type}} \to \mathcal{L}_\mu$)
  - the *n*-ary fixpoint binder is used for mutually recursive definitions
  - only forward programs $\alpha \in \{1, 2\}$ are used

Formulating Decision Problems to be Solved

- $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$ and $t_{\text{schema}}(T) : \mathcal{L}_{\text{type}} \to \mathcal{L}_\mu$
- $\mathcal{L}_\mu$ closed under boolean operations

- XPath expressions $e_1, ..., e_n$ and schemas $T_1, ..., T_n$
- $\gamma^\bullet$ for comparing XPath expresssions from the same context

Many Decision Problems Can be Formulated

- XPath emptiness: $t_{\text{xpath}}(e_1, \gamma^\bullet \wedge t_{\text{schema}}(T_1))$
- XPath typing: $t_{\text{xpath}}(e_1, \gamma^\bullet \wedge t_{\text{schema}}(T_1)) \wedge \neg t_{\text{schema}}(T_2)$
  - if the formula is unsatisfiable then all nodes selected by e_1 under type constraint T_1 are included in the type T_2
- XPath containment:
  $t_{\text{xpath}}(e_1, \gamma^\bullet \wedge t_{\text{schema}}(T_1)) \wedge \neg t_{\text{xpath}}(e_2, \gamma^\bullet \wedge t_{\text{schema}}(T_2))$
- XPath equivalence, XPath overlap

Deciding $\mathcal{L}_\mu$ Satisfiability

- Does a formula $\psi \in \mathcal{L}_\mu$ admit a satisfying finite binary tree?

Principles

- Enumerate finite binary trees, look for a node on which ψ holds
- The truth status of a formula ψ can be determined from the status of a few of its subformulas
- The Fisher-Ladner Closure cl(ψ) $=$ $\{$ subformula of ψ where fixpoints are unwounded once $\}$
- We focus on a subset Lean(ψ) $\subseteq$ cl(ψ):
  - atomic propositions (alphabet symbols in ψ)
  - existential formulas

Example

Emptiness of XPath expression: `self::b/parent::a`

- $\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X$
- $\exp(\varphi) = b \vee \langle 2 \rangle \varphi$

$$\text{Lean}(\psi) = \left\{ \begin{array}{l} \langle 1 \rangle \top, \\ \langle \bar{1} \rangle \top, \\ \langle 2 \rangle \top, \\ \langle \bar{2} \rangle \top, \\ \sigma, \\ a, \\ b, \\ \langle 1 \rangle \varphi, \\ \langle 2 \rangle \varphi \end{array} \right\}$$

- The atomic proposition "σ" simulates an infinite alphabet ($\sigma \equiv \neg a \wedge \neg b$)

Nodes of the Searched Binary Tree

- The satisfiability-testing algorithm attempts to build a satisfying finite binary tree such that some node satisfies ψ

- A node is a ψ-type: a set $t \subseteq \text{Lean}(\psi)$ which satisfies constraints, e.g.:
  - modal consistency: $\forall \langle\alpha\rangle\,\varphi \in \text{Lean}(\psi), \langle\alpha\rangle\,\varphi \in t \Rightarrow \langle\alpha\rangle \top \in t$
  - tree node: $\left\langle\bar{1}\right\rangle \top \notin t \vee \left\langle\bar{2}\right\rangle \top \notin t$
  - labeled with exactly one atomic proposition $\sigma \in t$
- a ψ-type valuates any formula in $\text{cl}(\psi)$ via a relation $\dot{\in}$
  - for instance $\varphi_1 \wedge \varphi_2 \mathrel{\dot{\in}} t$ iff $\varphi_1 \mathrel{\dot{\in}} t$ and $\varphi_2 \mathrel{\dot{\in}} t$

Satisfiability-Testing Algorithm: Principles

Bottom-up Construction of a Tree of ψ-types

- A set T of ψ-types is repeatedly updated (least fixpoint computation)
  - Initially: $\emptyset$
  - Step 1 : all possible leaves are added
  - Step i : all possible parent nodes of current nodes are added

Termination

- If ψ is present in some root node, then ψ is satisfiable
- The algorithm returns a satisfying model as soon as it is found
- Otherwise, it terminates when no more node can be added
  - all roots of all buidable finite trees have been added

Example

$\text{Lean}(\psi) = \{\langle 1 \rangle \top,\ \langle \overline{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \overline{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$$\text{Lean}(\psi) = \left\{ \langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi \right\}$$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^0 = \emptyset$

$$\text{Lean}(\psi) = \left\{\langle 1\rangle \top,\ \left\langle \overline{1}\right\rangle \top,\ \langle 2\rangle \top,\ \left\langle \overline{2}\right\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\right\}$$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^1 = ?$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1 \rangle \top, \langle \bar{1} \rangle \top, \langle 2 \rangle \top, \langle \bar{2} \rangle \top, \sigma, a, b, \langle 1 \rangle \varphi, \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \quad \}$$

$$T^0 = \emptyset$$

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \text{exp}(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^2 = ?$

$T^1 = \{ \ \sigma \ \ \sigma \ \ \sigma \ \ a \ \ a \ \ a \ \ b \ \ b \ \ b \ \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $b\ \langle 2 \rangle\, \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{\ \sigma\quad \sigma\quad \sigma\quad a\quad a\quad a\quad b\quad b\quad b\ \}$

$T^0 = \emptyset$

$\mathrm{Lean}(\psi) = \{\langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $b \, \langle 2 \rangle \, \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{ \; \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \; \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1 \rangle \top, \; \langle \bar{1} \rangle \top, \; \langle 2 \rangle \top, \; \langle \bar{2} \rangle \top, \; \sigma, \; a, \; b, \; \langle 1 \rangle \varphi, \; \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $b \, \langle 2 \rangle \, \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{ \; \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \; \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top, \; \langle \bar{1} \rangle \top, \; \langle 2 \rangle \top, \; \langle \bar{2} \rangle \top, \; \sigma, \; a, \; b, \; \langle 1 \rangle \varphi, \; \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $b \, \langle 2 \rangle \varphi$ belong to T^2 ? Yes!

$T^2 = ?$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1 \rangle \top, \langle \bar{1} \rangle \top, \langle 2 \rangle \top, \langle \bar{2} \rangle \top, \sigma, a, b, \langle 1 \rangle \varphi, \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does b satisfy ψ ?

$T^2 = ?$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top, \ \langle \bar{1} \rangle \top, \ \langle 2 \rangle \top, \ \langle \bar{2} \rangle \top, \ \sigma, \ a, \ b, \ \langle 1 \rangle \varphi, \ \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X . b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^2 = ?$ Does b satisfy ψ ? No!

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi\}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $a \, \langle 1 \rangle \, \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{ \sigma \;\; \sigma \;\; \sigma \;\; a \;\; a \;\; a \;\; b \;\; b \;\; b \}$

$T^0 = \emptyset$

$\mathrm{Lean}(\psi) = \{ \langle 1 \rangle \top, \; \langle \bar{1} \rangle \top, \; \langle 2 \rangle \top, \; \langle \bar{2} \rangle \top, \; \sigma, \; a, \; b, \; \langle 1 \rangle \varphi, \; \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $a \, \langle 1 \rangle \, \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{ \; \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \; \}$

$T^0 = \emptyset$

$\mathrm{Lean}(\psi) = \{ \langle 1 \rangle \top, \; \langle \bar{1} \rangle \top, \; \langle 2 \rangle \top, \; \langle \bar{2} \rangle \top, \; \sigma, \; a, \; b, \; \langle 1 \rangle \varphi, \; \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X . b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does a $\langle 1 \rangle \varphi$ belong to T^2 ?

$T^2 = ?$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top,\ \langle \bar{1} \rangle \top,\ \langle 2 \rangle \top,\ \langle \bar{2} \rangle \top,\ \sigma,\ a,\ b,\ \langle 1 \rangle \varphi,\ \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

Does $a \, \langle 1 \rangle \, \varphi$ belong to T^2 ? Yes!

$T^2 = ?$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{ \langle 1 \rangle \top, \ \langle \bar{1} \rangle \top, \ \langle 2 \rangle \top, \ \langle \bar{2} \rangle \top, \ \sigma, \ a, \ b, \ \langle 1 \rangle \varphi, \ \langle 2 \rangle \varphi \}$

$\psi = a \wedge \langle 1 \rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2 \rangle X \equiv \exp(\varphi) = b \vee \langle 2 \rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^2 = ?$

Does a satisfy ψ?

$T^1 = \{ \ \sigma \ \ \sigma \ \ \sigma \ \ a \ \ a \ \ a \ \ b \ \ b \ \ b \ \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

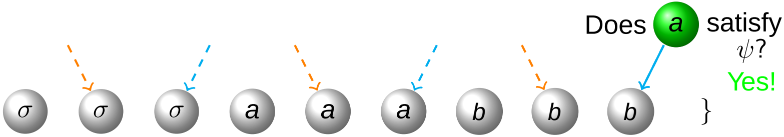
$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^2 = ?$

$T^1 = \{$



$\}$

$T^0 = \emptyset$

$\mathrm{Lean}(\psi) = \{\langle 1\rangle\top,\ \langle\bar{1}\rangle\top,\ \langle 2\rangle\top,\ \langle\bar{2}\rangle\top,\ \sigma,\ a,\ b,\ \langle 1\rangle\varphi,\ \langle 2\rangle\varphi\}$

$\psi = a \wedge \langle 1\rangle\varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle\varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$$\left|T^2\right| = 81$$

$$T^1 = \{\ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b\ \}$$

$$T^0 = \emptyset$$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$\longrightarrow$ return satisfiable!

$\left|T^2\right| = 81$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^3 = \ldots$

$\left|T^2\right| = 81$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Example

$T^n = T^{n-1}$ (fixpoint)

$T^3 = \ldots$

$\left|T^2\right| = 81$

$T^1 = \{ \sigma \quad \sigma \quad \sigma \quad a \quad a \quad a \quad b \quad b \quad b \quad \}$

$T^0 = \emptyset$

$\text{Lean}(\psi) = \{\langle 1\rangle \top,\ \langle \bar{1}\rangle \top,\ \langle 2\rangle \top,\ \langle \bar{2}\rangle \top,\ \sigma,\ a,\ b,\ \langle 1\rangle \varphi,\ \langle 2\rangle \varphi\}$

$\psi = a \wedge \langle 1\rangle \varphi$ with $\varphi = \mu X.b \vee \langle 2\rangle X \equiv \exp(\varphi) = b \vee \langle 2\rangle \varphi$

Emptiness check of XPath expression `self::b/parent::a`

Correctness & Complexity

Theorem

The satisfiability problem for a formula $\psi \in \mathcal{L}_\mu$ is decidable in time $2^{O(n)}$ where $n = |Lean(\psi)|$.

Theorem

For $e \in \mathcal{L}_{XPath}$ and a regular tree type expression T, the translations of e and T in $\mathcal{L}_\mu$ are linear in the size of e and T.

Corollary

XPath decision problems (e.g., typing, containment, emptiness, equivalence) in presence of schemas can be decided in time complexity $2^{O(n)}$.

Correctness & Complexity

Theorem

The satisfiability problem for a formula $\psi \in \mathcal{L}_\mu$ is decidable in time $2^{O(n)}$ where $n = |Lean(\psi)|$.

Theorem

For $e \in \mathcal{L}_{XPath}$ and a regular tree type expression T, the translations of e and T in $\mathcal{L}_\mu$ are linear in the size of e and T.

Corollary

XPath decision problems (e.g., typing, containment, emptiness, equivalence) in presence of schemas can be decided in time complexity $2^{O(n)}$.

Solver Implementation Techniques

Idea: Implicit Representation

- The T^is can be represented by boolean expressions
- Set-theoretic operations: composition of boolean expressions

- A set of ψ-types is encoded by a Binary Decision Diagram (BDD) [Bryant, 1986]

Critical Optimizations

- Conjunctive partitioning and early quantification (aims at composing smaller BDDs)
- Good initial order of BDD variables

Implementation

- System fully implemented (Java + Buddy C BDD library)
- Implementation available: `http://wam.inrialpes.fr/xml/`

Some Examples and Demo

- DTD of the W3C SMIL 1.0 recommendation

Question	Answer	Time (ms)
`/descendant::video` $\subseteq$ `/descendant::video[parent::seq]`?	no	125
`/descendant::audio[preceding-sibling::video]` $\neq \emptyset$?	yes	109
`child::switch[ancestor::head]` $\subseteq$ `descendant::switch`?	yes	105

- Demo

Main Contributions

A tree logic offering an interesting balance

- expressiveness: regular tree types + multi-directional navigation + finite recursion
- complexity for satisfiability: $2^{O(n)}$ where $n = |\text{Lean}(\psi)|$
- Compilation of main XML concepts: linear
- extensibility warranted (sublogic of the AFMC with converse)

A system for solving basic decision problems involving XPath queries and schemas

- The largest XPath fragment effectively treated so far for static analysis
- The system benefits from the boolean closure of the logic
- Efficient implementation in practice

Future Work

Extending the Tree Logic

- Decidable counting constraints
- Decidable data-value comparisons

Applications

- Static type-checking of XSLT, XQuery
- Optimization
- XML Security
- Checking integrity constaints in XML databases
- Query comparison in P2P networks

Thank you!

`http://wam.inrialpes.fr/xml/`

Bryant, R. E. (1986).
Graph-based algorithms for boolean function manipulation.
IEEE Transactions on Computers, 35(8):677–691.

Murata, M., Lee, D., Mani, M., and Kawaguchi, K. (2005).
Taxonomy of XML schema languages using formal language theory.
ACM Transactions on Internet Technology, 5(4):660–704.