Introduction to Tree Logics

Pierre Genevès
CNRS

(slides mostly based on the ones by W. Martens and T. Schwentick)

University of Grenoble Alpes, 2016–2017

Why Logic?

- Tree automaton $\leftrightsquigarrow$ algorithm
- Logical formula $\leftrightsquigarrow$ specification $\leftrightsquigarrow$ appropriate for queries
- Most expressive logics are tightly related to tree automata

Outline

1. Trees seen as logical structures
2. First-order logic over trees
3. Monadic second-order logic over trees

What is a Logic?

Mathematical logic

- A tool that allows reasoning (show *properties*) about *things* (e.g. arithmetic, set theory...)
- Before talking about a logic, one must define the *thing* we are talking about!
- The *thing* is usually a *structure* (unconstrained, word, tree...)

What is a Logic?

Mathematical logic

- A tool that allows reasoning (show *properties*) about *things* (e.g. arithmetic, set theory...)
- Before talking about a logic, one must define the *thing* we are talking about!
- The *thing* is usually a *structure* (unconstrained, word, tree...)

Recall that in logic:

- A *relational vocabulary* is a sequence of relation names $(R, ..., T)$ with associated arities $\text{arity}(R), ..., \text{arity}(T)$
- A *structure* $\mathcal{S}$ over $(R, ..., T)$ is a tuple

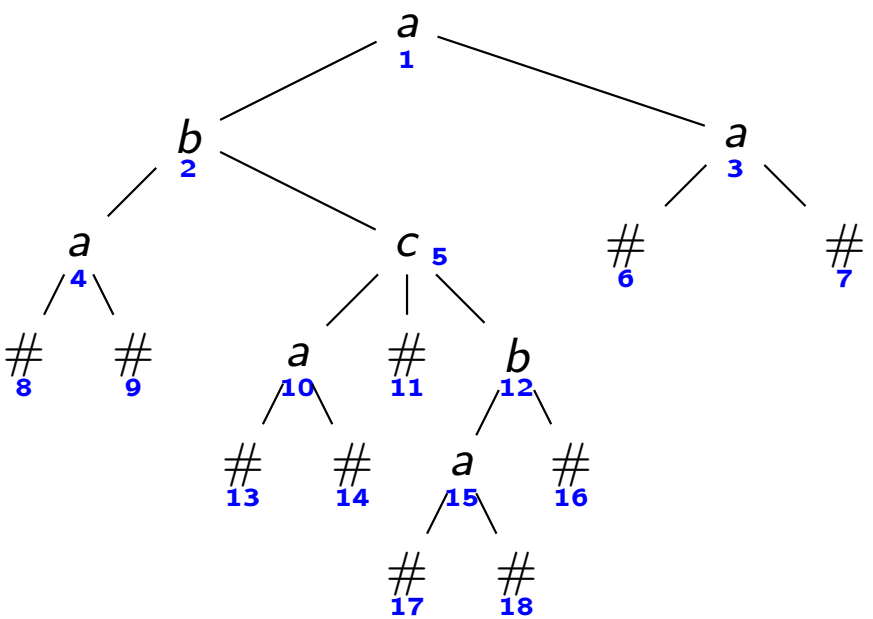
$$(D, R^{\mathcal{S}}, ..., T^{\mathcal{S}})$$

where D is a finite set, and $R^{\mathcal{S}} \subseteq D^{\text{arity}(R)}, ..., S^{\mathcal{S}} \subseteq D^{\text{arity}(S)}$

Trees as relational structures

A tree t over a **ranked alphabet** $\Sigma = \{a, ..., b\}$ naturally corresponds to a structure $\underline{t}$ over the vocabulary $V_\Sigma = (\texttt{child}, <, L_a, ..., L_b)$

Example



Structure

- $D^{\underline{t}} = \{1, 2, 3, ...\}$
- L_σ: unary relations
  - $L_a = \{1, 3, 4, 10, 15\}$
  - $L_b = \{2, 12\}$
  - ...
- `child`, $<$ (ordering) are binary relations
  - 1 $\texttt{child}^{\underline{t}}$ 2
  - 1 $\texttt{child}^{\underline{t}}$ 3
  - 2 $\texttt{child}^{\underline{t}}$ 4

    ...
  - $2 <^{\underline{t}} 3$
  - $4 <^{\underline{t}} 5$

    ...

Outline

1. Trees seen as relational structures
2. First-order logic over trees
3. Monadic second-order logic over trees

First-Order Logic over Trees

FO over the vocabulary $V_\Sigma = (\texttt{child}, <, L_a, ..., L_b)$:

$$\begin{aligned}\varphi ::= \quad & x = y \mid x \texttt{ child } y \mid x < y \mid L_a(x) \mid ... \mid L_b(y) \\ & \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\varphi\end{aligned}$$

with usual abbreviations $\varphi \vee \varphi, \varphi \rightarrow \varphi, \forall x\varphi, ...$

Example

All nodes labeled a in t have a child labeled b, iff:

$$\underline{t} \models \forall x(L_a(x) \rightarrow \exists y(L_b(y) \wedge x \texttt{ child } y))$$

First-Order Logic over Trees

FO over the vocabulary $V_\Sigma = (\texttt{child}, <, L_a, ..., L_b)$:

$$\begin{aligned}\varphi ::= \quad & x = y \mid x \texttt{ child } y \mid x < y \mid L_a(x) \mid ... \mid L_b(y) \\ & \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi\end{aligned}$$

with usual abbreviations $\varphi \vee \varphi, \varphi \rightarrow \varphi, \forall x \varphi, ...$

Notation/terminology

- If φ is a formula over V_Σ then

$$\text{Language}(\varphi) := \{t \mid t \text{ ranked tree } \Sigma \text{ such that } \underline{t} \models \varphi\}$$

- φ is *satisfiable* iff $\text{Language}(\varphi) \neq \emptyset$, *valid* iff $\text{Language}(\neg\varphi) = \emptyset$
- We note FO[$\texttt{child}, <$] as a shorthand for "FO over V_Σ"
- A tree language L is definable in FO[$\texttt{child}, <$] if there exists φ with $L = \text{Language}(\varphi)$

Applications to XML Trees

- A **closed** formula φ defines a tree type (Language(φ)), for example:

$$\forall x(L_a(x) \rightarrow \exists y(L_b(y) \wedge x \texttt{ child } y))$$

- A formula $\varphi(x, y)$ **with two free variables** defines a relation between two tree nodes: a (relative) query

Applications to XML Trees

- A **closed** formula φ defines a tree type (Language(φ)), for example:

$$\forall x(L_a(x) \rightarrow \exists y(L_b(y) \wedge x \texttt{ child } y))$$

- A formula $\varphi(x, y)$ **with two free variables** defines a relation between two tree nodes: a (relative) query

Exercise:

Is the following query definable in FO[child, <]?

- `child::a[parent::b]/child::c`
- `self::a/descendant::b`

Applications to XML Trees

- A **closed** formula φ defines a tree type (Language(φ)), for example:

$$\forall x(L_a(x) \rightarrow \exists y(L_b(y) \wedge x \texttt{ child } y))$$

- A formula $\varphi(x, y)$ **with two free variables** defines a relation between two tree nodes: a (relative) query

Exercise:

Is the following query definable in FO[child, <]?

- `child::a[parent::b]/child::c`
- `self::a/descendant::b`

Observation about FO Expressivity

Transitive Closure

- "x $\texttt{child}^+$ y" is not expressible in FO[$\texttt{child}, <$]

Observation about FO Expressivity

Transitive Closure

- "x $\texttt{child}^+$ y" is not expressible in FO[$\texttt{child}, <$]

$\rightarrow$ We can consider FO[descendant, $<$] instead ☺

Observation about FO Expressivity

Transitive Closure

- "x $\texttt{child}^+$ y" is not expressible in FO[$\texttt{child}$, <]

$\rightarrow$ We can consider FO[descendant, <] instead ☺

Example

- `descendant::p[not child::q]/child::*`
- $\exists z(x \text{ descendant } z \wedge L_p(z) \wedge \neg\exists w(z \text{ child } w \wedge L_q(w)) \wedge z \text{ child } y))$

Observation about FO Expressivity

Transitive Closure

- "x $\texttt{child}^+$ y" is not expressible in FO[$\texttt{child}, <$]

$\rightarrow$ We can consider FO[descendant, $<$] instead ☺

Example

- `descendant::p[not child::q]/child::*`
- $\exists z(x \text{ descendant } z \wedge L_p(z) \wedge \neg\exists w(z \text{ child } w \wedge L_q(w)) \wedge z \text{ child } y))$
- Here, z child w stands for:
  z descendant $w \wedge \neg\exists v(z \text{ descendant } v \wedge v \text{ descendant } y)$

Property

- Every XPath expression of the "CoreXPath" fragment is equivalent to a formula with 2 free variables in FO^2[descendant, $<$]
  [Marx, 2004, Genevès and Vion-Dury, 2004]

CoreXPath

query	::=		
		$/\,path$	absolute path
	|	*path*	relative path
	|	*query* | *query*	union
	|	*query* $\cap$ *query*	intersection
path	::=		
		path/*path*	path composition
	|	*path*[*qualifier*]	qualified path
	|	*axis*::*nodetest*	step
qualifier	::=		
		qualifier and *qualifier*	conjunction
	|	*qualifier* or *qualifier*	disjunction
	|	not(*qualifier*)	negation
	|	*path*	path
nodetest	::=		
		σ	node label
	|	$*$	any node label
axis	::=		tree navigation axis
		self | child | parent | descendant | preceding	
	|	descendant-or-self | ancestor | ancestor-or-self	
	|	following-sibling | preceding-sibling | following	

... what about the reciprocal property?

- Is any $FO^2[\text{descendant}, <]$ formula equivalent to a CoreXPath expression?

$\rightarrow$ A desirable property for a query language

  - if true, CoreXPath is expressively complete with respect to paths definable in FO:
  - "if a path is definable in FO, then there exists a CoreXPath expression that defines it".

... what about the reciprocal property?

- Is any $FO^2[\text{descendant}, <]$ formula equivalent to a CoreXPath expression?

$\rightarrow$ A desirable property for a query language

  - if true, CoreXPath is expressively complete with respect to paths definable in FO:
  - "if a path is definable in FO, then there exists a CoreXPath expression that defines it".

... No!

- CoreXPath is not complete with respect to $FO^2[\text{descendant}, <]$
- But a relatively simple extension of CoreXPath is sufficient for obtaining the reciprocal [Marx, 2004] ...

Example (CoreXPath extended)

$$e_1 \stackrel{\text{def}}{=} \texttt{self::*[}\underbrace{\texttt{(child::*[}F_1\texttt{])}^*}_{e_2}\texttt{/child::*[}F_2\texttt{]]}$$

- F_1 and F_2 are filtering expressions
- e_2 is not a CoreXPath expression, it defines the transitive-reflexive closure of the relation expressed by `child::*[`F_1`]`

Example (CoreXPath extended)

$$e_1 \stackrel{\text{def}}{=} \texttt{self::*[}\underbrace{\texttt{(child::*[}F_1\texttt{])}^*}_{e_2}\texttt{/child::*[}F_2\texttt{]]}$$

- F_1 and F_2 are filtering expressions
- e_2 is not a CoreXPath expression, it defines the transitive-reflexive closure of the relation expressed by `child::*[`F_1`]`
- e_2 is equivalent to:

$$x = y \vee (x \text{ descendant } y \wedge F_1(y) \wedge \forall z(x \text{ descendant } z \wedge z \text{ descendant } y \rightarrow F_1(z)))$$

- e_1 defines the set of paths (x, x) such that x has a descendant y for which F_2 is true and for all z strictly between x and y, F_1 is true.
- e_1 is not expressible in CoreXPath:
  - Any CoreXPath qualifier is equivalent to a FO formula with only 2 variables
  - It can be shown that the main qualifier of e_1 requires 3 variables

Example (CoreXPath extended)

$$e_1 \stackrel{\text{def}}{=} \texttt{self::*}[\underbrace{(\texttt{child::*}[F_1])^*}_{e_2}/\texttt{child::*}[F_2]]$$

- e_2 is called a conditional axis (different from $\texttt{descendant-or-self}=\texttt{child}^*$ since it depends on F_1)
- Conditional XPath is CoreXPath extended with all conditional axes in the 4 directions (`descendant`, `ancestor`, `following-sibling`, `preceding-sibling`)

Properties

- Conditional XPath is expressively complete with respect to FO[descendant, <] [Marx, 2004]
- Most CoreXPath properties (like linear-time evaluation) generalize to Conditional XPath

Is FO the Ideal Logic for XML?

Assessment for XML

- FO (and variants) are appropriate for characterizing the expressive power and (to some extent) the complexity of query languages...

Decidability and Complexity

- Over an unconstrained relational structure: FO is undecidable [Church, 1936, Turing, 1937]
- Over trees: FO is decidable but not elementary
- FO^2 (the fragment of FO restricted to only 2 distinct variables) : 2-NEXPTIME [Mortimer, 1975], NEXPTIME [Grädel et al., 1997]

But...

FO Expressivity is Limited

Limitation

- FO does not capture all regular tree languages
- Example:
  - The language L composed of all trees with an even number of a (recall tree automata!)
  - L is not definable in FO
    (exercise: intuitively, FO cannot "count")
- FO is not expressive enough for reasoning over all DTDs, XML Schemas, Relax NGs

Outline

1. Trees seen as relational structures
2. First-order logic over trees
3. Monadic second-order logic over trees

Monadic Second-Order Logic

MSO is the extension of FO with set variables X:

$$\varphi ::= x = y \mid x \texttt{ child } y \mid x < y \mid L_a(x) \mid \ldots \mid L_b(y)$$
$$\mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\varphi \mid X(x) \mid \exists X\varphi$$

with usual abbreviations $\varphi \vee \varphi, \varphi \rightarrow \varphi, \forall x\varphi, \forall X\varphi, \ldots$

Example: Boolean Expressions

$$\begin{aligned}\text{true expression} \equiv\ & \exists X\ .\ X(root) \wedge \forall x \\ & (L_1(x) \rightarrow X(x)) \wedge \\ & (L_0(x) \rightarrow \neg X(x)) \wedge \\ & ((L_\wedge(x) \wedge X(x)) \rightarrow (\forall y[x \texttt{ child } y \rightarrow X(y)])) \wedge \\ & ((L_\vee(x) \wedge X(x)) \rightarrow (\exists y[x \texttt{ child } y \wedge X(y)]))\end{aligned}$$

Monadic Second-Order Logic

MSO is the extension of FO with set variables X:

$$\varphi ::= x = y \mid x \texttt{ child } y \mid x < y \mid L_a(x) \mid \ldots \mid L_b(y)$$
$$\mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\varphi \mid X(x) \mid \exists X\varphi$$

with usual abbreviations $\varphi \vee \varphi, \varphi \to \varphi, \forall x\varphi, \forall X\varphi, \ldots$

Notation and Terminology

- If φ is a closed formula over V_Σ then

$$\text{Language}(\varphi) := \{t \mid t \text{ ranked tree over } \Sigma \text{ such that } \underline{t} \models \varphi\}$$

- A tree language is definable in MSO if there exists a MSO formula φ with $S = \text{Language}(\varphi)$

MSO Variants: WSkS

- Recall the definition of our relational vocabulary
- We define the logical data model (ordered trees: **structure** + **order**)
- Instead of `child` and $<$, one can use the relations ch_i modeling the parent to i-th child relationship:

$$\mathrm{x}\ \mathrm{ch}_i\ \mathrm{y} \text{ iff y is the } i\text{-th child of x}$$

- Then, instead of MSO[`child`,$<$], we can consider MSO$[\mathrm{ch}_1, \mathrm{ch}_2, \ldots, \mathrm{ch}_k]$

$\rightarrow$ Exercise: write some formulas

- This is WSkS: Weak monadic Second-order logic of k Successors

$\rightarrow$ WSkS is not so weak... ☺

WS2S: A Variant of Special Interest

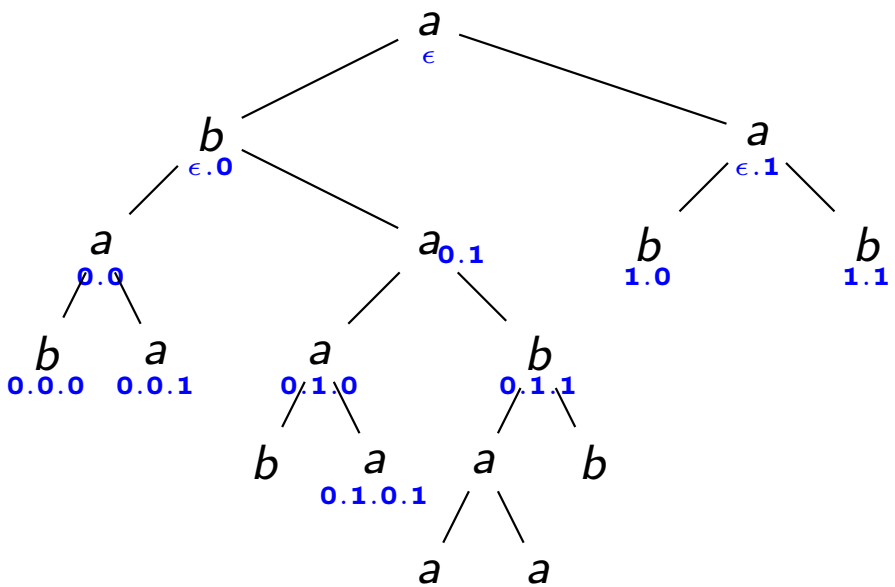
- Zoom on WSkS $= \mathrm{MSO}[\mathrm{ch}_1, \mathrm{ch}_2, ..., \mathrm{ch}_k]$ where $k = 2$
- Recall the encoding of unranked trees as binary trees

$\rightarrow$ Considering ch_1 and ch_2 is sufficient wlog!

$\rightarrow$ We can even call them first successor and second successor or "fc" (for first-child) and "ns" (next-sibling), explicitly referring to the unranked case

- Anyway, this is WS2S.

The logical tree structure: the WS2S perspective



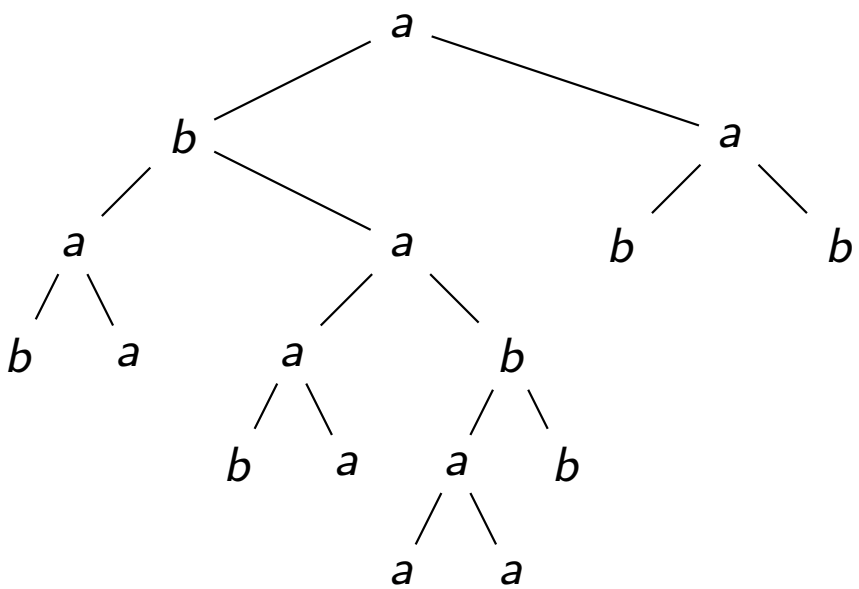
$$\mathcal{L}_{\mathsf{WS2S}} \ni \varphi ::= x = y \mid \mathsf{firstchild}(x, y) \mid \mathsf{nextsibling}(x, y) \mid L_a(x) \mid \ldots \mid L_b(y)$$
$$\mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\varphi \mid X(x) \mid \exists X\varphi$$

Logical Variants

- Now you know how to build and clearly define plenty of (different?) logical formalisms
- Most importantly: you understand why it is important to define and mention the **relational vocabulary** over the **logical structure**
- FO[`child`, <], FO[descendant, <], FO[fc, ns], FO[fc^+, ns^+], FO^2[fc^+, ns^+]
  - Exercise(s): what are the connections between those logics?
  - Btw: what was the problem again with FO[fc,ns]?
- MSO[`child`, <], MSO[fc, ns]=WS2S
  - Exercise: finding MSO formulas not in any FO: e.g. true boolean expressions, other ones?

Exercise with MSO

Construct a WS2S formula φ which describes the set of trees with an even number of a nodes, e.g.:



$$\mathcal{L}_{\mathsf{WS2S}} \ni \varphi ::= x = y \mid \mathsf{firstchild}(x, y) \mid \mathsf{nextsibling}(x, y) \mid L_a(x) \mid \ldots \mid L_b(y)$$
$$\mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x\varphi \mid X(x) \mid \exists X\varphi$$

Expressivity and Decidability of MSO

Theorem [Thatcher and Wright, 1968, Doner, 1970]

- MSO exactly captures regular tree languages
- The satisfiability-problem for a MSO formula is decidable

Principle: the logic-tree automata connection

- Show that MSO is as expressive as tree automata
- We associate a tree automaton to each formula, and reciprocally
- For the direction logic$\longrightarrow$ automata, induction:
  - basic tree automata
  - $\varphi \vee \varphi'$: composition of tree automata (recall: closure of regular tree languages)
- Satisfiability of the logical formula: emptiness check of the final automaton

Expressivity and Decidability of MSO

Theorem [Thatcher and Wright, 1968, Doner, 1970]

- MSO exactly captures regular tree languages
- The satisfiability-problem for a MSO formula is decidable

Principle: the logic-tree automata connection

- Show that MSO is as expressive as tree automata
- We associate a tree automaton to each formula, and reciprocally
- For the direction logic$\longrightarrow$ automata, induction:
  - basic tree automata
  - $\varphi \vee \varphi'$: composition of tree automata (recall: closure of regular tree languages)
- Satisfiability of the logical formula: emptiness check of the final automaton

$\rightarrow$ We can compute the tree automaton (the algorithm) that corresponds to a given logical formula (specification)!

Complexity

What is the size of the tree automaton corresponding to φ?

(a) smaller than φ
(b) same size as φ
(c) larger then φ
(d) HUGE

Complexity

What is the size of the tree automaton corresponding to φ?

(a) smaller than φ
(b) same size as φ
(c) larger then φ
(d) HUGE

Hints

- $\neg\varphi$: construction of the complement of the automaton for φ (determinisation)
- $\forall x\varphi \equiv \neg\exists x\neg\varphi$
- Each quantifier alternation (e.g. $\exists\forall\exists$) introduces a determinisation...

Complexity

What is the size of the tree automaton corresponding to φ?

(a) smaller than φ
(b) same size as φ
(c) larger then φ
(d) HUGE

Hints

- $\neg\varphi$: construction of the complement of the automaton for φ (determinisation)
- $\forall x\varphi \equiv \neg\exists x\neg\varphi$
- Each quantifier alternation (e.g. $\exists\forall\exists$) introduces a determinisation...

Complexity

What is the size of the tree automaton corresponding to φ?

(a) smaller than φ
(b) same size as φ
(c) larger then φ
(d) HUGE

Hints

- $\neg\varphi$: construction of the complement of the automaton for φ (determinisation)
- $\forall x\varphi \equiv \neg\exists x\neg\varphi$
- Each quantifier alternation (e.g. $\exists\forall\exists$) introduces a determinisation...

Complexity

Theorem

- The number of states of the automaton associated with a formula with n quantifier alternations is in the worst case at least:

$$2^{2^{\cdots^{2^{c \cdot n}}}}$$

  where c is a constant. [Stockmeyer and Meyer, 1973]

- That's a lower bound!
- MSO satisfiability is decidable in non-elementary time (i.e. not bounded by any stack of exponentials)
- Nevertheless, there exists an implementation of this decision procedure: MONA [Klarlund and Møller, 2001]

Complexity

Theorem

- The number of states of the automaton associated with a formula with n quantifier alternations is in the worst case at least:

$$2^{2^{\cdots^{2^{c \cdot n}}}}$$

  where c is a constant. [Stockmeyer and Meyer, 1973]
- That's a lower bound!
- MSO satisfiability is decidable in non-elementary time (i.e. not bounded by any stack of exponentials)
- Nevertheless, there exists an implementation of this decision procedure: MONA [Klarlund and Møller, 2001]

Complexity

Theorem

- The number of states of the automaton associated with a formula with n quantifier alternations is in the worst case at least:

$$2^{2^{\cdots^{2^{c \cdot n}}}}$$

  where c is a constant. [Stockmeyer and Meyer, 1973]
- That's a lower bound!
- MSO satisfiability is decidable in non-elementary time (i.e. not bounded by any stack of exponentials)
- Nevertheless, there exists an implementation of this decision procedure: MONA [Klarlund and Møller, 2001]

Complexity

Theorem

- The number of states of the automaton associated with a formula with n quantifier alternations is in the worst case at least:

$$2^{2^{\cdots^{2^{c \cdot n}}}}$$

  where c is a constant. [Stockmeyer and Meyer, 1973]
- That's a lower bound!
- MSO satisfiability is decidable in non-elementary time (i.e. not bounded by any stack of exponentials)
- Nevertheless, there exists an implementation of this decision procedure: MONA [Klarlund and Møller, 2001]

Summary: 2 Yardsticks for Measuring Expressivity

FO

- ✓ FO[fc^+, ns^+] is expressively equivalent to Conditional XPath
- × No FO variant can capture regular tree languages entirely

Summary: 2 Yardsticks for Measuring Expressivity

FO

- ✓ FO[fc^+, ns^+] is expressively equivalent to Conditional XPath
- × No FO variant can capture regular tree languages entirely

MSO

- ✓ One of the most expressive (yet decidable) logic known today
- × You pay for what you get:
  1. Satisfiability is non-elementary
  2. Unless $P{=}NP$, there does not exist any elementary f (i.e. bounded by a stack of exponential) such that MSO formulas can be **evaluated** in time $f(\text{size}(\varphi)) \cdot p(\text{size}(t))$ with p polynomial
     [Frick and Grohe, 2002]
- × This makes MSO almost useless as a query language :(

Summary: 2 Yardsticks for Measuring Expressivity

FO

- ✓ FO[fc^+, ns^+] is expressively equivalent to Conditional XPath
- × No FO variant can capture regular tree languages entirely

MSO

- ✓ One of the most expressive (yet decidable) logic known today
- × You pay for what you get:
  1. Satisfiability is non-elementary
  2. Unless $P{=}NP$, there does not exist any elementary f (i.e. bounded by a stack of exponential) such that MSO formulas can be **evaluated** in time $f(\text{size}(\varphi)) \cdot p(\text{size}(t))$ with p polynomial [Frick and Grohe, 2002]
- × This makes MSO almost useless as a query language :(

→ There exists tree logics as expressive as MSO and for which both evaluation (model-checking) and satisfiability-checking can be implemented more efficiently (less complex): μ-calculus for trees [Genevès et al., 2007]


Church, A. (1936).
A note on the entscheidungsproblem.
Journal of Symbolic Logic, 1:101–102.


Doner, J. (1970).
Tree acceptors and some of their applications.
Journal of Computer and System Sciences, 4:406–451.


Frick, M. and Grohe, M. (2002).
The complexity of first-order and monadic second-order logic revisited.
In *LICS*, pages 215–224.


Genevès, P., Layaïda, N., and Schmitt, A. (2007).
Efficient static analysis of XML paths and types.
In *PLDI '07: Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 342–351, New York, NY, USA. ACM Press.


Genevès, P. and Vion-Dury, J.-Y. (2004).
XPath formal semantics and beyond: A Coq-based approach.
In *TPHOLs '04: Emerging Trends Proceedings of the 17th International Conference on Theorem Proving in Higher Order*

Logics, pages 181–198, Salt Lake City, Utah, United States. University Of Utah.


Grädel, E., Kolaitis, P. G., and Vardi, M. Y. (1997).
On the decision problem for two-variable first-order logic.
Bulletin of Symbolic Logic, 3(1):53–69.


Klarlund, N. and Møller, A. (2001).
MONA Version 1.4 User Manual.
BRICS Notes Series NS-01-1, Department of Computer Science, University of Aarhus.


Marx, M. (2004).
Conditional XPath, the first order complete XPath dialect.
In *PODS '04: Proceedings of the twenty-third ACM Symposium on Principles of Database Systems*, pages 13–22, New York, NY, USA. ACM Press.


Mortimer, M. (1975).
On language with two variables.
Zeit für Math. Logik und Grund. der Math., 21:135–140.


Stockmeyer, L. and Meyer, A. (1973).
Word problems requiring exponential time.

In *STOC '73: Proceedings of the 5th ACM symposium on Theory of computing*, pages 1–9, New York, NY, USA. ACM Press.

Thatcher, J. W. and Wright, J. B. (1968).
Generalized finite automata theory with an application to a decision problem of second-order logic.
Mathematical Systems Theory, 2(1):57–81.

Turing, A. (1937).
On computable numbers, with an application to the entscheidungsproblem.
Proc. London Math. Soc. Correction in vol. 43, pp. 544-546., 42:230–265.