A Tree Logic...

... and an Application for the Analysis of Cascading Style Sheets

Pierre Genevès

pierre.geneves@inria.fr

EPFL, Oct. 23^{th} 2012

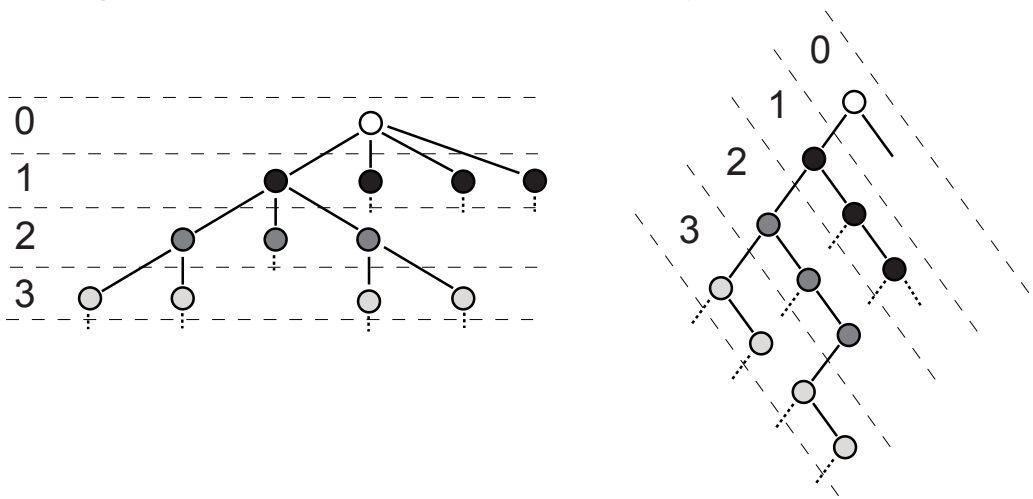
Outline

1. Insights on the $\mathcal{L}_\mu$ Tree Logic
2. Brief Overview of some Applications
3. Zoom on the Analysis of CSS

Data Model for the Logic

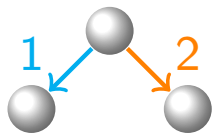
Trees: the logic was originally designed for XML trees

- Specifically: finite binary labeled trees
- They model finite ordered unranked labeled trees wlog
- Bijective encoding of unranked trees as binary trees:



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

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



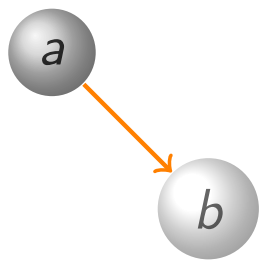
$\mathcal{L}_\mu \ni \varphi, \psi$	$::=$	formula
	$\top$	true
$\mid$	$p \mid \neg p$	atomic prop (negated)
$\mid$	$n \mid \neg n$	nominal (negated)
$\mid$	$\varphi \vee \psi \mid \varphi \wedge \psi$	disjunction (conjunction)
$\mid$	$\langle \alpha \rangle \varphi \mid \neg \langle \alpha \rangle \top$	existential (negated)
$\mid$	$\mu X.\varphi$	unary fixpoint (finite recursion)
$\mid$	$\mu \overline{X_i.\varphi_i}$ in ψ	n-ary fixpoint

Sample Formula and Satisfying Tree

a

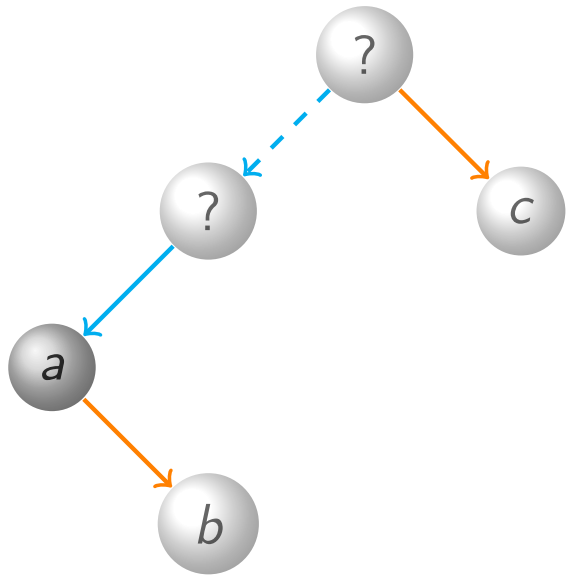
Sample Formula and Satisfying Tree

$a \wedge \langle 2 \rangle\, b$



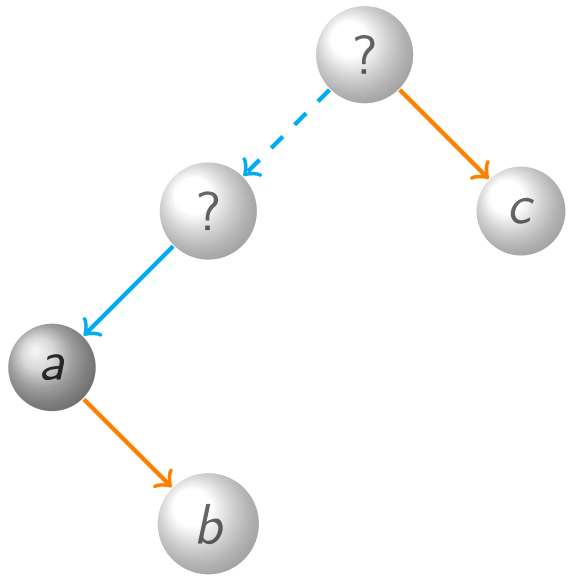
Sample Formula and Satisfying Tree

$$a \wedge \langle 2 \rangle\, b \wedge \mu X.\, \langle 2 \rangle\, c \vee \langle \overline{1} \rangle\, X$$



Sample Formula and Satisfying Tree

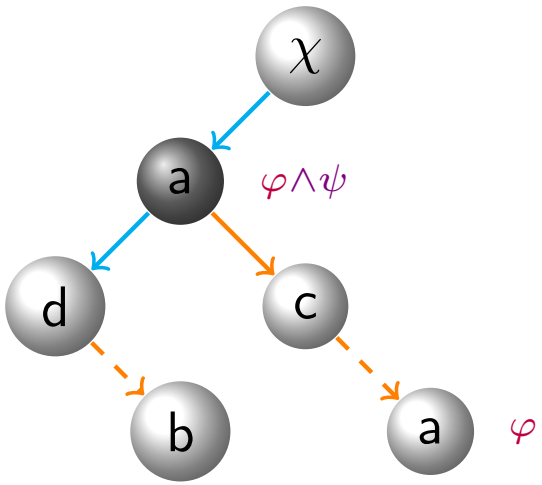
$$a \wedge \langle 2 \rangle\, b \wedge \mu X.\, \langle 2 \rangle\, c \vee \langle \overline{1} \rangle\, X$$



- Semantics: models of φ are finite trees for which φ holds at some node

✓ Nice balance between succinctness and expressive power: XPath, CSS selectors, and XML types can be translated into the logic, linearly

Example: Translation of an XPath Expression into $\mathcal{L}_\mu$



- Formula holds at selected nodes
- $\mu Z.\varphi$: finite recursion
- Converse programs are crucial
- More generally, we have a compiler:
  - $t_{\text{xpath}}(e, \chi) : \mathcal{L}_{\text{XPath}} \times \mathcal{L}_\mu \to \mathcal{L}_\mu$
  - χ is the latest navigation step
  - initially, $\chi = \neg \langle \overline{1} \rangle \top \wedge \neg \langle \overline{2} \rangle \top$ for absolute expressions

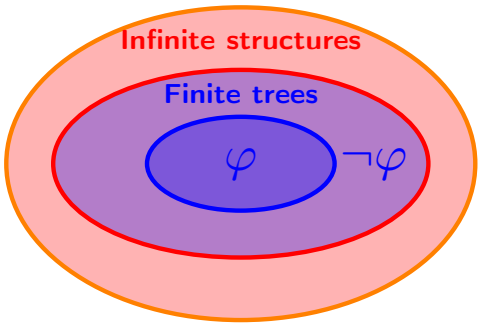
Translated query: child::a [child::b]

$$\underbrace{a \wedge (\mu Z. \langle \overline{1} \rangle \chi \vee \langle \overline{2} \rangle Z)}_{\varphi} \quad \wedge \quad \underbrace{\langle 1 \rangle \mu Y. b \vee \langle 2 \rangle Y}_{\psi}$$

$\mathcal{L}_\mu$ Closure under Negation

Cycle-freeness: A key property

- If both a program and its converse occur between a $\mu X.$ binder and X, formula has a cycle, e.g.: $\mu X.\, \langle \alpha \rangle\, X \vee \langle \overline{\alpha} \rangle\, X$
- Otherwise the formula is cycle-free
- in practice, most (all?) formulas are cycle-free (e.g. XPath translations are always cycle-free)



- Cycle-freeness of $\mathcal{L}_\mu$ implies closure under negation
  - The negation of finite recursion is finite recursion (see paper)
  - $\neg\varphi$ is easily (linearly) expressible in $\mathcal{L}_\mu$ for all $\varphi \in \mathcal{L}_\mu$
- Crucial for BC: implication (subtyping, containment tests...)
- Crucial for implementation

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

Is a formula $\psi \in \mathcal{L}_\mu$ satisfiable?

- Given ψ, determine whether there exists a finite tree that satisfies ψ
- Validity: test $\neg\psi$

Principles: Automatic Theorem Proving

- Search for a proof tree
- Build the proof bottom up:

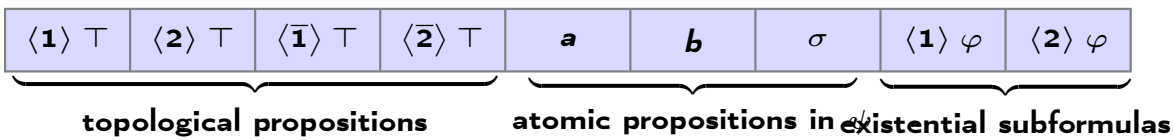
  "*if* ψ *holds then it is necessarily somewhere up*"

Search Space Optimization

Idea: Truth Status is Inductive

- The truth status of ψ can be expressed as a function of its subformulas
- For boolean connectives, it can be deduced (truth tables)
- Only base subformulas really matter: Lean(ψ)

Lean(ψ) :



A Tree Node: Truth Assignment of Lean(ψ) Formulas

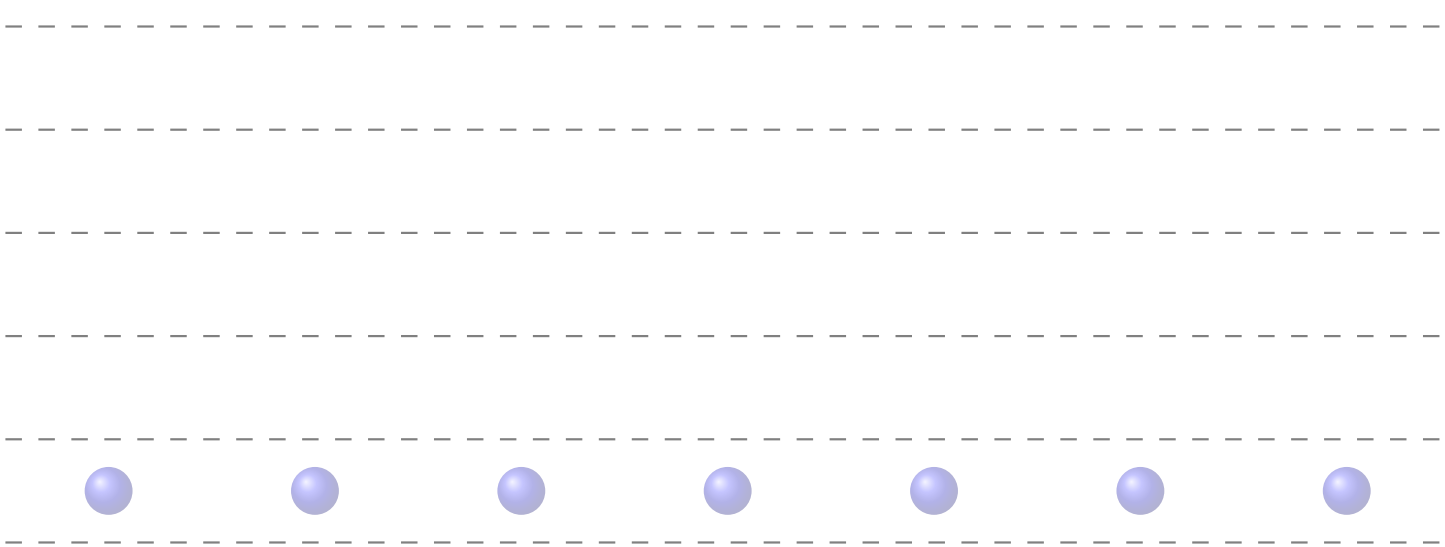
- With some additional constraints, e.g. $\neg \langle \bar{1} \rangle \top \vee \neg \langle \bar{2} \rangle \top$

Satisfiability-Testing Algorithm: Principles

Bottom-up construction of proof tree

- A set of nodes is repeatedly updated (fixpoint computation)

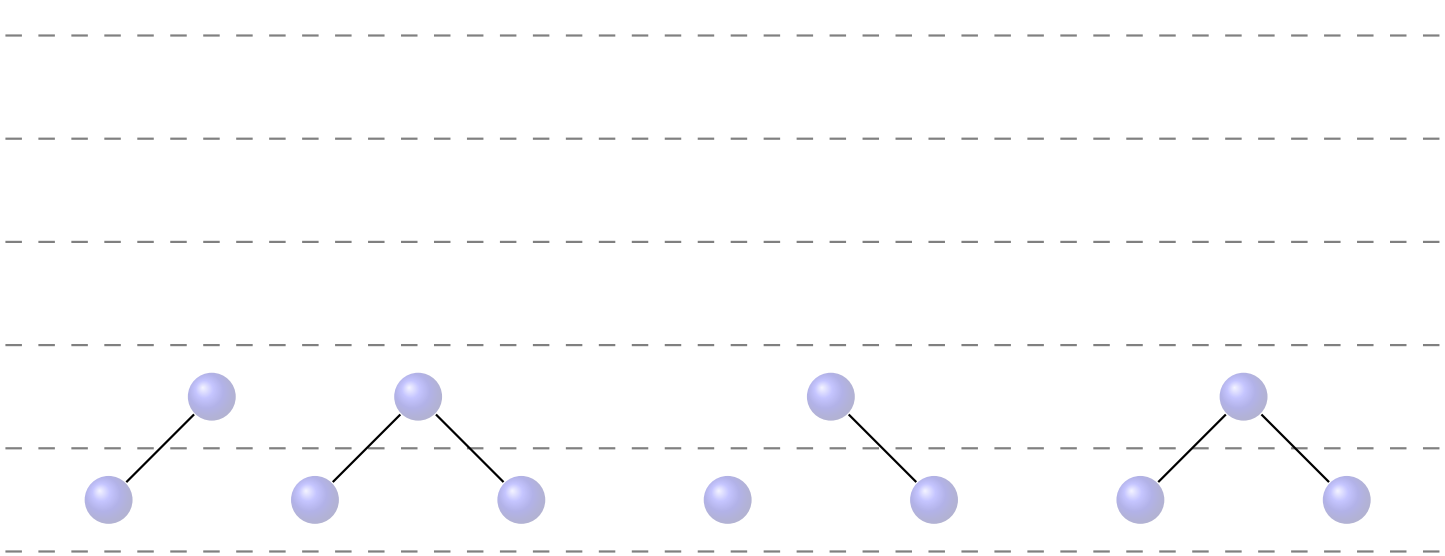
Satisfiability-Testing Algorithm: Principles



Bottom-up construction of proof tree

- Step 1: all possible leaves are added

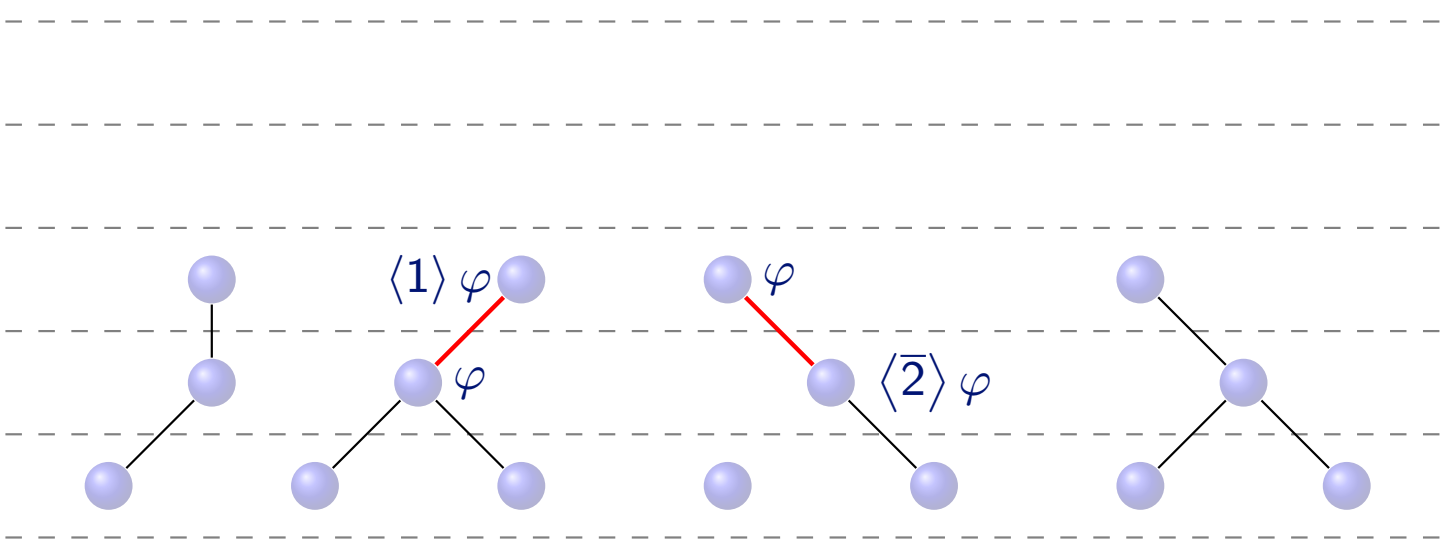
Satisfiability-Testing Algorithm: Principles



Bottom-up construction of proof tree

- Step $i > 1$: all possible parents of previous nodes are added

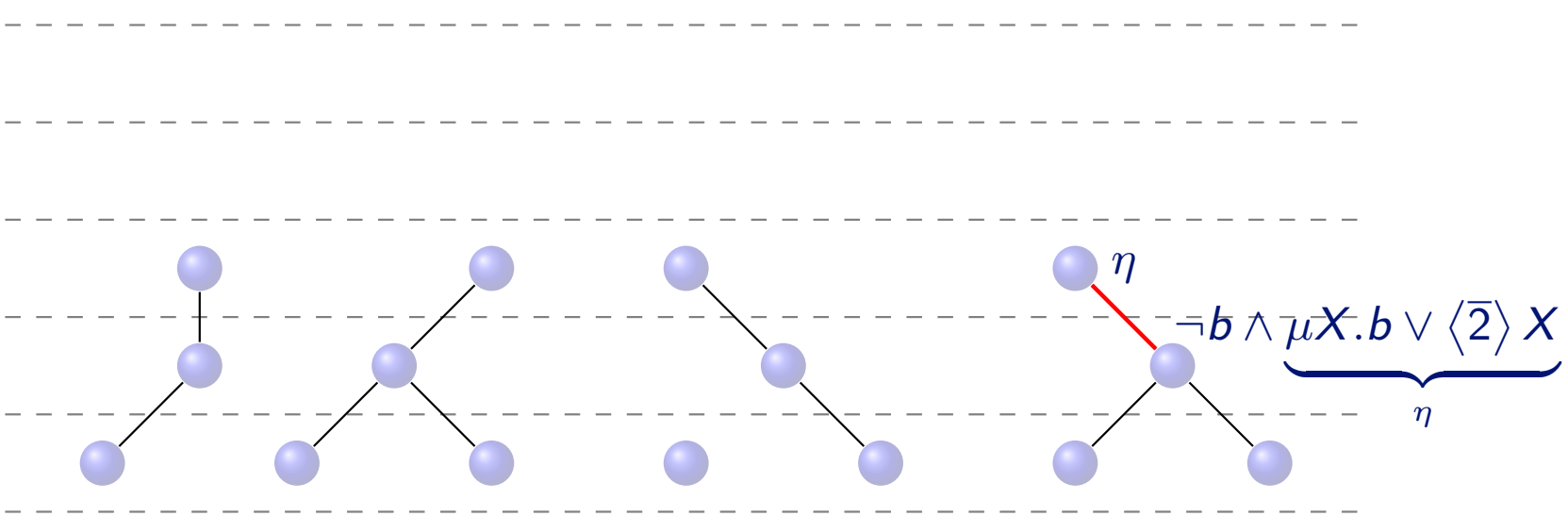
Satisfiability-Testing Algorithm: Principles



Compatibility relation between nodes

- Nodes from previous step are proof support:
  $\langle\alpha\rangle\,\varphi$ is added if φ holds in some node added at previous step

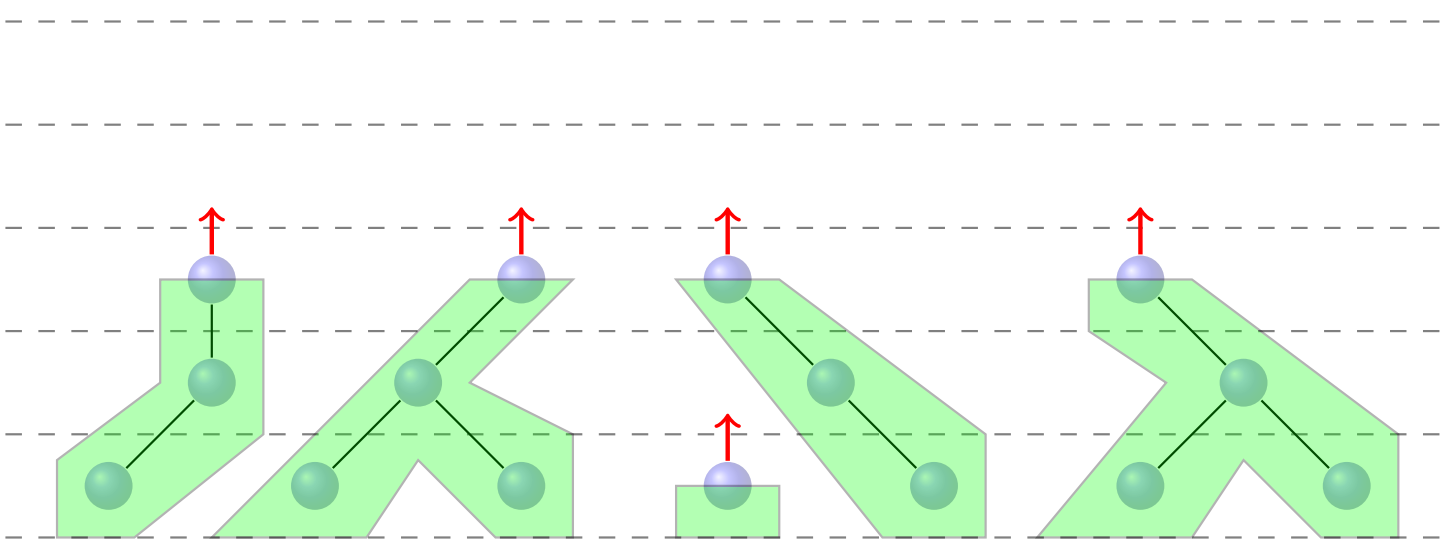
Satisfiability-Testing Algorithm: Principles



Compatibility relation between nodes

- Nodes from previous step are proof support:
  $\langle \alpha \rangle \varphi$ is added if φ holds in some node added at previous step

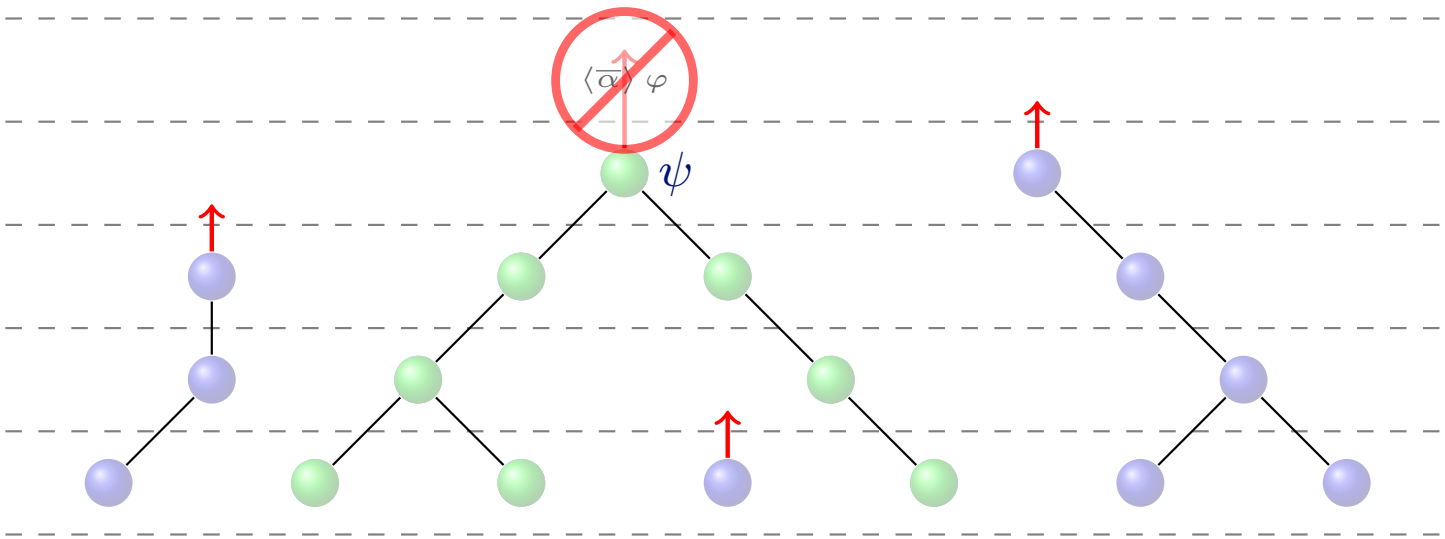
Satisfiability-Testing Algorithm: Principles



Progressive bottom-up reasoning (partial satisfiability)

- $\langle \overline{\alpha} \rangle \varphi$ are left unproved until a parent is connected

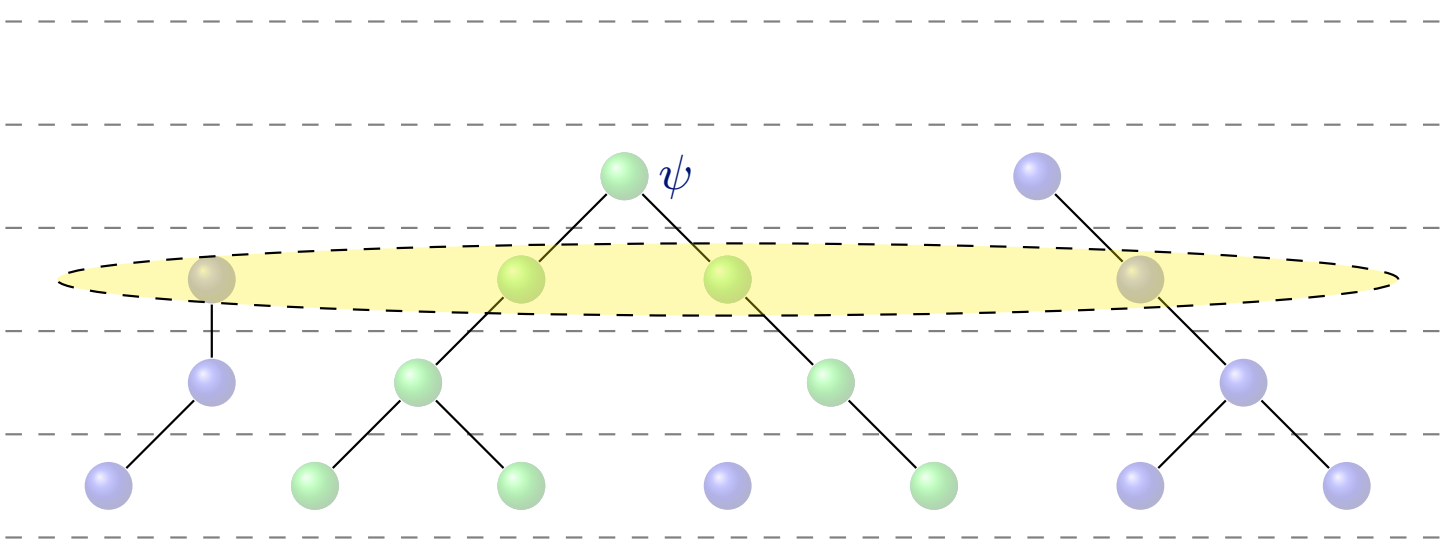
Satisfiability-Testing Algorithm: Principles



Termination

- If ψ is present in some root node, then ψ is satisfiable
- Otherwise, the algorithm terminates when no more nodes can be added

Satisfiability-Testing Algorithm: Principles



Implementation techniques

- Crucial optimization: symbolic representation

Correctness & Complexity

Theorem

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

System fully implemented

- decision procedure
- compilers (XPath, DTD, XML Schema, CSS selectors, ...)

Overview of Experiments

DTD	Symbols	Binary type variables
SMIL 1.0	19	11
XHTML 1.0 Strict	77	325

Table: Types used in experiments.

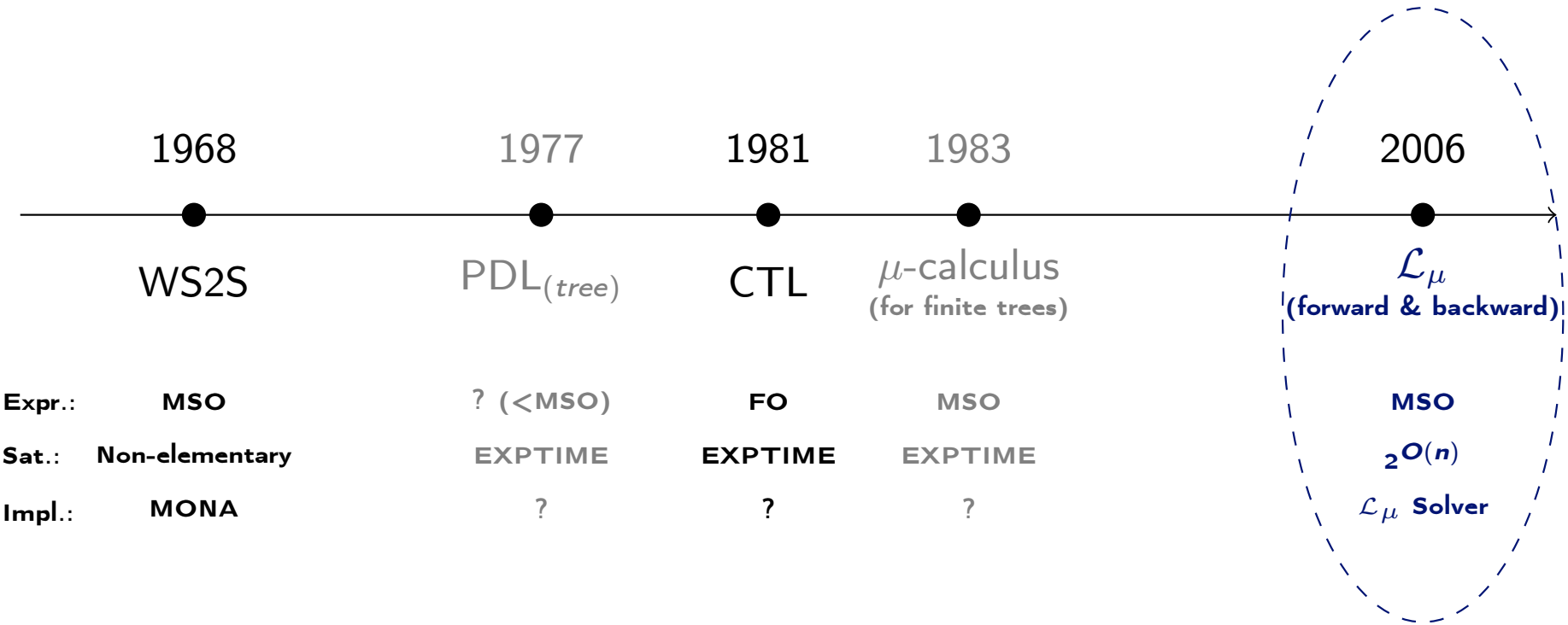
XPath decision problem	XML type	Time (ms)
$e_1 \subseteq e_2$ and $e_2 \not\subseteq e_1$	none	353
$e_4 \subseteq e_3$ and $e_4 \subseteq e_3$	none	45
$e_6 \subseteq e_5$ and $e_5 \not\subseteq e_6$	none	41
e_7 is satisfiable	SMIL 1.0	157
e_8 is satisfiable	XHTML 1.0	2630
$e_9 \subseteq (e_{10} \cup e_{11} \cup e_{12})$	XHTML 1.0	2872

Table: Some decision problems and corresponding results.

For the last test, size of the Lean is 550. The search space is $2^{550} \approx 10^{165}$... more than the square number of atoms in the universe 10^{80}

Tree Logics: an Overview

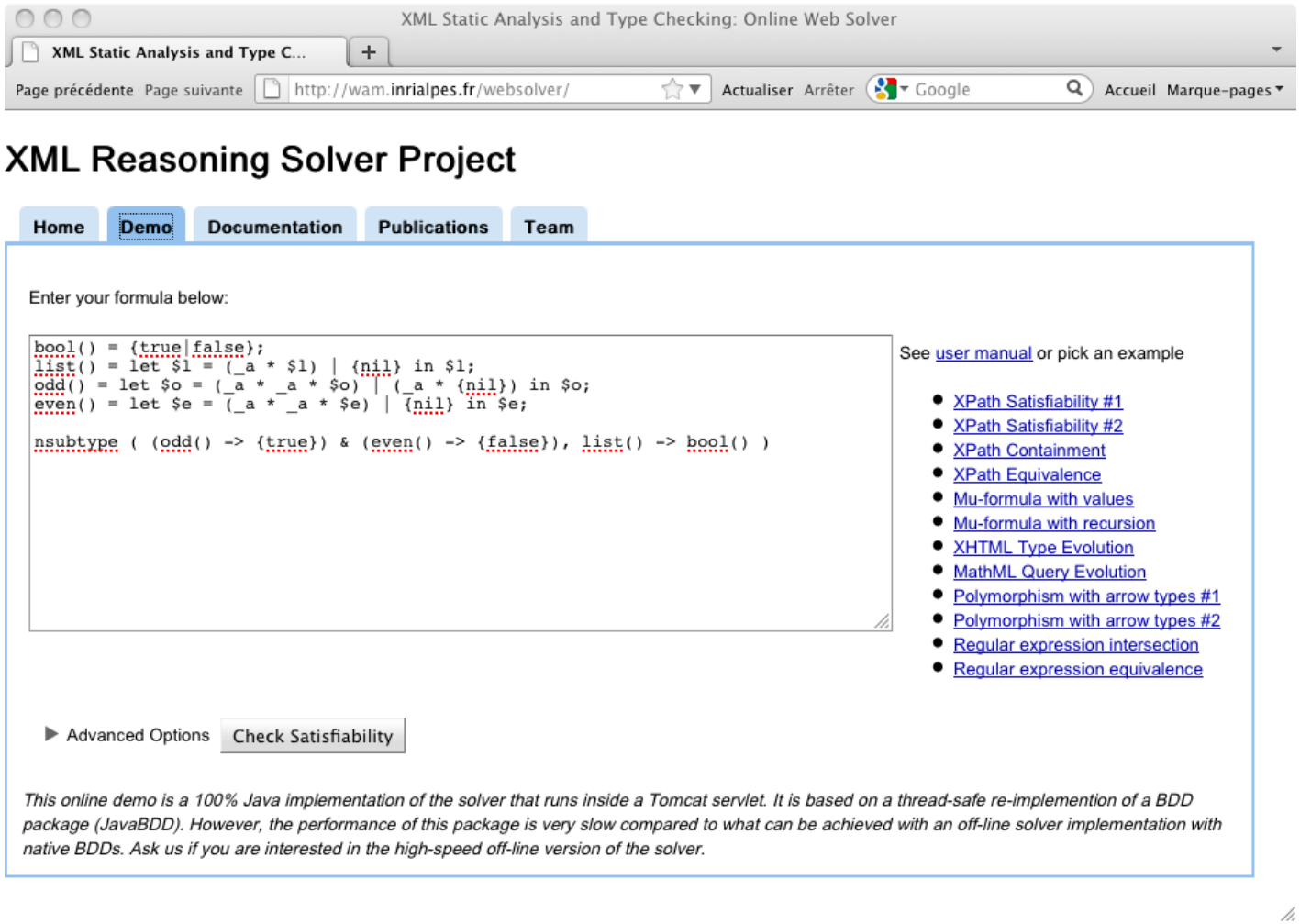
- On the theoretical side: $\mathcal{L}_\mu$ offers an interesting expressivity, succinctness, optimal complexity bound



On the practical side:

- except (hyperexponential) MONA, this is the only one implementation of a satisfiability solver for such an expressive logic
- It can be useful for graphs too: the sublogic with no backward modalities has the finite tree model property

Try it online*: http://wam.inrialpes.fr/websolver



* or offline if performance is critical: the offline version is much faster (native BDD library, further optimizations like compression of symbols)

Some Applications

- Containment for XML queries [PLDI'07, ICDE'10, IJCAI'11]
  - → checking equivalence of monadic queries q_1 and q_2:
    $\forall t, \forall n \in t, \quad q_1(t,n) \stackrel{?}{=} q_2(t,n)$
- Dead code analysis for XQuery [ICSE'10, ICSE'11]
- Impact of schema evolution [ICFP'09, TOIT'11]
  - → Schema S evolves into S': impact on a query written against S?
- Deciding subtyping for rich type algebras [ICFP'11]
  - → Intersection, negation, function, and polymorphic types
  - → A purely logical resolution for deciding subtyping with polymorphism
  - → A time complexity bound of $2^{\mathcal{O}(|\tau_1|+|\tau_2|)}$ for checking $\tau_1 \leq \tau_2$
- Containment for SPARQL queries (polyadic, graphs) [AAAI'12, IJCAR'12]
- CSS Analysis [WWW'12]
- ...

Some Applications

- Containment for XML queries [PLDI'07, ICDE'10, IJCAI'11]
  - → checking equivalence of monadic queries q_1 and q_2:
    $\forall t, \forall n \in t, \quad q_1(t, n) \stackrel{?}{=} q_2(t, n)$
- Dead code analysis for XQuery [ICSE'10, ICSE'11]
- Impact of schema evolution [ICFP'09, TOIT'11]
  - → Schema S evolves into S': impact on a query written against S?
- Deciding subtyping for rich type algebras [ICFP'11]
  - → Intersection, negation, function, and polymorphic types
  - → A purely logical resolution for deciding subtyping with polymorphism
  - → A time complexity bound of $2^{\mathcal{O}(|\tau_1|+|\tau_2|)}$ for checking $\tau_1 \leq \tau_2$
- Containment for SPARQL queries (polyadic, graphs) [AAAI'12, IJCAR'12]
- CSS Analysis [WWW'12]
- ...