# Extended Tree Automata Models for the Verification of Infinite State Systems

Florent Jacquemard

INRIA Saclay & LSV (UMR CNRS/ENS Cachan)

florent.jacquemard@inria.fr
http://www.lsv.ens-cachan.fr/~jacquema

# Executive Summary

- several models extending tree automata
  - extension with global/local constraints
  - extension with auxiliary memory
  - different kinds of trees (ranked / unranked)
  - modulo equational theories
- application to different verification problems

# Concurrent readers/writers

Example from [Clavel et al. 2007 LNCS 4350]

1. $\mathsf{state}(0, 0) \to \mathsf{state}(0, s(0))$
2. $\mathsf{state}(r, 0) \to \mathsf{state}(s(r), 0)$
3. $\mathsf{state}(r, s(w)) \to \mathsf{state}(r, w)$
4. $\mathsf{state}(s(r), w) \to \mathsf{state}(r, w)$

(1) writers can access the file if nobody else is accessing it

(2) readers can access the file if no writer is accessing it

(3,4) readers and writers can leave the file at any time

Properties expected:

- mutual exclusion between readers and writers
- mutual exclusion between writers

## ⟨Concurrent readers/writers⟩

- state$(x_R, x_W)$ reflects the state of a file, currently accessed by $x_R$ readers and $x_W$ writers
- equation: applied in direction left-to-right $=$ rewrite rules
- Réécriture: formalisme de calcul symbolique par remplacement de sous termes défini par une ensemble fini d'équations orientées.

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

initial configuration:

$\mathsf{state}(0,0)$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

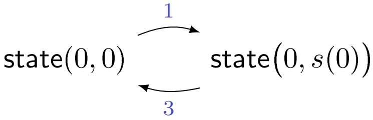
reachable configurations:

$\mathsf{state}(0,0)$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

reachable configurations:



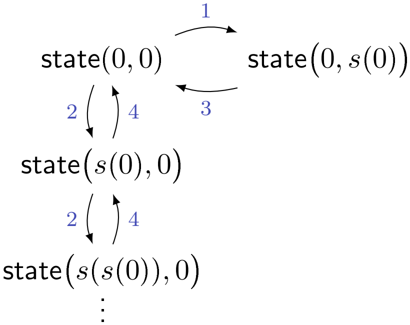
$\mathsf{state}(0,0)$

$\mathsf{state}\big(0,s(0)\big)$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

reachable configurations:



$\mathsf{state}(0,0)$
$\mathsf{state}\big(0,s(0)\big)$
1
3
2
4
$\mathsf{state}\big(s(0),0\big)$
2
4
$\mathsf{state}\big(s(s(0)),0\big)$
$\vdots$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \rightarrow \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \rightarrow \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \rightarrow \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \rightarrow \mathsf{state}(r,w)$

reachable configurations:

$\mathsf{state}(0,0)$

tree automaton:

$$\begin{aligned} 0 &\rightarrow q_0 \\ \mathsf{state}(q_0,q_0) &\rightarrow q \end{aligned}$$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

reachable configurations:

$$\begin{tikzcd}
\mathsf{state}(0,0) \arrow[r, bend left, "1"] & \mathsf{state}\big(0,s(0)\big) \arrow[l, bend left, "3"]
\end{tikzcd}$$

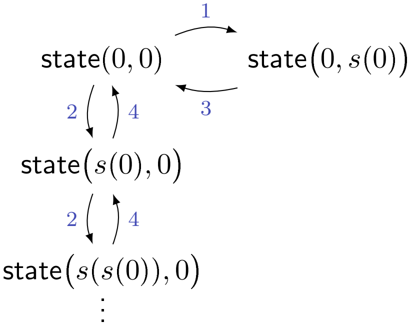
tree automaton:

$$\begin{aligned}
0 &\to q_0 \\
\mathsf{state}(q_0,q_0) &\to q \\
s(q_0) &\to q_1 \\
\mathsf{state}(q_0,q_1) &\to q
\end{aligned}$$

# Concurrent readers/writers: reachable configurations

1. $\mathsf{state}(0,0) \to \mathsf{state}(0,s(0))$
2. $\mathsf{state}(r,0) \to \mathsf{state}(s(r),0)$
3. $\mathsf{state}(r,s(w)) \to \mathsf{state}(r,w)$
4. $\mathsf{state}(s(r),w) \to \mathsf{state}(r,w)$

reachable configurations:



tree automaton:

$$
\begin{aligned}
0 &\to q_0\\
\mathsf{state}(q_0,q_0) &\to q\\
s(q_0) &\to q_1\\
\mathsf{state}(q_0,q_1) &\to q\\
\mathsf{state}(q_1,q_0) &\to q\\
s(q_1) &\to q_2\\
\mathsf{state}(q_2,q_0) &\to q\\
s(q_2) &\to q_2
\end{aligned}
$$

System Timbuk [Genet Tong 2004 JAR].

Automated construction, guess the *acceleration* $s(q_2) \to q_2$

# ⟨Concurrent readers/writers: reachable configurations⟩

- reachable states characterized by a tree automaton:
- set of productions rules = recursive definitions
- every states in red = variable representing set of terms
- if $q_0$ contains terms $t_1$ and $t_0$, then $q$ contains state$(t_1, t_2)$
- least solution: the variable in lhs is the smallest set containing union of sets defined in rhs

# Concurrent readers/writers: verification

Properties expected:

1. mutual exclusion between readers and writers
   excluded pattern: $\text{state}\big(s(x), s(y)\big)$
2. mutual exclusion between writers
   excluded pattern: $\text{state}\big(x, s(s(y))\big)$

Set of excluded configurations = union of

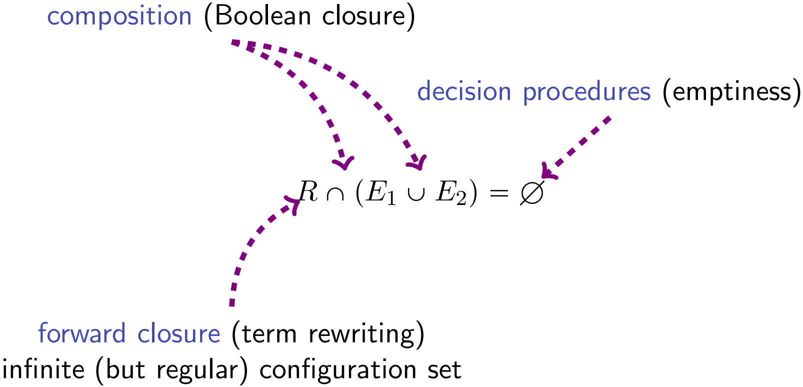
$$E_1 = \big\{\text{state}\big((q_1 \mid q_2), (q_1 \mid q_2)\big)\big\}$$

$$E_2 = \big\{\text{state}\big((q_0 \mid q_1 \mid q_2), q_2\big)\big\}$$

with $0 \rightarrow q_0$, $s(q_0) \rightarrow q_1$, $s(q_1) \rightarrow q_2$, $s(q_2) \rightarrow q_2$.

Verification: The intersection between the set of reachable configurations and excluded configurations is empty.

# Regular Model Checking



composition (Boolean closure)

decision procedures (emptiness)

$$R \cap (E_1 \cup E_2) = \emptyset$$

forward closure (term rewriting)
infinite (but regular) configuration set

# Limitation: Non-Regular Configuration Set

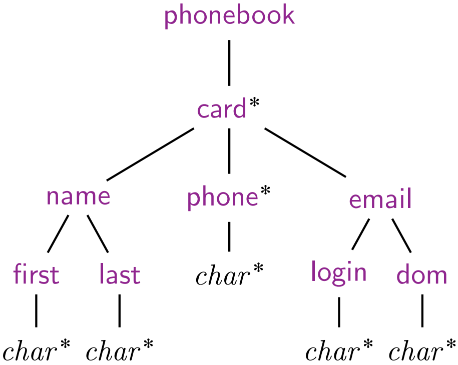
two files, configurations of the form: state$(x_1, y_1, x_2, y_2)$

- both files have the same number of readers

$$\mathsf{state}(x, y_1, x, y_2)$$

This set cannot be represented by tree automata
(pumping lemma)

# Type Definition for XML Data

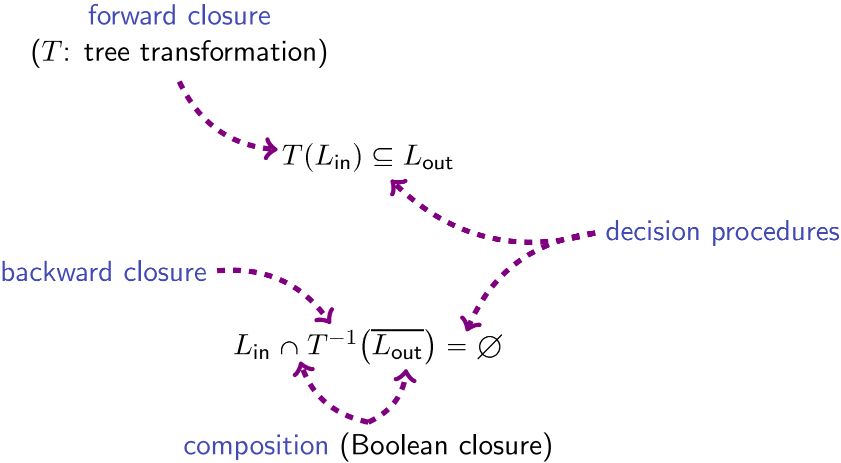


Defines an unranked ordered tree automata language.

Tree automata capture all type formalisms in use for XML data.

# Static Typechecking

[Milo Suciu Vianu 2003 JCSS]



forward closure
($T$: tree transformation)

$$T(L_{\mathsf{in}}) \subseteq L_{\mathsf{out}}$$

decision procedures

backward closure

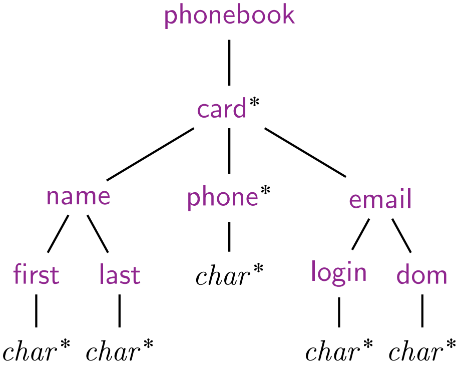
$$L_{\mathsf{in}} \cap T^{-1}\big(\overline{L_{\mathsf{out}}}\big) = \varnothing$$

composition (Boolean closure)

# ⟨Static Typechecking⟩

- verification that a program defining tree transformations always converts valid input data into valid output data,
- input and output types are defined by tree automata
- reduction to inclusion decision if the image of $L_{\mathsf{in}}$ by $T$ can be characterized by a tree automaton
- if this is not the case, a second option is to consider the inverse image of the complement of $L_{\mathsf{out}}$. For instance, in the above article (where $T$ is expressed in a fragment of XSLT) the inverse image is a tree automata language, and type checking is done by testing that the intersection with the input type is empty.

# Limitation: XML Integrity Constraints



- email is a key (ID)

Cannot be expressed with unranked tree automata

# Overcoming the Limitations of Tree Automata

find extensions of standard tree automata
preserving the good properties (as much as possible)

- closure under Boolean operations
- decision procedures (in particular emptiness)
- effective forward or backward closure by transformations

several models

- extension with global/local constraints
- extension with auxiliary memory
- different kinds of trees (ranked / unranked)
- modulo equational theories

motivated by different applications in verification

# Extended Models: Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints

FO Th. Proving

Tree automata with global constraints

XML integrity constraints

Horn Clauses with equality

dynamic properties: forward/backward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

Unranked tree rewriting

XML updates

XML r/w ACP

# Ranked Tree Automata: Definition

A (ranked) tree automaton is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

- $\Sigma$ is a ranked signature,
- $Q$ is a finite set of states,
- $F \subseteq Q$ is the subset of final states,
- $\Delta$ is a set of transitions of the form $f(q_1, \ldots, q_n) \to q$.

$$\begin{array}{rcl} \Sigma & = & \{0:0, s:1, \mathsf{state}:2\} \\ Q & = & \{q_0, q_1, q_2, q\} \\ F & = & \{q\} \end{array}$$

$$\Delta = \left\{ \begin{array}{rcl} 0 & \to & q_0 \\ \mathsf{state}(q_0, q_0) & \to & q \\ s(q_0) & \to & q_1 \\ \mathsf{state}(q_0, q_1) & \to & q \\ \mathsf{state}(q_1, q_0) & \to & q \\ s(q_1) & \to & q_2 \\ \mathsf{state}(q_2, q_0) & \to & q \\ s(q_2) & \to & q_2 \end{array} \right\}$$

# Ranked Tree Automata: Definition

A (ranked) tree automaton is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

$\Sigma$ is a ranked signature,

$Q$ is a finite set of states,

$F \subseteq Q$ is the subset of final states,

$\Delta$ is a set of transitions of the form $f(q_1, \ldots, q_n) \to q$.

## Regular languages

$$\begin{array}{rcl}
\mathcal{L}(\mathcal{A}, q) & = & \left\{ a \in \Sigma_0 \;\middle|\; a \to q \in \Delta \right\} \\
& \cup & \left\{ f(t_1, \ldots, t_n) \;\middle|\; \begin{array}{l} f \in \Sigma_n \\ t_1 \in \mathcal{L}(\mathcal{A}, q_1), \ldots, t_n \in \mathcal{L}(\mathcal{A}, q_n) \\ f(q_1, \ldots, q_n) \to q \in \Delta \end{array} \right\}
\end{array}$$

$$\mathcal{L}(\mathcal{A}) \quad = \quad \bigcup_{q \in F} \mathcal{L}(\mathcal{A}, q)$$

# Ranked Tree Automata: Main Properties

- Boolean closure of regular languages
- membership $t \in \mathcal{L}(\mathcal{A})$ is decidable in PTIME
- emptiness $\mathcal{L}(\mathcal{A}) = \varnothing$ is decidable in linear time
- finiteness is decidable in PTIME
- emptiness of intersection $\mathcal{L}(\mathcal{A}_1) \cap \ldots \cap \mathcal{L}(\mathcal{A}_n) = \varnothing$ is EXPTIME-complete
- universality $\mathcal{L}(\mathcal{A}) = \mathcal{T}(\Sigma)$ is EXPTIME-complete
- inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ is EXPTIME-complete
- equivalent to
  - parse trees of a context-free grammar
  - well-formed terms over a sorted signature
  - regular tree grammars
  - sentences of monadic second-order logic of the tree

# Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints

FO Th. Proving

Tree automata with global constraints

XML integrity constraints

Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

Unranked tree rewriting

XML updates

XML r/w ACP

# Tree Automata with Local Constraints (Siblings)

- both files have the same number of readers

$$\mathsf{state}(x, y_1, x, y_2)$$

$$\begin{array}{rcl}
0 & \rightarrow & q_0 \\
\mathsf{state}(q_0, q_0, q_0, q_0) & \rightarrow & q \\
s(q_0) & \rightarrow & q_1 \\
\mathsf{state}(q_0, q_1, q_0, q_1) & \rightarrow & q
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{state}(q_1, q_0, q_1, q_0) & \rightarrow & q \\
s(q_1) & \rightarrow & q_2 \\
\mathsf{state}(q_2, q_0, q_2, q_0) & \xrightarrow{1=3} & q \\
s(q_2) & \rightarrow & q_2
\end{array}$$

Tree automata with sibling = and ≠ constraints
[Bogaert Tison 1992 STACS]

# Tree Automata with Local Constraints

- both files have the same number of readers

$$\mathsf{pair}\big(\mathsf{state}(x, y_1), \mathsf{state}(x, y_2)\big)$$

$$\begin{array}{rcl} 0 & \rightarrow & q_0 \\ \mathsf{state}(q_0, q_0) & \rightarrow & p_1 \mid p_2 \\ s(q_0) & \rightarrow & q_1 \\ \mathsf{state}(q_0, q_1) & \rightarrow & p_1 \mid p_2 \end{array} \qquad \begin{array}{rcl} \mathsf{state}(q_1, q_0) & \rightarrow & p_1 \mid p_2 \\ s(q_1) & \rightarrow & q_2 \\ \mathsf{state}(q_2, q_0) & \rightarrow & p_1 \mid p_2 \\ s(q_2) & \rightarrow & q_2 \\ \mathsf{pair}(p_1, p_2) & \xrightarrow{1.1=2.1} & q \end{array}$$

Tree automata with $=$ and $\neq$ (path) constraints

[Mongy 1981 PhD], [Caron 1993 PhD]

# Tree Automata with Local Constraints: Definition

A (ranked) tree automaton with local constraints is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

- $\Sigma$ is a ranked signature,
- $Q$ is a finite set of states,
- $F$ $\subseteq Q$ is the subset of final states,
- $\Delta$ is a set of transitions of the form $f(q_1, \ldots, q_n) \xrightarrow{c} q$. where $c$ is a conjunction of equalities $p = p'$ and disequalities $p \neq p'$, for paths $p$, $p'$.

## Languages

$$
\begin{aligned}
\mathcal{L}(\mathcal{A}, q) &= \left\{ a \in \Sigma_0 \;\middle|\; a \to q \in \Delta \right\} \\
&\cup \left\{ f(t_1, \ldots, t_n) \;\middle|\; \begin{array}{l} f \in \Sigma_n \\ t_1 \in \mathcal{L}(\mathcal{A}, q_1), \ldots, t_n \in \mathcal{L}(\mathcal{A}, q_n) \\ f(q_1, \ldots, q_n) \xrightarrow{c} q \in \Delta \\ f(t_1, \ldots, t_n) \models c \end{array} \right\} \\
\mathcal{L}(\mathcal{A}) &= \bigcup_{q \in F} \mathcal{L}(\mathcal{A}, q)
\end{aligned}
$$

# Tree Automata with Local Constraints: Emptiness Decision

- emptiness is undecidable in general [Mongy 1981 PhD], [Bogaert 1990 PhD], for cousins: [Tommasi 1992 MT]
- emptiness EXPTIME-complete for sibling constraints [Bogaert Tison 1992 STACS]
- emptiness is decidable for deterministic and complete reduction automata (RA): ordering $<$ on states, for all $f(q_1, \ldots, q_n) \xrightarrow{c} q$, $q$ upper bound of $\{q_1, \ldots, q_n\}$ [Dauchet et al 1995 JSC]
- emptiness is undecidable for reduction automata

  J Rusinowitch Vigneron 2006 IJCAR
- emptiness is in EXPTIME with restriction to $\neq$ $O(|Q|.|\Delta|^{P(\mathcal{A})})$

  Comon-Lundh J 2003 IC
- emptiness decidable when arbitrary $\neq$, no overlap between $=$ [Godoy et al 2010 STOC]

## ⟨Tree Automata with Local Constraints: Emptiness Decision⟩

- RA: strict upper bound if c contains an equality constraint
- the number of equalities tested along a computation path in a run of $\mathcal{A}$ is bounded (by the number of states of $\mathcal{A}$)
- $P(\mathcal{A})$ is a polynomial in the size of the constrains of $\mathcal{A}$

# Tree Automata with Local Constraints: Regularity

regularity is undecidable

- for RA (reduction of universality)
- for $\mathrm{TA}_{\neq}$ (local $\neq$ constraints only) (id)
- for $\mathrm{TA}_{=}$ (local $=$ constraints only) (reduction of emptiness)

# First Order Theorem Proving

Term Rewriting System (TRS) $\mathcal{R}$: finite set of rewrite rules $\ell \to r$.

rewrite relation $\xrightarrow[\mathcal{R}]{*}$: smallest relation containing $\mathcal{R}$ closed under application of substitution and contexts.

- FO encompassment theory, predicate $E_t(s)$ if $s$ embeds $t$: compilation into deterministic reduction automata [Dauchet et al 1995 JSC]
- FO of $\xrightarrow[\mathcal{R}]{*}$ decidable for $\mathcal{R}$ ground, or $\ell$, $r$ linear and don't share variable for all $\ell \to r$ [Dauchet Tison 1990 LICS]
- FO of $\xleftrightarrow[\mathcal{R}_1]{*}$,. . . ,$\xleftrightarrow[\mathcal{R}_n]{*}$ decidable with same assumptions on $\mathcal{R}_1$,. . . ,$\mathcal{R}_n$, application to model-checking problem of $A\pi\mathcal{L}$ (a spatial equational logic for the applied $\pi$-calculus) J Lozes Treinen Villard 2011 TOSCA

# Inductive Theorem Proving

automatic proof of FO conjectures in Herbrand structures

*ex:* $x + 0 = x$ inductive th. of $\{0 + x = x, s(x) + y = s(x + y)\}$.

- inductionless induction / proof by consistency [Huet Hullot 1982 JCSS], [Jouannaud Kounalis 1986 LICS], [Kapur Musser 1987 AIJ].

  inductive completion [Fribourg 1989 JSC]:
  condition for *consistency* is ground reducibility
- g.r. is EXPTIME-complete Comon-Lundh J 2003 IC
  based on emptiness decision for TA with local $\neq$ constraints

## ⟨Inductive Theorem Proving⟩

- Herbrand structure: quotient of $\mathcal{T}(\Sigma)$
- inductive th. $=$ FO sentences : for all Herbrand structure, Horn clauses : smallest Herbrand structure

# Inductive Theorem Proving

- implicit induction: automatic computation of induction schemes
  - cover-sets [Kapur Zhang 1995 RRL]
  - test-sets [Bouhoula Rusinowitch 1995 JAR]
  - normal-form (standard) tree automata [Bouhoula Jouannaud 2001 IC] Horn clauses specifications with equational left-linear constructor relations

    $$s(p(x)) \to x,\ p(s(x)) \to x$$

  - NF constrained TA Bouhoula J 2008 IJCAR, Bouhoula J 2007 FCS-ARSPA, Bouhoula J 2011 JAL specifications with constrained constructor relations (axioms for complex data structures).

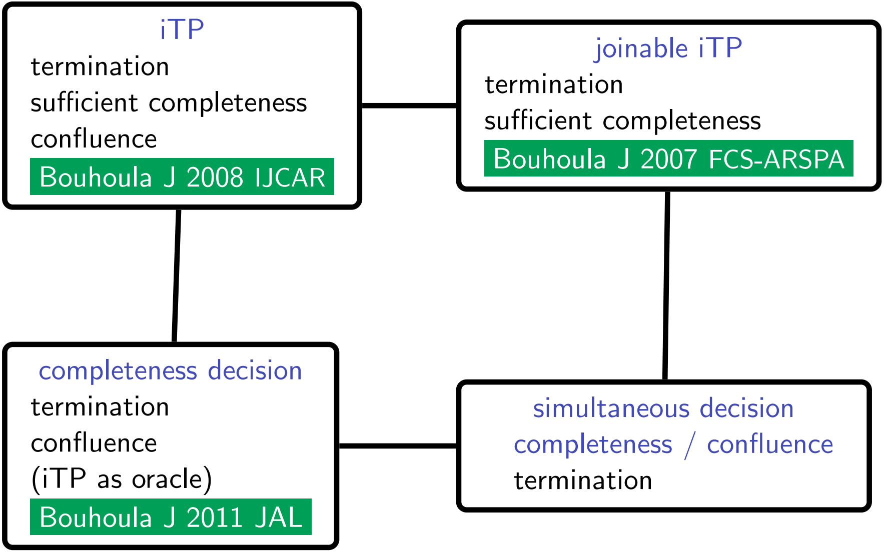
    $$\begin{array}{rcll} \mathsf{cons}(x, \mathsf{cons}(x, y)) & \to & \mathsf{cons}(x, y) & \\ \mathsf{cons}(x_1, \mathsf{cons}(x_2, y)) & \to & \mathsf{cons}(x_2, \mathsf{cons}(x_1, y)) & \parallel x_1 > x_2 \end{array}$$

    constrained TA are also decision tool (inconsistency detection)

## ⟨Inductive Theorem Proving⟩

- procedure sound and refutationally complete when R is sufficiently complete and the constructor subsystem RC is terminating
- refutationally complete: any conjecture that is not valid in the initial model will be disproved
- without the above hypotheses, it still remains sound and refutationally complete for conjectures, where all the variables are constrained to belong to the language of NF
- if R is strongly complete (a stronger condition for sufficient completeness) and ground confluent, then when the procedure fails, it follows that the conjecture is not an inductive theorem

# Automata Based Inductive Theorem Proving



iTP
termination
sufficient completeness
confluence
Bouhoula J 2008 IJCAR

joinable iTP
termination
sufficient completeness
Bouhoula J 2007 FCS-ARSPA

completeness decision
termination
confluence
(iTP as oracle)
Bouhoula J 2011 JAL

simultaneous decision
completeness / confluence
termination

# Outline

static properties: composition and decision (constraint solving)

- Tree automata with local constraints — FO Th. Proving
- Tree automata with global constraints — XML integrity constraints
- Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

- Ranked tree rewriting
- Rewriting strategies
- Unranked tree rewriting — XML updates, XML r/w ACP

# Unranked Ordered Trees & Hedges

$\Sigma$ unranked alphabet

hedge = finite sequence of unranked trees (possibly $\varepsilon$)
unranked tree = variable
$a$(hedge) with $a \in \Sigma$

## ⟨Unranked Ordered Trees & Hedges⟩

- the term 'hedge' was introduced by Courcelle
- $\varepsilon$ is the empty hedge

# Unranked Tree Automata: Definition

A hedge automaton (HA [Murata 2000]) is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

- $\Sigma$ is an (unranked) alphabet,
- $Q$ is a finite set of states,
- $F \subset Q$ is the subset of final states,
- $\Delta$ is a set of transitions of the form $f(L) \to q$ where $L$ is a regular word language over $Q^*$.

## Regular languages

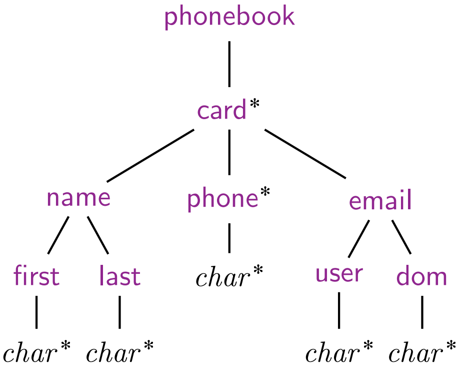
$$\begin{array}{rcl}
\mathcal{L}(\mathcal{A}, q) & = & \left\{ a \in \Sigma_0 \;\middle|\; a \to q \in \Delta \right\} \\
 & \cup & \left\{ f(t_1, \ldots, t_n) \;\middle|\; \begin{array}{l} f \in \Sigma_n \\ t_1 \in \mathcal{L}(\mathcal{A}, q_1), \ldots, t_n \in \mathcal{L}(\mathcal{A}, q_n) \\ f(L) \to q \in \Delta,\; q_1 \ldots q_n \in L \end{array} \right\}
\end{array}$$

$$\mathcal{L}(\mathcal{A}) = \bigcup_{q \in F} \mathcal{L}(\mathcal{A}, q)$$

Equivalent to ranked tree automata via binary encodings

# Type Definition for XML Data

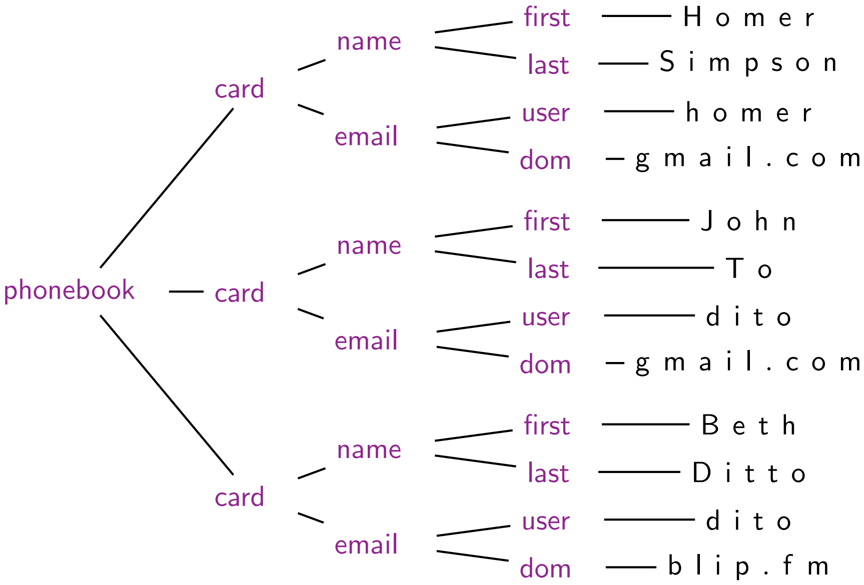
DTD



Hedge Automaton

$$
\begin{aligned}
\mathsf{phonebook}(p_{\mathsf{c}}{}^*) &\to p_{\mathsf{b}} \\
\mathsf{card}(p_{\mathsf{n}}\ p_{\mathsf{h}}{}^*\ p_{\mathsf{m}}) &\to p_{\mathsf{c}} \\
\mathsf{name}(p_{\mathsf{f}}\ p_{\mathsf{l}}) &\to p_{\mathsf{n}} \\
\mathsf{first}(p^*) &\to p_{\mathsf{f}} \\
\mathsf{last}(p^*) &\to p_{\mathsf{l}} \\
\mathsf{phone}(p^*) &\to p_{\mathsf{h}} \\
\mathsf{email}(p_{\mathsf{u}}\ p_{\mathsf{d}}) &\to p_{\mathsf{m}} \\
\mathsf{user}(p^*) &\to p_{\mathsf{u}} \\
\mathsf{dom}(p^*) &\to p_{\mathsf{d}} \\
\mathsf{a} &\to p \\
\mathsf{b} &\to p \\
&\vdots
\end{aligned}
$$

# Hedge Automaton Run



phonebook

card

name

first Homer

last Simpson

email

user homer

dom gmail.com

card

name

first John

last To

email

user dito

dom gmail.com

card

name

first Beth

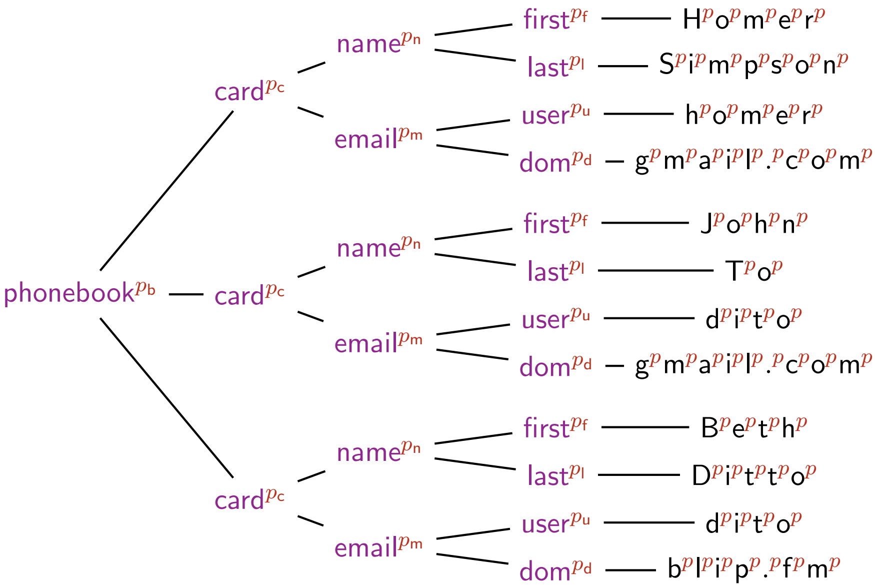
last Ditto

email

user dito

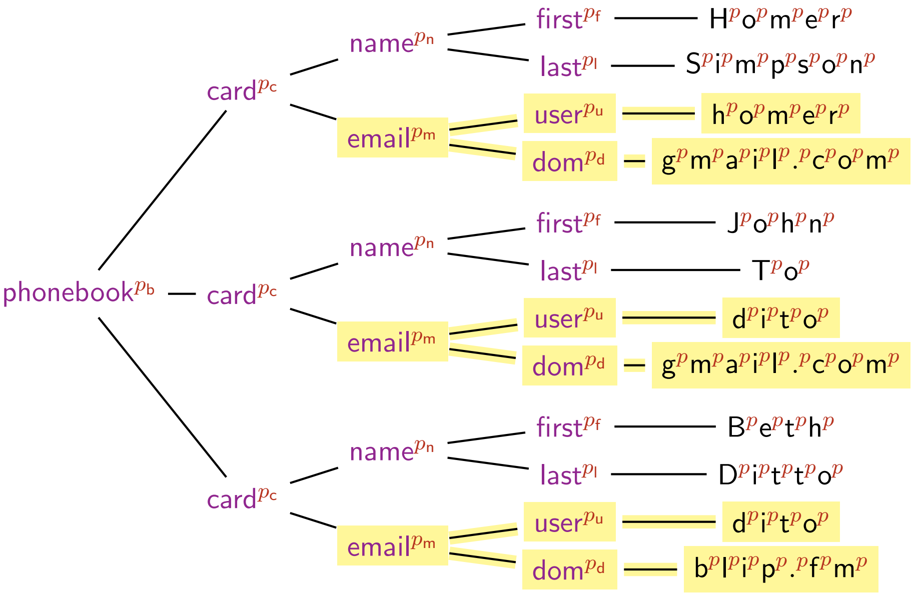
dom blip.fm

# Hedge Automaton Run



# Key Constraint

email is a key: $p_{\mathsf{m}} \not\approx p_{\mathsf{m}}$ $\quad \forall x, y\ p_{\mathsf{m}}(x) \wedge p_{\mathsf{m}}(y) \wedge x \neq y \Rightarrow t|_x \neq t|_y$



phonebook$^{p_{\mathsf{b}}}$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — H$^{p}$o$^{p}$m$^{p}$e$^{p}$r$^{p}$

last$^{p_{\mathsf{l}}}$ — S$^{p}$i$^{p}$m$^{p}$p$^{p}$s$^{p}$o$^{p}$n$^{p}$

email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — h$^{p}$o$^{p}$m$^{p}$e$^{p}$r$^{p}$

dom$^{p_{\mathsf{d}}}$ — g$^{p}$m$^{p}$a$^{p}$i$^{p}$l$^{p}$.$^{p}$c$^{p}$o$^{p}$m$^{p}$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — J$^{p}$o$^{p}$h$^{p}$n$^{p}$

last$^{p_{\mathsf{l}}}$ — T$^{p}$o$^{p}$

email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — d$^{p}$i$^{p}$t$^{p}$o$^{p}$

dom$^{p_{\mathsf{d}}}$ — g$^{p}$m$^{p}$a$^{p}$i$^{p}$l$^{p}$.$^{p}$c$^{p}$o$^{p}$m$^{p}$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — B$^{p}$e$^{p}$t$^{p}$h$^{p}$

last$^{p_{\mathsf{l}}}$ — D$^{p}$i$^{p}$t$^{p}$t$^{p}$o$^{p}$

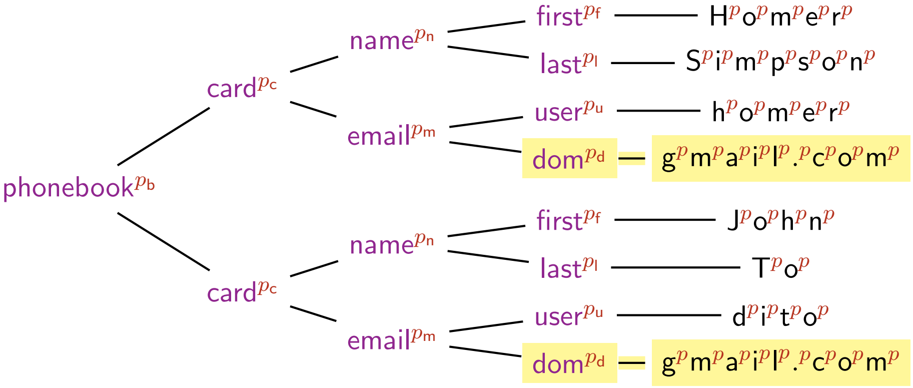
email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — d$^{p}$i$^{p}$t$^{p}$o$^{p}$

dom$^{p_{\mathsf{d}}}$ — b$^{p}$l$^{p}$i$^{p}$p$^{p}$.$^{p}$f$^{p}$m$^{p}$

# Global Equality Constraint

all domain's coincide $p_\mathsf{d} \approx p_\mathsf{d}$ $\quad \forall x, y\ p_\mathsf{d}(x) \land p_\mathsf{d}(y) \Rightarrow t|_x = t|_y$



phonebook$^{p_\mathsf{b}}$

card$^{p_\mathsf{c}}$

name$^{p_\mathsf{n}}$

first$^{p_\mathsf{f}}$ — H$^p$o$^p$m$^p$e$^p$r$^p$

last$^{p_\mathsf{l}}$ — S$^p$i$^p$m$^p$p$^p$s$^p$o$^p$n$^p$

email$^{p_\mathsf{m}}$

user$^{p_\mathsf{u}}$ — h$^p$o$^p$m$^p$e$^p$r$^p$

dom$^{p_\mathsf{d}}$ — g$^p$m$^p$a$^p$i$^p$l$^p$.$^p$c$^p$o$^p$m$^p$

card$^{p_\mathsf{c}}$

name$^{p_\mathsf{n}}$

first$^{p_\mathsf{f}}$ — J$^p$o$^p$h$^p$n$^p$

last$^{p_\mathsf{l}}$ — T$^p$o$^p$

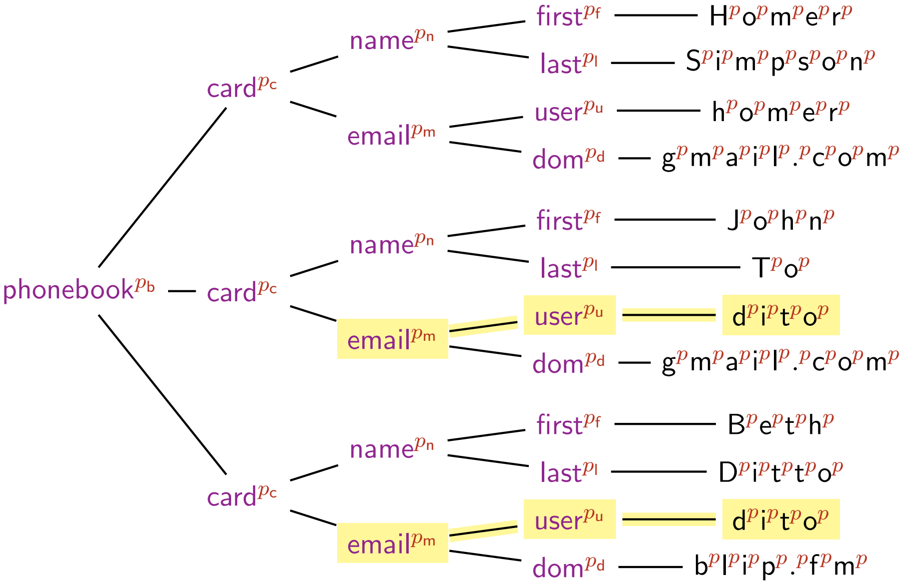
email$^{p_\mathsf{m}}$

user$^{p_\mathsf{u}}$ — d$^p$i$^p$t$^p$o$^p$

dom$^{p_\mathsf{d}}$ — g$^p$m$^p$a$^p$i$^p$l$^p$.$^p$c$^p$o$^p$m$^p$

# Negation

user is a not a key $\neg p_{\mathsf{m}} \not\approx p_{\mathsf{m}}$ $\exists x, y\; p_{\mathsf{u}}(x) \wedge p_{\mathsf{u}}(y) \wedge x \neq y \wedge t|_x = t|_y$



phonebook$^{p_{\mathsf{b}}}$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — H$^p$o$^p$m$^p$e$^p$r$^p$

last$^{p_{\mathsf{l}}}$ — S$^p$i$^p$m$^p$p$^p$s$^p$o$^p$n$^p$

email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — h$^p$o$^p$m$^p$e$^p$r$^p$

dom$^{p_{\mathsf{d}}}$ — g$^p$m$^p$a$^p$i$^p$l$^p$.$^p$c$^p$o$^p$m$^p$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — J$^p$o$^p$h$^p$n$^p$

last$^{p_{\mathsf{l}}}$ — T$^p$o$^p$

email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — d$^p$i$^p$t$^p$o$^p$

dom$^{p_{\mathsf{d}}}$ — g$^p$m$^p$a$^p$i$^p$l$^p$.$^p$c$^p$o$^p$m$^p$

card$^{p_{\mathsf{c}}}$

name$^{p_{\mathsf{n}}}$

first$^{p_{\mathsf{f}}}$ — B$^p$e$^p$t$^p$h$^p$

last$^{p_{\mathsf{l}}}$ — D$^p$i$^p$t$^p$t$^p$o$^p$

email$^{p_{\mathsf{m}}}$

user$^{p_{\mathsf{u}}}$ — d$^p$i$^p$t$^p$o$^p$

dom$^{p_{\mathsf{d}}}$ — b$^p$l$^p$i$^p$p$^p$.$^p$f$^p$m$^p$

## ⟨Negation⟩

- $\neg p_{\mathsf{m}} \not\approx p_{\mathsf{m}}$ and $p_{\mathsf{m}} \approx p_{\mathsf{m}}$ have different semantics
- $p_{\mathsf{m}} \approx p_{\mathsf{m}}$ does not hold here

# Tree Automata with Global Constraints: Definition

A tree automaton with global constraints (TAGC[$\approx, \not\approx$]) is a tuple $\mathcal{A} = \langle \Sigma, Q, F, \Delta, C \rangle$ where

- $\langle \Sigma, Q, F, \Delta \rangle$ is a HA,
- $C$ is a Boolean combination of atomic constraints
  $C := q_1 \approx q_2 \mid q_1 \not\approx q_2 \mid \neg C \mid C \vee C \mid C \wedge C$ with $q_1, q_2 \in Q$

run $r$ of $\mathcal{A}$ on $t$: function $dom(t) \to Q$ compatible with $\Delta$,
successful if $r(root) \in F$

language: $\mathcal{L}(\mathcal{A}) = \{t \mid \exists r$ successful run of $\mathcal{A}$ on $t$, $\langle t, r \rangle \models C\}$
$\langle t, r \rangle \models q_1 \approx q_2$ iff $\forall x, y \in dom(t)\ q_1(x) \wedge q_2(y) \wedge x \neq y \Rightarrow t|_x = t|_y$
$\langle t, r \rangle \models q_1 \not\approx q_2$ iff $\forall x, y \in dom(t)\ q_1(x) \wedge q_2(y) \wedge x \neq y \Rightarrow t|_x \neq t|_y$

## $\langle$Tree Automata with Global Constraints: Definition$\rangle$

$\approx$ and $\not\approx$ are symmetric

# TAGED

The original model [Filiot et al 2007 CSL], [Filiot et al 2008 DLT]
TAGED = positive TAGC$[\approx, \not\approx_{\mathsf{irr}}]$
↳ restriction to $q_1 \not\approx q_2$ with $q_1 \neq q_2$

- closure $\cup$ (polynomial), $\cap$ (exponential), not $\neg$
- membership is NP-complete
- emptiness
  - EXPTIME-complete for PCTAGC$[\approx]$
  - NEXPTIME for PCTAGC$[\not\approx_{\mathsf{irr}}]$ (set constraints with negation)
  - decidable for subclasses of TAGED bounding # of tests
- universality, inclusion undecidable
- finiteness EXPTIME for PCTAGC$[\approx]$

# ⟨TAGED⟩

TAGED introduced as a tool for deciding satisfiability of a fragment of the spatial logic TQL of Cardelli et al

# RTA

Rigid Tree Automata (RTA) $=$ subclass PCTAGC$[\approx_{\text{ref}}]$ of TAGED
J Klay Vacher 2009 LATA, J Klay Vacher 2011 IC

[Filiot et al 2008 DLT]: positive TAGC$[\approx]$ $\equiv$ positive TAGC$[\approx_{\text{ref}}]$
($\approx_{\text{ref}}$: restriction to $q \approx q$) (exponential blowup for $\rightarrow$)

- closure $\cup$ (polynomial), $\cap$ (exponential), not $\neg$
- TA $\subsetneq$ DRTA $\subsetneq$ RTA
- membership is NP-complete
- emptiness decidable in linear time
- universality, inclusion undecidable
- finiteness decidable PTIME

Godoy et al 2010 LICS

regularity is undecidable for RTA, TAGED, TAGC$[\approx]$

# ⟨RTA⟩

RTA introduced as a tool for reachability analysis of communicating processes (applied $\pi$-calculus)

# Full TAGC

**Godoy et al 2010 LICS**

emptiness is decidable for TAGC$[\approx, \not\approx]$

- One tree is accepted iff a tree of *"small"* height is accepted
- global pumping: replace all subtrees of height $h$ by selected subtrees of height $< h$ while preserving all the relative $\approx$, $\not\approx$
- accepted tree $t \mapsto$ sequence of measures $e_0, e_1, \ldots, e_{h(t)}$ st if $e_i \preceq e_j$ for $i < j$ then there exists a global pumping
- Higman's Lemma, König's Lemma: exists a bound $B$ on the maximal length of sequences (for any $t$) without $e_i \preceq e_j$, $i < j$
- every $t$ of height $> B$ can be reduced by a global pumping.

## ⟨Full TAGC⟩

- the decidability of emptiness for the full class TAGED has been open (and considered difficult) for 3 years
- roughly $e_i$ contains tuples on number of occurrences of trees of height $\leqslant i$ for each state $q$
- intuitively, it is a minimal information to keep in order to preserve the constraints
- $e_i \preceq e_j$ implies that there is an injection of terms of height $h(t) - i$ into terms of height $h(t) - j$
- König's Lemma: finite number of possible $e_0$, the tree of all possible decreasing sequences is finite because branching is finite

# Arithmetic Constraints

linear inequality $\sum_{q \in Q} a_q \cdot |q| \geqslant a$ or $\sum_{q \in Q} a_q \cdot \|q\| \geqslant a$, $a_q, a \in \mathbb{Z}$

for a run $r$ on a tree $t$, 
$$\begin{array}{rcl} |q| & = & |r^{-1}(q)| \\ \|q\| & = & \Big|\{t|_x \mid x \in dom(t), r(x) = q\}\Big| \end{array}$$

natural inequality (type '$\mathbb{N}$') when all $a_q, a$ have the same sign

Presburger automata [Seidl et al 2003 PODS, 2008], [Dal Zilio Lugiez 2006]: count the siblings of unranked trees (*local cstr*).

- emptiness decidable in NPTIME for TAGC[ $|.|_{\mathbb{Z}}$]
- emptiness undecidable TAGC[$\approx, |.|_{\mathbb{Z}}$] Godoy et al 2010 LICS
- TAGC[$\approx, \not\approx, |.|_{\mathbb{N}}, \|.\|_{\mathbb{N}}$] $\equiv$ TAGC[$\approx, \not\approx$] id

## ⟨Arithmetic Constraints⟩

- actually, the emptiness decision algorithm is for positive TAGC
- PCTAGC$[\approx, \not\approx, |.|_{\mathbb{N}}, \|.\|_{\mathbb{N}}]$ is used as an intermediate class, for the elimination of $\neg$ in global constraint
- the complete equivalence result is TAGC$[\approx, \not\approx, \mathbb{N}] =$ positive PCTAGC$[\approx, \not\approx, \mathbb{N}] =$ positive TAGC$[\approx, \not\approx]$, where $\mathbb{N}$ stands for $|.|_{\mathbb{N}}, \|.\|_{\mathbb{N}}$

# Other Decidable Extensions

Godoy et al 2010 LICS
and extended version

on ranked trees, emptiness is still decidable for

- TAGC$[\approx, \not\approx]$ extended with local $=$ and $\neq$ constraints between siblings, à la [Bogaert Tison 1992 STACS]
- TAGC$[\approx, \not\approx]$ where $\approx$ and $\not\approx$ are interpreted modulo flat equational theories

## ⟨Other Decidable Extensions⟩

- it must be mentioned that emptiness decision for these extensions work with the same technique as before
- this shows the robustness of the decision procedure

# TAGED and DAG Automata

DA: tree automata computing on DAGs representing ranked trees
emptiness is NP-complete for DA [Charatonik 1999]

[Vacher 2010 PhD]

- positive TAGC[$\not\approx_{\mathsf{irr}}$] $\equiv$ DA (exponential blowup for $\rightarrow$)
- positive TAGC[$\approx, \not\approx_{\mathsf{irr}}$] $\equiv$ DA[$\approx$]
- emptiness is decidable in NEXPTIME for TAGED

Ongoing work with A. Muscholl, C. Vacher, I. Walukiewicz:
generalization to PCTAGC[$\approx, \not\approx$]
(elementary upper bound for emptiness decision)

# ⟨TAGED and DAG Automata⟩

- DA: a run is a labeling of DAG nodes with states: one note can only receive one state when TA is ND
- definition DA$[\approx]$: for run $r$ on DAG $d$ $\langle d, r\rangle \models q_1 \approx q_2$ iff $\forall x, y \in dom(d)\ q_1(x) \wedge q_2(y) \Rightarrow x = y$
- the goal of ongoing work is the definition of a further extension of DA DA$[\approx, \not\approx_{\mathsf{ref}}] \equiv$ TAGC$[\approx, \not\approx]$

# Monadic Second Order Logic

MSO$[+1, \approx, \not\approx, |.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$ monadic second-order with predicates

$a(x)$ ($x$ labeled by $a \in \Sigma$ in $t$)

$+1$ $S_{\downarrow}(x, y)$ ($y$ child of $x$) and $S_{\rightarrow}(x, y)$ ($y$ next sibling of $x$)

$\approx$ $X \approx Y$ (for all $x \in X$, $y \in Y$, $t|_x = t|_y$)

$\not\approx$ $X \approx Y$ (for all $x \in X$, $y \in Y$, $t|_x \neq t|_y$)

$|.|_{\mathbb{N}}$ $\sum a_i \cdot |X_i| \geqslant a$ $a_i, a \in \mathbb{Z}$, same sign ($|X_i|$ is cardinality of $X_i$)

$\|.\|_{\mathbb{N}}$ $\sum a_i \cdot \|X_i\| \geqslant a$ ($\|X_i\|$ is cardinality of $\{t|_x \mid x \in X_i\}$)

- MSO$[+1]$ $\equiv$ tree automata [Thatcher Wright 1968]
- MSO$[+1, \approx]$ undecidable
- MSO$[+1, \mathbb{Z}]$ undecidable [Klaedtke Ruess 2002]
- EMSO$[+1, \mathbb{Z}]$ decidable [Klaedtke Ruess 2002]
- fragment of EMSO$[+1, \approx, \not\approx]$ decidable [Filiot et al 2008 DLT]
- EMSO$[+1, \approx, \not\approx, |.|_{\mathbb{N}}, \|.\|_{\mathbb{N}}]$ decidable Godoy et al 2010 LICS

# ⟨Monadic Second Order Logic⟩

- jump to last line
- decidability of a fragment of MSO logic interpreted on tree
- (interpretation domain for formulas is set of position in a tree)
- the predicates are the following
- navigation $S$...
- (dis)equalities...
- arithmetic (natural linear inequalities)...
- existential quantification of the variables involved in $\approx$, $\not\approx$, $|.|_{\mathbb{N}}$, $\|.\|_{\mathbb{N}}$
- proof by compilation of formulas into TAGC
- the transformation also works in the other direction
  (expression of the existence of a run in logic)
- EMSO$[+1, \approx]$ strictly more expressive than MSO (express subtree equality)
- no closure under projection under components (see Treinen 2000)
  → no Thatcher Wright like construction for quantifiers

# Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints

FO Th. Proving

Tree automata with global constraints

XML integrity constraints

Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

Unranked tree rewriting

XML updates

XML r/w ACP

# Tree Automata as Horn Clause Sets

[Frühwirth et al 1991 LICS]

Ranked tree automaton $\langle \Sigma, Q, F, \Delta \rangle$

$$f(q_1, \ldots, q_n) \to q \in \Delta$$

Finite set $\mathcal{A}$ of Horn clauses over $\Sigma$ and $Q$ (monadic)

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \in \mathcal{A}$$

$\mathcal{A}$ admit a smallest Herbrand model $\mathcal{H}_{\mathcal{A}}$ (all clauses are definite)

Language $\mathcal{L}(\mathcal{A}, q) = \{t \mid \mathcal{H}_{\mathcal{A}} \models q(t)\}$

## ⟨Tree Automata as Horn Clause Sets⟩

- Frühwirth et al: type inference in logic programming
- le plus petit modèle de Herbrand $\mathcal{H}_{\mathcal{A}}$ associe un ensemble de termes clos à chaque symbole de prédicat
- Cela permet la définition de languages pour les ensembles de clauses, qui coïncident avec les langages d'automates
- there are several advantages of this Horn clause representation of TA
- one is that it permits to describe several different TA models in one uniform formalism

# Clauses/Automata Models

standard tree automata ($x_1, \ldots, x_n$ pairwise distinct)

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \tag{reg}$$

$\varepsilon$-transitions

$$q_1(x) \Rightarrow q(x) \tag{$\varepsilon$}$$

alternating clauses

$$q_1(x), \ldots, q_n(x) \Rightarrow q(x) \tag{alt}$$

2-ways (bidirectional) clauses ($x_1, \ldots, x_n$ pairwise distinct)

$$q\big(f(x_1, \ldots, x_n)\big) \Rightarrow q_i(x_i) \tag{bidi}$$

Tree automata with sibling = constraints ($x_1, \ldots, x_n$ may have duplicates)

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \tag{sibling}$$

- $\varepsilon$-transitions can be suppressed with polynomial procedure
- les clauses alternantes sont aussi appelées clauses intersection
- elle définissent l'état $Q$ comme contenant l'intersection de $Q_1$,...,$Q_n$
- cela (l'état $Q$) correspond à une conjonction dans une transition d'automate alternant
- automates alternants = (reg) + (alt)
- i.e. they can be suppressed too but at price of exponential blowup
- clauses bidirectionelles: introduites pour l'analyse de programme logiques dans [Frühwirth, Shapiro, Vardi, Yardeni. Logic programs as types for logic programs. LICS 1991]
- cet article propose la représentation d'automates d'arbres par clauses de Horn
- lien également avec contraintes ensembliste (projection)
- automates bidirectionnels = (reg) + (bidi)
- il existe d'autres définitions des automates bidirectionnels strictement plus expressifs que les automates d'arbres
- les automates bidirectionnel alternants, définis par union de clauses des 4 types ci-dessus sont équivalents en expressivité aux automates d'arbres finis
- i.e. ∃ transformation reg+alt+bidi → reg seul

# Decision Problems, Satisfiability

TA $\mathcal{A}$ = finite set of Horn clauses (reg)

- membership: $t \in \mathcal{L}(\mathcal{A}, q)$ iff $\mathcal{A} \cup \{q(t) \Rightarrow \bot\}$ is unsatisfiable
- emptiness: $\mathcal{L}(\mathcal{A}, q) \neq \emptyset$ iff $\mathcal{A} \cup \{q(x) \Rightarrow \bot\}$ is unsatisfiable
- emptiness of intersection: $\mathcal{L}(\mathcal{A}, q_1) \cap \ldots \cap \mathcal{L}(\mathcal{A}, q_k) \neq \emptyset$ iff $A \cup \{q_1(x), \ldots, q_k(x) \Rightarrow \bot\}$ is unsatisfiable
- MII: membership to the intersection of instances
  $\exists$ substitution $\sigma$ s.t. $\sigma(t_i) \in \mathcal{L}(\mathcal{A}, q_i)$ for all $1 \leqslant i \leqslant k$ iff $A \cup \{q_1(t_1), \ldots, q_k(t_k) \Rightarrow \bot\}$ is unsatisfiable
- MI: membership of an instance = MII with $k = 1$

- use automated deduction techniques for these problems

## ⟨Decision Problems, Satisfiability⟩

another advantage of representing TA as Horn clause is that is permits to reduce many decision problems on TA to Horn clauses satisfiability

# Decision by Saturation

[Goubault-Larrecq 2005 IPL]

ordered resolution with selection and $\varepsilon$-splitting terminates on an instance of MII with (reg) clauses

- number of step at most exponential
- exact complexity for MII

This technique also permits to cast alternating and 2-way TA into TA (reg + alt + bidi = reg) with exponential blowup

J Rusinowitch Vigneron 2006 IJCAR

J Rusinowitch Vigneron 2008 JLAP

- tree automata clauses with
  - positive equalities (equational theories)
  - negative equalities (constraints)
- saturation with basic ordered paramodulation with selection and $\varepsilon$-splitting

# ⟨Decision by Saturation⟩

- ordering and selection function are parameters of the calculus
- careful choices for these parameters can ensure termination
- Resolution:

$$\frac{C \vee +Q(s) \qquad -Q(t) \vee D}{C\sigma \vee D\sigma}$$

  where $\sigma$ is the most general unifier (*mgu*) of $s$ and $t$
- unificateur de $s$ et $t =$ subtitution $\sigma$ telle que $s\sigma = t\sigma$ $\sigma < \sigma'$ ($\sigma$ plus général que $\sigma'$) si $\exists\theta\, \sigma' = \sigma\theta$ deux termes unifiables ont un unificateur le plus général, unique modulo renomage de variables
- correction: si $S'$ obtenu de $S$ par résolution, alors tout modèle de $S$ est un modèle de $S'$.
- donc si $S'$ n'a pas de modèle, alors $S$ n'a pas de modèle.
- donc si $\perp$ est déduit à partir de $S$ par résolution, $S$ est insatisfiable.
- complétude: si $S$ insatisfiable, alors on déduit la clause vide à partir de $S$ en un nombre fini d'étapes de résolution.

- la résolution seule n'est pas complète pour les clauses générales.
- dans le cas des clauses générales, il faut ajouter une règle de factorisation:

$$\underline{C \vee +P(s) \vee +P(t)}$$

# TA with Equality Constraints modulo Equational Theories

$$\begin{array}{lll}
 & \Rightarrow \ell = r & \text{(eq)} \\
q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k & \Rightarrow q\big(f(x_1, \ldots, x_n)\big) & \text{(test)} \\
\text{for all } i \leqslant n, q \succ q_i & & \\
q_1(x_1), \ldots, q_n(x_n) & \Rightarrow q\big(f(x_1, \ldots, x_n)\big) & \text{(reg}'\text{)} \\
\multicolumn{3}{l}{q, q_1, \ldots, q_n \text{ minimal or } q_i = q,\ q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \text{ minimal}}
\end{array}$$

| | no clauses (eq)<br>(empty eq. theory) | clauses (eq) |
|---|---|---|
| std TA<br>(reg) | TA | TAE |
| TA with constraints<br>(test)+(reg') | TAD | TADE |

# TA with Equality Constraints modulo Equational Theories

$$\begin{array}{lll} & \Rightarrow \ell = r & \text{(eq)} \\ q_1(x_1),\ldots,q_n(x_n), u_1 = v_1,\ldots,u_k = v_k & \Rightarrow q\big(f(x_1,\ldots,x_n)\big) & \text{(test)} \\ \text{for all } i \leqslant n, q \succ q_i & & \\ q_1(x_1),\ldots,q_n(x_n) & \Rightarrow q\big(f(x_1,\ldots,x_n)\big) & \text{(reg}'\text{)} \end{array}$$

$q, q_1, \ldots, q_n$ minimal or $q_i = q$, $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n$ minimal

| | no clauses (eq)<br>(empty eq. theory) | clauses (eq) |
|---|---|---|
| std TA<br>(reg) | TA | TAE |
| TA with constraints<br>(test)+(reg') | TAD | TADE |

for equational theory $\succ$-convergent, and monadic: $\ell = x$ or $\ell = g(z_1, \ldots, z_n)$, $z_1, \ldots, z_n$ distinct

$$\begin{array}{rcl} \mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) & = & \mathsf{eq}(x, y) \\ \mathsf{cons}(x, \mathsf{cons}(x, y)) & = & \mathsf{cons}(x, y) \end{array}$$

# TA with Equality Constraints modulo Equational Theories

$$\Rightarrow \ell = r \qquad \text{(eq)}$$

$$q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \quad \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(test)}$$

for all $i \leqslant n, q \succ q_i$

$$q_1(x_1), \ldots, q_n(x_n) \quad \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg}'\text{)}$$

$q, q_1, \ldots, q_n$ minimal or $q_i = q$, $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n$ minimal

| | no clauses (eq)<br>(empty eq. theory) | clauses (eq) |
|---|---|---|
| std TA<br>(reg) | TA | TAE |
| TA with constraints<br>(test)+(reg') | TAD | TADE |

for equational theory $\succ$-convergent, collapsing: $\ell = x$, and sublinear: $f(\ell_1, \ldots, \ell_n) = x$, $\quad \ell_1, \ldots, \ell_n$ linear

$$\mathsf{car}(\mathsf{cons}(x, y)) = x, \quad \mathsf{cdr}(\mathsf{cons}(x, y)) = y, \quad \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)) = x$$

$$\mathsf{dec}\big(\mathsf{enc}(x, y), y\big) = x, \quad \mathsf{adec}\big(\mathsf{enc}(x, \mathsf{pub}(y)), \mathsf{priv}(y)\big) = x$$

# TA with Equality Constraints modulo Equational Theories

$$\Rightarrow \ell = r \qquad \text{(eq)}$$

$$q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(test)}$$

for all $i \leqslant n, q \succ q_i$

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg}'\text{)}$$

$q, q_1, \ldots, q_n$ minimal or $q_i = q$, $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n$ minimal

| | no clauses (eq)<br>(empty eq. theory) | clauses (eq) |
|---|---|---|
| std TA<br>(reg) | TA | TAE |
| TA with constraints<br>(test)+(reg') | TAD | TADE |

prototype http://tace.gforge.inria.fr/

## ⟨TA with Equality Constraints modulo Equational Theories⟩

- (reg′) similar to the condition of RA, after patch for the problem for ND
- proof TAE by finite (exponential) invariant type. saturation steps:

1. superposition of equations into (reg) clauses → 2-way like clauses
2. elimination of 2-way clauses
   i.e. closure is regular
   use of basic strategy is crucial

- proof TAD: every paramodulation step returns either a clause
  smaller than its premises (because of the state ordering $\succ$)
  or a clause of an invariant finite type
- proof TADE: same principle as TADE
  more involved (count number variables, unification lemma for sub linear)
  (negative) equations are eliminated first
- CSQ: MII decidable

# TA with Equality Constraints modulo Equational Theories

J Meyer Weidenbach 1998 RTA

TA with sibling equalities constraints = finite set of Horn clauses

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(sibling)}$$

- languages not closed under rewriting with flat TRS
  *ex.* closure of $\{f^n(a) \mid n \geqslant 0\}$ by $f(x) \to g(x, x)$
- solution: sibling constraint modulo

$$p_1(t), \ldots, p_m(t), q_1(x_1), \ldots, q_n(x_n) \Rightarrow q(s) \mid \ell = r \qquad \text{(fb)}$$

($t$, $s$, $\ell$, $r$ flat)

- sorted superposition terminates on finite a set of clauses (fb).

## ⟨TA with Equality Constraints modulo Equational Theories⟩

- (fb) is a generalization of (sibling)
- csq termination: unification decidable in (fb) theories
- and generalization *semi-linear* theories (transformable into (fb) theories)

# Disequality Constraints

[Reuss Seidl 2010 LPAR]

tree automata with term constraints (TCA)

$$\begin{aligned} q_1(x_1), \ldots, q_n(x_n), \quad & x_{i_1} = s_1, \ldots, x_{i_k} = s_k, \\ & x_{j_1} \neq t_1, \ldots, x_{j_l} \neq t_l \Rightarrow q\big(f(x_1, \ldots, x_n)\big) \quad \text{(tca)} \end{aligned}$$

$$\mathsf{vars}(s_1, \ldots, s_k, t_1, \ldots, t_l) \subseteq \{x_1, \ldots, x_n\}$$

- generalize the tree automata with sibling constraints of [Bogaert Tison 1992 STACS]
- Boolean closure
- emptiness decidable

question: combination of TCA with (test) and (eq)?

## Pushdown Automata

Extension des automates de mots finis (NFA) avec une mémoire non bornée = pile.

$\mathcal{A} = (\Sigma, \Gamma, Q, Q^{\mathsf{i}}, Q^{\mathsf{f}}, \delta)$, where $\Sigma$: alphabet d'entrée, $\Gamma$: alphabet de pile.

$$\delta \subseteq \quad \underset{(push)}{Q \times \Sigma \times Q \times \Gamma} \quad \cup \quad \underset{(pop)}{Q \times \Sigma \times \Gamma \times Q} \quad \cup \quad \underset{(internal)}{Q \times \Sigma \cup \{\varepsilon\} \times Q}$$

On peut généraliser les 3 types de transitions à

$$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \times Q \times \Gamma^*$$

transition $(q, a, \gamma, q', u)$: dans l'état $q$, lisant $a$, avec $\gamma$ en haut de la pile, l'automate: passe dans l'état $q'$, passe aucaractère suivant dans le mot d'entrée, retire $\gamma$, pousse $u$ sur la pile.
acceptation: en arrivant dans la pile vide.
Example: $\Sigma = \{a, b\}$, $\Gamma = \{\alpha\}$.

$$\begin{array}{rl} push: & q^{\mathsf{i}} \xrightarrow{a} q^{\mathsf{i}}, \alpha \\ internal: & q^{\mathsf{i}} \xrightarrow{b} q^{\mathsf{f}} \\ pop: & q^{\mathsf{f}} \xrightarrow{a, \alpha} q^{\mathsf{f}} \end{array}$$

reconnaît: $\{a^n b a^n \mid n \geqslant 0\}$.

## Automates à pile (propriétés)

- expressivité $\supsetneq$ NFA
- même expressivité que les grammaires hors-contexte ($N := N_1 N_2$, $N := a$)
- vide décidable
- pas clos par intersection, complément
- la restriction visibly: $\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int}$ a les clôtures Booléennes.

restriction visibly clos par intersection, complément

**Lemma :**

Le langage des piles accessibles est régulier.

⟨Disequality Constraints⟩

Pushdown Tree Automata with One Tree Memory [[Guessarian 83, Schimpf Gallier 85, Coquidé et al 94]]

- $\Sigma$: input signature; on input: terms of $\mathcal{T}(\Sigma)$
- $\Gamma$: stack alphabet; the auxiliary memory is a stack of $\Gamma^*$

$$Q_1(x_1, s_1), \ldots, Q_n(x_n, s_n) \Rightarrow Q\big(f(\overline{x}), h(y_1, \ldots, y_m)\big) \qquad \text{(read)}$$

where $f \in \Sigma$, $h \in \Gamma$, $s_1, \ldots, s_n \in \mathcal{T}(\Gamma, \{y_1, \ldots, y_m\})$

$$Q_1(x, s) \Rightarrow Q\big(x, h(y_1, \ldots, y_m)\big) \qquad \text{(pda-}\varepsilon\text{)}$$

where $h \in \Gamma$, $s \in \mathcal{T}(\Gamma, \{y_1, \ldots, y_m\})$.
⟨Disequality Constraints⟩

equivalent to the pushdown tree automata with stack

- il s'agit d'un cas particulier des automates de [Guessarian 83] qui sont descendants avec un arbre comme mémoire auxiliaire et des transitions:

  $$q\big(e(y_1, \ldots, y_m)\big) \to f\big(q_1(s_1), \ldots, q_n(s_n)\big)$$

  où $s_1, \ldots, s_n \in \mathcal{T}(\Gamma, \{y_1, \ldots, y_m\})$
- Il y a équivalence en expressivité entre
- ces automates d'arbres (top-down) avec un arbre en mémoire,
- les automates d'arbres (top down) à pile du transparent et
- les grammaires d'arbres hors-contexte (cf. chap. 2 TATA).

top-down transitions

$$\begin{array}{crcl}
push & q(y) & \to & f\big(q_1(e_1(y)), \ldots, q_n(e_n(y))\big) \\
pop & q(e(y)) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
pop & q(\bot) & \to & f\big(q_1(\bot), \ldots, q_n(\bot)\big) \\
int & q(y) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
\varepsilon & q(y) & \to & q'(y)
\end{array}$$

# Tree Automata with One Memory

[Guessarian 83 MST], [Schimpf Gallier 85 JCSS], [Coquidé et al 94 TCS], [Comon Cortier 05 TCS]

$$q_1(x_1, y_1), q_2(x_2, y_2) \Rightarrow q\big(f(x_1, x_2), h(y_1, y_2)\big) \qquad \text{(push)}$$

$$q_1\big(x_1, h(y_{11}, y_{12})\big), q_2(x_2, y_2) \Rightarrow q\big(f(x_1, x_2), y_{11}\big) \qquad (\text{pop}_{11})$$

$$q_1(x_1, \bot), q_2(x_2, y_2) \Rightarrow q\big(f(x_1, x_2), \bot\big)$$

$\ldots$ (pop$_{12}$), (pop$_{21}$), (pop$_{22}$)

$$\Rightarrow q(a, \bot) \qquad (\text{int}_0)$$

$$q_1(x_1, y_1), q_2(x_2, y_2) \Rightarrow q\big(f(x_1, x_2), y_1\big) \qquad (\text{int}_1)$$

$$q_1(x_1, y_1), q_2(x_2, y_2) \Rightarrow q\big(f(x_1, x_2), y_2\big) \qquad (\text{int}_2)$$

where $h \in \Gamma_2$ (memory signature) $a \in \Sigma_0$, $f \in \Sigma_2$ (input sig.)

# ⟨Tree Automata with One Memory⟩

- on se limite ici aux symboles d'arité 0 ou 2 pour $\Sigma$ et $\Gamma$
- $\perp \in \Gamma_0$ est le symbole de mémoire vide
- generalize pushdown automata but not pushdown tree automata (with a stack)
- orthogonal en expressivité avec les automates descendante de [Guessarian 83] présentés ci-dessus
- transitions $q\big(e(y_1, \ldots, y_m)\big) \rightarrow f\big(q_1(s_1), \ldots, q_n(s_n)\big)$ with $s_1, \ldots, s_n \in \mathcal{T}(\Gamma, \{y_1, \ldots, y_m\})$

# Tree Automata with One Memory: Languages and Closure

(input) language

$$\mathcal{L}(\mathcal{A}, q) = \{t \mid q(t, s) \in \mathcal{H}_{\mathcal{A}}\}$$

memory language

$$\mathcal{M}(\mathcal{A}, q) = \{s \mid q(t, s) \in \mathcal{H}_{\mathcal{A}}\}$$

- closure of languages: $\cup$, not $\cap$, not $\neg$

# Visibly Tree Automata with One Memory

Comon-Lundh J Perrin 2007 FOSSACS

visibly condition à la [Alur Madhusudan 2004 STOC]

$\Sigma = \Sigma_{\mathsf{push}} \uplus \Sigma_{\mathsf{pop}_{11}} \uplus \ldots \uplus \Sigma_{\mathsf{int}_0} \uplus \Sigma_{\mathsf{int}_1} \uplus \Sigma_{\mathsf{int}_2}$

the input symbol determines the possible operations on memory

- closure of languages under $\cup$ and $\cap$ and $\neg$
- emptiness is PTIME-complete

$\mathcal{L}(\mathcal{A}, q) = \varnothing$ iff $\mathcal{M}(\mathcal{A}, q) = \varnothing$

$\mathcal{M}(\mathcal{A}, q)$ is in $\mathcal{H}_3$ language [Nielson Seidl 02 SAS] $O(n^3)$

- universality and inclusion are EXPTIME-complete

related: [Chabin Réty 2007 FroCoS]

## ⟨Visibly Tree Automata with One Memory⟩

- visibly: the computation depends on the shape of the input tree
- for all $\mathcal{A}$, $q$, $\mathcal{M}(\mathcal{A}, q)$ is a language of bidirectional alternating automata (clauses reg + alt + bidi = reg)
- definition by ”*projection*” of clauses on second component of states
- def $\mathcal{H}_1$: $B \Rightarrow h$
  - $h$ linear
  - variables connected in $B$ (in literals sharing variables) are siblings in $h$
- $\mathcal{H}_3$ decidable in cubic time
- def $\mathcal{H}_3$: $B \Rightarrow h$
  - $h$ and all literal of $B$ are linear
  - variable dependence graph for $B\,h$ is acyclic, adjacent literals have at most one variable in common

# Visibly Tree Automata with One Memory and Contraints

Comon-Lundh J Perrin 2008 LMCS

extension with constraints testing the contents of the memory

$$q_1(x_1, y_1), q_2(x_2, y_2), R(y_1, y_2) \Rightarrow q\big(f(x_1, x_2), y_1\big) \qquad (\mathsf{int}_1^=)$$

$$q_1(x_1, y_1), q_2(x_2, y_2), R(y_1, y_2) \Rightarrow q\big(f(x_1, x_2), y_2\big) \qquad (\mathsf{int}_2^=)$$

definition of constraints:

$$y_1 = y_1', y_2 = y_2' \Rightarrow g(y_1, y_2) = g(y_1', y_2') \qquad (=)$$

$$y_1 \equiv y_1', y_2 \equiv y_2' \Rightarrow g(y_1, y_2) \equiv h(y_1', y_2') \qquad (\equiv)$$

$$R_1(y_1, y_1'), R_2(y_2, y_2') \Rightarrow R_3\big(g(y_1, y_2), h(y_1', y_2')\big) \qquad (\mathsf{reg})$$

| constraints | membership | emptiness | $\cap, \neg$ |
|---|---|---|---|
| none | PTIME | PTIME | yes |
| $=$ | NP-complete | EXPTIME | no |
| $\equiv$ | PTIME | 2-EXPTIME | yes |
| (reg) | NP-complete | undec. | yes |

combination with local sibling constraints in input tree $(x_1 = x_2)$

# ⟨Visibly Tree Automata with One Memory and Contraints⟩

- constraints restricted to transitions int
- definition with constraints and negation
- for (reg), negation can be defined
- for $=$, $\equiv$, it must be added (but is still a particular case of (reg)
- sibling tests: when combined with $\equiv$, same results as $\equiv$ alone

# TA1M for Verification

Tree automata with one memory and constraints can recognize the following data-structures

- balanced binary trees
- powerlists
  (description and verification of data parallel algorithms)
- red-black trees (binary search trees)
  1. every node is black or red
  2. the root is black
  3. all the leaves are black
  4. the 2 children of a red node are black
  5. all paths have the same number of black nodes

# ⟨TA1M for Verification⟩

- remember: language of balanced binary trees is not regular
- powerlists (Misra): lists of length $2^n$, for $n \geqslant 0$, whose elements are stored in the leaves of a balanced binary tree
- data structure for data parallel algorithms with recursive structure: Fast Fourier Transform, Batcher's sorting schemes and prefix-sum. permits succinct descriptions of such algorithms, highlighting the roles of both parallelism and recursion

→ verification of properties of these algorithms, proof of equivalence of different algorithms that solve the same problem

- red-black = binary search tree well balanced
- efficient traversal, search and sort algorithms associated are verifiable with TA techniques
- red-black: properties 1-4 are regular
- red-black: property 5 en faisant push à la lecture d'un noeud noir et testant l'égalité à lecture d'un noeud rouge

# Accumulating Parameters

[Affeldt Comon-Lundh 2009 FPS]

Horn clauses with $m$ rigid variables

→ Horn clauses with $m$ accumulating parameters

$$q_{\boldsymbol{u}}(y_i, y_1, \ldots, y_m) \Rightarrow q_{\boldsymbol{u}'}(y_i, y_1, \ldots, y_m) \qquad \text{(write)}$$

$$q_{\boldsymbol{u}}(x_1, \overline{\boldsymbol{y}}), \ldots, q_{\boldsymbol{u}}(x_n, \overline{\boldsymbol{y}}) \Rightarrow q_{\boldsymbol{u}}\big(f(x_1, \ldots, x_n), \overline{\boldsymbol{y}}\big) \qquad \text{(rig)}$$

- termination of ordered resolution
- Horn presentation (and FO techniques) for RTA?
- RTA modulo equational theories?

## ⟨Accumulating Parameters⟩

- set $\mathcal{C}$ of clauses with rigid free variables $Y_1, \ldots, Y_m$ and flexible free variables $x_1, \ldots, x_n$
- satisfiable if $\exists$ $\Sigma$-algebra $\mathfrak{A}$ st $\forall$ $\sigma : \{X_1, \ldots, X_m\} \rightarrow \mathfrak{A}$, $\exists$ FO model $\mathcal{M}$ with domain $\mathfrak{A}$ st $\mathcal{M}, \sigma \models \forall x_1, \ldots, x_n \, \mathcal{C}$
- equivalently $\forall$ $\sigma : \{X_1, \ldots, X_m\} \rightarrow \mathcal{T}(\Sigma)$, $\exists$ Herbrand model $\mathcal{H}$ st $\mathcal{H} \models \forall x_1, \ldots, x_n \; \sigma(\mathcal{C})$.

# Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints

FO Th. Proving

Tree automata with global constraints

XML integrity constraints

Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

Unranked tree rewriting

XML updates
XML r/w ACP

# Term Rewriting Systems

TA are often used as decision tool for properties of TRS

key property: effective preservation of regularity
(forward closure of a TA is a TA that can be constructed)

- for ground TRS [Brainerd 1969 SWAT]
  (the rewriting relation is regular [Dauchet et al 1987 LICS])
- for linear and right-flat (*monadic*) TRS [Salomaa 1988 JCSS]
- for linear and right-shallow (*semi-monadic*) TRS
  [Coquidé et al 1994 TCS]
- for generalized semi-monadic TRS [Gyenizse Vágvölgyi 1998 TCS]

  for right-linear and right-shallow TRS [Nagaya Toyama 2002 IC]
- for right-linear finite path overlapping TRS [Takai et al 2000 RTA]

# Term Rewriting Systems: Decision Problems

decision problems for TRS

| | |
|---|---|
| reachability | given $s, t$, $s \xrightarrow[\mathcal{R}]{*} t$? |
| joinability | given $s, t$, $\exists u\ s \xrightarrow[\mathcal{R}]{*} u \xleftarrow[\mathcal{R}]{*} t$? |
| confluence | for all $s, t, u$ if $s \xleftarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{R}]{*} t$ then $\exists v\ s \xrightarrow[\mathcal{R}]{*} v \xleftarrow[\mathcal{R}]{*} t$ |

Jacquemard 2003 IPL

Mitsuhashi Oyamaguchi J 2006 AISC

- reachability,
- joinability,
- confluence are undecidable for flat TRS

(techniques of Ganzinger J Veanes 1998 ASIAN, 2000 IJFCS)

open question (M. Sakai): for flat non-collapsing TRS?

## ⟨Term Rewriting Systems: Decision Problems⟩

- preservation of regularity enables the decision of reachability, joinability and local confluence (reduction to emptiness decision).
- regularity of the rewrite relation (like for ground TRS) permits the decision of confluence
- rule of thumb for preservation of regularity: lhs is not important, rhs must be flat and linear
- right-flatness: simulation TM with $a(b(x)) \to c(d(x))$ (words)
- right-linearity: $f(x) \to g(x, x)$ applied to $\{f^n(0) \mid n \geqslant 0\}$
- commitment: flat TRS
- confluence is decidable for shallow right-linear TRS and for right-(ground or variable) TRS (Godoy and Tiwari)
- reachability dec. for symmetric of [Nagaya and Toyama 2002]: left-shallow, left-linear and non-collapsing
- reachability open for shallow and left linear TRS (BTTA modulo?)

# Term Rewriting Systems: Unicity of Normal Forms

UN for all $s$, there is at most 1 NF $n$ st $s \xrightarrow[\mathcal{R}]{*} n$

- PTIME for shallow and linear TRS Godoy J 2009 RTA

- undecidable for right-ground TRS [Verma 2008 FI]
- undecidable for flat TRS [Godoy Hernandez 2009 AAECS]
- undecidable for linear and right-flat TRS [Godoy Tison 2007 CADE]
- undecidable for flat and right-linear TRS Godoy J 2009 RTA

## ⟨Term Rewriting Systems: Unicity of Normal Forms⟩

many other natural properties of TRS like reachability, termination, confluence, weak normalization, etc. are decidable for the last class of TRS (flat and right-linear)

# Rewrite Closure of Tree Automata with Constraints

- TA with sibling equalities constraints not closed under rewriting with flat TRS

J Meyer Weidenbach 1998 RTA

J Klay Vacher 2011 IC

- closure of RTA by linear and collapsing TRS is not RTA
- membership to the closure is undecidable for RTA and linear and collapsing TRS
- membership to the closure is decidable for RTA and linear and invisibly pushdown TRS

"invisibly": $\underset{\text{push}}{\mathsf{adec}}\Big(\underset{\text{pop}}{\mathsf{enc}}(x, \underset{\text{int}}{\mathsf{pub}}(a)), \underset{\text{int}}{\mathsf{priv}}(a))\Big) \to x$

*see* [Chabin Réty 2007 FroCoS]:
closure of VPTA by linear visibly context-free TRS

- constraints (local or global) modulo and superposition calculi?

## ⟨Rewrite Closure of Tree Automata with Constraints⟩

- the motivation for studying was the decision of reachability properties
- bad new is that closure of RTA does not hold for very restricted TRS
- however, the weaker problem of decision of membership to the closure is sufficient for our purpose
- bad new again, it is undecidable for the same TRS
- restriction: invisibly = inverse visibly

# Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints

FO Th. Proving

Tree automata with global constraints

XML integrity constraints

Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

Unranked tree rewriting

XML updates
XML r/w ACP

# Rewriting Strategies

using rewriting strategies can improve preservation of ranked tree automata languages

bottom-up strategy inverse-preserve regularity

- for linear TRS [Durand Sénizergues 2007 RTA]

bounded strategy inverse-preserve regularity

- for linear TRS [Durand et al 2010 RTA]
- for left-linear TRS [Durand Sylvestre 2011 RTA]

context-sensitive strategy [OBJ2 1985 POPL] preserves regularity

- for linear right-shallow TRS [Kojima Sakai 2009 RTA]

# Rewriting Strategies (cntd)

Gascón Godoy J 2008 WRS
and extended version

innermost strategy (call-by-value)

TA = regular languages (ranked trees)

TAS = languages of tree automata with sibling constraints

- TA → TAS for shallow TRS

  innermost- reachability, joinability decidable for shallow TRS

  regularity of innermost-closure is decidable for shallow TRS
- TA → TA for linear and right-shallow TRS
  also in [Kojima Sakai 2009 RTA]
- TAS $\not\rightarrow$ TAS for linear and flat TRS
- TA $\not\rightarrow$ TA for right-linear and right-flat TRS

# ⟨Rewriting Strategies (cntd)⟩

### TAS construction

1. reduction to the reachable terms from a constant
2. construction TAS recognizing the reachable terms from a constant based on a representation of the set of reachable terms by constrained terms from [Godoy, Huntingford 2007]

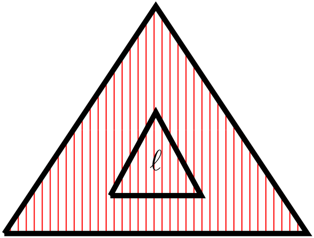
### contrast with plain rewriting

- reachability (for plain rewriting) is undecidable for shallow TRS
- regularity of (plain rewriting) closure is undecidable for shallow TRS
- right-linear and right-flat TRS preserve regular for plain rewriting

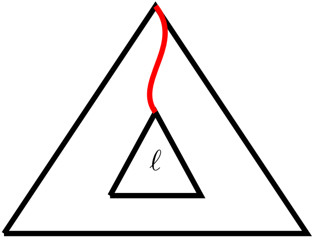
# Controlled Term Rewriting

J Kojima Sakai 2011 FroCoS

specification (for each rewrite rule) of rewrite positions with
selection automata [Frick Grohe Koch 2003 LICS]
(≡ 1 var. MSO & monadic Datalog [Gottlob Koch 2004 JACM])



CntTRS



pCntTRS
(prefix control)

- SA introduced in the context of evaluating unary (XPath) or node-selecting (monadic Datalog) queries on compressed unranked trees
- express innermost strategy for left-linear TRS
- express CS strategy

# Controlled Term Rewriting

J Kojima Sakai 2011 FroCoS

- closure of CSTG by monotonic CntTRS
  $\ell \to r$ with $\ell$, $r$ linear, vars$(\ell)$ = vars$(r)$, $|$dom$(\ell)| = |$dom$(r)|$
- reachability undecidable for flat word CntTRS
  $a(x) \to b(x)$, $b \to a(b)$, $a(b) \to b$
- reachability undecidable for ground CntTRS
- reachability PSPACE-complete for CF nc word CTRS
  = CS languages [Dassow et al 1997 handbook FL]
- reachability PSPACE-complete for CF nc word pCTRS
  NF of CS $AB := AC$, $A := BC$, $A := a$ [Penttonen 1974 IC]
- recursive control: regular model checking EXPTIME
  for linear right-shallow pCntTRS

## ⟨Controlled Term Rewriting⟩

- recursive: if you prefix-select a position $p$ in a term $t$, then $p$ is also selected in the descendants of $t$.

# Outline

static properties: composition and decision (constraint solving)

Tree automata with local constraints
FO Th. Proving

Tree automata with global constraints
XML integrity constraints

Horn Clauses with equality

dynamic properties: forward closure (regular model checking)

Ranked tree rewriting

Rewriting strategies

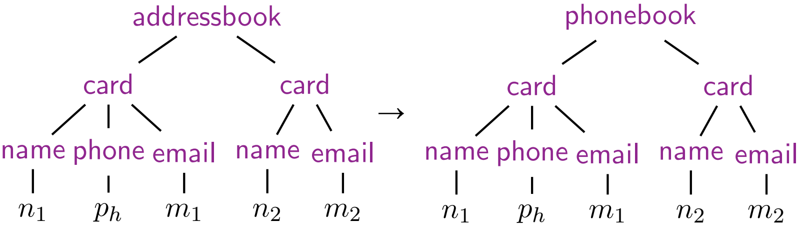
Unranked tree rewriting
XML updates
XML r/w ACP

# Unranked Tree Rewriting Systems (HRS)

[Löding Spelten 2007 MFCS], [Touili 2007 VECOS]

$$\text{addressbook}(x) \rightarrow \text{phonebook}(x)$$

- the rule can be applied to any node labeled by addressbook
- the variable $x$ represents a finite sequence of trees (hedge)



$\simeq$ term rewriting modulo A (via binary encoding)

# ⟨Unranked Tree Rewriting Systems (HRS)⟩

- [Löding and Spelten 2007] works on infinite TRS to unranked tree rewriting for the case of subtree and flat prefix rewriting which is a combination of standard ground tree rewriting and prefix word rewriting on the ordered leaves of subtrees of height 1

# HA and CF-HA

[de la Higuera PhD] [Ohsaki 2001 CSL]

Variants of the hedge automata of [Murata 2000]
A HA, resp. CF-HA is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

- $\Sigma$ is an (unranked) alphabet,
- $Q$ is a finite set of states,
- $F \subset Q$ is the subset of final states,
- $\Delta$ is a set of transitions of the form $f(L) \to q$ where
  - $L$ is a regular word language over $Q^*$
  - $L$ is a context-free CF word language over $Q^*$

HA $\equiv$ ranked tree automata

CF-HA $\equiv$ ranked tree automata modulo A

## $\langle$HA and CF-HA$\rangle$

- regular strictly included in A-TA
- A-TA not closed under intersection
- leaf$\big($A$(L_1) \cap$ A$(L_2)\big) =$ leaf$(L_1) \cap$ leaf$(L_2)$

(i) leaf$(L) =$ leaf$\big($A$(L)\big)$

(ii) $s =_A t$ implies leaf$(s) =$ leaf$(t)$
and converse if $\Sigma = \Sigma_\varnothing \uplus \{a\}$ ($\Sigma_A$ singleton).

## $\langle$HA and CF-HA$\rangle$

Correspondences $\mathcal{T}(\Sigma) \leftrightarrow \mathcal{O}(\Sigma)$ $\Sigma = \Sigma_{\varnothing} \uplus \{a\}$ where $a$ is the only associative symbol.

$$\begin{array}{lrcl} \mathit{flat} : & \mathcal{T}(\Sigma) & \rightarrow & \mathcal{O}(\Sigma) \\ \mathit{hflat} : & \mathcal{T}(\Sigma)^{*} & \rightarrow & \mathcal{H}(\Sigma) \\ \mathit{flat}^{-1} : & \mathcal{O}(\Sigma) & \rightarrow & \mathcal{T}(\Sigma) \end{array}$$

($\mathit{flat}^{-1}$ définie sur $\mathcal{O}(\Sigma) \cap \mathit{flat}(\mathcal{T}(\Sigma))$)

Definitions ($g \in \Sigma_n \backslash \Sigma_{\mathsf{A}}$):

$$\begin{array}{rcl} \mathit{flat}\big(g(t_1, \ldots, t_n)\big) & = & g(\mathit{flat}(t_1) \ldots \mathit{flat}(t_n)) \\ \mathit{flat}\big(a(t_1, t_2)\big) & = & a(\mathit{hflat}(t_1\, t_2)) \\[1ex] \mathit{hflat}\big(g(s_1, \ldots, s_n)\, t_2 \ldots t_m\big) & = & \mathit{flat}\big(g(s_1, \ldots, s_n)\big)\ \mathit{hflat}(t_2 \ldots t_m) \\ \mathit{hflat}\big(a(s_1, s_2)\, t_2 \ldots t_m\big) & = & \mathit{hflat}(s_1 s_2\, t_2 \ldots t_m) \\[1ex] \mathit{flat}^{-1}\big(g(t_1 \ldots t_n)\big) & = & g(\mathit{flat}^{-1}(t_1), \ldots, \mathit{flat}^{-1}(t_n)) \\ \mathit{flat}^{-1}\big(a(t_1 \ldots t_m)\big) & = & a(\mathit{flat}^{-1}(t_1), a(\mathit{flat}^{-1}(t_2), \ldots, \\ & & a(\mathit{flat}^{-1}(t_{m-1}), \mathit{flat}^{-1}(t_m)))) \\ & & (m \geqslant 2) \end{array}$$

## A-TA ↔ CF-HA via flattening

$\subseteq$ for all TA $\mathcal{A}$ there exists a CF-HA $\mathcal{A}'$ such that $L(\mathcal{A}') = \mathit{flat}\big(L(\mathcal{A})\big) = \mathit{flat}\big(\mathsf{A}(L(\mathcal{A}))\big)$.

| TA $\mathcal{A}$ | CF-HA $\mathcal{A}'$ |
|---|---|
| $a(q_1, q_2) \to q$ | $N_q := N_{q_1} N_{q_2}$, $N_q := q$ |
| $g(q_1, \ldots, q_k) \to q$ | $g(q_1 \ldots q_k) \to q$ |

$\supseteq$ for all CF-HA $\mathcal{A}'$ there exists a TA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathit{flat}^{-1}\big(L(\mathcal{A}')\big)$ (*i.e.* $\mathit{flat}\big(\mathsf{A}(L(\mathcal{A}))\big) = L(\mathcal{A}')$).

| CF-HA $\mathcal{A}'$ | TA $\mathcal{A}$ |
|---|---|
| $N := N_1 N_2$ | $a(q_{N_1}, q_{N_2}) \to q_N$ |
| $I := N_1 N_2$ | $a(q_{N_1}, q_{N_2}) \to q$ |
| $g(q_1 \ldots q_k) \to q$ | $g(q_1, \ldots, q_k) \to q$ |

# Closure Results (1)

J Rusinowitch 2008 RTA

1. inverse CF HRS: rules $\ell \to a(x)$, $x \in vars(\ell)$

$$\mathsf{users}\big(\mathsf{user}(\mathsf{id}(y), y_1), \mathsf{user}(\mathsf{id}(y), y_2), x\big) \to \mathsf{users}(x)$$

preserve HA

exponential construction (needs determinization)

The following rules are not preserving HA

- collapsing rules $g(a, x, b) \to x$
- linear & flat rules $g(x) \to g(a, x, b)$
- linear & flat rules $g(x, q, a, y) \to g(x, b, q', y)$

## ⟨Closure Results (1)⟩

encoding of HRS into TRS loose flatness.
e.g. $f(x) \to f(b\,x\,e)$ encoded into $f(x) \to f(a(b, a(x, e)))$, not right-flat ($a$ associative).

and we don't have TA construction for these kinds of TRS.

# Closure Results (2)

J Rusinowitch 2008 RTA

1. restricted CF HRS: rules $a(x) \to r$, $r$ linear, $x$ at depth $\leqslant 1$ in $r$

$$a(x) \to b(x), \quad a(x) \to a(\mathsf{card}(\mathsf{name}), x),$$

preserve CF-HA

polynomial construction (completion of CF grammars)

The following (CF) rules are not preserving CF-HA

- non-shallow rules $a(x) \to b(a(x, e))$
- non-linear linear CF rules $g(x) \to g(x, x)$

## ⟨Closure Results (2)⟩

- $a(x) \to b(a(x,e))$ produces (starting from a constant $c$)

$$\underbrace{b(\ldots b}_{n}(a(c,e^n)))$$

- $g(x) \to g(x,x)$ generates $g\left(a^{2^k}\right)$

# XQuery Update Facility (XQUF)

[W3C recommandation 2011]

extension of XQuery with XML update primitives

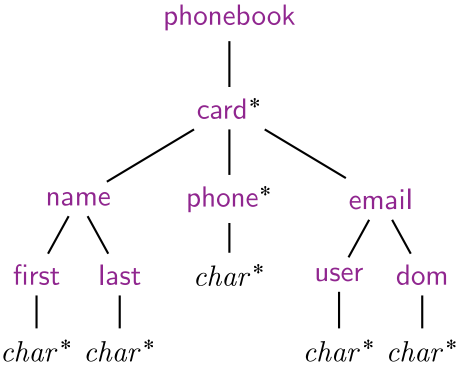
- [Fundulaki Maneth 2007 SACMAT] model XACU
- [Bravo Cheney Fundulaki 2008 EDBT] synthesis of schema, verification tool ACCoN
- [Gardner et al 08 PODS] local Hoare reasoning about W3C DOM update library (Context Logic).
- [Benedikt Cheney 2009 DBLP] formal model, op. semantics
- J Rusinowith 2010 PPDP
- [Boneva et al 2011 ICDT] translation of view updates

## ⟨XQuery Update Facility (XQUF)⟩

- W3C XQuery UF: content nodes (for insertion, replacement) are specified by positions within the tree in input (using XPath expressions)
- approximated by states of a tree automaton (equiv. type names of regular expression types)
- = parameters in rewrite rules
- (B C 09) the problem of synthesizing an output schema describing the result of an update applied to a given input schema. They show how to infer safe over-approximations for the results of both queries and updates
- [Boneva et al 2011 ICDT]: given an XML view definition and a user defined view update program, find a source update program that translates the view update without side effects on the view. update programs are compatible with the XQuery Update Facility (XQUF) snapshot semantics, a fragment of XQUF is expressible.
- PPDP 10:
  - formal model: parameterized rewrite rules
  - forward/backward type inference, typechecking, reachability verification regular tree model checking
  - verification of access control policies

# DTD and HA

## DTD



## Hedge Automaton

$$
\begin{array}{rcl}
\mathsf{phonebook}(p_{\mathsf{c}}{}^{*}) & \rightarrow & p_{\mathsf{b}} \\
\mathsf{card}(p_{\mathsf{n}}\ p_{\mathsf{h}}{}^{*}\ p_{\mathsf{m}}) & \rightarrow & p_{\mathsf{c}} \\
\mathsf{name}(p_{\mathsf{f}}\ p_{\mathsf{l}}) & \rightarrow & p_{\mathsf{n}} \\
\mathsf{first}(p^{*}) & \rightarrow & p_{\mathsf{f}} \\
\mathsf{last}(p^{*}) & \rightarrow & p_{\mathsf{l}} \\
\mathsf{phone}(p^{*}) & \rightarrow & p_{\mathsf{h}} \\
\mathsf{email}(p_{\mathsf{u}}\ p_{\mathsf{d}}) & \rightarrow & p_{\mathsf{m}} \\
\mathsf{user}(p^{*}) & \rightarrow & p_{\mathsf{u}} \\
\mathsf{dom}(p^{*}) & \rightarrow & p_{\mathsf{d}} \\
\mathsf{a} & \rightarrow & p \\
\mathsf{b} & \rightarrow & p \\
\vdots & &
\end{array}
$$

# XQUF Primitive Rename

"replace a label addressbook by phonebook"

$$\text{addressbook}(x) \rightarrow \text{phonebook}(x)$$

- the rule can be applied to any node labeled by addressbook
- the variable $x$ represents an hedge

# XQUF Primitive Insert First

"insert a tree of type $p_{\mathsf{c}}$ (card) as the first children of phonebook"

$$\mathsf{phonebook}(x) \to \mathsf{phonebook}(p_{\mathsf{c}}, x)$$

- $p_{\mathsf{c}}$ is a state of a given HA $\mathcal{A}$
- it stands for an arbitrary tree in $\mathcal{L}(\mathcal{A}, p_{\mathsf{c}})$
- this parametrized rule represents an infinity of rules.
  see also [Gilleron 91 STACS], [Löding 02 STACS]

# XQUF Primitive Insert Into

"insert a tree of type $p_{\mathsf{c}}$ as an arbitrary children of phonebook"

$$\mathsf{phonebook}(x, y) \rightarrow \mathsf{phonebook}(x, p_{\mathsf{c}}, y)$$

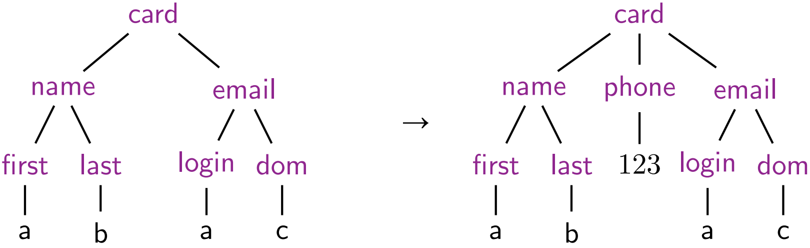
- each of the variables $x$ and $y$ represents an arbitrary hedge

# XQUF Primitive Insert After

"insert a tree of type $p_{\mathsf{h}}$ (phone) as sibling following name"

$$\mathsf{name}(x) \to \mathsf{name}(x), p_{\mathsf{h}}$$

- the right hand side of this rule is an hedge of length 2 (not a tree)



# XQUF Primitive Replace

"replace a subtree (headed by) card by sequence of $n$ trees of respective types $p_1$,. . . , $p_n$"

$$\text{address}(x) \to p_1, \ldots, p_n$$

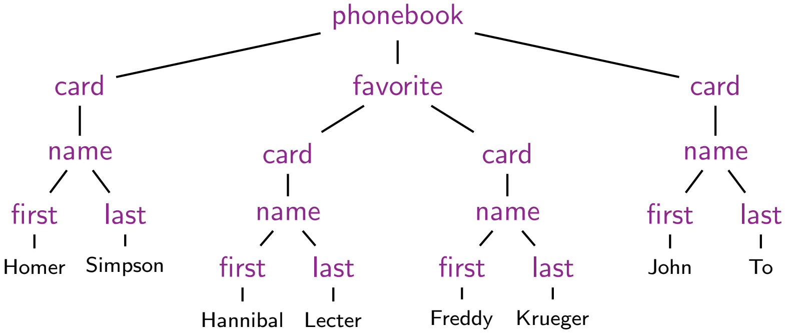
# XQUF Primitive Delete

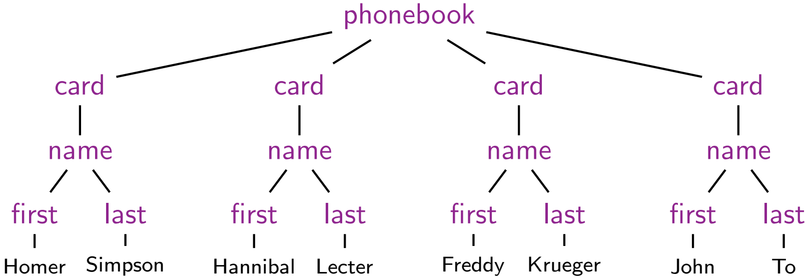
"delete a whole subtree headed by card"

$$\text{card}(x) \to \varepsilon$$

- $\varepsilon$ is the empty sequence of trees (empty hedge)

# Delete single node (not a XQUF Primitive)





# Extended XQUF Primitive Delete Single Node

"delete a single node labeled by favorite"

$$\text{favorite}(x) \to x$$

- the trees in the sequence of children $x$ are moved up to the position of the deleted node.
- *collapsing rule*
- useful for constructing security views of documents

# Forward and Backward Closure of XQUF Primitives

J Rusinowith 2010 PPDP

| | | | |
|---|---|---|---|
| $a(x)$ | $\rightarrow$ | $b(x)$ | REN |
| $a(x)$ | $\rightarrow$ | $a(p, x)$ | $\text{INS}_{\text{first}}$ |
| $a(x)$ | $\rightarrow$ | $a(x, p)$ | $\text{INS}_{\text{last}}$ |
| $a(x, y)$ | $\rightarrow$ | $a(x, p, y)$ | $\text{INS}_{\text{into}}$ |
| $a(x)$ | $\rightarrow$ | $p_1$ | $\text{RPL}_1$ |
| $a(x)$ | $\rightarrow$ | $()$ | DEL |

preserve HA

| | | | |
|---|---|---|---|
| $a(x)$ | $\rightarrow$ | $p, a(x)$ | $\text{INS}_{\text{before}}$ |
| $a(x)$ | $\rightarrow$ | $a(x), p$ | $\text{INS}_{\text{after}}$ |
| $a(x)$ | $\rightarrow$ | $p_1, \ldots, p_n$ | RPL |
| $a(x)$ | $\rightarrow$ | $x$ | $\text{DEL}_{\text{s}}$ |

preserve CF-HA
polynomial construction

inverse-preserve HA
exponential construction

# Towards XQUF Typechecking

[Benedikt Cheney 2009 DBLP]

| | | | |
|---|---|---|---|
| $u$ | XQueryUpdate expression | $L_{\text{in}}$ | input type (HA) |
| $t$ | unranked tree | $\Omega$ | regexp of primitives |
| $\omega$ | sequence of primitives | $L_{\text{out}}$ | output type (HA) |

$$\boxed{\begin{array}{lllllll} u,\, t & \mapsto & \omega & \mapsto & \omega' & \mapsto & t \xrightarrow{\omega'} t' \\ \\ u,\, L_{\text{in}} & \mapsto & \Omega & \mapsto & \Omega' & \mapsto & post^{\Omega}(L_{\text{in}}) \subseteq L_{\text{out}} \end{array}}$$

controlled rewriting

- $post^{\Omega}(L_{\text{in}})$ see matrix grammars [Dassow et al 1997 handbook FL]
- selection of position of application of primitives

# Rule Based Access Control Policies

## Access Control Policy (ACP)

$\mathcal{R}_+$ authorized operations of eXQUF (HRS)

$\mathcal{R}_-$ forbidden operations of eXQUF (HRS)

example

$$\mathcal{R}_+ = \left\{ \begin{array}{rcl} \mathsf{addressbook}(x) & \to & \mathsf{addressbook}(p_{\mathsf{c}}, x) \\ \mathsf{card}(x) & \to & \varepsilon \end{array} \right\}$$

- user can insert card with name, delete card

$$\mathcal{R}_- = \{\mathsf{name}(x) \to p_{\mathsf{n}}\}$$

- user cannot change a name

# Inconsistency

### Inconsistency

An ACP $\langle \mathcal{R}_+, \mathcal{R}_- \rangle$ is inconsistent
if there exists $t$, $t'$ such that $t \xrightarrow[\mathcal{R}_-]{} t'$ and $t \xrightarrow[\mathcal{R}_+]{*} t'$.

example: changing name in a card is simulated by deleting and then inserting.

[Fundulaki Maneth 2007 SACMAT] [Moore 11 LATA]
Inconsistency is undecidable for XQUF

# Local Inconsistency

### Local Inconsistency

An ACP $\langle \mathcal{R}_+, \mathcal{R}_- \rangle$ is locally inconsistent for $t$ if there exists $u$ such that $t \xrightarrow[\mathcal{R}_-]{} u$ and $t \xrightarrow[\mathcal{R}_+]{*} u$.

### J Rusinowitch 2010 PPDP

Local inconsistency is decidable in PTIME for eXQUF

- using the forward closure construction

## ⟨Local Inconsistency⟩

- compute a HA recognizing $L_- = \{u \mid t \xrightarrow[\mathcal{R}_-]{} u\}$
- compute a CF-HA recognizing $L_+ = \mathcal{R}_+^*\big(\{t\}\big)$
- check that $L_- \cap L_+ \neq \emptyset$.

# Extended Tree Automata Models

static properties: composition and decision (constraint solving)

Tree automata
with
local constraints

FO Th. Proving

Tree automata
with
global constraints

XML integrity
constraints

Horn Clauses
with
equality

dynamic properties: forward closure (regular model checking)

Ranked
tree rewriting

Rewriting
strategies

Unranked
tree rewriting

XML updates
XML r/w ACP

# Perspective 1: Generalizing Local Constraints

tree automata with local constraints other than equality

- constraints of equality modulo equational theories
  forward closure of constrained tree automata languages
- structural equality, equal depth:
  verification of algorithms on balanced structures (binary search trees, powerlists...)
- reduction ordering and other symbolic constraints
  automated induction on complex data structures
  languages of normal forms of constrained TRS
- hierarchical theories and constraints
  separated signatures for constraints (base, *e.g.* arithmetic) and input tree
  decision procedures for hierarchical first-order theorem proving

## ⟨Perspective 1: Generalizing Local Constraints⟩

- a is a well-founded ordering on terms, stable under application of substitutions and contexts, assumed total on ground terms
- ordering: for transforming a non terminating TRS into an equivalent orientable theory (containing rules with ordering constraints)
- inductive theorems: FO conjectures valid in Herbrand structures (term algebra, and modulo an equational theory for complex structure, e.g. sets, ordered lists)
- NF automata are good representation of these model and can serve as induction scheme for automated induction, as well as for decision procedure in ”proof by consistency”

# ⟨Perspective 1: Generalizing Local Constraints⟩

- hierarchical FO theories: confined to conservative extensions of the base models
- base signature $(\Sigma, \Omega)$, extended into $\Sigma \subset \Sigma'$, $\Omega \subset \Omega'$
- hierarchical rewrite system: $c \| \ell \to r$ where constraint $c$ over terms of $\mathcal{T}(\Sigma, \mathcal{X}_\Omega)$ and $\ell, r \in \mathcal{T}(\Sigma' \setminus \Sigma, \mathcal{X}_{\Omega'})$
- ex.: sorted lists
- CF TA: with constraints over separated signature
- Bachmair Ganzinger Waldmann 1994
- superposition calculus combine with specialized prover for constraints refutation (for the primitive base theory) is refutationaly complete, under the assumption of sufficiently completeness of the theory with respect to simple instances.
- modular combination of a superposition-based theorem prover with an arbitrary refutational prover for the primitive base theory, whose axiomatic representation in some logic may remain hidden.
- decision of sufficient completeness (sufficient condition for refutational completeness of above combined procedure)
- restrictions on base theory: congruence of finite index (compatible with constraints) → pumping in equivalence classes for emptiness decision

# Unified Approach: Tree Automata as Horn Clauses

...

# Perspective 2: Tools for XML Reasoning Tasks

- TAGC with key constraints

$$\forall x, y\; p(x) \wedge p(y) \wedge x \neq y \Rightarrow t|_x \neq t|_y \quad (x, y \in \text{positions})$$

- TAGC with inclusion constraints

$$\begin{array}{lcll} \forall x\, \exists y\; p(x) & \Rightarrow & (q(y) \wedge t|_x = t|_y) & (x, y \in \text{positions}) \\ \forall u\, \exists v\; p(u) & \Rightarrow & (q(v) \wedge u = v & (v, u \in \text{subtrees}) \end{array}$$

- TAGC with constraints in monadic FO over $q(y)$ and $x = y$ interpretation in the domain of subtrees (*see* automata on DAGs)

## ⟨Perspective 2: Tools for XML Reasoning Tasks⟩

- domain of subtrees = domain of DAGS
- keys cannot be expressed in this model

# Perspective 2: Combining Local/Global Constraints

ranked trees

- combination of TAGC$[\approx, \not\approx]$ with local $=$ and $\neq$ constraints between siblings à la [Bogaert Tison 1992 STACS]

unranked ordered trees

- unranked tree automata with local sibling constraints UTASC of [Löding Wong 2007 ICALP], [ 2009 FSTTCS]

$$\forall x, y\ \phi(x,y) \Rightarrow t|_x = t|_y \qquad \phi \text{ MSO over } q(x), S_{\rightarrow}(x,y)$$
$$\forall x, y\ \phi(x,y) \Rightarrow t|_x \neq t|_y$$

- combination of TAGC$[\approx, \not\approx]$ with UTASC?
- more generally: global monadic second order constraints

## ⟨Perspective 2: Combining Local/Global Constraints⟩

- UTASC: the interpretation domain is set of sibling positions (below current computation position)

# Perspective 2': Data Trees

unranked ordered trees labeled by $\Sigma \times \mathbb{N}$
EMSO$^2[+1, \sim]$ decidable [Bojańczyk et al 2009 JACM]

- encoding of data values as trees
  data equality $\sim$ expressed by subtree equality $\approx$
  - EMSO$^2[+1, \sim]$ and TAGC$[\approx, \not\approx]$ uncomparable
  - restricting application of $\approx$, $\not\approx$ by TAGC (PTIME)
- rigid tree automata J Klay Vacher 2009 LATA, 2011 IC
  and register tree automata [Jurdzinski Lazic 2007 LICS]
  Horn clauses with accumulating parameters
- TAGC computing on data trees: $\approx$ interpreted on data values
  emptiness TAGC$[\approx, \not\approx, |.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$ and TAGC$[|.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$?
  integer programming techniques, [David et al. 2011 ICDT]
  key constraint: $|q| = \|q\|$
- unranked/data tree rewriting

## ⟨Perspective 2': Data Trees⟩

- application TAGC: verification of consistency of combination of typing constraints (the HA) and integrity constraints (the global constraint)
- compare subtrees stronger than compare values (cf. H0 (i) and (iii) semi-srtuctured data: data is also in the structure
- restricted classes with better complexity: example data only in leaves
- EMSO$^2$ not TAGC: inclusion constraints $\forall x \exists y\ a(x) \Rightarrow (b(y) \wedge x \sim y)$.
- TAGC not EMSO$^2$ : subtrees equality
- (David 2011 ICDT)
- linear data constraints $|a|$ and $\|a\|$
- key constraint $|q| = \|q\|$
- set constraints for set of value under $a$
- emptiness NP with integer linear programming techniques