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. 07 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: 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:

$$\begin{tikzcd}
\mathsf{state}(0,0) \arrow[r, bend left, "1"] \arrow[d, bend right, "2"'] & \mathsf{state}\big(0,s(0)\big) \arrow[l, bend left, "3"] \\
\mathsf{state}\big(s(0),0\big) \arrow[u, bend right, "4"'] \arrow[d, bend right, "2"'] & \\
\mathsf{state}\big(s(s(0)),0\big) \arrow[u, bend right, "4"'] & \\
\vdots &
\end{tikzcd}$$

Concurrent readers/writers: reachable configurations

$$
\begin{array}{llll}
1. & \mathsf{state}(0,0) \to \mathsf{state}(0,s(0)) & 3. & \mathsf{state}(r,s(w)) \to \mathsf{state}(r,w) \\
2. & \mathsf{state}(r,0) \to \mathsf{state}(s(r),0) & 4. & \mathsf{state}(s(r),w) \to \mathsf{state}(r,w)
\end{array}
$$

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:

$$\begin{tikzcd}
\mathsf{state}(0,0) \arrow[r, bend left, "1"] \arrow[d, bend right, "2"'] & \mathsf{state}\big(0,s(0)\big) \arrow[l, bend left, "3"] \\
\mathsf{state}\big(s(0),0\big) \arrow[u, bend right, "4"'] \arrow[d, bend right, "2"'] & \\
\mathsf{state}\big(s(s(0)),0\big) \arrow[u, bend right, "4"'] & \\
\vdots &
\end{tikzcd}$$

tree automaton:

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

System Timbuk [Genet Tong 04 JAR]

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

Concurrent readers/writers: verification

Properties expected:

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

Set of excluded configurations = union of

$$E_1 = \big\{\mathsf{state}\big((q_1 \mid q_2), (q_1 \mid q_2)\big) \to e_1\big\}$$
$$E_2 = \big\{\mathsf{state}\big((q_0 \mid q_1 \mid q_2), q_2\big) \to e_2\big\}$$

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

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

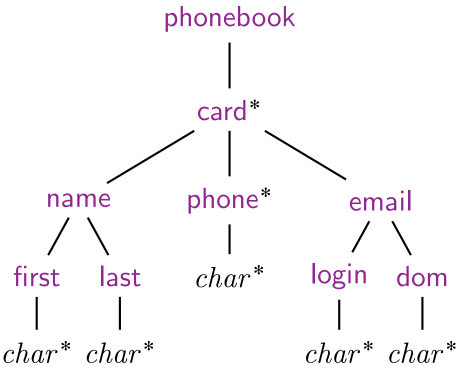
two files, configurations of the form: $\mathsf{state}(x_1, y_1, x_2, y_2)$

- 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 03 JCSS]

forward closure
(T: tree transformation)

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

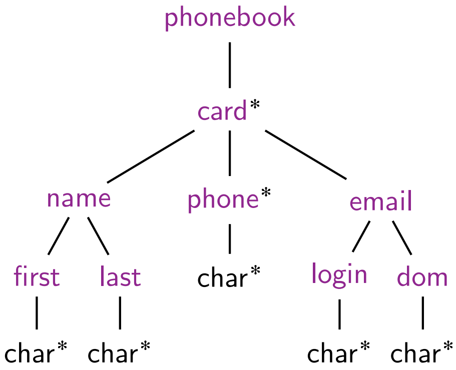
decision procedures

backward closure

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

composition (Boolean closure)

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

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

Unranked Ordered Trees & Hedges

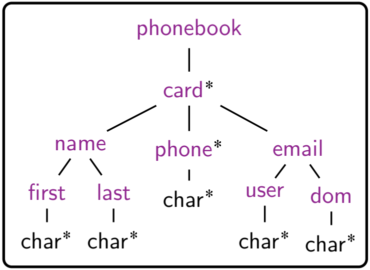
Σ unranked alphabet

$$\begin{array}{rcl} \text{hedge} & = & \text{finite sequence of unranked trees (possibly } \varepsilon\text{)} \\ \text{unranked tree} & = & \text{variable} \\ & & a(\text{hedge}) \text{ with } a \in \Sigma \end{array}$$

Unranked Tree Automata: Definition

A hedge automaton (HA [Murata 00])
is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

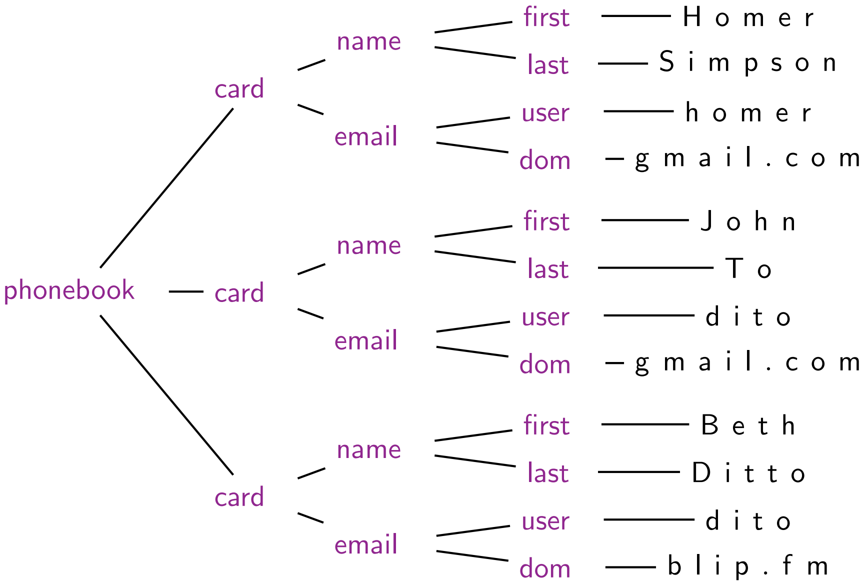
- Σ is an (unranked) alphabet
- Q is a finite set of states
- $F \subset Q$ is the subset of final states
- Δ is a set of transitions $a(L) \rightarrow q$ where $L \subseteq Q^*$ is regular



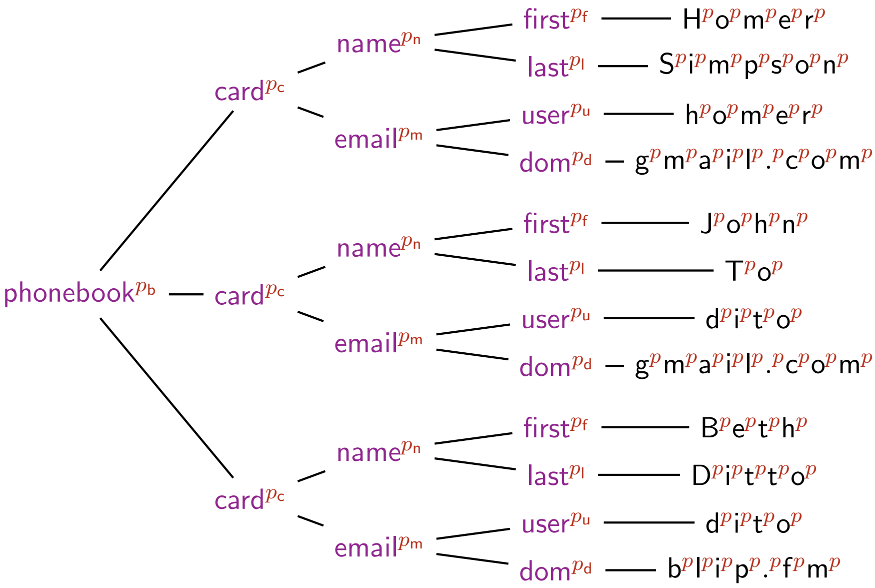
$$
\begin{array}{rcl}
\mathsf{phonebook}(p_{\mathsf{c}}{}^*) & \rightarrow & p_{\mathsf{b}} \\
\mathsf{card}(p_{\mathsf{n}}\ p_{\mathsf{h}}{}^*\ p_{\mathsf{m}}) & \rightarrow & p_{\mathsf{c}} \\
\mathsf{name}(p_{\mathsf{f}}\ p_{\mathsf{l}}) & \rightarrow & p_{\mathsf{n}} \\
\mathsf{first}(p^*) & \rightarrow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

Equivalent to ranked tree automata via binary encodings

Hedge Automaton Run

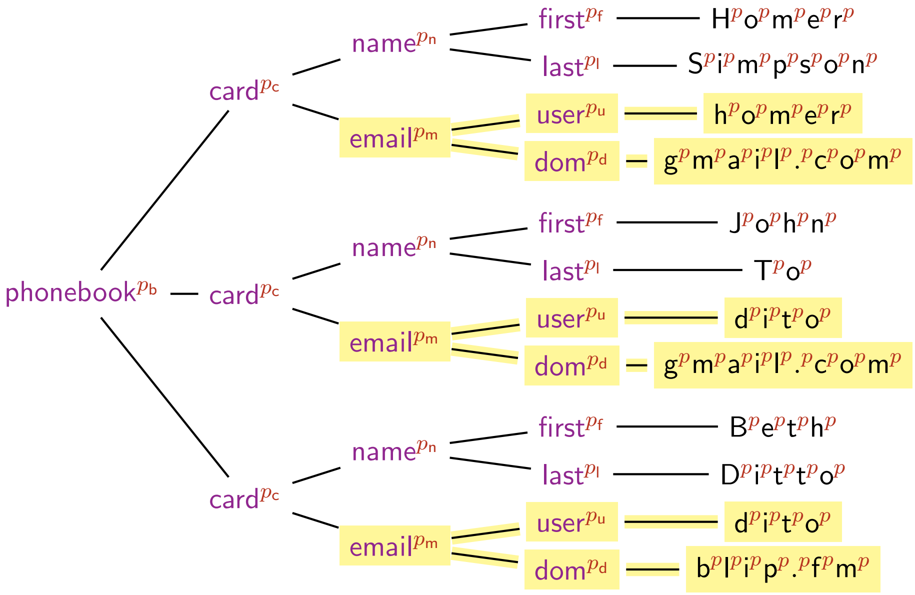


Hedge Automaton Run



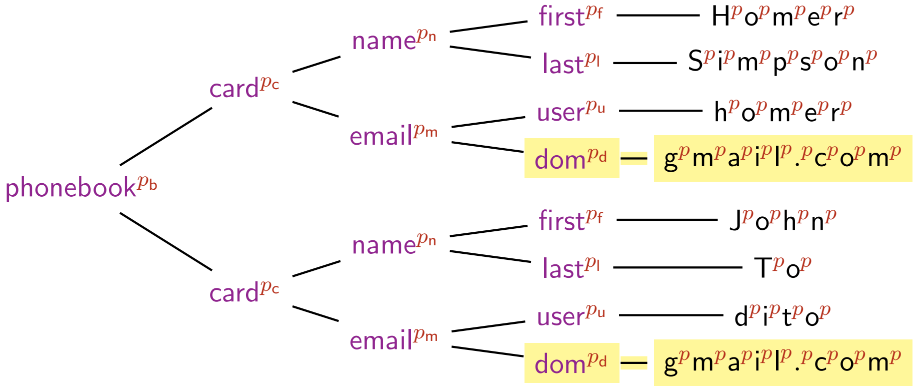
Key Constraint

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



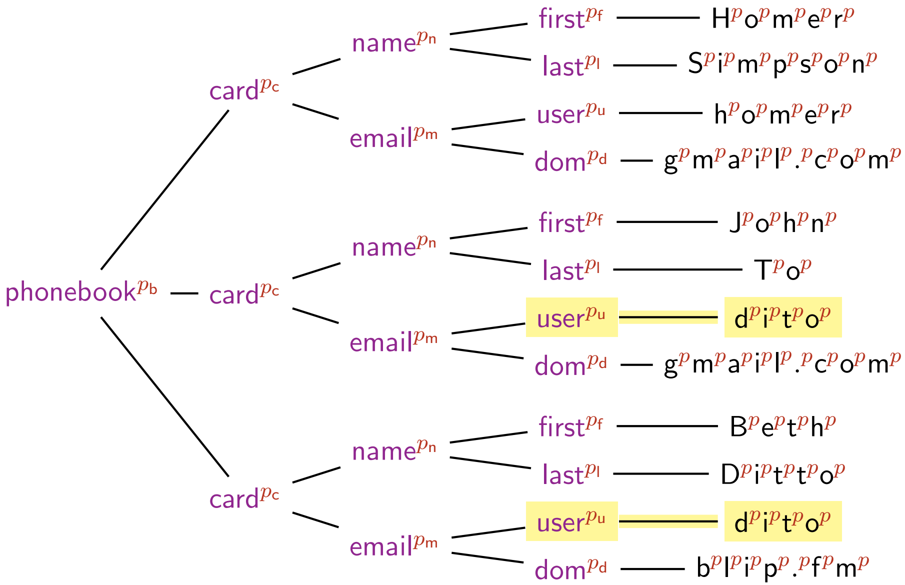
Global Equality Constraint

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



Negation

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



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 Δ,
successful if $r(root) \in F$
language: $\mathcal{L}(\mathcal{A}) = \{t \mid \exists r \text{ successful run of } \mathcal{A} \text{ on } t,\ \langle t, r \rangle \models C\}$

$$\langle t, r \rangle \models q_1 \approx q_2 \text{ iff } \forall x, y \in dom(t)\ q_1(x) \wedge q_2(y) \wedge x \neq y \Rightarrow t|_x = t|_y$$
$$\langle t, r \rangle \models q_1 \not\approx q_2 \text{ iff } \forall x, y \in dom(t)\ q_1(x) \wedge q_2(y) \wedge x \neq y \Rightarrow t|_x \neq t|_y$$

TAGED

The original model [Filiot et al 07 CSL], [Filiot et al 08 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
- universality, inclusion undecidable
- 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

Emptiness Decision TAGC

Godoy et al 10 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.

Decidable Extensions and Logic

Godoy et al 10 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 92 STACS]
- TAGC[$\approx, \not\approx$] where $\approx$ and $\not\approx$ are interpreted modulo flat equational theories
- TAGC[$\approx, \not\approx$] $+$ linear arithmetic constraints

decidability of EMSO extended with predicates on sets suitable to express $\approx$, $\not\approx$ and arithmetic constraints

TAGED and DAG Automata

DA: tree automata computing on DAGs representing ranked trees
emptiness is NP-complete for DA [Charatonik 99]

[Vacher 10 PhD]

- positive TAGC[$\not\approx_{\mathsf{irr}}$] $\equiv$ DA (exponential blowup for $\rightarrow$)
- TAGED $\equiv$ 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)

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

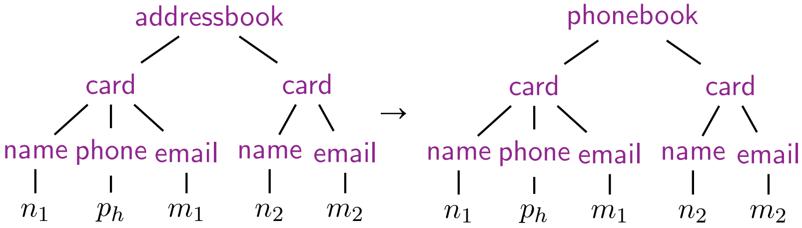
XML updates
XML r/w ACP

Unranked Ordered Tree Rewriting Systems (HRS)

[Löding Spelten 07 MFCS], [Touili 07 VECOS]

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

- 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)

XQuery Update Facility (XQUF)

[W3C recommandation 2011]

extension of XQuery with XML update primitives

- [Fundulaki Maneth 07 SACMAT] model XACU
- [Bravo Cheney Fundulaki 08 EDBT] synthesis of schema, verification tool ACCoN
- [Gardner et al 08 PODS] local Hoare reasoning about W3C DOM update library (Context Logic).
- [Benedikt Cheney 09 DBLP] formal model, op. semantics
- [Boneva et al 11 ICDT] translation of view updates
- J Rusinowith 10 PPDP
  - model of update primitives as parametrized rewrite rules
  - forward/backward closure

XQUF Primitive Insert First

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

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

- p_{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_{c} as an arbitrary children of phonebook"

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

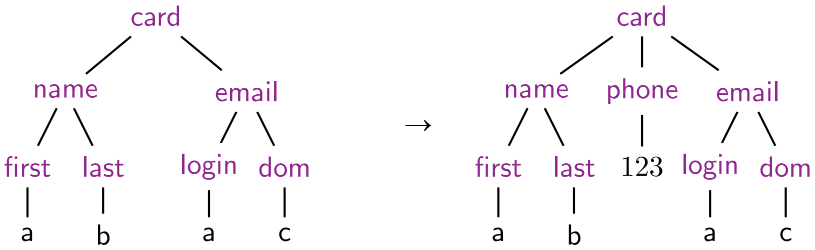
- each of the variables x and y represents an arbitrary hedge

XQUF Primitive Insert After

"insert a tree of type p_{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"

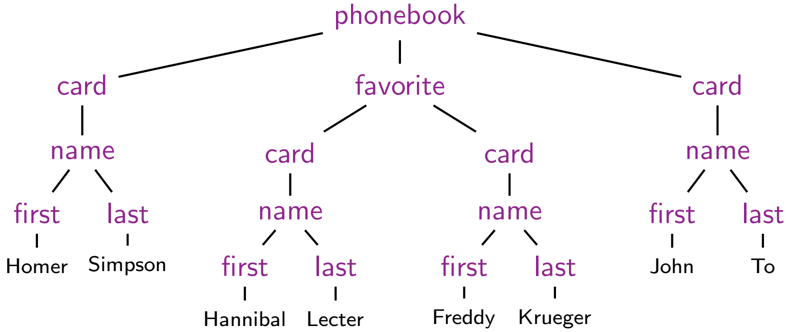
$$\text{address}(x) \to p_1, \dots, p_n$$

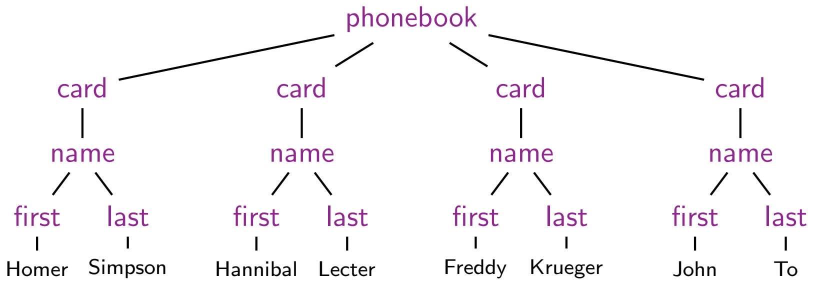
XQUF Primitive Delete

case $n = 0$
"delete a whole subtree headed by card"

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

Delete single node (not a XQUF Primitive)





Primitive Delete Single Node

"delete a single node labeled by favorite"

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

- 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

XQUF Primitives: Summary

$$
\begin{array}{rcllrcll}
a(x) & \rightarrow & b(x) & \mathsf{REN} & & & & \\
a(x) & \rightarrow & a(p, x) & \mathsf{INS}_{\mathsf{first}} & a(x) & \rightarrow & p, a(x) & \mathsf{INS}_{\mathsf{before}} \\
a(x) & \rightarrow & a(x, p) & \mathsf{INS}_{\mathsf{last}} & a(x) & \rightarrow & a(x), p & \mathsf{INS}_{\mathsf{after}} \\
a(x, y) & \rightarrow & a(x, p, y) & \mathsf{INS}_{\mathsf{into}} & & & & \\
a(x) & \rightarrow & p_1 & \mathsf{RPL}_1 & a(x) & \rightarrow & p_1, \ldots, p_n & \mathsf{RPL} \\
a(x) & \rightarrow & () & \mathsf{DEL} & a(x) & \rightarrow & x & \mathsf{DEL}_{\mathsf{s}}
\end{array}
$$

Forward Closure of XQUF Primitives

...does not preserve HA languages

e.g. $a(x) \to p_b, p_a, p_c$ (RPL)

$$\{d(a)\} \xrightarrow{*} \{d\big(b^n, a, c^n\big)\}$$

e.g. $a(x) \to x$ ($\mathsf{DEL_s}$)

$$\{a(b, a(b, \ldots, c), c)\} \xrightarrow{*} \{a\big(b^n, a, c^n\big)\}$$

- an extension of HA is needed

HA and CF-HA

[de la Higuera PhD] [Ohsaki 01 CSL]

Variants of the hedge automata of [Murata 00]

A HA, resp. CF-HA is a tuple $\langle \Sigma, Q, F, \Delta \rangle$ where

- Σ is an (unranked) alphabet,
- Q is a finite set of states,
- $F \subset Q$ is the subset of final states,
- Δ is a set of transitions of the form $a(L) \rightarrow q$ where
  - $L \subseteq Q^*$ is regular
  - $L \subseteq Q^*$ is context-free

HA $\equiv$ ranked tree automata

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

Forward and Backward Closure of XQUF Primitives

J Rusinowith 10 PPDP

$a(x)$	$\rightarrow$	$b(x)$	REN
$a(x)$	$\rightarrow$	$a(p, x)$	$\mathsf{INS}_{\mathsf{first}}$
$a(x)$	$\rightarrow$	$a(x, p)$	$\mathsf{INS}_{\mathsf{last}}$
$a(x, y)$	$\rightarrow$	$a(x, p, y)$	$\mathsf{INS}_{\mathsf{into}}$
$a(x)$	$\rightarrow$	p_1	RPL_1
$a(x)$	$\rightarrow$	$()$	DEL

preserve HA

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

preserve CF-HA
polynomial construction

inverse-preserve HA
exponential construction

Other Closure Results

J Rusinowitch 08 RTA

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

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

   exponential construction (needs determinization)

   not linear & flat rules $g(x, q, a, y) \to g(x, b, q', y)$

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

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

   polynomial construction (completion of CF grammars)

   not non-linear linear CF rules $g(x) \to g(x, x)$

   not non-shallow rules $a(x) \to b(a(x, e))$

Verification of XQUF Queries and ACP

- forward closure of some primitives is a crude approximation of XQUF semantics
- [Benedikt Cheney 09 DBLP] better approximation by regularly controlled forward closure
- J Kojima Sakai 11 FroCoS: selection of positions for application of primitives
  controlled term rewriting
- J Rusinowitch 10 PPDP decision of local consistency of rule based access control policies for XQUF

  in PTIME, using the forward closure construction of CF-HA

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

Perspectives

1. Unification of Extended Tree Automata Formalisms

Presburger Automata
[Seidl et al 03 PODS]
[Dal Zilio Lugiez 06 AAE]

TAGC$[\approx, \not\approx]$

Tree Automata
with 1 Memory

Register
Tree Automata

Unified
formalism?

Tree Automata
with
local constraints

Data Tree

unranked trees
labeled by $\Sigma \times \mathbb{N}$

FO$^2[+1, \sim]$
[Bojańczyk et al 09 JACM]

Type Definitions
$+$ Integrity Constraints

2. Extension of Current Models (for application purposes)

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \dots, q_n(x_n) \Rightarrow q\big(a(x_1, \dots, x_n)\big)$$

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local sibling = constraints when $x_1, \ldots, x_n$ may have duplicates

[Bogaert Tison 92 STACS]

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local = modulo eq. theories J Rusinowitch Vigneron 08 JLAP

$$\begin{array}{rl} & \Rightarrow \ell = r \\ q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k & \Rightarrow q\big(a(x_1, \ldots, x_n)\big) \end{array}$$

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local = modulo eq. theories J Rusinowitch Vigneron 08 JLAP

$$\begin{array}{rl} & \Rightarrow \ell = r \\ q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k & \Rightarrow q\big(a(x_1, \ldots, x_n)\big) \end{array}$$

local term constraints [Reuss Seidl 10 LPAR]

$$\begin{array}{lll} q_1(x_1), \ldots, q_n(x_n), & x_{i_1} = s_1, \ldots, x_{i_k} = s_k, & \\ & x_{j_1} \neq t_1, \ldots, x_{j_l} \neq t_l & \Rightarrow q\big(a(x_1, \ldots, x_n)\big) \end{array}$$

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local = modulo eq. theories J Rusinowitch Vigneron 08 JLAP

$$\begin{aligned} & \Rightarrow \ell = r \\ q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \quad & \Rightarrow q\big(a(x_1, \ldots, x_n)\big) \end{aligned}$$

local term constraints [Reuss Seidl 10 LPAR]

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

automata with 1 memory Comon-Lundh J Perrin 08 LMCS

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

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local = modulo eq. theories J Rusinowitch Vigneron 08 JLAP

$$\begin{aligned} & \Rightarrow \ell = r \\ q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \quad & \Rightarrow q\big(a(x_1, \ldots, x_n)\big) \end{aligned}$$

local term constraints [Reuss Seidl 10 LPAR]

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

automata with 1 memory Comon-Lundh J Perrin 08 LMCS

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

rigid variables/accumulating parameters

[Affeldt Comon-Lundh 09 FPS] J Klay Vacher 09 LATA, " 11 IC

$$\begin{aligned} q_{\boldsymbol{u}}(y_i, y_1, \ldots, y_m) & \Rightarrow q_{\boldsymbol{u}'}(y_i, y_1, \ldots, y_m) \\ q_{\boldsymbol{u}}(x_1, \overline{\boldsymbol{y}}), \ldots, q_{\boldsymbol{u}}(x_n, \overline{\boldsymbol{y}}) & \Rightarrow q_{\boldsymbol{u}}\big(a(x_1, \ldots, x_n), \overline{\boldsymbol{y}}\big) \end{aligned}$$

Unified Approach: Tree Automata as Horn Clauses

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

$$q_1(x_1), \ldots, q_n(x_n) \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local sibling = constraints when $x_1, \ldots, x_n$ may have duplicates

local = modulo eq. theories J Rusinowitch Vigneron 08 JLAP

$$\Rightarrow \ell = r$$

$$q_1(x_1), \ldots, q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \quad \Rightarrow q\big(a(x_1, \ldots, x_n)\big)$$

local term constraints [Reuss Seidl 10 LPAR]

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

automata with 1 memory Comon-Lundh J Perrin 08 LMCS

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

rigid variables/accumulating parameters

[Affeldt Comon-Lundh 09 FPS] J Klay Vacher 09 LATA, " 11 IC

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

$$q_{\boldsymbol{u}}(x_1, \overline{\boldsymbol{y}}), \ldots, q_{\boldsymbol{u}}(x_n, \overline{\boldsymbol{y}}) \Rightarrow q_{\boldsymbol{u}}\big(a(x_1, \ldots, x_n), \overline{\boldsymbol{y}}\big)$$

Perspective 1: Combining Local/Global Constraints

ranked trees

- combination of TAGC$[\approx, \not\approx]$ with local $=$ and $\neq$ constraints between siblings à la [Bogaert Tison 92 STACS]

unranked ordered trees

- unranked tree automata with local sibling constraints UTASC of [Löding Wong 07 ICALP], [" 09 FSTTCS]

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

Perspective 2: Generalized Global Constraints

- TAGC can express key constraints $q \not\approx q$

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

- extension of TAGC for inclusion constraints

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

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

Perspective 2: Generalized Local Constraints

tree automata with local constraints

- equalities, disequalities: matching $\mathsf{state}(x, y_1, x, y_2)$
- reduction ordering and other symbolic constraints
  automated induction on complex data structures
  [Bouhoula Jouannaud 01 IC], Bouhoula J 08 IJCAR, Bouhoula J 07 FCS-ARSPA, Bouhoula J 11 JAL

$$\begin{array}{rrcl} & \mathsf{cons}(x, \mathsf{cons}(x, y)) & \to & \mathsf{cons}(x, y) \\ x_1 > x_2 \quad \| & \mathsf{cons}(x_1, \mathsf{cons}(x_2, y)) & \to & \mathsf{cons}(x_2, \mathsf{cons}(x_1, y)) \end{array}$$

- structural equality, equal depth:
  verification of algorithms on balanced structures (binary search trees, powerlists...)
- constraints of equality modulo equational theories
  forward closure of constrained tree automata languages

Merci!