Tree Automata with Constraints: a brief survey

Emmanuel Filiot, Florent Jacquemard, Sophie Tison

Tree Automata and Constraints: a brief survey

Emmanuel Filiot Florent Jacquemard Sophie Tison

[1]University of Paris 12, LACL

[2]INRIA Paris-Rocquencourt, Ircam

[3]University of Lille 1, LIFL

Dagstuhl Seminar 13192
Tree Transducers and Formal Methods

Tree Automata and non-linearity

Non-linearity phenomena are non captured by tree automata, e.g. :

- The set of ground instances of a non-linear term is not recognizable.
- The class of recognizable tree languages is not closed under non-linear homomorphisms.

How to enrich transitions by equality and disequality constraints ?

Two main ideas

- Add **local** tests, e.g. to capture the set of instances of $f(g(x), x)$
  Why: pattern matching, rewriting, representing finitely image by non-linear transformations of recognizable tree languages (type checking?), ...
  When: from 80's to now

- Add **global** tests, e.g. to capture "all the subterms rooted by f are different".
  Why: integrity constraints (XML), non linear query languages, ...
  When: more recently

Leitmotiv: keeping as far as possible good closure and decision properties.

Tree Automata with Local Constraints

Equality and Disequality Constraints

An **equality constraint** (resp. a **disequality constraint**) is a predicate on $T(\Sigma)$ written $\pi = \pi'$ (resp. $\pi \neq \pi'$) where $\pi, \pi' \in \{1, \ldots, k\}^*$.

Such a predicate is satisfied on a tree t, which we write $t \models \pi = \pi'$, if $\pi, \pi' \in \mathcal{P}ost$ and $\models t|_\pi = t|_{\pi'}$ (resp. $\pi \neq \pi'$ is satisfied on t if $\pi = \pi'$ is not satisfied on t).

Equality and Disequality Constraints

An **equality constraint** (resp. a **disequality constraint**) is a predicate on $T(\Sigma)$ written $\pi = \pi'$ (resp. $\pi \neq \pi'$) where $\pi, \pi' \in \{1, \ldots, k\}^*$.

Such a predicate is satisfied on a tree t, which we write $t \models \pi = \pi'$, if $\pi, \pi' \in \mathcal{P}ost$ and $\models t|_\pi = t|_{\pi'}$ (resp. $\pi \neq \pi'$ is satisfied on t if $\pi = \pi'$ is not satisfied on t).

These constraints are indeed unlabeled path constraints. More general disequality constraints have been defined by H. Seidl and A. Reuß(LPAR 2010, FOSSACS 2012).

Automata with Equality and Disequality Constraints

The general class

An **automaton with equality and disequality constraints** is a tuple (Q, Σ, Q_f, Δ) where

- Σ is a finite ranked alphabet
- Q is a finite set of states
- Q_f is a subset of Q of final states
- Δ is a set of transition rules of of the form:

$$f(q_1, \ldots, q_n) \xrightarrow{c} q$$

where $f \in \Sigma$, $q_1, \ldots, q_n, q \in Q$, and c is a Boolean combination of equality and disequality constraints.

Automata with Equality and Disequality Constraints

The general class

- The move relation $\rightarrow_{\mathcal{A}}$ is defined by:
  $t \rightarrow_{\mathcal{A}} t'$ if and only

$$\begin{array}{l} t = C[f(q_1(u_1), \ldots, q_n(u_n)], \\ t' = C[q(f(u_1, \ldots, u_n))] \\ f(q_1, \ldots, q_n) \xrightarrow{c} q \in \Delta, \\ f(u_1, \ldots, u_n) \models c. \end{array}$$

- $\xrightarrow{*}_{\mathcal{A}}$ is the reflexive and transitive closure of $\rightarrow_{\mathcal{A}}$.
- $\mathcal{A}$ **accepts** a tree $t \in T(\Sigma)$ if $t \xrightarrow{*}_{\mathcal{A}} q$ for some final state q.
- The **language accepted**, or **recognized**, is the set $L(\mathcal{A})$ of trees $t \in T(\Sigma)$ accepted by $\mathcal{A}$.

Automata with Equality and Disequality Constraints

The general class

- The move relation $\rightarrow_{\mathcal{A}}$ is defined by:
  $t \rightarrow_{\mathcal{A}} t'$ if and only

$$
\begin{array}{l}
t = C[f(q_1(u_1), \dots, q_n(u_n)], \\
t' = C[q(f(u_1, \dots, u_n))] \\
f(q_1, \dots, q_n) \xrightarrow{c} q \in \Delta, \\
f(u_1, \dots, u_n) \models c.
\end{array}
$$

- $\xrightarrow{*}_{\mathcal{A}}$ is the reflexive and transitive closure of $\rightarrow_{\mathcal{A}}$.
- $\mathcal{A}$ **accepts** a tree $t \in T(\Sigma)$ if $t \xrightarrow{*}_{\mathcal{A}} q$ for some final state q.
- The **language accepted**, or **recognized**, is the set $L(\mathcal{A})$ of trees $t \in T(\Sigma)$ accepted by $\mathcal{A}$.
- Class with only equalities defined in 1981 (Dauchet & Mongy)

Automata with Equality and Disequality Constraints

Example 1

- $Q = \{q\}$,
- $\Sigma = \{f, a\}$
- final states: $\{q\}$
- Δ consists of the following rules:

$$\begin{aligned} r_1: &\quad a \rightarrow q \\ r_2: &\quad f(q, q) \xrightarrow{1=2} q \end{aligned}$$

Automata with Equality and Disequality Constraints

Example 1

- $Q = \{q\}$,
- $\Sigma = \{f, a\}$
- final states: $\{q\}$
- Δ consists of the following rules:

$$\begin{aligned} r_1 : &\quad a \to q \\ r_2 : &\quad f(q,q) \xrightarrow{1=2} q \end{aligned}$$

The automaton recognizes the set of balanced trees.

Automata with Equality and Disequality Constraints

Example 2

- $Q = (\{q, q_g, q_{fin}\},$
- $\Sigma = \{f, g, a\}$
- final states: $\{q_{fin}\}$
- Δ consists of the following rules:

$$
\begin{array}{rc}
r_1: & a \rightarrow q \\
r_2: & f(q, q) \rightarrow q \\
r_3: & f(q_g, q) \xrightarrow{11=2} q_{fin} \\
r_4: & g(q) \rightarrow q_g \\
r_5: & g(q) \rightarrow q
\end{array}
$$

Automata with Equality and Disequality Constraints

Example 2

- $Q = (\{q, q_g, q_{fin}\},$
- $\Sigma = \{f, g, a\}$
- final states: $\{q_{fin}\}$
- Δ consists of the following rules:

$$\begin{array}{rc} r_1: & a \to q \\ r_2: & f(q,q) \to q \\ r_3: & f(q_g, q) \xrightarrow{11=2} q_{fin} \\ r_4: & g(q) \to q_g \\ r_5: & g(q) \to q \end{array}$$

The automaton recognizes the ground instances of $f(g(x), x)$.

Automata with Equality and Disequality Constraints

Example 3

- $Q = (\{q\},$
- $\Sigma = \{f, a\}$
- final states: $\{q\}$
- Δ consists of the following rules:

$$\begin{aligned} r_1 : &\quad a \to q_a \\ r_2 : &\quad f(q_a, q_a) \to q \\ r_3 : &\quad f(q, q) \xrightarrow{12=21} q \end{aligned}$$

Automata with Equality and Disequality Constraints

Example 3

- $Q = (\{q\},$
- $\Sigma = \{f, a\}$
- final states: $\{q\}$
- Δ consists of the following rules:

$$\begin{aligned} r_1 : & \quad a \to q_a \\ r_2 : & \quad f(q_a, q_a) \to q \\ r_3 : & \quad f(q, q) \xrightarrow{12=21} q \end{aligned}$$

The automaton recognizes "grids".

Automata with Equality and Disequality Constraints

Properties

- Can be determinized (and completed) (exponential blow up)

Automata with Equality and Disequality Constraints

Properties

- Can be determinized (and completed) (exponential blow up)
- Closure under boolean operations

Automata with Equality and Disequality Constraints

Properties

- Can be determinized (and completed) (exponential blow up)
- Closure under boolean operations
- Emptiness is undecidable for this class -e.g. use Post problem or previous encoding of grids -: undecidable even with only equalities between cousins.

Automata with Equality and Disequality Constraints

Properties

- Can be determinized (and completed) (exponential blow up)
- Closure under boolean operations
- Emptiness is undecidable for this class -e.g. use Post problem or previous encoding of grids -: undecidable even with only equalities between cousins.
- capture homomorphic images of recognizable languages (only equality constraints are needed) and images by bottom-up transducers.

Undecidability comes in some sense from superposition of (equality) tests. To get a decidable class, we can try to limit this superposition...

Automata with Constraints

Restrictions

- restrict the form of the tests
- limit the number of equality tests
- limit the superposition of equality tests

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.

$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.

$f(q_1, q_2, q_3) \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$:

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.
$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$: if the automaton is not deterministic, we have to check whether there is some tree leading both to q_1 and q_2: we can determinize

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.
$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$: if the automaton is not deterministic, we have to check whether there is some tree leading both to q_1 and q_2: we can determinize
  - $f(q, q) \rightarrow q, 1 \neq 2$:

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.
$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$: if the automaton is not deterministic, we have to check whether there is some tree leading both to q_1 and q_2: we can determinize
  - $f(q, q) \rightarrow q, 1 \neq 2$: We need to count, but we just have to count up to maximal arity.

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.

$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$: if the automaton is not deterministic, we have to check whether there is some tree leading both to q_1 and q_2: we can determinize
  - $f(q, q) \rightarrow q, 1 \neq 2$: We need to count, but we just have to count up to maximal arity.
    The unranked case: W. Karianto and C. Löding, ICALP 2007.

Automata with Constraints between brothers

Constraints are reduced to constraints between brothers (siblings)

Allowed constraints are $i = j$ or $i \neq j$.
$f(q_1, q_2, q_3) \ \rightarrow_{1=2,\ 1\neq 3}$ allowed, $f(q_1, q_2) \ \rightarrow_{1=21}$ forbidden.

- can be determinized (exponential blow up)
- stable under boolean operations.
- Emptiness $EXPTIME-$complete but polynomial in the deterministic case.
  Proof for "normal" TA: compute by fix-point the set of reachable states and check it contains a final state. Does it work here?
  - $f(q_1, q_2) \rightarrow q, 1 = 2$: if the automaton is not deterministic, we have to check whether there is some tree leading both to q_1 and q_2: we can determinize
  - $f(q, q) \rightarrow q, 1 \neq 2$: We need to count, but we just have to count up to maximal arity.
    The unranked case: W. Karianto and C. Löding, ICALP 2007.
  - Recognizability is decidable.

Reduction automata

Limit the number of equality tests along a path

A **reduction automaton** $\mathcal{A}$ is an automaton with equality and disequality constraints with an ordering on the states such that for each rule $f(q_1, \ldots, q_n) \xrightarrow{c} q$,

- q is maximal in $q, q_1, \ldots, q_n$
- q is a strict upper bound of $q_1, \ldots, q_n$ if c contains an equality constraint.

Idea: Bound the number of equalities tested along a path.
Defined/studied by Dauchet, Caron, Coquidé, Comon and Jacquemard (94 and following)

Reduction automata

An example

How to define the set of ground instances of $f(f(x,x),f(x,y))$?

Reduction automata

An example

How to define the set of ground instances of $f(f(x,x), f(x,y))$?

- $\Delta =$

$$
\begin{array}{ll}
a \rightarrow q & \\
f(q,q) \rightarrow q_f & f(q,q_f) \rightarrow q_f \\
f(q_f,q) \rightarrow q_f & f(q_f,q_f) \rightarrow q_f \\
f(q_f,q_f) \xrightarrow{11=12 \wedge 11=21} q_{fin} &
\end{array}
$$

Reduction automata

An example

How to define the set of ground instances of $f(f(x,x),f(x,y))$?

- $\Delta =$

$$
\begin{array}{ll}
a \to q & \\
f(q,q) \to q_f & f(q,q_f) \to q_f \\
f(q_f,q) \to q_f & f(q_f,q_f) \to q_f \\
f(q_f,q_f) \xrightarrow{11=12 \wedge 11=21} q_{fin} &
\end{array}
$$

- $q < q_f < q_{fin}$

Reduction automata

An example

How to define the set of ground instances of $f(f(x,x),f(x,y))$?

- $\Delta =$

$$
\begin{array}{ll}
a \rightarrow q & \\
f(q,q) \rightarrow q_f & f(q,q_f) \rightarrow q_f \\
f(q_f,q) \rightarrow q_f & f(q_f,q_f) \rightarrow q_f \\
f(q_f,q_f) \xrightarrow{11=12 \wedge 11=21} q_{fin} &
\end{array}
$$

- $q < q_f < q_{fin}$
- q_{fin} final state

Reduction automata

An example

How to define the set of ground instances of $f(f(x,x),f(x,y))$ with $\Sigma = \{a,f\}$.

Deterministic version:

$\Delta =$

$$
\begin{array}{ll}
a \rightarrow q & f(q,q) \rightarrow q_f \\
f(q,q_f) \rightarrow q_f & f(q_f,q) \rightarrow q_f \\
f(q_f,q_f) \xrightarrow{11=12 \wedge 11=21} q_{fin} & f(q_f,q_f) \xrightarrow{11\neq 12 \vee 11\neq 21} q_f
\end{array}
$$

Reduction automata

Properties

- closed under union and intersection

Reduction automata

Properties

- closed under union and intersection
- closed under complement for deterministic automata

Reduction automata

Properties

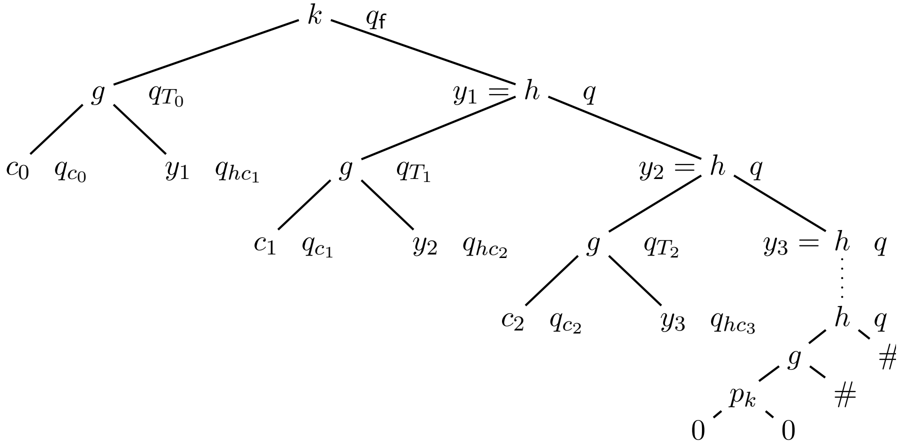
- closed under union and intersection
- closed under complement for deterministic automata
- emptiness decidable for **deterministic** automata (M. Dauchet and alt. 1993)

Reduction automata

Properties

- closed under union and intersection
- closed under complement for deterministic automata
- emptiness decidable for **deterministic** automata (M. Dauchet and alt. 1993)
- emptiness undecidable for non deterministic automata (F. Jacquemard and alt. 2008)

Deciding emptiness "needs" determinism



Automata with only one kind of tests

- $TA_{\neq}$: automata with only disequality tests
  - closed under union and intersection
  - emptiness

Automata with only one kind of tests

- $TA_{\neq}$: automata with only disequality tests
  - closed under union and intersection
  - emptiness decidable

Automata with only one kind of tests

- $TA_{\neq}$: automata with only disequality tests
  - closed under union and intersection
  - emptiness decidable in EXPTIME (Comon & Jacquemard, 97)
- $TA_{=}$: automata with only equality tests
  - closed under union and intersection
  - emptiness

Automata with only one kind of tests

- $TA_{\neq}$: automata with only disequality tests
  - closed under union and intersection
  - emptiness decidable in EXPTIME (Comon & Jacquemard, 97)
- $TA_{=}$: automata with only equality tests
  - closed under union and intersection
  - emptiness undecidable

Automata with only one kind of tests

- The complement of a $TA_=$ is a $TA_{\neq}$.

Automata with only one kind of tests

- The complement of a $TA_=$ is a $TA_{\neq}$.
- The complement of a $TA_{\neq}$ is a $TA_=$.

Sketch of the (effective) proof: a state reached by t in the automaton for the complement of L corresponds to the set of all states that t can't reach in the automaton for L. (folklore construction, formalized by Creus & alt. STOC 2010).

Automata with only one kind of tests

- The complement of a $TA_=$ is a $TA_{\neq}$.
- The complement of a $TA_{\neq}$ is a $TA_=$.

Sketch of the (effective) proof: a state reached by t in the automaton for the complement of L corresponds to the set of all states that t can't reach in the automaton for L. (folklore construction, formalized by Creus & alt. STOC 2010).

Universality is decidable for $TA_=$, undecidable for $TA_{\neq}$.

The homomorphism Problem

$Hom-$problem

Instance	A recognizable language R, an homomorphism H.
Answer	"yes" if and only $H(R)$ is recognizable

The homomorphism Problem

$Hom-$problem

Instance	A recognizable language R, an homomorphism H.
Answer	"yes" if and only $H(R)$ is recognizable

Is it decidable?

The homomorphism Problem

$Hom-$problem

Instance A recognizable language R, an homomorphism H.

Answer "yes" if and only $H(R)$ is recognizable

Is it decidable?

Remark: If H is shallow -i.e the images of symbols are shallow-,

The homomorphism Problem

$Hom-$problem

Instance A recognizable language R, an homomorphism H.

Answer “yes” if and only $H(R)$ is recognizable

Is it decidable?

Remark: If H is shallow -i.e the images of symbols are shallow-, $H(R)$ is recognizable by an automaton with tests between brothers. as recognizability is decidable for language recognizable by an automaton with tests between brothers: the homomorphism problem is decidable for shallow homomorphisms.

The homomorphism Problem

Proposition

The $HOM-$problem is decidable (Godoy & alt, STOC 2010) and EXPTIME-Complete (Creus & alt., LICS 2012)

- TA_{Hom}: equality test without superposition, no disequality tests: $t \xrightarrow{c} q$ where $t \in \mathcal{T}(\Sigma, Q)$, and c is a Boolean combination of equality constraints for positions of t labeled by states.
  - capture homomorphic images of recognizable tree languages
  - emptiness, finiteness, recognizability are decidable for this class.
- $TA_{Hom,\neq}$: equality and disequality tests, no superposition of equality tests : emptiness is decidable

Application to transducers?

- The image of a recognizable tree language by a bottom-up tree transducer can be represented finitely by a TA_{Hom}.
- E.g., we can decide whether $T_1(L_1) \subset T_2(L_2)$, L_1, L_2 recognizable tree languages, T_1, T_2 bottom-up tree transducers.

What about global constraints?

Tree Automata with Global Constraints

Local Constraints

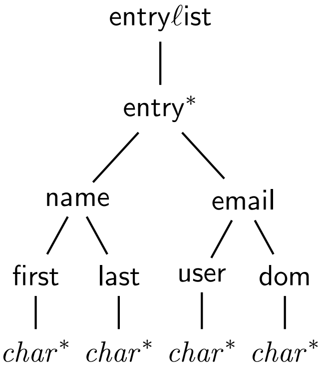
- non linear pattern matching
- non-linear transformation
- term rewriting, reachability analysis

Global Constraints

- tree structured data processing
- schema = type + integrity constraints

Type Definition for XML Data

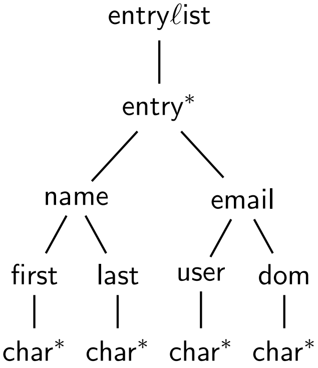
DTD



Tree Automaton

$$
\begin{array}{rcl}
\mathsf{entry}(p_{\mathsf{n}}, p_{\mathsf{m}}) & \rightarrow & p_{\mathsf{e}} \\
\mathsf{empty} & \rightarrow & p_{\mathsf{e}\ell} \\
\mathsf{entry}\ell\mathsf{ist}(p_{\mathsf{e}}, p_{\mathsf{e}\ell}) & \rightarrow & p_{\mathsf{e}\ell} \\
\hline
\mathsf{name}(p_{\mathsf{f}}\ p_{\mathsf{l}}) & \rightarrow & p_{\mathsf{n}} \\
\mathsf{first}(p) & \rightarrow & p_{\mathsf{f}} \\
\mathsf{last}(p) & \rightarrow & p_{\mathsf{l}} \\
\hline
\mathsf{email}(p_{\mathsf{u}}\ p_{\mathsf{d}}) & \rightarrow & p_{\mathsf{m}} \\
\mathsf{user}(p) & \rightarrow & p_{\mathsf{u}} \\
\mathsf{dom}(p) & \rightarrow & p_{\mathsf{d}} \\
\hline
\mathsf{a}(p) & \rightarrow & p \\
\vdots & & \\
\varepsilon & \rightarrow & p
\end{array}
$$

Integrity Constraint



- email is a key (ID)
  subtrees below email are pairwise distinct

Cannot be expressed with standard tree automata

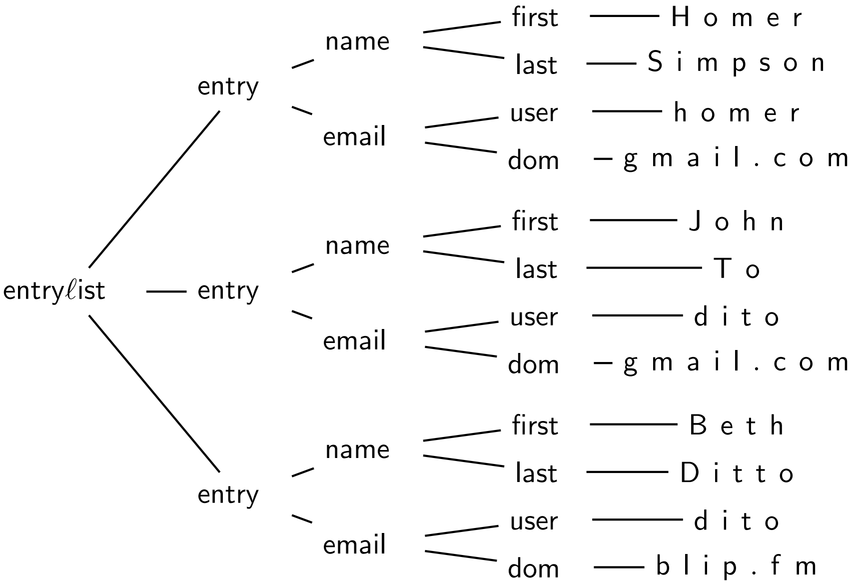
Tree Automata with Local Constraints

- equality and disequality test in transitions
- checked during computation

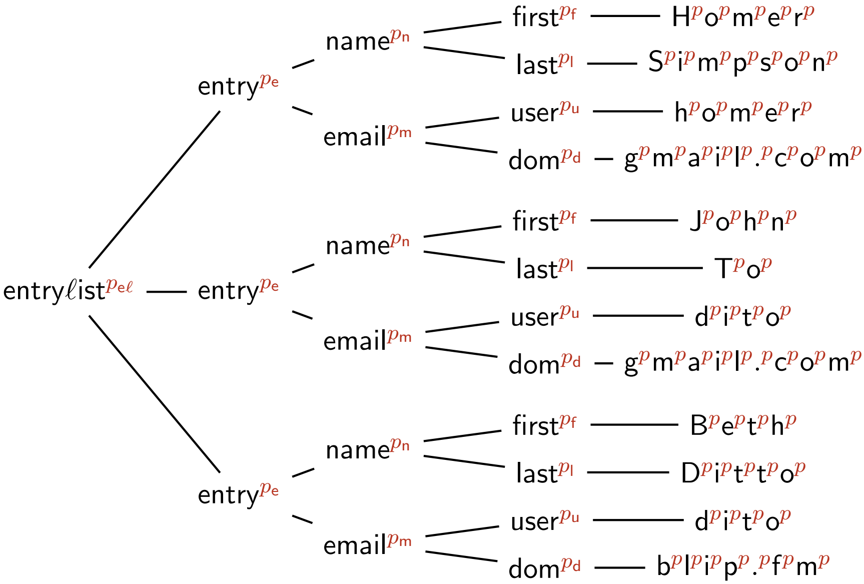
Tree Automata with Global Constraints

- one standard tree automaton + one separate constraint
- checked on run, after computation

Tree

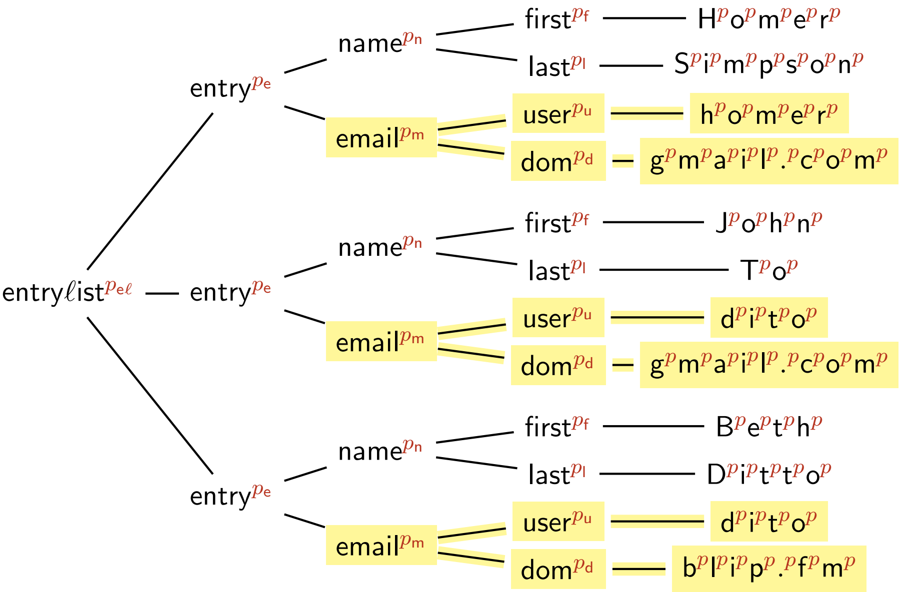

entryℓist
entry
entry
entry
name
email
name
email
name
email
first
last
user
dom
first
last
user
dom
first
last
user
dom
H o m e r
S i m p s o n
h o m e r
g m a i l . c o m
J o h n
T o
d i t o
g m a i l . c o m
B e t h
D i t t o
d i t o
b l i p . f m

Tree Automaton Run



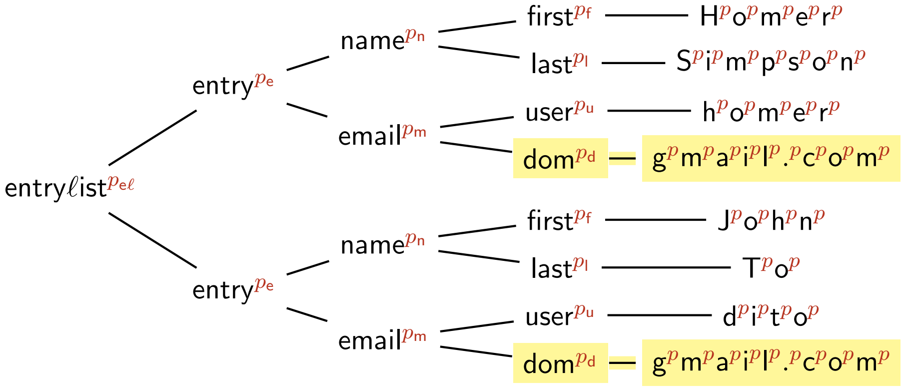
Key Constraint

email is a key: $\forall x, y\ p_{\mathsf{m}}(x) \wedge p_{\mathsf{m}}(y) \wedge x \neq y \Rightarrow t|_x \neq t|_y$



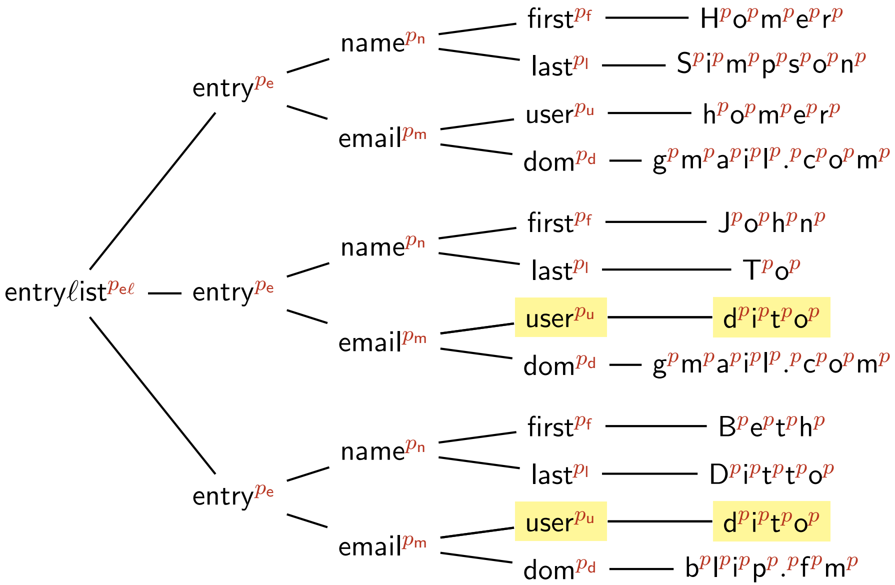
Global Equality Constraint

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



Negation

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



Tree Automata with Global Constraints

A TAGC $\mathcal{A}$ is given by

- a tree automaton $ta(\mathcal{A}) = \langle \Sigma, Q, F, \Delta \rangle$
- a constraint ϕ: FO formula interpreted on a term $t \in \mathcal{T}(\Sigma)$ and a run r of $ta(\mathcal{A})$ on t, build with predicates
  - $x = y$ (node equality)
  - $p(x)$ (the label of x in r is p)
  - $t|_x = t|_y$ (subterm equality)

A relabeling $r : dom(t) \to Q$ is a successful run of $\mathcal{A}$ on t iff

- r run of $ta(\mathcal{A})$ on t
- $r(root) \in F$
- $t, r \models \phi$.

Language $\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\Sigma) \mid \exists r$ successful run of A on $t\}$.

Constraints: Notations

type	formula
$p_1 \approx p_2$	$\forall x, y\ p_1(x) \wedge p_2(y) \Rightarrow t\|_x = t\|_y$
$p \approx_{\mathsf{ref}} p$	subcase with same states
$p_1 \approx_{\mathsf{irr}} p_2$	subcase with distinct states
$p_1 \not\approx p_2$	$\forall x, y\ p_1(x) \wedge p_2(y) \wedge x \neq y \Rightarrow t\|_x \neq t\|_y$
$p \not\approx_{\mathsf{ref}} p$	subcase with same states (key)
$p_1 \not\approx_{\mathsf{irr}} p_2$	subcase with distinct states

note that $\neg p_1 \approx p_2$ and $p_1 \not\approx p_2$ have different semantics.

notation $\mathsf{TAGC}[\tau_1, \ldots, \tau_n]$: the constraint is a Boolean combination of atomic constraints of types $\tau_1, \ldots, \tau_n$.

$\mathsf{TAGC}^{\wedge}[\tau_1, \ldots, \tau_n]$: when constraint is positive conjunctive.

Rest of the talk

some subclasses of $\mathsf{TAGC}[\approx, \not\approx]$ studied

TAGED	$= \mathsf{TAGC}^{\wedge}[\approx, \not\approx_{\mathsf{irr}}]$	[Filiot, Talbot, Tison 2007, 08]
RTA	$\equiv \mathsf{TAGC}^{\wedge}[\approx_{\mathsf{ref}}]$	[J Klay Vacher 2009, 11]
Dag Automata	$\equiv \mathsf{TAGC}^{\wedge}[\not\approx_{\mathsf{irr}}]$	[Charatonik 1999]
$\mathsf{TAGC}[\approx, \not\approx]$		[Godoy et al 2010]

decidable extensions

$\mathsf{TAGC}[\approx, \not\approx, \mathbb{N}]$	arithmetic	[Godoy et al 2010,12]
$\mathsf{TACB}[\approx, \not\approx, \mathbb{N}]$	with local equality tests	
	$+$ modulo	

open classes

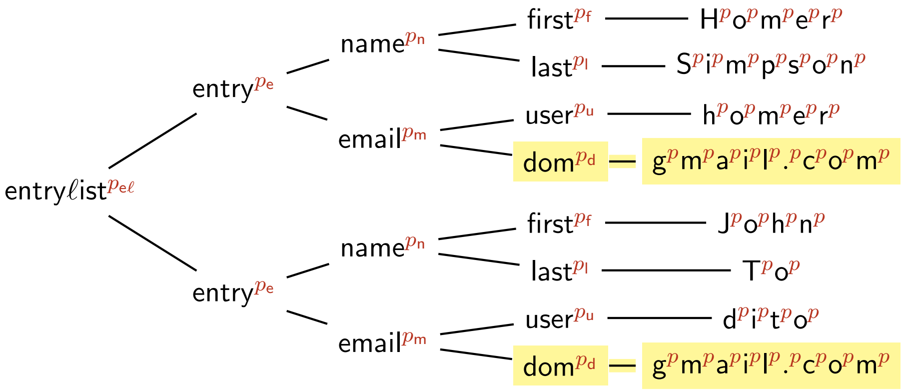
inclusion constraints

TAGED [Filiot, Talbot, Tison 2007, 08]

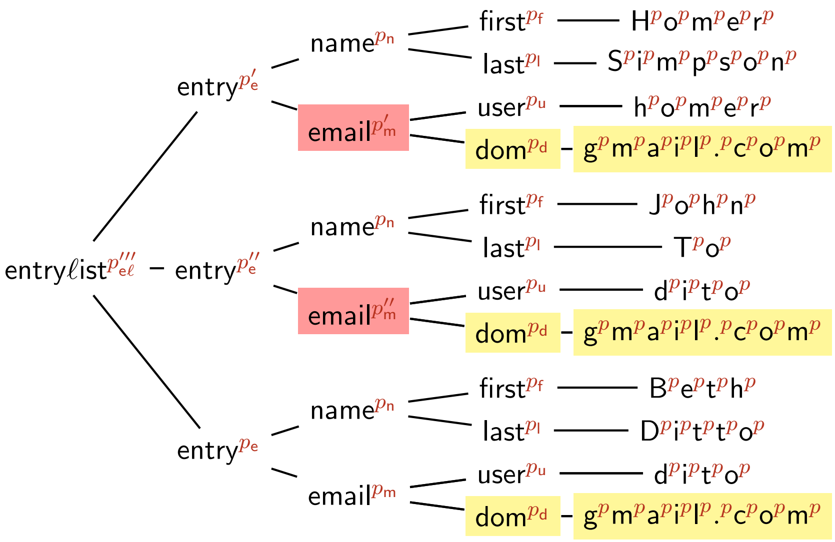
TAGED := TAGC$^\wedge[\approx, \not\approx_{\mathsf{irr}}]$ (positive conjunctive)

decision tool for TQL spatial logic [Cardelli, Ghelli 2002].

Example: all domain's coincide: with $p_{\mathsf{d}} \approx p_{\mathsf{d}}$



all domain's coincide, there are at least two different email's: with $p_{\mathsf{d}} \approx p_{\mathsf{d}} \wedge p'_{\mathsf{m}} \not\approx p''_{\mathsf{m}}$ and email$(p_{\mathsf{u}}\ p_{\mathsf{d}}) \to p_{\mathsf{m}} \mid p'_{\mathsf{m}} \mid p''_{\mathsf{m}}$, and transitions such that p'_{m}, p''_{m} occur exactly once in a successful run.



TAGED: Closure Properties

The class of TAGED- ($\mathrm{TAGC}^{\wedge}[\approx, \not\approx_{\mathsf{irr}}]$) languages is effectively closed under

- $\cup$: linear construction
  disjoint union of automata and global constraints
- $\cap$: quadratic construction (Cartesian product)

$\mathcal{A}_1 \cap \mathcal{A}_2$		$\mathcal{A}_1$		$\mathcal{A}_2$
$Q_1 \times Q_2$		Q_1		Q_2
$\langle p_1, p_2 \rangle \approx \langle p'_1, p'_2 \rangle$	iff	$p_1 \approx p'_1$	or	$p_2 \approx p'_2$
$\langle p_1, p_2 \rangle \not\approx \langle p'_1, p'_2 \rangle$	iff	$p_1 \not\approx p'_1$	or	$p_2 \not\approx p'_2$

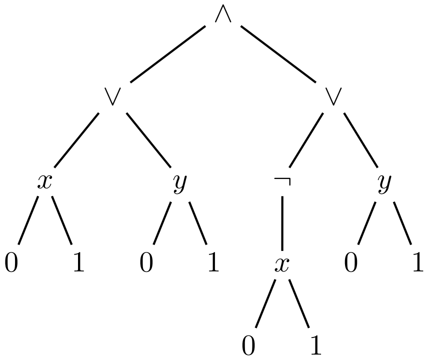
- not $\neg$
  $\{$trees with subtree $\mathsf{g}(x, y) \mid x \neq y\}$ is in TAGED, not its complement.
- not determinizable

TAGED: Decision I

Membership is NP-complete [Filiot et al 2007 CSL]

hardness: reduction of satisfiability of Boolean expressions.

Ex. $(x \vee y) \;\wedge\; (\neg x \vee y)$



$$
\begin{array}{rcl}
0 & \to & q_x \mid q \\
1 & \to & q_x \mid q \\
x(q, q_x) & \to & q_1 \\
x(q_x, q) & \to & q_0 \\
0 & \to & q_y \mid q \\
1 & \to & q_y \mid q \\
\dots & &
\end{array}
$$

$$q_x \approx q_x \wedge q_y \approx q_y$$

This TAGED is in $\mathsf{TAGC}^{\wedge}[\approx_{\mathsf{ref}}]$

TAGED: Decision II

Universality and inclusion are undecidable

Reduction of PCP into $\text{TAGC}^{\wedge}[\approx_{\text{ref}}]$ universality.

Emptiness

- EXPTIME-complete for $\text{TAGC}^{\wedge}[\approx]$
- NEXPTIME for $\text{TAGC}^{\wedge}[\not\approx_{\text{irr}}]$
- decidable for subclass of TAGED testing bounded number of $\approx$, $\not\approx_{\text{irr}}$ [Filiot et al 2007 CSL]
- decidable for subclass of TAGED testing bounded number of $\not\approx_{\text{irr}}$ [Filiot et al 2008 DLT]

Rigid Tree Automata [J Klay Vacher 2009, 11]

RTA $=$ subclass $\mathsf{TAGC}^{\wedge}[\approx_{\mathsf{ref}}]$ of TAGED

for data flow analysis of security protocols

$\approx$-constraints are sufficient for non-linear pattern matching
(bounded number of local equality tests)

Ex. for the pattern $\mathsf{f}(x, x)$,

$$\begin{array}{rcl} \mathsf{a} & \to & q|q_x, \\ \mathsf{b} & \to & q|q_x, \\ \mathsf{f}(q, q) & \to & q|q_x, \\ \mathsf{f}(q_x, q_x) & \to & q_{\mathsf{f}} \\ & q_x \approx q_x & \end{array}$$

successful run of $\mathcal{A}$ on $\mathsf{f}(\mathsf{f}(\mathsf{a}, \mathsf{b}), \mathsf{f}(\mathsf{a}, \mathsf{b}))$: $q_{\mathsf{f}}\big(q_x(q, q), q_x(q, q)\big)$.

RTA: Expressiveness

RTA $\subsetneq$ TAGED

TAGC$^{\wedge}[\approx] \equiv$ TAGC$^{\wedge}[\approx_{\mathsf{ref}}]$ [Filiot et al 2008 DLT]

construction $\rightarrow$ with exponential blowup.

Determinism

TA $\subsetneq$ DRTA $\subsetneq$ RTA

$\{\mathsf{f}(x, x) \mid x \in \mathcal{T}(\Sigma)\}$ not recognizable by DRTA.

Subclass of visibly RTA determinizable.

Rewrite closure
(under equational specification of cryptographic operators)

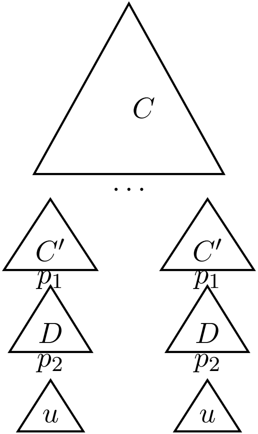
RTA: Expressiveness II

Pumping Lemma

If $t \in L(\mathcal{A})$ and $depth(t) > (|Q| + 1)|R|$, then there exist

- a context C,
- two contexts C' and D, with D non-trivial,
- a term u

such that $t = C\big[C'[D[u]], \ldots, C'[D[u]]\big]$ and for all $n \geq 0$, $C\big[C'[D^n[u]], \ldots, C'[D^n[u]]\big] \in L(\mathcal{A})$.



The set of balanced binary trees is not a RTA language.

RTA: Closure

like TAGED

- $\cup$ (polynomial)
- $\cap$ (exponential):
  2 RTA $\to$ 1 TAGC$^\wedge[\approx]$ for $\cap$ $\to$ 1 RTA for $\cap$
  It is a lower bound (reduction of the emptiness of intersections for tree automata).
- not $\neg$

The set of balanced binary trees is not RTA.
Its complement is recognized by

$$
\begin{array}{rcl@{\qquad}rcl@{\qquad}l}
\mathsf{a} & \to & q|q_r & \mathsf{f}(q,q) & \to & q|q_r & \\
\mathsf{f}(q_r,q) & \to & q'_r & \mathsf{f}(q'_r,q) & \to & q'_r & \\
\mathsf{f}(q,q'_r) & \to & q'_r & \mathsf{f}(q,q_r) & \to & q' & \\
\mathsf{f}(q_r,q'_r) & \to & q_\mathsf{f} & \mathsf{f}(q'_r,q_r) & \to & q_\mathsf{f} & \\
\mathsf{f}(q_\mathsf{f},q) & \to & q_\mathsf{f}, & \mathsf{f}(q,q_\mathsf{f}) & \to & q_\mathsf{f} & q_r \approx q_r
\end{array}
$$

RTA: Decision

Decision

- membership is NP-complete
- universality, inclusion undecidable
- finiteness:
  - PTIME for RTA ($TAGC^{\wedge}[\approx_{ref}]$),
  - EXPTIME for $TAGC^{\wedge}[\approx]$.
- emptiness decidable in linear time

For emptiness: same state marking algorithm as for tree automata.

- regularity is undecidable for RTA, TAGED, $TAGC^{\wedge}[\approx]$

Global vs Local Constraints

TAGED (RTA) and automata with local constraints are orthogonal.

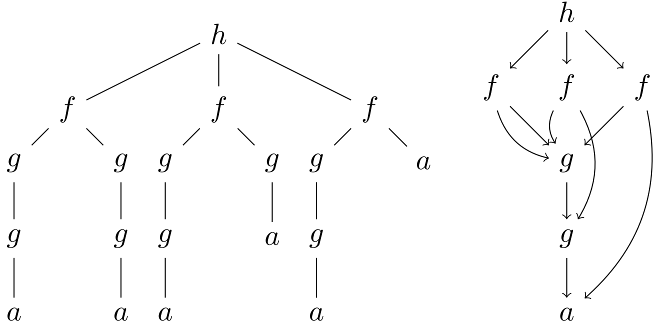
- balanced binary trees $\notin$ TAGED
- "all domain's coincide" $\notin$ local constraints automata.

Lets call two nodes equivalent in a successful run if they have been "successfully tested" to be equal in the run.

	# of eq. classes	cardinality of eq. classes
local constraints	unbounded	bounded
TAGEDs	bounded	unbounded

DAG Automata

DA: tree automata computing (relabeling of nodes by states) on DAGs representations of ranked trees (maximal sharing).



DAG Automata

DA: tree automata computing (relabeling of nodes by states) on DAGs representations of ranked trees (maximal sharing).

Membership is NP-complete for DA
[Charatonik 1999], [Anantharaman et al 2005]

Membership is PTIME for deterministic DA

Tree membership is PTIME complete [Lohrey Maneth 2006]

DAG Automata and Global $\approx$ Constraints

- DA: a unique state is associated to equal subtrees (same node in the DAG representation)
- RTA: a unique subtree is associated to every occurrence of the same state q $\boxed{\text{if } q \approx q}$

DA and RTA are orthogonal.

$\{f(x, x)\}$ is not recognizable by DA.

DAG Automata and Global $\not\approx$ Constraints

Emptiness is NP-complete for DA [Charatonik 1999]

[Vacher 2010 PhD]

- DA $\equiv$ TAGC$^\wedge[\not\approx_{\mathsf{irr}}]$
  - $\Rightarrow$ $q_1 \not\approx q_2$ iff $q_1 \neq q_2$.
  - $\Leftarrow$ states $2^Q \setminus \{P \mid \exists q_1, q_2 \in P, q_1 \not\approx q_2\}$

DAG Automata and Global $\not\approx$ Constraints

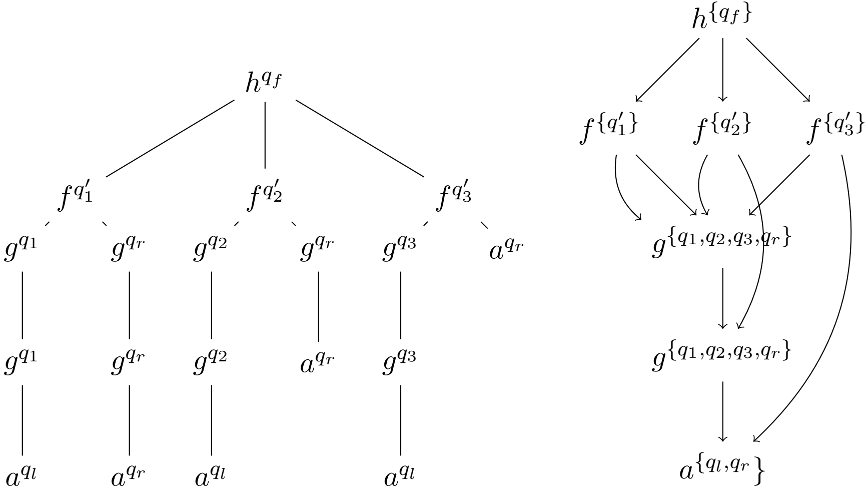
Emptiness is NP-complete for DA [Charatonik 1999]

[Vacher 2010 PhD]

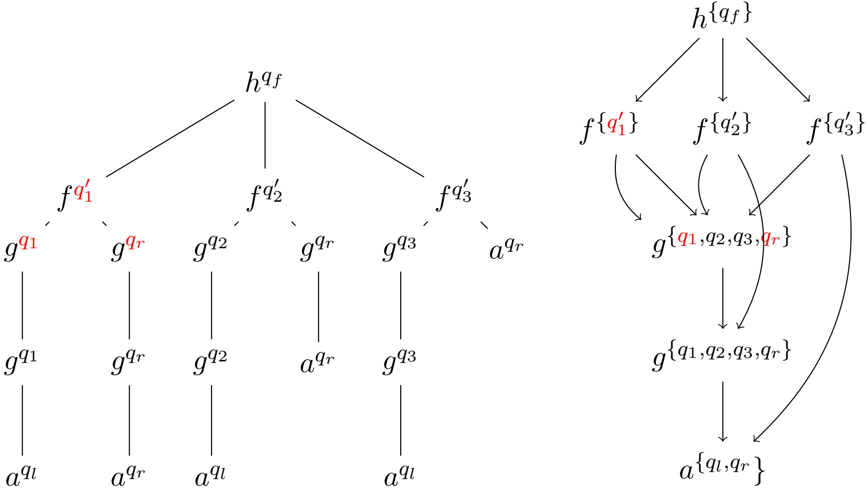
- DA $\equiv$ TAGC$^{\wedge}[\not\approx_{\mathsf{irr}}]$
  - $\Rightarrow$ $q_1 \not\approx q_2$ iff $q_1 \neq q_2$.
  - $\Leftarrow$ states $2^Q \setminus \{P \mid \exists q_1, q_2 \in P, q_1 \not\approx q_2\}$
- TAGC$^{\wedge}[\approx, \not\approx_{\mathsf{irr}}] \equiv$ DA$[\approx]$

  DA$[\approx]$: $q_1 \approx q_2 := \forall x, y \in dom(d)\ q_1(x) \wedge q_2(y) \Rightarrow x = y$
- Emptiness is still NP-complete for DA$[\approx]$
- Emptiness is decidable in NEXPTIME for TAGED

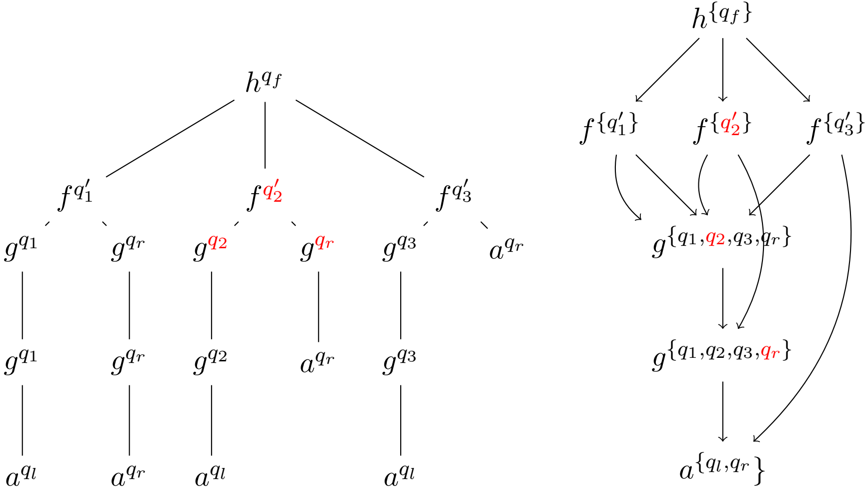
Open problem: generalization to TAGC$^{\wedge}[\approx, \not\approx]$
(elementary upper bound for emptiness decision)



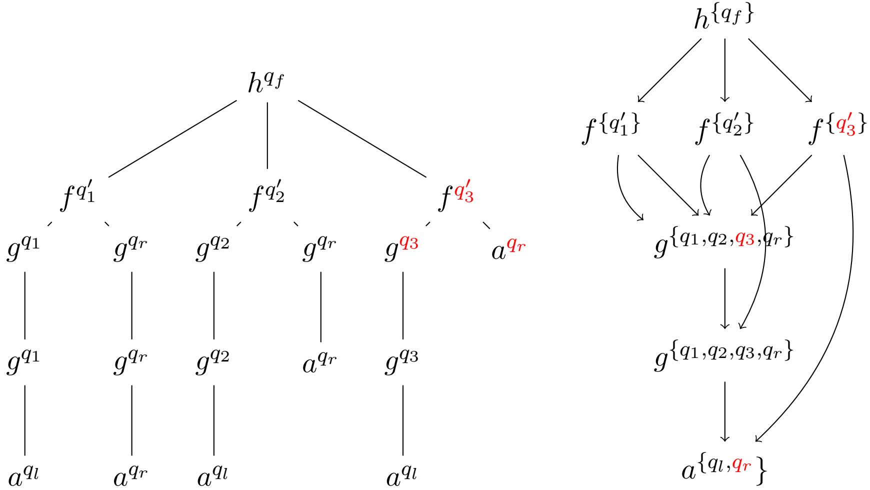
h^{q_f}
$f^{q'_1}$
$f^{q'_2}$
$f^{q'_3}$
g^{q_1}
g^{q_r}
g^{q_2}
g^{q_r}
g^{q_3}
a^{q_r}
g^{q_1}
g^{q_r}
g^{q_2}
a^{q_r}
g^{q_3}
a^{q_l}
a^{q_r}
a^{q_l}
a^{q_l}
$h^{\{q_f\}}$
$f^{\{q'_1\}}$
$f^{\{q'_2\}}$
$f^{\{q'_3\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$a^{\{q_l,q_r\}}$



h^{q_f}
$f^{q'_1}$
$f^{q'_2}$
$f^{q'_3}$
g^{q_1}
g^{q_r}
g^{q_2}
g^{q_r}
g^{q_3}
a^{q_r}
g^{q_1}
g^{q_r}
g^{q_2}
a^{q_r}
g^{q_3}
a^{q_l}
a^{q_r}
a^{q_l}
a^{q_l}
$h^{\{q_f\}}$
$f^{\{q'_1\}}$
$f^{\{q'_2\}}$
$f^{\{q'_3\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$a^{\{q_l,q_r\}}$



h^{q_f}
$f^{q_1'}$
$f^{q_2'}$
$f^{q_3'}$
g^{q_1}
g^{q_r}
g^{q_2}
g^{q_r}
g^{q_3}
a^{q_r}
g^{q_1}
g^{q_r}
g^{q_2}
a^{q_r}
g^{q_3}
a^{q_l}
a^{q_r}
a^{q_l}
a^{q_l}
$h^{\{q_f\}}$
$f^{\{q_1'\}}$
$f^{\{q_2'\}}$
$f^{\{q_3'\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$a^{\{q_l,q_r\}}$



h^{q_f}
$f^{q'_1}$
$f^{q'_2}$
$f^{q'_3}$
g^{q_1}
g^{q_r}
g^{q_2}
g^{q_3}
a^{q_r}
a^{q_l}
$h^{\{q_f\}}$
$f^{\{q'_1\}}$
$f^{\{q'_2\}}$
$f^{\{q'_3\}}$
$g^{\{q_1,q_2,q_3,q_r\}}$
$a^{\{q_l,q_r\}}$

Closure and Decision Results

	TA TAGC[ ]	RTA TAGC$^\wedge[\approx_{\mathsf{ref}}]$	DA TAGC$^\wedge[\not\approx_{\mathsf{irr}}]$	TAGED TAGC$^\wedge[\approx,\not\approx_{\mathsf{irr}}]$
$\cup$	PTIME	PTIME	PTIME	PTIME
$\cap$	PTIME	EXPTIME	not	EXPTIME
$\neg$	EXPTIME	not	not	not
emptiness	linear time	linear time	NP-c.	NEXPTIME-c
membership	PTIME	NP-c.	NP-c.	NP-c.
$\cap$-emptiness	EXPTIME-c.	EXPTIME-c.	EXPTIME-c.	
universality	EXPTIME-c.	undec.	undec.	undec.
inclusion	EXPTIME-c.	undec.	undec.	undec.
finiteness	PTIME	EXPTIME	PTIME	

TAGC[$\approx, \not\approx$] [Godoy et al 10,12]

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

we use TAGC[$\approx, \not\approx$] $\equiv$ TAGC$^{\wedge}$[$\approx, \not\approx$]

- 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 $e_0, e_1, \ldots, e_{\mathsf{h}(t)}$ 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 of TAGC[$\approx, \not\approx$]

extension to unranked trees immediate, using encoding into binary trees [Godoy et al 2012]

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]
- TAGC[$\approx, \not\approx$] where $\approx$ and $\not\approx$ are interpreted modulo flat equational theories

Arithmetic Constraints

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

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, 2008], [Dal Zilio Lugiez 2006]: count the siblings of unranked trees (*local cstr*).

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

Monadic Second Order Logic

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

- first order variables x: position in a tree
- second order variables X: finite set of positions

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 $x \in X$, $y \in Y$, $t\|_x \neq t\|_y$)
$\|.\|_{\mathbb{Z}}$	$\sum a_i \cdot \|X_i\| \geq a$, $a_i, a \in \mathbb{Z}$ ($\|X_i\|$ is cardinality of X_i)
$\|.\|_{\mathbb{N}}$	when a_i, a have same sign
$\\|.\\|_{\mathbb{Z}}$	$\sum a_i \cdot \\|X_i\\| \geq a$ ($\\|X_i\\|$ is cardinality of $\{t\|_x \mid x \in X_i\}$)
$\\|.\\|_{\mathbb{N}}$	when a_i, a have same sign

Monadic Second Order Logic: satisfiability

- MSO$[+1] \equiv$ tree automata [Thatcher Wright 1968]
- MSO$[+1, \approx]$ undecidable
- MSO$[+1, \mathbb{Z}]$ undecidable [Klaedtke Ruess 2002]

EMSO: $\exists X_1 \ldots \exists X_n \, \phi(X_1, \ldots, X_n) \wedge \psi(X_1, \ldots, X_n)$ where

- $\phi(X_1, \ldots, X_n)$ in MSO$[+1]$
- $\psi(X_1, \ldots, X_n)$ in MSO$[+1, \approx, \not\approx, |.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$, free

- EMSO$[+1, \mathbb{Z}]$ decidable [Klaedtke Ruess 2002]
- fragment of EMSO$[+1, \approx, \not\approx]$ decidable [Filiot et al 2008]
- EMSO$[+1, \approx, \not\approx, |.|_{\mathbb{N}}, \|.\|_{\mathbb{N}}]$ decidable [Godoy et al 2010]

Perspectives

- Combining Local/Global Constraints for unranked ordered trees
  - combination of TAGC$[\approx, \not\approx]$
  - with UTASC: unranked tree automata with local sibling constraints [Löding Wong 2007, 09]
- 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)

Thank you

Thank you