Unranked Tree Automata with Sibling Equalities and Disequalities

Wong Karianto Christof Löding

Lehrstuhl für Informatik 7, RWTH Aachen, Germany

34th International Colloquium, ICALP 2007

Outline

Unranked trees

Why unranked tree?

- application of such tree as models of semi-structured data(queries)
- automata-related and logic-related notions for the unranked trees
- many suit for ranked trees, for the unranked ones too.

Unranked trees

Why unranked tree?

- application of such tree as models of semi-structured data(queries)
- automata-related and logic-related notions for the unranked trees
- many suit for ranked trees, for the unranked ones too.

logic, automation models vs finite automata'framework

- more expressive than finite automata
- good algorithmic properties
- constraints
  - numerical constraints
  - structures(label and data value)

Unranked Tree Automata with Sibling Constraints
——deterministic UTACS

Nonemptiness

nonemptiness problem for the deterministic automata is decidable

Unranked Tree Automata with Sibling Constraints
——deterministic UTACS

Nonemptiness

nonemptiness problem for the deterministic automata is decidable

Expressive

the nondeterministic automata are more expressive than the deterministic ones.

Learn from the Ranked ones

now, we get

nonemptiness problem for the deterministic/non-deterministic automata of the ranked tree is decidabe

come into the unrank ones

- The number of pairs of sibling subtrees to be compared is not a priori bounded.
- Can we find the bound?
  - Yes, then, find it, then done!
  - No, pursue another methods ... ...
- deterministic vs non-deterministic, they are diff.

Lugiez

Automata on multitrees(unranked, unordered trees). Constraints: numerical and inclusion relations among multisets of (multi)trees. Closed under Boolean operations, determinizable Have a decidable nonemptiness problem.

Lugiez

Automata on multitrees(unranked, unordered trees). Constraints: numerical and inclusion relations among multisets of (multi)trees. Closed under Boolean operations, determinizable Have a decidable nonemptiness problem.

Diff. to Lugiez's

Evaluating a constraint in an unbounded(unordered) sequence of (multi)trees is reduced to evaluating the constraint in an (unordered) sequence of multisets of trees whose length is bounded by the number of states of the underlying automation.

Lugiez

Automata on multitrees(unranked, unordered trees). Constraints: numerical and inclusion relations among multisets of (multi)trees. Closed under Boolean operations, determinizable Have a decidable nonemptiness problem.

Advantages

- Equality tests are imposed between multisets of trees(in our setting: between trees)
- The number of equality tests depends on the number of states of the automation instead of the size of input (muti)tree.

tuple and word

κ-tuple

- $\mathbb{N}(\mathbb{N}_+)$: set of (positive) natural numbers
- $\mathbb{N}^\kappa(\mathbb{N}_+^\kappa)$: κ-tuples over the sets $\mathbb{N}(\mathbb{N}_+)$
- denoted by $\bar{d}, \bar{e}$ ... ...
- ordered by comparing them componentwise
- $\widehat{m}$:κ-tuple (m, ..., m), when κ is clear from the context

Set and words

- A: a set
- A^*: the set of all(finite) words over A
- ε: empty word
- write A^+ for $A^* \setminus \varepsilon$
- $|w|$: the length of word w

word structure and MSO-formula

MSO

- A: a finite, nonempty alphabet
- $< \{1, ..., |w|\}, S, <, (\chi_\alpha)_{\alpha \in A} >$ is a word structure, where
  - S and $<$ denote the successor and the order relation
  - $\{1, ..., |w|\}$: set of positions in w
  - χ_α: for $\forall \alpha \in A$, is the set of α positions in w
- formulas of MSO over A are built up from
  - first-order variables x, y, z, ..., which range over positions
  - MSO variables X, Y, Z, ..., which range over sets of positions
  - atomic formulas: x = y, x < y, S(x,y), X(x), and $\chi(x)$, for $\forall \alpha \in A$ and for all variables x, y, X
  - Boolean connectives
  - first-order as well as set quantifiers.

word structure and MSO-formula

MSO

- $\varphi(x_1, ..., x_n, \mathrm{X}_1, ..., \mathrm{X}_m)$ indicate that the MSO-formula φ may contain free occurrences of the variables $x_1, ..., x_n, \mathrm{X}_1, ..., \mathrm{X}_m$.

Tree

- Σ: a nonempty, finite (tree-labeling) alphabet
- Tree domain D: a nonempty, prefix-closed subset of $\mathbb{N}_+^*$, such that
  - $\forall$ u $\in$ D and i $>$ 0, if ui $\in$ D, then uj $\in$ D $\forall$ j $\in$ {1, ..., i }
- A finite unranked tree t over Σ is a mapping t:$dot_t \rightarrow \Sigma$
- The elements of dom_t are called the nodes of t, node ε is called the root of t
- A node u $\in$ dom_t is said to have k$\geq$0 successors if uk $\in$ dom_t but u(k+1)$\notin$ dom_t
  - call ui the i-th successor of u
  - ui and uj are sibling nodes, for each i, j $\in$ {1, ..., k}
  - leaf of t is a node without any successor.

Tree cont.

- subtree at u of t
  - Tree: $t|_u$, $dom_{t_u} = \{v \in \mathbb{N}_+^* | uv \in dom_t\}$ and $t|_u$ = t(uv), for all v $\in dom_{t|_u}$
  - $t|_u$ is call a direct subtree of t, if |u| = 1.
  - write t as $a(t_1, ..., t_\kappa)$ to indicate that its root is labeled with a and that it has κ successors at with the subtrees $t_1, ..., t_\kappa$ are rooted
- $\mathtt{T}_\Sigma$ as set of all Σ-labeled trees

Constraints

Look ahead

1. bottom-up fashion labeling algorithm
2. the application of a transition on a node of the input tree is subject to some equality and diseauqlity contraints between the direct subtrees of that particular node.
3. for example: "1 = 2 $\wedge$ 1 $\neq$ 3" and $\wedge_{1 \leq i,j \leq k}$(i = j)

Constraints

Four kinds of constraints

1. $\eta = \exists_{EQ}$: there exist $\kappa, \lambda \in \{1, ..., |w|\}$ such that $w \vDash \varphi(\kappa, \lambda)$ and $t_\kappa \neq t_\lambda$
2. $\eta = \exists_{NEQ}$: there exist $\kappa, \lambda \in \{1, ..., |w|\}$ such that $w \vDash \varphi(\kappa, \lambda)$ and $t_\kappa = t_\lambda$
3. $\eta = \forall_{EQ}$: for all $\kappa, \lambda \in \{1, ..., |w|\}$ such that $w \vDash \varphi(\kappa, \lambda)$ and $t_\kappa = t_\lambda$
4. $\eta = \forall_{NEQ}$: for all $\kappa, \lambda \in \{1, ..., |w|\}$ such that $w \vDash \varphi(\kappa, \lambda)$ and $t_\kappa \neq t_\lambda$

UTACS definition

A UTACS $\mathfrak{A}$ is a tuple $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ s.t.

- Q is a finite, nonempty set of states
- $F \subseteq Q$ is the set of final or accepting states,
- $\Lambda \subseteq \Sigma \times Q$ contains the leaf node transitions
- $\Delta \subseteq Reg_+(Q) \times CONS_Q \times \Sigma \times Q$, inner-node transition
  - where $Reg_+(Q)$ denotes the set of regular subset of Q^+

UTACS definition

A UTACS $\mathfrak{A}$ is a tuple $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ s.t.

A run of UTACS $\mathfrak{A}$ on t is defined as:

a Q-labeled tree ρ: $dom_t \rightarrow Q$ with the following property:

1. leaf nodes: $\forall u \in dom_t$, we have $(t(u), \rho(u)) \in \Lambda$
2. Inner nodes: $\forall u \in dom_t$ with $k \geq 1$ successors, there exists a transition $(L, \alpha, t(u), \rho(u)) \in \Delta$, s.t. the word $\rho(u1)...\rho(uk)$ belong to L and, together with the tree sequence $t|_{u1}...t|_{uk}$, satisfies α

UTACS definition

A UTACS $\mathfrak{A}$ is a tuple $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ s.t.

A run of UTACS $\mathfrak{A}$ on t is defined as:

a Q-labeled tree ρ: $dom_t \rightarrow Q$ with the following property:

1. leaf nodes: $\forall u \in dom_t$, we have $(t(u), \rho(u)) \in \Lambda$
2. Inner nodes: $\forall u \in dom_t$ with $k \geq 1$ successors, there exists a transition $(L, \alpha, t(u), \rho(u)) \in \Delta$, s.t. the word $\rho(u1)...\rho(uk)$ belong to L and, together with the tree sequence $t|_{u1}...t|_{uk}$, satisfies α

A run ρ exists, write $t- > \rho(\varepsilon)$
The run ρ is said to be acceptingif $\rho \in \mathrm{F}$.
The tree t is accepted by $\mathfrak{A}$ if there an accepting run of $\mathfrak{A}$ on t.
The set of trees accepted by $\mathfrak{A}$ is denoted by $T(\mathfrak{A})$

UTACS definition

A UTACS $\mathfrak{A}$ is a tuple $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ s.t.

A run of UTACS $\mathfrak{A}$ on t is defined as:

a Q-labeled tree ρ: $dom_t \rightarrow Q$ with the following property:

1. leaf nodes: $\forall u \in dom_t$, we have $(t(u), \rho(u)) \in \Lambda$
2. Inner nodes: $\forall u \in dom_t$ with $k \geq 1$ successors, there exists a transition $(L, \alpha, t(u), \rho(u)) \in \Delta$, s.t. the word $\rho(u1)...\rho(uk)$ belong to L and, together with the tree sequence $t|_{u1}...t|_{uk}$, satisfies α

Example 1.

The set of well-balanced trees over the alphabet {a} can be recognized by a UTACS by taking Q = F = {q},
$\Lambda = (a, q)$,$\Delta = \{(Q^+, \alpha, a, q)\}$ with $\alpha = (true, \forall_{EQ})$

Example 2:Let $\Sigma = \{a, b, \bullet, \perp\}$. For each positive integer n, let $\underline{n}$ be the unary tree of height n over $\{\bullet\}$; that is:

$$\underline{n} = \underbrace{\bullet(\bullet(...(\bullet(\bullet)...)))}_{n-times}.$$

Example 2:Let $\Sigma = \{a, b, \bullet, \perp\}$. For each positive integer n, let $\underline{n}$ be the unary tree of height n over $\{\bullet\}$; that is:

(1 (2 *b b a*) 2 *a* (3 *a a*) 3) 1

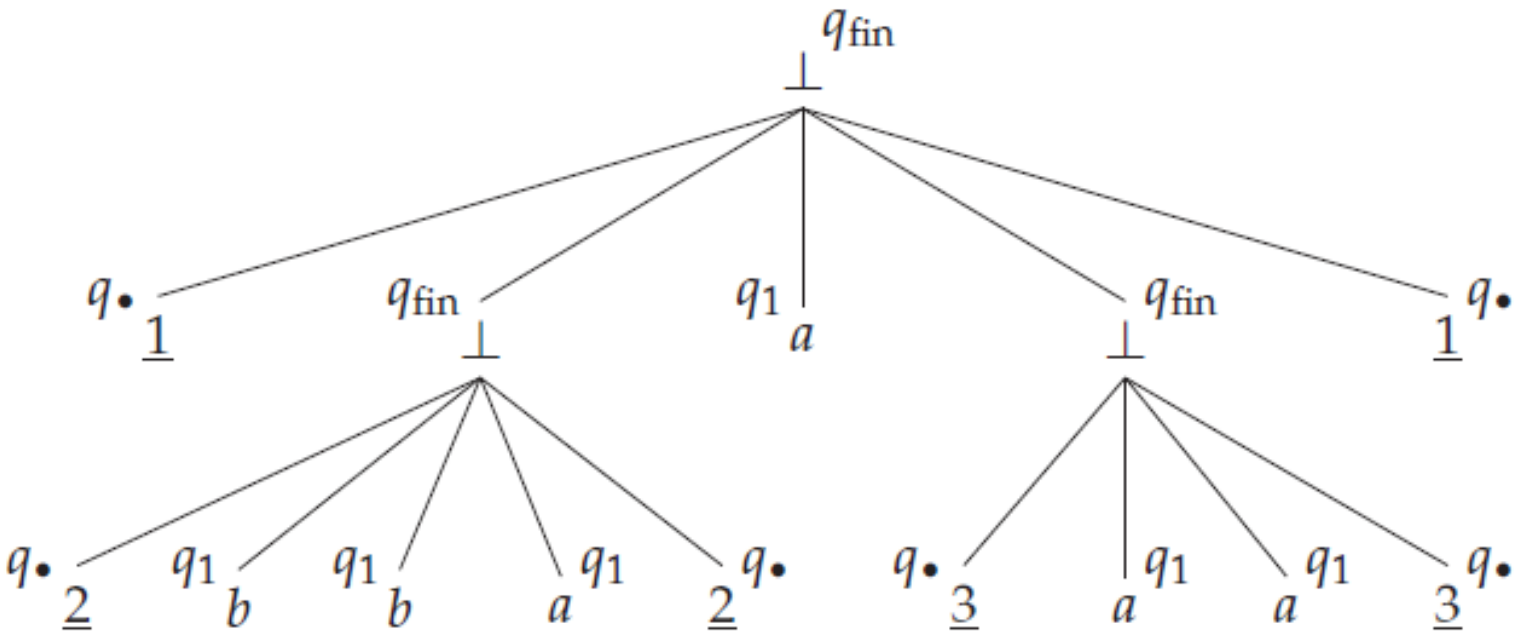
- The root is labeled with $\perp$.
- Every inner node is labeled with either $\perp$ *or* $\bullet$.
- Every inner node labeled with $\perp$ has at least two children. Exactly the first and the last of these are unary trees over $\bullet$, which, moreover, are equal.
- Every node labeled with $\bullet$ has rand at most 1 and may appear only as the first or the last child of an inner node.
- Every leaf if labeled with either a, or b, or $\bullet$.

Example 2:Let $\Sigma = \{a, b, \bullet, \perp\}$. For each positive integer n, let $\underline{n}$ be the unary tree of height n over $\{\bullet\}$; that is:

Three language T is recognized by UTACS

- $Q = \{q_0, q_1, q_{fin}\}$.
- $F = \{q_{fin}\}$.
- $\Lambda = \{(\bullet, q_0), (a, q_1), (b, q_1)\}$.
- $\Delta = \{(q_0, true, \bullet, q_0), (q_0(q_1 + q_{fin})^* q_0, \alpha, \perp, q_{qin})\}$
  - where α requires that "the first and the last subtree are equal, but they are different from all the other subtrees"(that is α is the sibling constraint)

Example 2:Let $\Sigma = \{a, b, \bullet, \perp\}$. For each positive integer n, let $\underline{n}$ be the unary tree of height n over $\{\bullet\}$; that is:



Deter. VS Nondeter. UTACSP

The UTACS $\mathfrak{A}$ is called deterministic(nondeterministic) if, for each tree $t \in \mathrm{T}_\Sigma$, there exists at most one state q(at least one state q) with $t \to q$.

Proposition 2

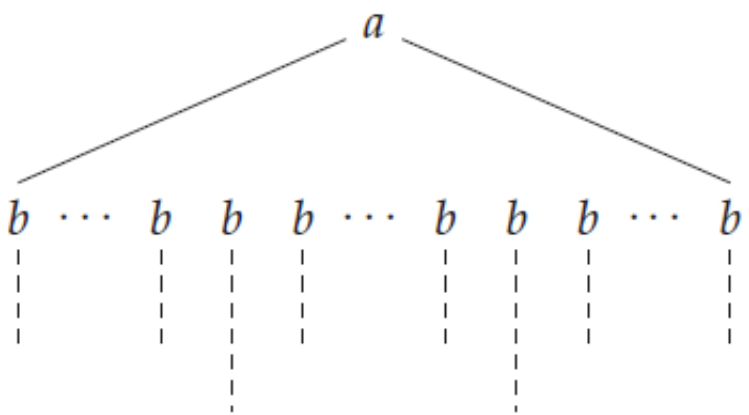
There exists a tree language that is recognizable by a nondeterministic UTACS, but by no deterministic UTACS.

Deter. VS Nondeter. UTACSP

The UTACS $\mathfrak{A}$ is called deterministic(nondeterministic) if, for each tree $t \in T_\Sigma$, there exists at most one state q(at least one state q) with $t \to q$.

Proposition 2

There exists a tree language that is recognizable by a nondeterministic UTACS, but by no deterministic UTACS.



nondeterminism

we would guess the positions of the latter b-strands and mark them by means of a special state. Then, using this particular state, we can address the appropriate pairs of positions that should be equal and those that should be distinct.

determinism

Tt is no longer possible; the fact that there are b-strands of arbitrary length prevents the possibility of using a special state to mark the positions of the two special b-strands and thus also of addressing their positions in the constraints.

Learn from ranked settings

ranked:deterministic

- In the ranked setting, it has been shown that the nonemptiness problem for deterministic automata with sibling constraints is decidable,
- carries over into nondeterministic automata since the it can be determinized.

Learn from ranked settings

ranked ones' standard marking algorithm

- one constructs trees that are accepted by the automaton under consideration, in a bottom-up fashion, by applying the transitions of the automaton.
- In order to apply a transition, needs to find, for each state occurring in the transition.
- With disequality constraints, need more than one tree evaluating to a state
- Thus, if the number of successors is bounded, then this bound gives an upper bound on the sufficient number of distinct trees needed for each state in order to apply a transition.

Learn from ranked settings

Unranked case

lies in the unrankedness aspect: as the number of successors of a tree node is not a priori bounded, we first need to find out how we can bound the sufficient number of distinct trees needed to satisfy a sibling constraint

Learn from ranked settings

Unranked case

lies in the unrankedness aspect: as the number of successors of a tree node is not a priori bounded, we first need to find out how we can bound the sufficient number of distinct trees needed to satisfy a sibling constraint

If we can assert the existence of such a bound:

1. For each transition, if this transition is applicable, then as many distinct trees as given by this bound are sufficient in order to apply this transition.
2. Using this bound, we can then devise, a nonemptiness decision procedure for deterministic UTACS's.

Notations

Let $\mathfrak{A} = (Q, \Sigma, \Delta, \Lambda, F)$ be a deterministic UTACS and $\tau = (L, \alpha, a, q)$ be a transition of $\mathfrak{A}$.

- $w \in Q_+$ *suitable for* τ
  if it can be used in an application of τ, thus resulting in a tree that evaluates to q, provided that, for each state occurring in w, there are plenty of distinct trees evaluating to this state q.
- S_τ: denote the set of words that are suitable for τ
  Moreover, in the following exposition we can assume that S_τ is not empty as τ otherwise cannot be applied at all and can thus be removed from Δ.

Notations

Let $\mathfrak{A} = (Q, \Sigma, \Delta, \Lambda, F)$ be a deterministic UTACS and $\tau = (L, \alpha, a, q)$ be a transition of $\mathfrak{A}$.

- $[|w, \tau|] \in \mathbb{N}^{|Q|}$: a tuple of natural numbers that indicates, for each state, the number of distinct trees that are used for a particular application of τ that uses w.
- $[|w, \tau|]$ can be seen as a mapping $[|w, \tau|] : Q \to \mathbb{N}$ where $[|w, \tau|](p)$ is assigned the p-component of $[|w, \tau|]$ for each p$\in$Q.
- $[|w, \tau|](p)$ does not need to exceed $|w|$

Notations

Let $\mathfrak{A} = (Q, \Sigma, \Delta, \Lambda, F)$ be a deterministic UTACS and $\tau = (L, \alpha, a, q)$ be a transition of $\mathfrak{A}$.

Aim

Our aim is to show the existence of a bound N such that for each word w that is suitable for τ, if $[|w, \tau|](p)$ exceeds N, for some p$\in$Q, then we can find another τ-suitable word w' such that $[|w, \tau|]$ is less than or equal to $\widehat{N}$.

Notations

Let $\mathfrak{A} = (Q, \Sigma, \Delta, \Lambda, F)$ be a deterministic UTACS and $\tau = (L, \alpha, a, q)$ be a transition of $\mathfrak{A}$.

$S_{\tau,R,\overline{d}}$

given a set R$\subseteq$Q and a tuple $\overline{d} \in \mathbb{N}_{|R|}$, the word w is said to be *suitable for τ with respect to R and $\overline{d}$* if the transition τ can be applied under the assumption that for each state p occurring in w:

- there are $\overline{d}(p)$ many distinct trees that evaluate to p, if p$\in$R, and
- there are plenty of distinct trees that evaluate to p, if p$\notin$R.

We denote the set of all words that are suitable for τ with respect to R and $\overline{d}$ by $S_{\tau,R,\overline{d}}$.

Bound Lemma, S_τ and $S_{\tau,R,\overline{d}}$

Bound Lemma 3

There exists some N$\geq$0 such that, for each transition τ of $\mathfrak{A}$ and for each word $w \in S_\tau$, there exists a word $w' \in S_\tau$ such that the following holds:

1. $[|w', \tau|] \leq \widehat{N}$
2. $[|w', \tau|] \leq [|w, \tau|]$
3. For any p$\in$Q, if $[|w, \tau|](p) >$ N, then $[|w', \tau|](p) > 0$.

Bound Lemma, S_τ and $S_{\tau,R,\overline{d}}$

Lemma 4

The sets S_τ and $S_{\tau,R,\overline{d}}$, for all R$\subseteq$Q and $\overline{d} \in N^{|R|}$, are regular subsets of Q^+. In particular, the nonemptiness problem for these sets is decidable.

Bound Lemma, S_τ and $S_{\tau,R,\overline{d}}$

Lemma 4 Proof sketch

- Roughly speaking, a word w belongs to S_τ iff it belongs to L and the constraints in α do not cause conflicts in w;
  - example: any pair (κ, λ) of positions in w satisfying a $\forall_{EQ}$-constraint of τ may not satisfy any $\forall_{NEQ}$-constraints of τ.
- since $\mathfrak{A}$ is supposed to be deterministic, if a pair of positions is declared to have equal subtrees by α, then the Q-labels of those positions must be equal.

Bound Lemma, S_τ and $S_{\tau,R,\overline{d}}$

Lemma 4 Proof sketch cont.

- Since atomic constraints are built from MSO-formulas, we can write an MSO-formula that captures all these requirements, which shows the regularity of S_τ.
- To show the regularity of $S_{\tau,R,\overline{d}}$, need to require that the occurrences of p$\in$R can be partitioned into $\overline{d}(p)$-many sets of positions such that this partitioning does not cause conflict in w.
  - for instance,if a pair (κ, λ) of positions in w that are labeled with a state from R satisfies a $\forall_{EQ}$-constraint, then both positions must lie in the same partition.□

A Normal Form for Transitions

Let $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ be a deterministic UTACS. In the next proofs, consider preprocessing of the transitions of a (deterministic) UTACS, the normal form:

Proposition 8

Each (deterministic) UTACS is equivalent to one where each constraint occurring therein is a conjunction consisting of an $\forall_{EQ}$-constraint $\theta_{\forall_{EQ}}$, an $\forall_{NEQ}$-constraint $\theta_{\forall_{NEQ}}$, $\exists_{EQ}$-constraints $\varphi_1, ..., \varphi_k$, and $\exists_{NEQ}$-constraints $\psi_1, ..., \psi_\ell$.
w and $t_1, ...t_{|w|}$ satisfy $\varphi \wedge \psi$ when φ and ψ belong to $\theta_{\forall_{EQ}}$.

- iff $\forall\kappa\forall\lambda \wedge \kappa, \lambda \in \{1, \ldots, |w|\}$, if $w \models \varphi(\kappa, \lambda)$, then $t_\kappa = t_\lambda$, and if $w \models \psi(\kappa, \lambda)$, then $t_\kappa = t_\lambda$,
- iff $\forall\kappa\forall\lambda \wedge \kappa, \lambda \in \{1, \ldots, |w|\}$, if $w \models \varphi(\kappa, \lambda) \vee \psi(\kappa, \lambda)$, then $t_\kappa = t_\lambda$.

Determinism versus Nondeterminism

example

find one UTACS s.t. ...

The set $S_\tau \subseteq Q^+$ is regular

a word $w \in Q^+$ is suitable for τ iff all of the following requirements are met

1. The word w belong to L
2. There exist some pairs of positions in w, say, $(x_1, y_1), ..., (x_k, y_k), (x_1^{'}, y_1^{'}), ..., (x_\ell^{'}, y_\ell^{'})$, s.t.
   - the sets $\{(x_1, y_1), ..., (x_k, y_k)\}$and $\{(x_1^{'}, y_1^{'}), ..., (x_l^{'}, y_\ell^{'})\}$ do not overlap,that is, for each i=1, ..., k and j = 1, ..., ℓ, $\{x_i, y_i\} \neq \{x_j^{'}, y_j^{'}\}$
   - for each i=1,...,k, the pair (x_i, y_i) satisfies $\varphi_i(x_i, y_i) \wedge \neg(\theta_{\forall_{NEQ}}(x_i, y_i) \vee \theta_{\forall_{NEQ}}(y_i, x_i)) \wedge \bigwedge_{p \in Q}(\mathrm{X}_p(x_i) \leftrightarrow \mathrm{X}_p(y_i))$
   - for each j =1, ..., ℓ, the pair $(x_j^{'}, y_j^{'})$ satisfies $\psi_i(x_i^{'}, y_i^{'}) \wedge \neg(\theta_{\forall_{EQ}}(x_i^{'}, y_i^{'}) \vee \theta_{\forall_{EQ}}(y_i^{'}, x_i^{'})) \wedge \neg(x_j^{'} = y_j^{'})$

The set $S_\tau \subseteq Q^+$ is regular cont.

Three...

For each pair (x, y) of positions in w, the formulas:

$$\theta_{\forall_{EQ}}(x,y) \rightarrow \neg(\theta_{\forall_{NEQ}}(x,y) \vee \theta_{\forall_{NEQ}}(y,x)) \wedge \bigwedge_{p \in Q} (\mathrm{X}_p(x) \leftrightarrow \mathrm{X}_p(y))$$

and

$$\theta_{\forall_{NEQ}}(x,y) \rightarrow (\theta_{\forall_{EQ}}(x,y) \vee \theta_{\forall_{EQ}}(y,x)) \wedge \neg(x = y)$$

are satisfied

$S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular

The set $S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular. By Proposition 8, we can assume that α is a conjunction of an $\forall_{EQ}$-constraint $\theta_{\forall_{EQ}}$, an $\forall_{NEQ}$-constraint $\theta_{\forall_{NEQ}}$, $\exists_{EQ} - constraints \varphi_1, ...\varphi_\kappa$, and $\exists_{NEQ} - constraints \psi_1, ..., \psi_\ell$.

Proof

Here, we need a finer analysis of the suitable words. A word $w \in S_\tau$ respects R and $\overline{d}$ iff the occurrences of $p \in R$ can be partitioned into $\overline{d}(p)$-many sets of positions, and moreover, this partitioning may not cause conflict in w. As an illustration, if a pair (κ, λ) of positions in w satisfy $\theta_{\forall_{EQ}}$, and if they are labeled with a state from R, then both positions must lie in the same partition.

$S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular

The set $S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular. By Proposition 8, we can assume that α is a conjunction of an $\forall_{EQ}$-constraint $\theta_{\forall_{EQ}}$, an $\forall_{NEQ}$-constraint $\theta_{\forall_{NEQ}}$, $\exists_{EQ} - constraints \varphi_1, ...\varphi_\kappa$, and $\exists_{NEQ} - constraints \psi_1, ..., \psi_\ell$.

a word $w \in Q^+$ is suitable for τ with respect to R and $\overline{d}$ iff all of the following requirements are met:

1. the word w belongs to L.
2. There exists a family of sets of positions in w, say $(C^p_{j_p})_{p\in R, j_p=1,...,d_p}$, s.t. :
   ... ...
3. for each pair (x, y) of positions in w, the formulas:

The set $S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular

Remark 14

If $\overline{e} \in \mathbb{N}^{|R|}$ is a tuple of natural numbers with $\overline{d} \leq \overline{e}$, then we have $S_{\tau,R,\overline{d}} \subseteq S_{\tau,R,\overline{e}}$.

The set $S_{\tau,R,\bar{d}} \subseteq Q^+$ is regular

Bound Lemma 3

There exists some N$\geq$0 such that, for each transition τ of $\mathfrak{A}$ and for each word $w \in S_\tau$, there exists a word $w' \in S_\tau$ such that the following holds:

1. $[|w', \tau|] \leq \widehat{N}$
2. $[|w', \tau|] \leq [|w, \tau|]$
3. For any p$\in$Q, if $[|w, \tau|](p) >$ N, then $[|w', \tau|](p) > 0$.

The set $S_{\tau,R,\overline{d}} \subseteq Q^+$ is regular

Further restriction

In order to deal with the third condition of Lemma 3, we introduce a further restriction of the sets of suitable words. Let M be a subset of Q. We denote by $S_{\tau,M}$ and $S_{\tau,R,\overline{d},M}$ the restriction of both sets, respectively, to those words in which each state in M occurs at least once. As the former sets are regular, the latter sets also are. Also, Remark 14 still holds for the restriction to M: given a tuple $\overline{e} \in \mathbb{N}^{|R|}$ with $\overline{d} \leq \overline{e}$, we have $S_{\tau,R,\overline{d},M} \subseteq S_{\tau,R,\overline{e},M}$.

Note that all the sets of suitable words we have introduced so far have been shown to be regular. Hence, *it is decidable whether these sets are empty or not*.

Notation used in Finding Algorithms

- $m = |Q|$: the number of states of $\mathfrak{A}$
- $n \geq 0$: a natural number
- $\mathbb{N}_{\leq n}$: the set of natural numbers that are less than or equal to n
- $(\mathbb{N}_{\leq n})^m$: a set of m-tuples built from $\mathbb{N}_{\leq n}$
- $\overline{z} \restriction_R$: the restriction of $\overline{z}$ with respect to R, when $R \subseteq Q$ and $\overline{z} \in \mathbb{N}^m$.
- $\overline{z} \restriction_R : R \to \mathbb{N}$: an |R|-tuple with $\overline{z} \restriction_R (p) = \overline{z}(p)$, for all $p \in R$, and $\overline{z} \restriction_R (p)$ is undefined for all $p \in Q \setminus R$.
- $\mathrm{I} \restriction_R = \overline{z} \restriction_R | \overline{z} \in \mathrm{I}$: the restriction of I with respect to R, when $\mathrm{I} \subseteq (\mathbb{N}_{\leq n})^m$ of m-tuples

Finding the Bound Algorithm

1:**function** BOUND($\mathfrak{A}, \tau, M$)
2: $m \leftarrow |Q|$
3: if$S_{\tau,M} = \emptyset$ **then return 0**
4: get a word $u_{\tau,M}$ from $S_{\tau,M}$
5: $n \leftarrow |u_{\tau,M}|$ //see Remark 11
6: $\mathrm{I} \leftarrow (\mathbb{N}_{\leq n})^m$ //I contains the m-tuples to be checked
7: (for all R$\subseteq$Q do)
8: $I_R^+ \leftarrow \emptyset$ //contains the tuples $\overline{e}$*fromI* $\restriction_R$ that
9: $I_R^- \leftarrow \emptyset$ //have proven successful (i.e, $S_{\tau,R,\overline{e},M} \neq \emptyset$)
//and unsuccessful($S_{\tau,R,\overline{e},M} = \emptyset$), respectively
10: **while I** $\neq \emptyset$ **do**
11: get a tuple $\overline{z}$ from I and remove it from I
12: **for all R** $\subseteq$ **Q do**
13: $\overline{d} \leftarrow \overline{z} \restriction_R$
14: **if** there is no $\overline{e} \leq \overline{d}$ with $\overline{e} \in \mathrm{I}_R^+$ **then**

Find Cont.

15: **if** there is no $\overline{e} \geq \overline{d}$ with $\overline{e} \in I_R^-$ **then**
16: **if** $S_{\tau,R,\overline{d},M} = \emptyset$ **then**
17: $I_R^- \leftarrow I_R^- \cup \overline{d}$
18: **else**
19: $I_R^+ \leftarrow I_R^+ \cup \overline{d}$
20: get a word $v_{\tau,R,\overline{d},M}$ from $S_{\tau,R,\overline{d},M}$
21: $n' \leftarrow max\{n, |v_{\tau,R,\overline{d},M}|\}$ //update n and
22: $I \leftarrow I \cup ((\mathbb{N}_{\leq n'})^m \setminus (\mathbb{N}_{\leq n})^m)$ //put the new tuples into I
23: $n \leftarrow n'$
24: **return n**

algorighm BOUND terminates

Proof

- suppose Bound does not terminate. i.e. while-loop in Line $10 \sim 23$ executed infinitely often.
- Line11 ensures that after we have fetched a tuple from I, we also remove it from I.
- Line 22 ensures that only new tuples are put into I.
- in every execution of the while-loop we will always get a new tuple to consider.
- while-loop is executed infinitely often $\rightarrow$ new tuples are added to I infinitely often, otherwise:I would eventually be empty and the computation would eventually terminate

algorighm BOUND terminates

Proof Cont.

- Putting new tuples into I is done in Line 22, so this line and, more generally, Line 19 $\sim$ Line 23
- exist some i and j with i < j such that $\overline{z}_i \leq \overline{z}_j$,
- In the j-th iteration, however, $\overline{z} \restriction_R$ belongs to I_R^+, so the condition of the **if**-statement in Line 14 is violated, so Line19-Line23 not executed, a contradiction.

Proof Lemma 3

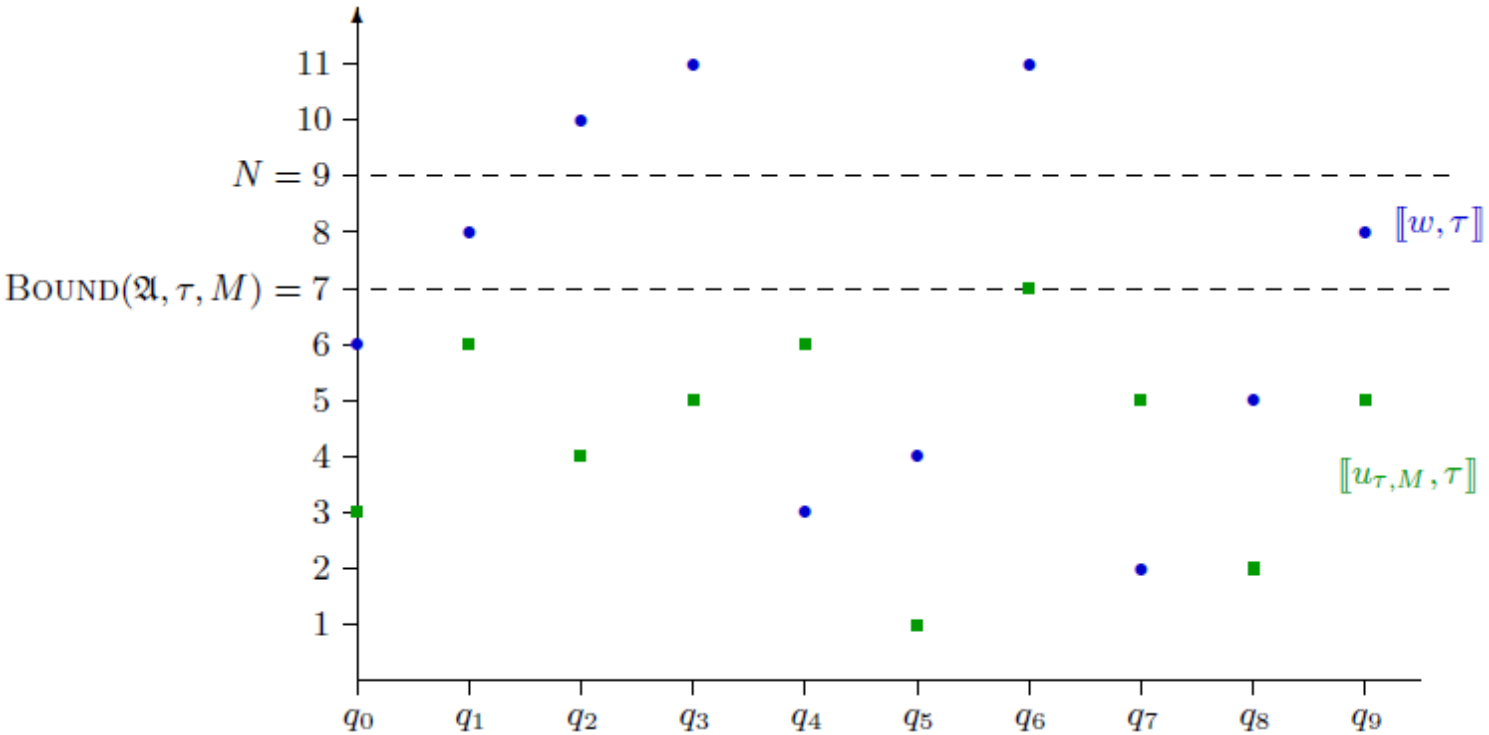
Define. Bound N to be **max{BOUND($\mathfrak{A}, \tau, M$) | $\tau \in \Delta, M \subseteq Q$}.**

Let τ be a transition of $\mathbb{A}$ and w be a word in S_τ . Our task is now to find a word $w' \in S_\tau$ that meets the three requirements given in the lemma. In particular, if τ can be applied by using w, then this must also hold for w'.

- If $[|w, \tau|] \leq \widehat{N}$, then we are done: we just need to take w as w', and all three requirements of the lemma are trivially met.
- Otherwise, there exist some states for which the corresponding values of $[|w, \tau|]$ exceed N. Let $M \subseteq Q$ be the (nonempty) set of these states; i.e.: $M = \{p \in Q | [|w, \tau|](p) > N\}$, then, run the BOUND algorithm

Proof Cont.

- we obtain the word $u_{\tau,M}$ from Line 4
- if $[|u_{\tau,M}, \tau|] \leq [|w, \tau|]$, then we are done by takeing this word as w'



- states: $q_0, ..., q_9$
- $[|u_{\tau,M}, \tau|]$=(3,6,4,5,,1,7,5,2,5)
- $[|w, \tau|]$=(6,8,10,11,3,4,11,2,5,8)
- The states for which $[|w, \tau|]$ exceeds N are q_2, q_3, q_6, so these states constitute the set M
- R=$\{q_0, q_4, q_5, q_7, q_8\}$
- $\overline{d} = (6, 3, 4, 2, 5)$

Formally, let us assume that$[|u_{\tau,M}, \tau|] \neq [|w, \tau|]$, that is, there exists some state $s \in Q$ such that: $[|u_{\tau,M}, \tau|](s) > [|w, \tau|](s)$

- if $[|w, \tau|](p) \leq BOUND(\mathfrak{A}, \tau, M)$, then among the words resulting from the computation of Bound($[|w, \tau|](p) \leq BOUND(\mathfrak{A}, \tau, M), \tau, M$), we should only consider those words whose p-component does not exceed $[|w, \tau|](p)$.
- $[|w, \tau|](p) > BOUND(\mathfrak{A}, \tau, M)$, then the p-component of the words resulting from the computation of Bound$(\mathfrak{A}, \tau, M)$ will never exceed $[|w, \tau|](p)$; thus, intuitively, we can assume that there are sufficiently many distinct trees evaluating to p
  Define R = $\{ p \in Q | [|w, \tau|](p) \leq BOUND(\mathfrak{A}, \tau, M) \}$
  And tuple $\overline{d}$ $in\mathbb{N}^{|R|}$ by setting
  $\overline{d}(p) = [|w, \tau|](p)$ fall all $p \in R$.

Formally, let us assume that$[|u_{\tau,M},\tau|] \neq [|w,\tau|]$, that is, there exists some state $s \in Q$ such that: $[|u_{\tau,M},\tau|](s) > [|w,\tau|](s)$

- Note that, due to (11) and because $Bound(A,\tau,M) \geq [|u_{\tau,M},\tau|](s)$,the set R is not empty.
- by the definition of R and $\overline{d}$, the word w belongs to $S_{\tau,R,\overline{d},M}$, so this set is not empty either
- exists some $\overline{e} \leq \overline{d}$ that has proven successful in the computation of Bound$(\mathfrak{A},\tau,M)$.
- assume that $\overline{d}$ itself has proven successful
- at some point of the computation of Bound$(\mathfrak{A},\tau,M)$ the part Line19-23 is executed,
  thereby giving a word $v_{\tau,R,\overline{d},M}$, from Line 20.
- Now, it is easy to verify: latter word indeed satisfies the the lemma, take it as the desired word w'

Unranked Tree Automata with Sibling Constraints
——deterministic UTACS

Nonemptiness

nonemptiness problem for the deterministic automata is decidable

Unranked Tree Automata with Sibling Constraints
——deterministic UTACS

Nonemptiness

nonemptiness problem for the deterministic automata is decidable

Expressive

the nondeterministic automata are more expressive than the deterministic ones.

Thanks!