Automata and Logic on Trees: Algorithms

Wim Martens[1] Stijn Vansummeren[2]

[1]University of Dortmund, Germany
[2]Hasselt University, Belgium

Natural Questions

General questions:

- Are non-deterministic and deterministic ranked tree automata equivalent? **Yes.**
- Are regular tree languages closed under Bool. operations? **Yes.**
- Does it matter whether we read trees top-down or bottom-up? **Yes.**
- Do we have a pumping lemma? **Yes.**
- Can tree automata be minimized? **Yes.**

Natural Questions

Complexity questions:
What is the complexity of deciding whether...

- automaton A accepts a tree t?
- automaton A accepts a tree at all?
- automaton A accepts a finite number of trees?
- automaton A accepts all trees?
- automaton A accepts all trees of automaton B?
- automaton A accepts precisely the trees of B?
- a set of automata accept a common tree?
- computing the smallest automaton B equivalent to A?

Natural Questions

We'll call these questions

- Membership
- (non)-Emptiness
- Finiteness
- Universality
- Containment / Inclusion
- Equivalence
- Intersection (non)-Emptimess
- Minimization

Why do we do this?

All of these questions have an immediate application in XML

- Membership
- (non)-Emptiness
- Finiteness
- Universality
- Containment / Inclusion
- Equivalence
- Intersection (non)-Emptimess
- Minimization

Why do we do this?

In XML...

tree automaton $\equiv$ database schema

Automaton A accepts tree t

$\equiv_{XML}$ database d adheres to schema S

Automaton B accepts all trees of A

$\equiv_{XML}$ schema B is more general than schema A

Compute the smallest automaton B equivalent to A

$\equiv_{XML}$ schema optimization

Outline

Membership

Definition (Membership Problem)

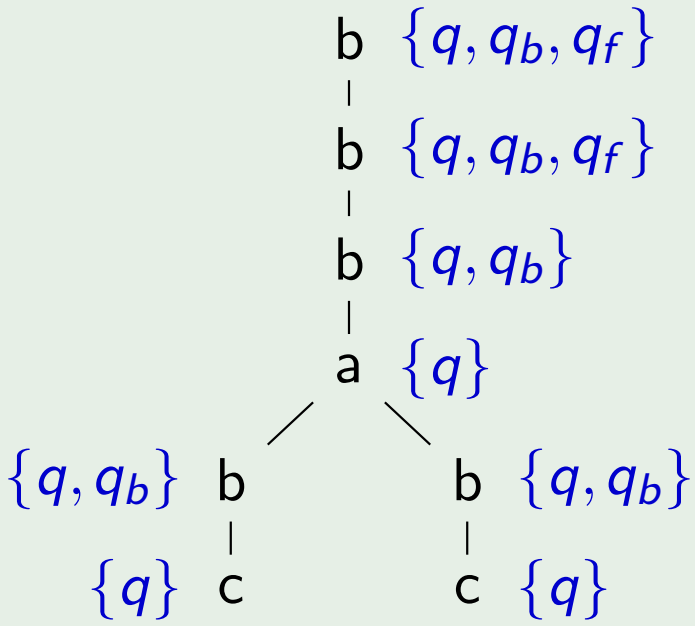
Given an automaton A and a tree t, is $t \in \text{Language}(A)$?

Membership

Example

Automaton with $\text{Final}(A) = q_f$ and rules

$$\varepsilon \xrightarrow{c} q \quad q \xrightarrow{b} q_b \quad q \xrightarrow{b} q \quad q_b \xrightarrow{b} q_f \quad (q,q) \xrightarrow{a} q$$



Membership: Precise Complexity

It's definitely in **PTIME**

- non-deterministic automaton: **logCFL**-complete
- deterministic automaton: in **logDCFL**, precise complexity unknown

Outline

Non-Emptiness

Definition (Non-Emptiness)

Given a tree automaton A, is $\text{Language}(A) \neq \emptyset$?

Non-Emptiness

Example

$\mathsf{Final}(A) = (0,1,2)$

$\varepsilon \xrightarrow{c} (0,0,0)$

$(0,0,0) \xrightarrow{a} (1,0,0) \quad (0,0,0) \xrightarrow{a} (1,0,1) \quad (0,0,1) \xrightarrow{a} (1,0,2)$

$(1,0,0) \xrightarrow{a} (0,0,0) \quad (1,0,0) \xrightarrow{a} (0,0,1) \quad (1,0,1) \xrightarrow{a} (0,0,2)$

$(0,1,0) \xrightarrow{a} (1,1,0) \quad (0,1,0) \xrightarrow{a} (1,1,1) \quad (0,1,1) \xrightarrow{a} (1,1,2)$

$(1,1,0) \xrightarrow{a} (0,1,0) \quad (1,1,0) \xrightarrow{a} (0,1,1) \quad (1,1,1) \xrightarrow{a} (0,1,2)$

$\big((0,0,0),(0,0,0)\big) \xrightarrow{b} (0,1,0) \quad \big((0,0,0),(1,0,0)\big) \xrightarrow{b} (1,1,0)$

$\big((1,0,0),(0,0,0)\big) \xrightarrow{b} (1,1,0) \quad \big((1,0,0),(1,0,0)\big) \xrightarrow{b} (0,1,0)$

Non-Emptiness Algorithm

It's a kind of reachability algorithm

Non-Emptiness Algorithm

Input: Tree automaton A
$Marked = \emptyset$;
repeat
  if $\exists a^{(k)} \in \mathrm{alph}(A), q_1, \ldots, q_k \in Marked$: $(q_1, \ldots, q_k) \xrightarrow{a} q$ **then**
    add q to $Marked$;
  end if
until No more state can be added to $Marked$
return $\mathrm{Init}(A) \cap Marked \neq \emptyset$

Non-Emptiness Algorithm: Complexity

Complexity: **PTIME**-complete

- in **PTIME**: At most a quadratic number of iterations over Rules(A)
- **PTIME**-hard: reduction from Path Systems

Non-Emptiness: **PTIME** Lower Bound

Definition (Path Systems)

Given

- a set of propositions $\{p_1, \ldots, p_n\}$;
- a set of axioms $\{a_1, \ldots, a_\ell\} \subseteq \{p_1, \ldots, p_n\}$;
- a set of rules $(p_i, p_j) \rightarrow p_k$ (p_k is provable from p_i and p_j); and
- a target proposition p,

is p provable from the axioms by using the rules?

Non-Emptiness: **PTIME** Lower Bound

Reduction from PATH SYSTEMS

Given instance

- propositions $\{p_1, \ldots, p_n\}$, axioms $\{a_1, \ldots, a_\ell\} \subseteq \{p_1, \ldots, p_n\}$,
- $(p_i, p_j) \rightarrow p_k$, and target proposition p,

of PATH SYSTEMS,

define automaton A such that

- for each axiom a_i: $\varepsilon \xrightarrow{\#} q_{a_i} \in \mathrm{Rules}(A)$;
- for each rule $(p_i, p_j) \rightarrow p_k$: $(q_{p_i}, q_{p_j}) \xrightarrow{a} q_{p_k} \in \mathrm{Rules}(A)$; and
- $\mathrm{Final}(A) = \{q_p\}$.

Hence, $\mathrm{Language}(A) \neq \emptyset$ iff

p provable from the axioms by using the rules

Non-Emptiness: Complexity Refined

Complexity Refined

- in **PTIME** for non-deterministic tree automata
- **PTIME**-hard for (bottom-up) deterministic tree automata

Outline

Finiteness

Definition

Finiteness Given a tree automaton A, is Language(A) finite?

Observation

Language(A) is infinite if and only if there is a loop on a useful state

useful state: state that appears in some accepting run

Finiteness

- Step 1: find the useful states
- Step 2: search for loops

Definition

State Reachability

- A state q is reachable from q
- If p is reachable from q, and there is a rule

$$(p_1, \ldots, p_n) \xrightarrow{a} p,$$

then $p_1, \ldots, p_n$ are reachable from q

Finiteness

- Step 1: find the useful states
- Step 2: search for loops

Notation

For $q \in \text{States}(A)$, let A_q be A with $\text{Final}(A) = \{q\}$

Finding useful states

- Test, for each q, whether $\text{Language}(A_q) = \emptyset$
- If $\text{Language}(A_q) = \emptyset$, remove q from A:
  - remove q from $\text{States}(A)$ and
  - remove each rule from $\text{Rules}(A)$ in which q occurs
- For all remaining states, test whether they are reachable from a state in $\text{Final}(A)$
- Remove all non-reachable q from A

Finiteness

- Step 1: find the useful states
- Step 2: search for loops

Searching for Loops

For each pair of useful states $p \neq q$ test whether

- p is reachable from q and
- q is reachable from p

(Or, alternatively, use a smarter cycle detection algorithm)

Finiteness

Finiteness: Complexity

- in **PTIME**
- **PTIME**-hard for (bottom-up) deterministic tree automata

Outline

Universality

Definition (Universality)

Given automaton A, is

Language(A) the set of all ranked trees over Alphabet(A)?

Universality

Test whether the language of A's complement is empty

Three steps:

- Determinize A (exponential time)
- Complete and complement the result (poly time)
- Test emptiness

Universality: Complexity

Universality: Complexity

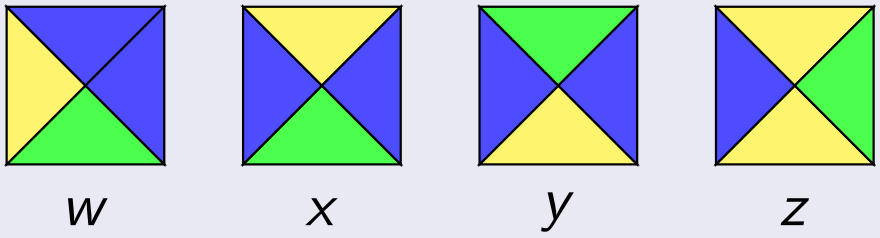
- in **EXPTIME**
- **EXPTIME**-hard (reduction from 2-player corridor tiling)

Universality: **EXPTIME** lower bound

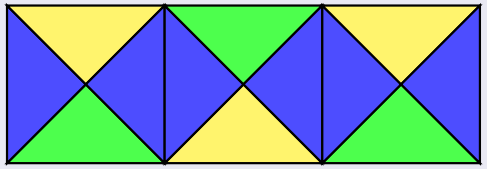
Reduction from 2-player corridor tiling

Corridor Tiling:

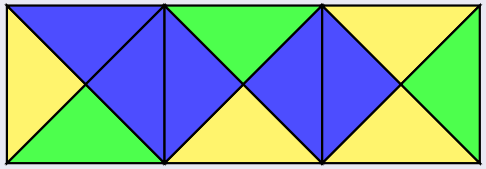
- Set of tiles T:



- The *bottom* row of tiles:



- The *top* row of tiles:

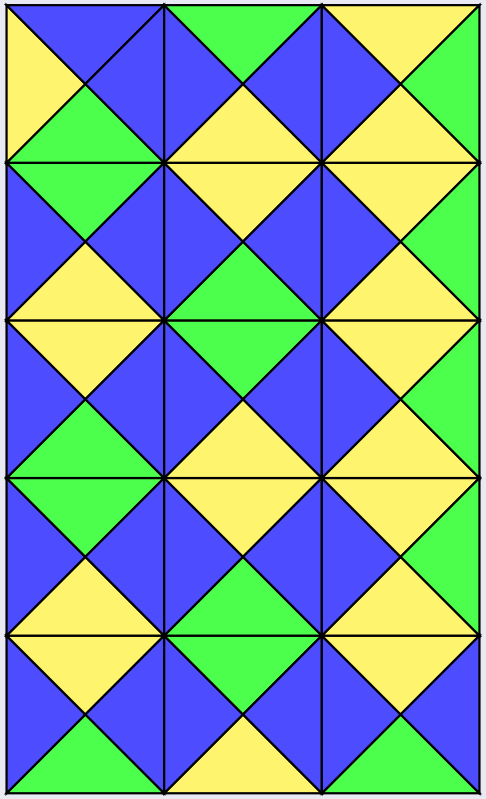


Universality: **EXPTIME** lower bound

Reduction from 2-player corridor tiling

Corridor Tiling:

Can we make a correct corridor tiling?



Universality: **EXPTIME** lower bound

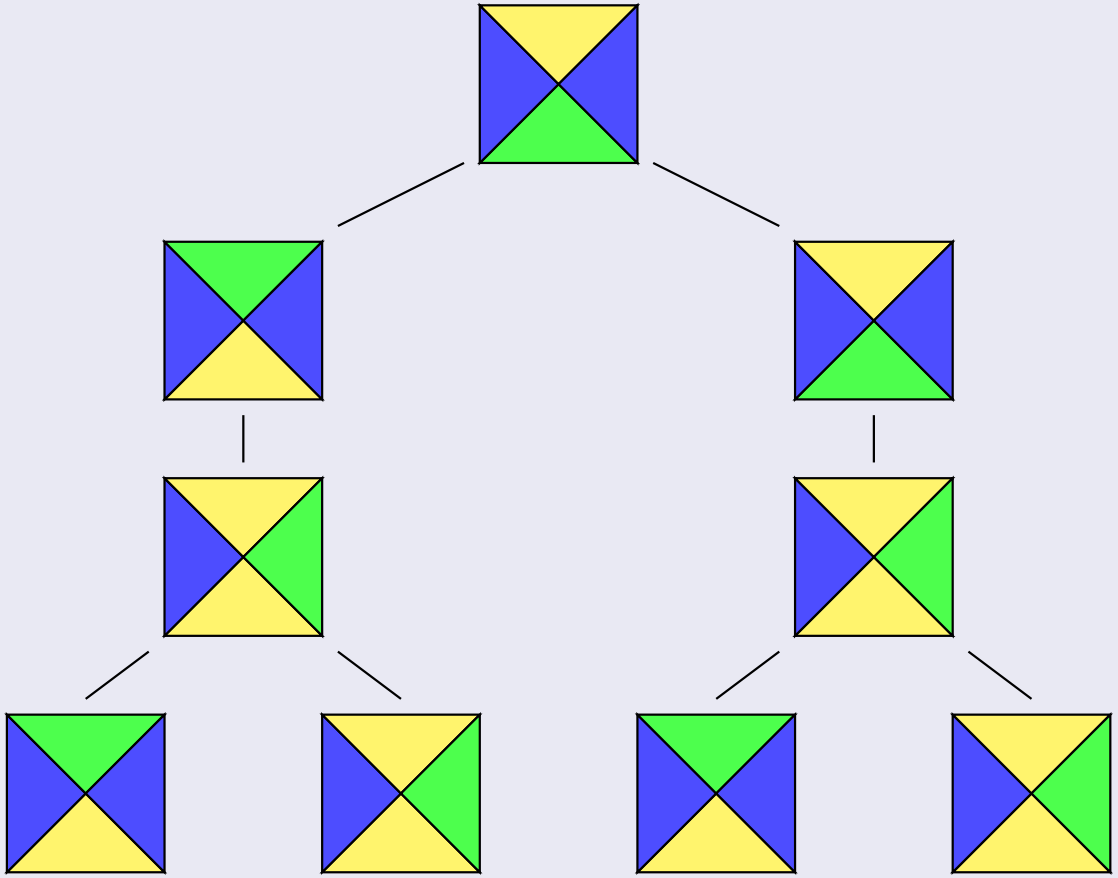
2-Player Corridor Tiling

- CONSTRUCTOR: tries to build a corridor tiling
- SPOILER: tries to prevent CONSTRUCTOR from doing so

Universality: **EXPTIME** lower bound

2-Player Corridor Tiling

Does CONSTRUCTOR have a winning strategy?



2-Player Corridor Tiling: the Reduction

How do we encode tilings?

Three sets of symbols:

- leaves: #
- for each tile t: unary symbol a_t (CONSTRUCTOR moves)
- for each tile t: binary symbol b_t (SPOILER moves)

So our trees look like

2-Player Corridor Tiling: the Reduction

We want to...

...accept all trees iff no winning strategy exists

Automaton A should accept if...

- there is a symbol b_t at odd depth;
- there is a symbol a_t at even depth;
- the top n levels of the tree don't encode the start row;
- there is a path on which the lowermost n nodes don't encode the final row;
- there is a path in which the number of tiles is not a multiple of n;
- there is a binary symbol *in the middle of the tree* with two equally labeled children;
- there is a horizontal mistake; or
- there is a vertical mistake.

Universality: Complexity Revisited

Universality: Complexity Revisited

- in **EXPTIME**
- **EXPTIME**-hard for non-deterministic tree automata
- in **PTIME** for deterministic tree automata!

Outline

Containment

Definition (Containment)

Given tree automata A and B, is $\text{Language}(A) \subseteq \text{Language}(B)$?

I.e., is $\text{Language}(A) \cap \overline{\text{Language}(B)} = \emptyset$?

Containment Complexity

- in **EXPTIME**
- **EXPTIME**-hard (from Universality)

Outline

Equivalence

Definition (Equivalence)

Given tree automata A and B, is $\text{Language}(A) = \text{Language}(B)$?

I.e., is

- $\text{Language}(A) \subseteq \text{Language}(B)$ and
- $\text{Language}(B) \subseteq \text{Language}(A)$?

Equivalence Complexity

- in **EXPTIME** (from Containment)
- **EXPTIME**-hard (from Universality)

Outline

Intersection Non-Emptiness

Definition (Intersection Non-Emptiness)

Given tree automata $A_1, \ldots, A_n$, is $\bigcap_i \text{Language}(A_i) \neq \emptyset$?

- Construct the product automaton $A_1 \cap \cdots \cap A_n$
  (size: product of the sizes of A_i)
- Test non-emptiness of the product automaton

Intersection non-Emptiness: Complexity

- in **EXPTIME**
- **EXPTIME**-hard (from 2-player corridor tiling)

Intersection Non-Emptiness: **EXPTIME** Lower Bound

Tiling system S: construct $A_1, \ldots, A_{n+2}$ such that

CONSTRUCTOR has a winning strategy iff $A_1 \cap \cdots \cap A_{n+2} \neq \emptyset$

Reduction

- Automaton A_1: accept all strategy trees
- Automaton A_2: accept if all horizontal constraints satisfied
- For each column $i = 1, \ldots, n$: Automaton A_{i+2} accept if vertical constraints satisfied in column i

Outline

Minimization

Definition (Minimization)

Given tree automaton A and integer k,

is there an equivalent automaton B such that

B is smaller than k?

Minimization

Minimization in general : **EXPTIME**-complete
But interesting for deterministic automata

Minimization: Theory

Recall...

Theorem (Myhill-Nerode for Trees)

The following are equivalent:

(a) *L is a regular tree language*

(b) *L is the union of some equivalence classes of finite index*

(c) *the relation $\equiv_L$ is a congruence of finite index*

Minimal automaton A for L follows from this classification: size of A = number of equivalence classes of $\equiv_L$

Minimization Algorithm

Minimization Algorithm

Input: Reduced deterministic automaton A

$R = R_0 = \{\text{Final}(A), \text{States}(A) - \text{Final}(A)\}$

repeat

$R = R_0$

define $p_1 R_0 p_2$ iff

- $p_1 \equiv_P p_2$ and
- $\forall a^{(n)} \in \text{Alphabet}(A)$, $\forall q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \in \text{States}(A)$ $r_1 R r_2$, where
  - $(q_1, \ldots, q_{i-1}, p_1, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_1$ and
  - $(q_1, \ldots, q_{i-1}, p_2, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_2$

until $R_0 = R$

$\text{States}(A_{\text{min}}) =$ equivalence classes of P

$\text{Rules}(A_{\text{min}}) = \{([q_1], \ldots, [q_n]) \xrightarrow{a^{(n)}} [q]\}$

$\text{Final}(A_{\text{min}}) = \{[q] \mid q \in \text{Final}(A)\}$

return A_{min}

Minimization: Complexity

Complexity Analysis

- Number of repeat-until-loops: linear
- Cost of refining the equivalence relation: cubic

Refining equivalence relation

Define $p_1 R_0 p_2$ iff

- $p_1 R p_2$ and
- $\forall a^{(n)} \in \text{Alphabet}(A)$, $\forall q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \in \text{States}(A)$ $r_1 R r_2$, where
  - $(q_1, \ldots, q_{i-1}, p_1, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_1$ and
  - $(q_1, \ldots, q_{i-1}, p_2, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_2$

Minimization: Complexity

Complexity Analysis

- Number of repeat-until-loops: linear
- Cost of refining the equivalence relation: cubic

Refining equivalence relation

Define $\neg(p_1 R_0 p_2)$ iff

- $\neg(p_1 R p_2)$ or
- $\exists a^{(n)} \in \text{Alphabet}(A)$, $\exists q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \in \text{States}(A)$ $\neg(r_1 R r_2)$, where
  - $(q_1, \ldots, q_{i-1}, p_1, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_1$ and
  - $(q_1, \ldots, q_{i-1}, p_2, q_{i+1}, \ldots, q_n) \xrightarrow{a} r_2$

Overview

- Membership: **PTIME**
- (non)-Emptiness: **PTIME**
- Finiteness: **PTIME**
- Universality: **EXPTIME** (**PTIME** for deterministic)
- Containment / Inclusion: **EXPTIME** (**PTIME** for deterministic)
- Equivalence: **EXPTIME** (**PTIME** for deterministic)
- Intersection (non)-Emptimess: **EXPTIME** (also for deterministic)
- Minimization: **EXPTIME** (**PTIME** for deterministic)