Multiple tree automata
a new model of tree automata

Gwendal Collet (TU Wien), Julien David (LIPN)

Séminaire CALIN, 24 mars 2015
① Introduction to automata: definitions and motivation
② Description of the model: Multiple Tree Automata
③ Minimization
④ Closure properties
⑤ Yield of a MTA: Link with language theory
Finite alphabet: $a, b, c...$

Set of transitions: $\Delta \subset Q \times \Sigma \times Q$

Finite set of states: initial, final...

$A = (\Sigma, Q, I, F, \delta)$

$L_A = (bc)^*(1 + a^+b)$

e.g.: $bcaabab \in L_A$
Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$$\mathcal{A} = (\Sigma = \bigcup_{k \geq 0} \Sigma_k, Q, I, \Delta)$$

Set of transitions:
$$\Delta \subset \bigcup_{k \geq 0} Q \times \Sigma_k \times Q^k$$

Finite set of states: initial, final...
Introduction: Regular Tree Automata

Finite ranked alphabet: \( a(0), b(1), c(1), d(2) \ldots \)

\[
\mathcal{A} = (\Sigma = \bigcup_{k \geq 0} \Sigma_k, Q, I, \Delta)
\]

Finite set of states: initial, final...

Set of transitions:
\[
\Delta \subset \bigcup_{k \geq 0} Q \times \Sigma_k \times Q^k
\]

\( \mathcal{L}_A = (b(c(\ldots b(c(d(a, d(a, a)))))\ldots)) \)

Independence

\( a \in \Sigma_0 \) (leaf), \( b, c \in \Sigma_1, d \in \Sigma_2 \)

\( i \in I \)

\((i, b, r), (q, a, \epsilon), (s, d, (q, q)), \ldots \in \Delta \)

\( \epsilon \) (empty word)

\( e.g.: \frac{a}{bcda} \in \mathcal{L}_A \)
Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$A = (\Sigma = \bigcup_{k \geq 0} \Sigma_k, Q, I, \Delta)$

Set of transitions: 
$\Delta \subset \bigcup_{k \geq 0} Q \times \Sigma_k \times Q^k$

Finite set of states: initial, final...
Random sampling of trees controlling the number of occurrences of a given pattern.
Introduction: Motivation

Random sampling of trees controlling the number of occurrences of a given pattern

Pattern

When reading the tree top-down:

Dependencies between nodes at a same height

Idea (C., David, Jacquot 2014):

- Use refined tree automata which count occurrences of a given pattern → need to handle dependencies
- Translate the associated tree grammar into a system of equations on generating series
- Design a bivariate Boltzmann sampler with the GS
Multiple Tree Automata (MTA)

Finite ranked alphabet: \( a(0), b(1), c(1), d(2) \ldots \)

\[
\mathcal{A} = (\Sigma = \bigcup_{k \geq 0} \Sigma_k, Q = \bigcup_{\ell \geq 1} Q_\ell, I, \Delta)
\]

Finite ranked set of states

Initial states \( \in Q_1 \)

Set of transitions: \( \Delta \subset \bigcup_{\ell \geq 1} Q_\ell \times \Sigma^\ell \times Part \times Q^* \)

\((q, (a_1, \ldots, a_\ell), P = (p_1, \ldots, p_r), (q_1, \ldots, q_r))\)

such that: \( |P| = \sum_{i=1}^\ell rank(a_i) \)

\[\forall 1 \leq j \leq r, rank(q_j) = |p_j|\]

\((q, (1, 2, 3, 4, 5, 6), \{\{2\}, \{1, 3, 4, 6\}, \{5, 7\}\}, (q_1, q_2, q_3))\)
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\text{node} \
, \quad n \geq 0
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\right\} \]
Multiple Tree Automata (MTA)

\[ A \]

\[ \mathcal{L}_A = \left\{ n \geq 0, \quad m \geq 0 \right\} \]
Multiple Tree Automata (MTA)

\[ A = \{ t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8 \} \]

\[ L_A = \{ \underbrace{\text{\textbullet}}_{n \geq 0}, \underbrace{\text{\textbullet}}_{m \geq 0} \} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \{ \begin{array}{c} n \geq 0, m \geq 0 \\ n \geq 0 \end{array} \} \]
Multiple Tree Automata (MTA)

\[ A \]

\[ \mathcal{L}_A = \begin{cases} \quad \begin{array}{c} n \geq 0 \\ m \geq 0 \end{array}, & n \geq 0 \end{cases} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
  \text{a tree with } n \geq 0 \text{ and } m \geq 0
\end{array} \right\} \]
$\mathcal{A}$

$\mathcal{L}_\mathcal{A} =$

\[
\begin{cases}
\quad, \quad n \geq 0 \\
\quad, \quad m \geq 0 \\
\end{cases}
\]
$\mathcal{L}_A =$ \begin{cases} 
\begin{array}{c}
\scalebox{1.5}{\begin{tikzpicture}
    \node (i) at (0,0) [red, fill=red, circle] {$i$};
    \node (q) at (0,-1) [red, fill=red, circle] {$q$};
    \node (r) at (-2,-2) [red, fill=red, circle] {$r$};
    \node (s) at (0,-2) [red, fill=red, circle] {$s$};
    \node (t) at (2,-2) [red, fill=red, circle] {$t$};
    \path (i) edge (q);
    \path (q) edge (r);
    \path (q) edge (s);
    \path (q) edge (t);
    \path (r) edge (t_6);
    \path (s) edge (t_7);
    \path (t) edge (t_8);
\end{tikzpicture}}
\end{array} \\
, \quad n \geq 0 \\
\begin{array}{c}
\scalebox{1.5}{\begin{tikzpicture}
    \node (root) at (0,0) [red, fill=red, circle] {};\draw[-latex] (root) -- ++(0,-2) node [below] {$m \geq 0$};
\end{tikzpicture}}
\end{array} \\
, \quad n \geq 0 
\end{cases}
Multiple Tree Automata (MTA)

\( \mathcal{L}_A = \{ \text{merge} \}, n \geq 0 \cup m \geq 0 \cup n \geq 0 \)
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{l}
\begin{array}{c}
\text{a tree with } n \geq 0 \\
\text{and } m \geq 0
\end{array}
\end{array} \right\} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{l}
A \\
(t_1, n \geq 0, m \geq 0) \end{array} \right\} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
    \begin{array}{c}
        \begin{array}{c}
            \text{Tree } n \geq 0 \\
            \text{Tree } m \geq 0
        \end{array}
    \end{array}
\end{array} \right\} \]
Multiple Tree Automata (MTA)

\[ \mathcal{A} \]

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
\text{tree, } n \geq 0 \\
\text{tree, } m \geq 0 \\
n \geq 0
\end{array} \right\} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c} \mathcal{A} \\
\end{array} \right\} \]

\[ \mathcal{A} = \left( \right. \]

\[ i \]

\[ t_1 \]

\[ q \]

\[ t_2 \]

\[ r \]

\[ t_6 \]

\[ t_3 \]

\[ s \]

\[ t_7 \]

\[ t_4 \]

\[ t_5 \]

\[ t_8 \]

\[ \left. \right\} \]

\[ n \geq 0 \]

\[ m \geq 0 \]

\[ n \geq 0 \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\text{tree}, \ n \geq 0
\end{array}
\end{array}
\end{array}\right\} \]

\[ A = (MTA) \]
Multiple Tree Automata (MTA)

\[ \mathcal{A} = (M, T, \delta, q_0, F) \]

\[ \mathcal{L}_A = \left\{ \begin{array}{l}
\text{for } n \geq 0, \quad m \geq 0
\end{array} \right\} \]
Multiple Tree Automata (MTA)

\[ \mathcal{L}_A = \left\{ \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\quad \\
\quad \\
\end{array}
\end{array}
\end{array}\right\} \quad n \geq 0 \]

\[ \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\quad \\
\quad \\
\end{array}
\end{array}
\end{array}\right\} \quad m \geq 0 \]
Multiple Tree Automata (MTA)

\[ \mathcal{A} \]  

\[ \mathcal{L}_\mathcal{A} = \left\{ \begin{array}{c} \begin{array}{c} \begin{array}{c} \text{nodes at the same height} \\ t_1, t_2, \ldots, t_8 \end{array} \\ \begin{array}{c} \text{dependencies} \\ n \geq 0, m \geq 0 \end{array} \end{array} \end{array} \right\} \]

same \( n \): can handle dependencies between a **bounded** number of nodes at the same height
\[ \mathcal{A} = \text{same} \] 
\[ \mathcal{L}_\mathcal{A} = \left\{ \begin{array}{l}
\begin{array}{c}
\text{same } n: \text{ can handle dependencies between a } \textbf{bounded} \text{ number of nodes at the same height }
\end{array}
\end{array} \right\} \]

**Def [Non-determinism]:**
Non-deterministic MTA iff \(|I| > 1\) or \(\exists q \in Q_k, (a_1, \ldots, a_k) \in \Sigma^k, (q, (a_1, \ldots, a_k), P, \vec{p}) \text{ and } (q, (a_1, \ldots, a_k), P', \vec{p}') \in \Delta\)

Deterministic MTA otherwise.
Minimization: size of a MTA

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Number of transitions → Not enough anymore!

\[ \mathcal{L}_{A_1} = \mathcal{L}_{A_2} = \mathcal{L}_{A_3} = \{ \text{Binary trees of height less than 3} \} \]
Minimization: size of a MTA

Minimize = Compute the smallest equivalent Deterministic MTA

Size = Number of transitions $\rightarrow$ Not enough anymore!

Size = Total length of transitions

$A_1$

$A_2$

$A_3$

$L_{A_1} = L_{A_2} = L_{A_3} = \{ \text{Binary trees of height less than 3} \}$
Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

Size = Number of transitions → Not enough anymore!

Size = Total length of transitions

\[ A_1 \]

\[ A_2 \]

\[ A_3 \]

size = 12

size = 8 equivalent

size = 5
Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

Size = Number of transitions → Not enough anymore!
Size = Total length of transitions

\( A_1 \)

\( q_1 \)  
\( q_2 \)  
\( i \)

\( A_2 \)

\( r_1 \)  
\( r_2 \)  
\( i \)  
\( r_3 \)  
\( r_4 \)

\( A_3 \)

\( s_1 \)  
\( s_2 \)  
\( i \)

size = 12
size = 8
merge
size
size = 5
Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Number of transitions → Not enough anymore!
Size = Total length of transitions

\[ A_1 \]
\[ i \rightarrow q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow \]
size = 12

\[ A_2 \]
\[ i \rightarrow r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow s_1 \rightarrow \]
size = 8

\[ A_3 \]
\[ i \rightarrow s_1 \rightarrow s_2 \rightarrow \]
size = 5
Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

Size = Number of transitions → Not enough anymore!

Size = Total length of transitions

\[ A_1 \]

\[ A_2 \]

\[ A_3 \]

size = 12

size = 8

size = 5

⚠️ No equivalent states
Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Number of transitions → Not enough anymore!
Size = Total length of transitions

No equivalent states!
New operation: splitting

\[ q \in Q_k \quad \sum_{i=1}^{n} k_i = k \]

\[ q_1 \in Q_{k_1} \quad q_n \in Q_{k_n} \]
Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Total length of transitions → Not enough anymore!
Size = Number of transitions

$A_1$

$q_1 \in Q_k$

$q \in Q_k \quad q_1 \in Q_{k_1} \quad \ldots \quad q_n \in Q_{k_n}$

$\sum k_i = k$

size = 12

⚠️ No equivalent states

New operation: splitting
Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Number of transitions → Not enough anymore!
Size = Total length of transitions

\( A_1 \)

\[ i \rightarrow q_1 \rightarrow q_2 \]

size = 12

\( A_2 \)

\[ i \rightarrow r_1 \rightarrow r_2 \rightarrow q_2 \]

size = 7

No equivalent states

New operation: splitting

\[ q \in Q_k \quad \vdots \quad \sum k_i = k \]

\[ q_1 \in Q_{k_1} \quad q_n \in Q_{k_n} \]
Minimization: minimal DMTA

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Total length of transitions

Theorem

A MTA without equivalent or splittable states is minimal. This minimal automaton can be computed for any DMTA.

Sketch of the minimization algorithm

- Compute and merge any equivalent states.
- Compute and split any splittable states.
- Repeat until a fixpoint is reached.
Closure properties of the tree languages

Theorem

1. MTA are closed under union and concatenation.
2. MTA are not closed under complementation.
3. Non-deterministic MTA are strictly more powerful than deterministic ones.

Proof:

1. Straightforward
2. Language of unary-binary trees with exactly one unary node
   non-deterministic!
Closure properties of the tree languages

Theorem
1. MTA are closed under union and concatenation.
3. Non-deterministic MTA are strictly more powerful than deterministic ones.
2. MTA are not closed under complementation.

Proof:
1. Straightforward
2. Language of unary-binary trees with exactly one unary node

non-deterministic!
Yield of an MTA $\mathcal{A}$:

Word language $Yield(\mathcal{A}) = \{\text{border}(T) : T \in \mathcal{L}_\mathcal{A}\}$

**Theorem**

$Yield(\text{MTA})$ are equivalent to **LCFRS** languages.

**Context-free** $\subset$ **Mildly context-sensitive** $\subset$ **Context-sensitive**

[Diagram of word language and border]

$\text{border}(T) = aeeaf a$
**Conjecture:** MTA are closed under intersection.

→ Semi-algorithm by computing joint dependences, believed to terminate eventually...
Conjecture: MTA are closed under intersection.
   → Semi-algorithm by computing joint dependences, believed to terminate eventually...

What about **Bottom-up** MTA?
   → useful for parsing
   → more expressive in Deterministic Regular TA
Further works

Conjecture: MTA are closed under intersection.

→ Semi-algorithm by computing joint dependences, believed to terminate eventually...

What about **Bottom-up** MTA?

→ useful for parsing
→ more expressive in Deterministic Regular TA

Characterize the tree languages recognized by MTA

→ Regular TL ⊂ Multiple TL ⊂ Context-free TL
→ Pumping lemma, swapping lemma, other tools?
Thank you!