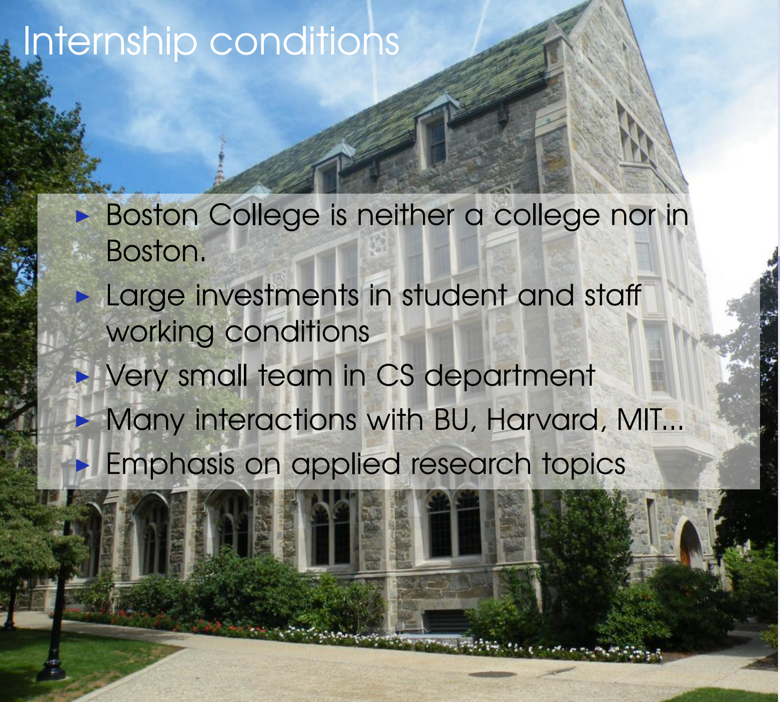
# An automaton model for forest algebras.

Antoine Delignat-Lavaud

Advised by Howard Straubing
Department of Computer Science, Boston College

September 1, 2010

# Internship conditions



- Boston College is neither a college nor in Boston.
- Large investments in student and staff working conditions
- Very small team in CS department
- Many interactions with BU, Harvard, MIT...
- Emphasis on applied research topics

# Internship goals and context

## Regular tree languages

- Starting point: finite ranked trees over a signature $A = \{a_1 : k_1 ... a_n : k_n\}$
- Ranked tree over $\{a : 0, b : 3, c : 1\}$

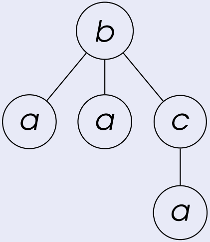



- Automaton model: $(A, Q, F \subseteq Q, \Delta)$. Transition in $\Delta$ are written $a_i(q_1, ..., q_{k_i}) \to q$.
- Trees are written in prefix form $b(a, a, c(a))$ but read bottom-up.

# Internship goals and context

## Regular tree languages

- Starting point: finite ranked trees over a signature $A = \{a_1 : k_1 ... a_n : k_n\}$
- Ranked tree over $\{a : 0, b : 3, c : 1\}$

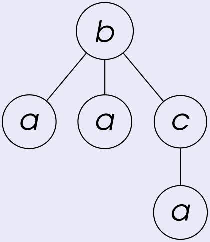



- Automaton model: $(A, Q, F \subseteq Q, \Delta)$. Transition in $\Delta$ are written $a_i(q_1, ..., q_{k_i}) \to q$.
- Trees are written in prefix form $b(a, a, c(a))$ but read bottom-up.

# Internship goals and context

## Regular tree languages

- Starting point: finite ranked trees over a signature $A = \{a_1 : k_1 ... a_n : k_n\}$
- Ranked tree over $\{a : 0, b : 3, c : 1\}$

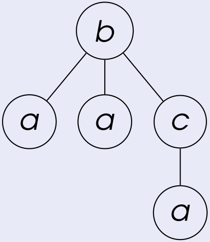



- Automaton model: $(A, Q, F \subseteq Q, \Delta)$. Transition in $\Delta$ are written $a_i(q_1, ..., q_{k_i}) \to q$.
- Trees are written in prefix form $b(a, a, c(a))$ but read bottom-up.

# Internship goals and context

## Regular tree languages

- Starting point: finite ranked trees over a signature $A = \{a_1 : k_1 ... a_n : k_n\}$
- Ranked tree over $\{a : 0, b : 3, c : 1\}$




- Automaton model: $(A, Q, F \subseteq Q, \Delta)$. Transition in $\Delta$ are written $a_i(q_1, ..., q_{k_i}) \to q$.
- Trees are written in prefix form $b(a, a, c(a))$ but read bottom-up.

## The word case

- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/{\equiv_L}$: there is a unique syntactic monoid $M_L$.

## The word case

- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/\equiv_L$: there is a unique syntactic monoid $M_L$.

## The word case

- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/\equiv_L$: there is a unique syntactic monoid $M_L$.

## The word case

- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/\equiv_L$: there is a unique syntactic monoid $M_L$.

## The word case

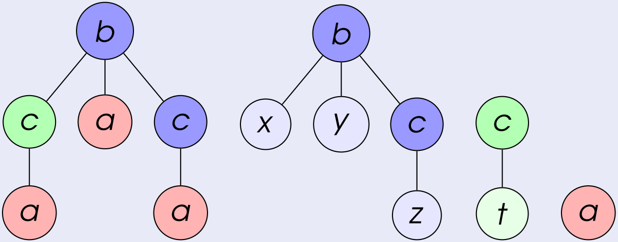
- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/\equiv_L$: there is a unique syntactic monoid $M_L$.

## The word case

- Language over $A$ = subset of the free monoid over $A$
- Universal property of the free monoid: any map $A \to S$ is uniquely extended to a morphism $A^* \to S$
- In a complete DFA accepting $L$, each letter $a \in A$ defines a transformation $f_a \in Q^Q$
- $\varphi : a \mapsto f_a$ is uniquely extended to a morphism from $A^*$ to $M = \langle f_a \rangle$. $L = \varphi^{-1}(X \subseteq M)$.
- Syntactic congruence: $w \equiv_L w'$ if $xwy \in L \Leftrightarrow xw'y \in L$
- Any morphism accepting $L$ factors through $A^* \to A^*/\equiv_L$: there is a unique syntactic monoid $M_L$.

# Back to trees

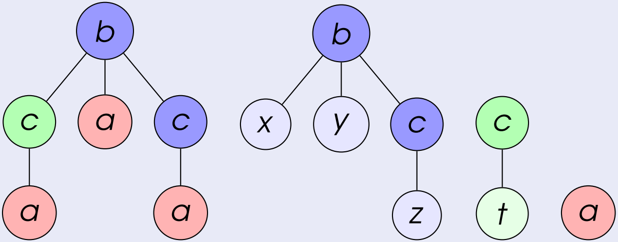
- What is a prefix, suffix, factor of a tree?



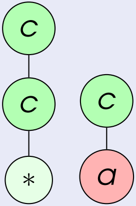
- Contexts: trees in wich one leaf is labelled by a variable $*$.

# Back to trees

- What is a prefix, suffix, factor of a tree?




- Contexts: trees in wich one leaf is labelled by a variable $*$.



## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/{\equiv_L}$, $F = [L]$. $a_i([t_1], ..., [t_{k_i}]) \rightarrow [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\not\Rightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \rightarrow Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/\equiv_L$, $F = [L]$. $a_i([t_1], \ldots, [t_{k_i}]) \rightarrow [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\not\Rightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \rightarrow Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/{\equiv_L}$, $F = [L]$. $a_i([t_1], \ldots, [t_{k_i}]) \rightarrow [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\nRightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \rightarrow Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/\equiv_L$, $F = [L]$. $a_i([t_1], ..., [t_{k_i}]) \to [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\not\Rightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \to Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/\equiv_L$, $F = [L]$. $a_i([t_1], ..., [t_{k_i}]) \rightarrow [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\not\Rightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \rightarrow Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

## Minimization and syntactic object

- Myhill-Nerode congruence: $t \equiv_L t'$ if for all contexts $p \in \mathcal{C}(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- Equivalently, coarsest congruence that refines $L$.
- Minimal automaton: $Q = \mathcal{T}(A)/\equiv_L$, $F = [L]$. $a_i([t_1], ..., [t_{k_i}]) \to [a_i(t_1 \cdots t_{k_i})]$.
- Minimal $\not\Rightarrow$ unique up to isomorphism!
- Syntactic structure: finite set $Q$ with maps $f_{a_i} : Q^{k_i} \to Q$
- Well understood if $k_i \leq 2$, impractical otherwise.

# Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$




Figure: Concatenation of forests

## Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$




Figure: Concatenation of forests

## Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$




Figure: Concatenation of forests

## Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$




Figure: Concatenation of forests

# Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$




Figure: Concatenation of forests

## Unranked trees

- Alphabet $A$ is a finite set of symbols
- Forest: $t ::= \varepsilon \mid a \in A \mid t + t \mid at$
- Unranked tree: $at$, $t$ a forest
- Important case for many applications (HTML, XML, databases)
- New horizontal dimension: $t + t \mid at$

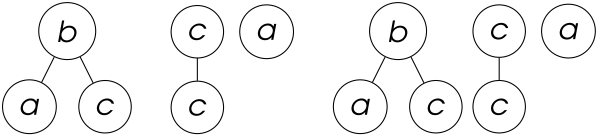



Figure: Concatenation of forests

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \rightarrow \mathcal{V}$ is bijective

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \rightarrow \mathcal{V}$ is bijective

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \rightarrow \mathcal{V}$ is bijective

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \to \mathcal{V}$ is bijective

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \to \mathcal{V}$ is bijective

# Usefulness of a simple syntactic object

## A tool to study properties of languages

- Schützenberger, McNaughton: $L$ star-free $\Leftrightarrow$ $M_L$ aperiodic $\Leftrightarrow$ $L$ $FO[<]$-definable
- Simon: $L$ piecewise-testable (union of $A^*a_1A^* \cdots A^*a_nA^*$) $\Leftrightarrow$ $M_L$ $\mathcal{J}$-trivial, i.e. $M_LmM_L = M_Lm'M_L \Rightarrow m = m'$

## Varieties

- Variety of monoids: family closed under submonoid, quotients and direct products
- Pseudovariety: variety $\mathbf{V}$ of finite monoids
- Variety of languages: family $\mathcal{V}(\Sigma)$ closed under booleans operations, inverses of morphisms and residuals
- Eilemberg's theorem: $\mathbf{V} \rightarrow \mathcal{V}$ is bijective

# Forest algebras

## Definition

A forest algebra is a tuple $(H, V, \cdot, \mathrm{in}_L, \mathrm{in}_R)$ such that:

- $H$ is the horizontal monoid denoted $(H, 0, +)$.
- $V$ is the vertical monoid denoted $(V, 1, \circ)$.
- $\cdot$ is a left monoidal action of $V$ on $H$.
- $\mathrm{in}_L, \mathrm{in}_R : H \to V$ are such that $\mathrm{in}_L(g) \cdot h = g + h$ and $\mathrm{in}_R(g) \cdot h = h + g$ for all $g, h \in H$.
- $\cdot$ is faithful, $\forall v, w \in V, \exists h \in H, v \cdot h \neq w \cdot h$.

Forest algebras were proposed by Walukiewicz and Bojańczyk in a 2007 paper.

## Forest algebra morphism

A morphism of forest algebras from $(H, V)$ to $(G, W)$ is a pair of monoid morphisms $(\alpha : H \to G, \beta : V \to W)$ such that

$$\forall h \in H, v \in V, \alpha(v \cdot h) = \beta(v) \cdot \alpha(h)$$

## Example

$(\mathcal{F}(A), \mathcal{C}(A), \cdot, \mathrm{in_L}, \mathrm{in_R})$ where

- $\mathcal{F}(A)$ is the monoid of forests over $A$ with concatenation
- $\mathcal{C}(A)$ is the monoid of contexts over $A$ with composition
- $\cdot$ is forest substitution in contexts

is the free forest algebra denoted $A^\Delta$. Universal property: any map $A \to V$ extends uniquely to a morphism $A^\Delta \to (H, V)$ such that $\beta(a(*)) = f(a)$

## Forest algebra morphism

A morphism of forest algebras from $(H, V)$ to $(G, W)$ is a pair of monoid morphisms $(\alpha : H \to G, \beta : V \to W)$ such that

$$\forall h \in H, v \in V, \alpha(v \cdot h) = \beta(v) \cdot \alpha(h)$$

## Example

$(\mathcal{F}(A), \mathcal{C}(A), \cdot, \mathrm{in_L}, \mathrm{in_R})$ where

- $\mathcal{F}(A)$ is the monoid of forests over $A$ with concatenation
- $\mathcal{C}(A)$ is the monoid of contexts over $A$ with composition
- $\cdot$ is forest substitution in contexts

is the free forest algebra denoted $A^\Delta$. Universal property: any map $A \to V$ extends uniquely to a morphism $A^\Delta \to (H, V)$ such that $\beta(a(*)) = f(a)$

# Recognizability

## Definition

$L \subseteq \mathcal{F}(A)$ is a recognized by a morphism $(\alpha, \beta) : A^\Delta \to (H, V)$ if $L = \alpha^{-1}(F \subseteq H)$. It is recognizable if there exists such a morphism such that $(H, V)$ is finite.

## Syntactic congruence

We define the relation $\equiv_L$ on forests and contexts:

$$t \equiv_L t' \iff \forall p \in \mathcal{C}(A), p \cdot t \in L \Leftrightarrow p \cdot t' \in L$$
$$p \equiv_L p' \iff \forall t \in \mathcal{F}(A), p \cdot t \in L \Leftrightarrow p' \cdot t \in L$$

The forest algebra $(H^L = \mathcal{F}(A)/{\equiv_L}, V^L = \mathcal{C}(A)/{\equiv_L})$ recognizes $L$ and any morphism recognizing $L$ factors through $(\alpha^L, \beta^L) : A^\Delta \to (H^L, V^L)$.

# Recognizability

## Definition

$L \subseteq \mathcal{F}(A)$ is a recognized by a morphism $(\alpha, \beta) : A^\Delta \to (H, V)$ if $L = \alpha^{-1}(F \subseteq H)$. It is recognizable if there exists such a morphism such that $(H, V)$ is finite.

## Syntactic congruence

We define the relation $\equiv_L$ on forests and contexts:

$$t \equiv_L t' \iff \forall p \in \mathcal{C}(A), p \cdot t \in L \Leftrightarrow p \cdot t' \in L$$
$$p \equiv_L p' \iff \forall t \in \mathcal{F}(A), p \cdot t \in L \Leftrightarrow p' \cdot t \in L$$

The forest algebra $(H^L = \mathcal{F}(A)/\equiv_L, V^L = \mathcal{C}(A)/\equiv_L)$ recognizes $L$ and any morphism recognizing $L$ factors through $(\alpha^L, \beta^L) : A^\Delta \to (H^L, V^L)$.

# Algebraic automaton model

## Definition

A forest algebra automaton is a tuple:

$$\mathcal{A} = (\langle Q, 0, + \rangle, A, \delta : A \times Q \to Q, F \subseteq Q)$$

$(Q, 0, +)$ is the finite state monoid, $\delta$ is the transition function and $F$ the set of accepting states.

## Run of the automaton

$\mathcal{A}$ induces a map $\mathcal{F}(A) \to Q$ written $\cdot^{\mathcal{A}}$ defined by induction on the structure of forests:

- $0^{\mathcal{A}} = 0$
- $(t_1 + t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} + t_2^{\mathcal{A}}$
- $(a \cdot t)^{\mathcal{A}} = \delta(a, t^{\mathcal{A}})$

$\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} \in F\}$.

# Algebraic automaton model

## Definition

A forest algebra automaton is a tuple:

$$\mathcal{A} = (\langle Q, 0, + \rangle, A, \delta : A \times Q \to Q, F \subseteq Q)$$

$(Q, 0, +)$ is the finite state monoid, $\delta$ is the transition function and $F$ the set of accepting states.

## Run of the automaton

$\mathcal{A}$ induces a map $\mathcal{F}(A) \to Q$ written $\cdot^{\mathcal{A}}$ defined by induction on the structure of forests:

- $0^{\mathcal{A}} = 0$
- $(t_1 + t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} + t_2^{\mathcal{A}}$
- $(a \cdot t)^{\mathcal{A}} = \delta(a, t^{\mathcal{A}})$

$\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} \in F\}$.

## Transition forest algebra

Transition forest algebra $(H, V)$ of $\mathcal{A} = \langle (Q, 0, +), A, \delta : A \times Q \to Q, F \subseteq Q \rangle$:

- Horizontal monoid $H = (Q, 0, +)$
- Vertical monoid $V = (H^H, \mathrm{id}_H, \circ)$
- Action is function application: $v \cdot h = v(h)$.
- Insertion functions are uniquely defined by the action.

The morphism defined by $\beta(a(*)) = \delta(a) \in H^H$ recognizes $L$.

## Transition forest algebra

Transition forest algebra $(H, V)$ of $\mathcal{A} = \langle (Q, 0, +), A, \delta : A \times Q \to Q, F \subseteq Q \rangle$:

- Horizontal monoid $H = (Q, 0, +)$
- Vertical monoid $V = (H^H, \mathrm{id}_H, \circ)$
- Action is function application: $v \cdot h = v(h)$.
- Insertion functions are uniquely defined by the action.

The morphism defined by $\beta(a(*)) = \delta(a) \in H^H$ recognizes $L$.

## Transition forest algebra

Transition forest algebra $(H, V)$ of $\mathcal{A} = \langle (Q, 0, +), A, \delta : A \times Q \to Q, F \subseteq Q \rangle$:

- Horizontal monoid $H = (Q, 0, +)$
- Vertical monoid $V = (H^H, \mathrm{id}_H, \circ)$
- Action is function application: $v \cdot h = v(h)$.
- Insertion functions are uniquely defined by the action.

The morphism defined by $\beta(a(*)) = \delta(a) \in H^H$ recognizes $L$.

## Transition forest algebra

Transition forest algebra $(H, V)$ of $\mathcal{A} = \langle (Q, 0, +), A, \delta : A \times Q \to Q, F \subseteq Q \rangle$:

- Horizontal monoid $H = (Q, 0, +)$
- Vertical monoid $V = (H^H, \mathrm{id}_H, \circ)$
- Action is function application: $v \cdot h = v(h)$.
- Insertion functions are uniquely defined by the action.

The morphism defined by $\beta(a(*)) = \delta(a) \in H^H$ recognizes $L$.

## Transition forest algebra

Transition forest algebra $(H, V)$ of $\mathcal{A} = \langle (Q, 0, +), A, \delta : A \times Q \rightarrow Q, F \subseteq Q \rangle$:

- Horizontal monoid $H = (Q, 0, +)$
- Vertical monoid $V = (H^H, \mathrm{id}_H, \circ)$
- Action is function application: $v \cdot h = v(h)$.
- Insertion functions are uniquely defined by the action.

The morphism defined by $\beta(a(*)) = \delta(a) \in H^H$ recognizes $L$.

# Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:

$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:

$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \rightarrow S, \lambda : S \times A \rightarrow Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:
$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \ldots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:

$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:
$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:

$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

## Definition

A bottom-up deterministic forest automaton (BUDFA) is a tuple:
$\mathcal{A} = \langle S, Q, s_0, \gamma : S \times Q \to S, \lambda : S \times A \to Q, F \rangle$

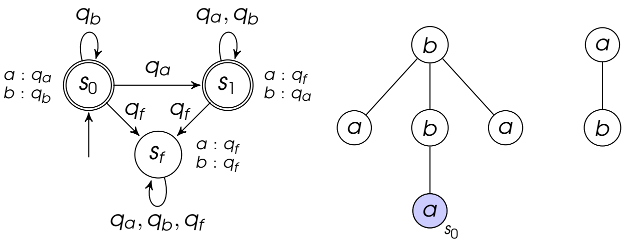
- $S$ is the finite set of horizontal states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of accepting states
- $Q$ is the finite set of vertical states
- $\gamma$ is a semiautomaton on $S$ over the alphabet $Q$
- $\lambda$ is the output function

$\gamma$ defines a left monoidal action of $Q^*$ on $S$, we will write $s \cdot q_1 \dots q_n$ instead of $\gamma(\gamma(\cdots\gamma(s, q_1)\cdots), q_{n-1}), q_n)$.

# Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

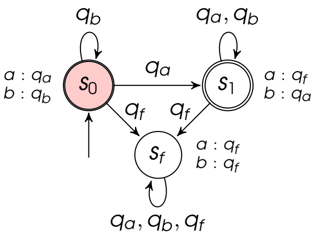
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

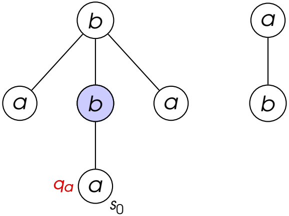


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

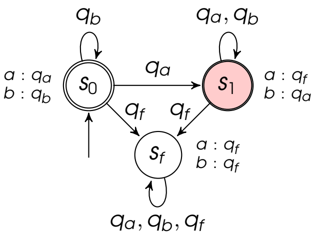


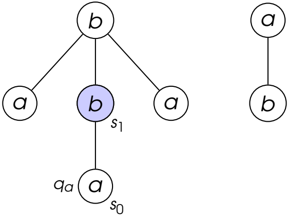



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

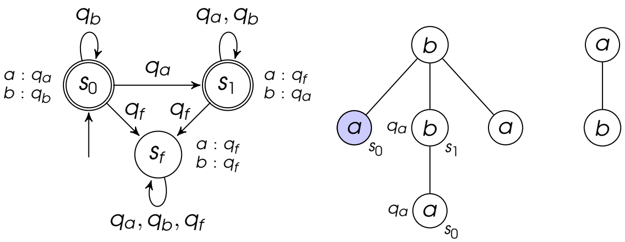


# Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

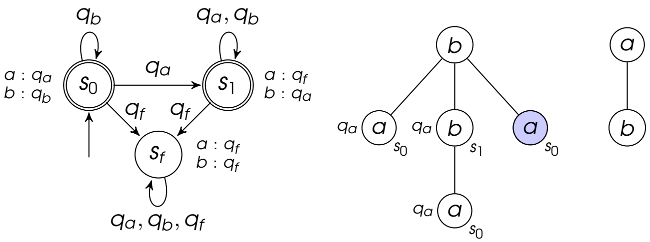
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

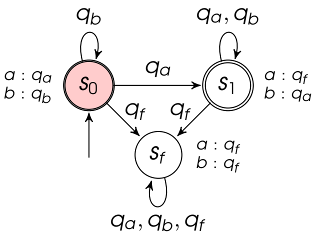
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

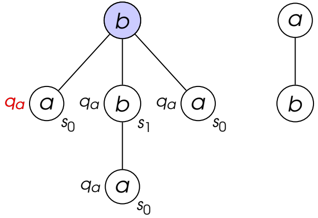


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

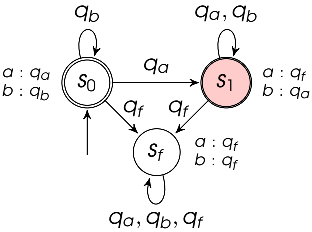


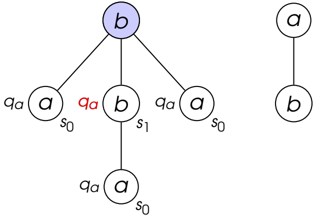



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

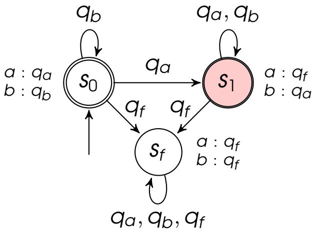


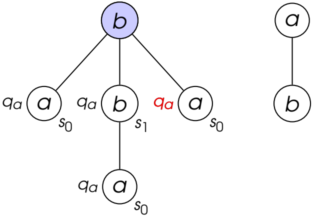



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

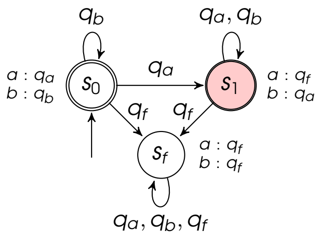


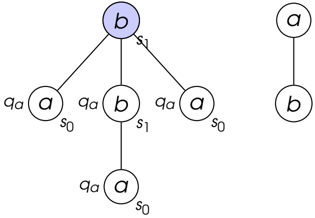



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

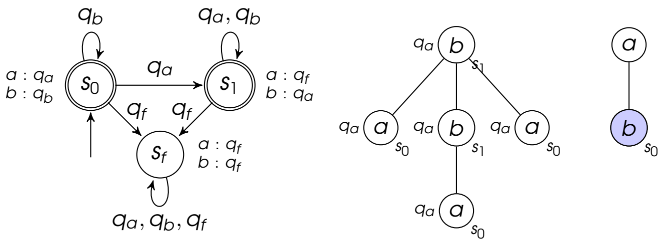


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

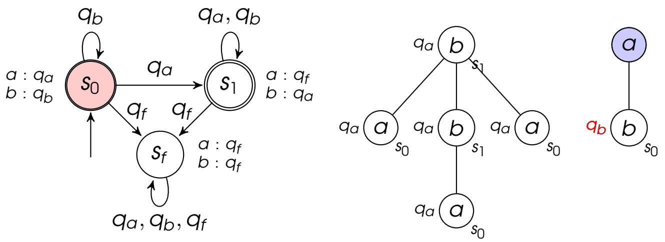
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

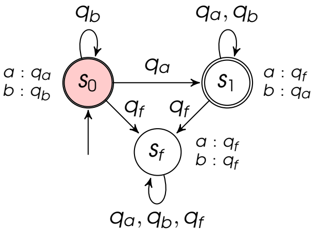
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

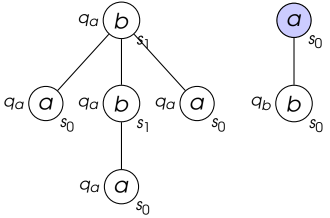


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

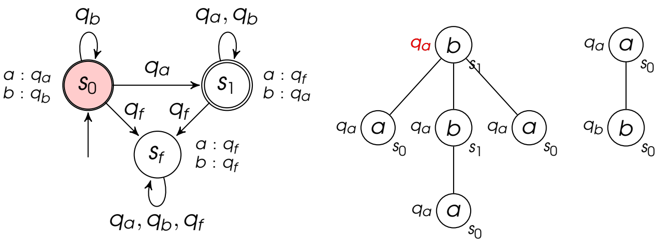


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

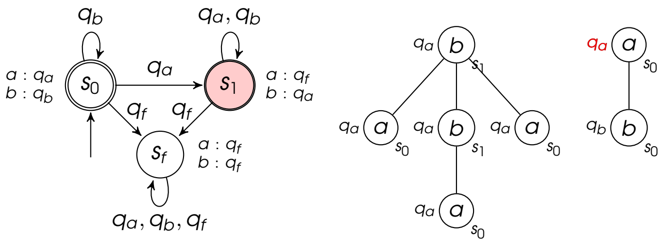
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.



## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

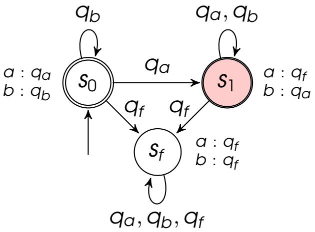
- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.

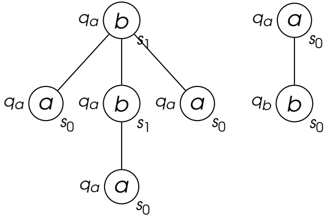


## Run on a BUDFA

$\mathcal{A}$ defines a map $\cdot^{\mathcal{A}} : \mathcal{F}(A) \to S$ defined by:

- $0^{\mathcal{A}} = s_0$.
- $(t_1 + a \cdot t_2)^{\mathcal{A}} = t_1^{\mathcal{A}} \cdot \lambda(t_2^{\mathcal{A}}, a)$.



## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0), a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in_L}(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in_R}(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in_R}(t)}, v_{\mathrm{in_L}(t)} \rangle$.

## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0), a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in}_L(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in}_R(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in}_R(t)}, v_{\mathrm{in}_L(t)} \rangle$.

## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0), a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in}_L(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in}_R(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in}_R(t)}, v_{\mathrm{in}_L(t)} \rangle$.

## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0),a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in}_L(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in}_R(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in}_R(t)}, v_{\mathrm{in}_L(t)} \rangle$.

## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0), a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in}_L(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in}_R(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in}_R(t)}, v_{\mathrm{in}_L(t)} \rangle$.

## Transition forest algebra

Let $\mathcal{A} = \langle S, Q, s_0, \gamma, \lambda, F \rangle$.

- $(S, Q, s_0, \gamma, F)$ is the horizontal DFA: let $H$ be its transition monoid
- For all $q \in Q$, let $\gamma_q = s \mapsto \gamma(s, q)$. $H$ is the submonoid of $(S^S, \mathrm{id}_S, *)$ generated by $(\gamma_q)_{q \in Q}$
- $X = \{f_w \in H \mid f_w(s_0) \in F\}$
- For all $a \in A$, let $v_a : h \mapsto \gamma_{\lambda(h(s_0), a)}$
- For all $t \in \mathcal{F}(A)$, let $v_{\mathrm{in_L}(t)} : h \mapsto \gamma_t * h$, $v_{\mathrm{in_R}(t)} : h \mapsto h * \gamma_t$
- $V$ is the submonoid of $(H^H, \mathrm{id}_H, \circ)$ generated by $\langle v_a, v_{\mathrm{in_R}(t)}, v_{\mathrm{in_L}(t)} \rangle$.

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

# Minimization of BUDFA

## Myhill-Nerode congruence

- Left contexts $\mathcal{C}_\ell(A)$: the hole has no sibling on its left
- Relation on $\mathcal{F}(A)$: $t \sim_L t'$ if $\forall p \in \mathcal{C}_\ell(A)$, $p \cdot t \in L \Leftrightarrow p \cdot t' \in L$.
- $\equiv_L \subseteq \sim_L$: $\sim_L$ is of finite index
- Restriction of $\equiv_L$ on $\mathcal{F}(A)$ to $\mathcal{T}(A)$ is an equivalence still denoted $\equiv_L$
- $\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$
- $\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$
- $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Myhill-Nerode congruence

- $t_1 \sim_L t_2 \Rightarrow at_1 \equiv_L at_2$:
  $p \in \mathcal{C}(A) \Rightarrow p \circ (a(*)) \in \mathcal{C}_\ell(A)$.
- $t_1 \sim_L t'_1$ and $t_2 \equiv_L t'_2 \Rightarrow t_1 + t_2 \sim_L t'_1 + t'_2$:
  $\forall p \in \mathcal{C}_\ell(A)$, since $p \circ \mathrm{in}_R(t_2) \in \mathcal{C}_\ell(A)$:
  $(p \circ \mathrm{in}_R(t_2)) \cdot t_1 \in L \Leftrightarrow (p \circ \mathrm{in}_R(t_2)) \cdot t'_1 \in L$
  $p \cdot (t_1 + t_2) \in L \Leftrightarrow p \cdot (t'_1 + t_2) \in L$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

## Myhill-Nerode congruence

- $t_1 \sim_L t_2 \Rightarrow at_1 \equiv_L at_2$:
  $p \in \mathcal{C}(A) \Rightarrow p \circ (a(*)) \in \mathcal{C}_\ell(A)$.
- $t_1 \sim_L t'_1$ and $t_2 \equiv_L t'_2 \Rightarrow t_1 + t_2 \sim_L t'_1 + t'_2$:
  $\forall p \in \mathcal{C}_\ell(A)$, since $p \circ \mathrm{in}_R(t_2) \in \mathcal{C}_\ell(A)$:
  $(p \circ \mathrm{in}_R(t_2)) \cdot t_1 \in L \Leftrightarrow (p \circ \mathrm{in}_R(t_2)) \cdot t'_1 \in L$
  $p \cdot (t_1 + t_2) \in L \Leftrightarrow p \cdot (t'_1 + t_2) \in L$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L}\rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Minimality proof

$T_s = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} = s\} \subseteq \mathcal{F}(A)$
$T_q = \{at \in \mathcal{F}(A) \mid \lambda(t^{\mathcal{A}}, a) = q\} \subseteq \mathcal{T}(A)$

- If $t_s, t'_s \in T_s$ then $t_s \sim_L t'_s$.
- If $t_q, t'_q \in T_q$ then $t_q \equiv_L t'_q$.
- $s \mapsto [T_s]_{\sim_L}$, $q \mapsto [T_q]_{\equiv_L}$ defines a surjective morphism from $\mathcal{A} = \langle S, s_0, Q, \gamma, \lambda, F\rangle$ to $\mathcal{A}_L$
- $s_0$ is mapped to $[0]_{\sim_L}$
- $F$ is mapped to $[L]_{\equiv_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Minimality proof

$T_s = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} = s\} \subseteq \mathcal{F}(A)$
$T_q = \{at \in \mathcal{F}(A) \mid \lambda(t^{\mathcal{A}}, a) = q\} \subseteq \mathcal{T}(A)$

- If $t_s, t'_s \in T_s$ then $t_s \sim_L t'_s$.
- If $t_q, t'_q \in T_q$ then $t_q \equiv_L t'_q$.
- $s \mapsto [T_s]_{\sim_L}, q \mapsto [T_q]_{\equiv_L}$ defines a surjective morphism from $\mathcal{A} = \langle S, s_0, Q, \gamma, \lambda, F \rangle$ to $\mathcal{A}_L$
- $s_0$ is mapped to $[0]_{\sim_L}$
- $F$ is mapped to $[L]_{\equiv_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Minimality proof

$T_s = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} = s\} \subseteq \mathcal{F}(A)$
$T_q = \{at \in \mathcal{F}(A) \mid \lambda(t^{\mathcal{A}}, a) = q\} \subseteq \mathcal{T}(A)$

- If $t_s, t'_s \in T_s$ then $t_s \sim_L t'_s$.
- If $t_q, t'_q \in T_q$ then $t_q \equiv_L t'_q$.
- $s \mapsto [T_s]_{\sim_L}$, $q \mapsto [T_q]_{\equiv_L}$ defines a surjective morphism from $\mathcal{A} = \langle S, s_0, Q, \gamma, \lambda, F \rangle$ to $\mathcal{A}_L$
- $s_0$ is mapped to $[0]_{\sim_L}$
- $F$ is mapped to $[L]_{\equiv_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Minimality proof

$T_s = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} = s\} \subseteq \mathcal{F}(A)$
$T_q = \{at \in \mathcal{F}(A) \mid \lambda(t^{\mathcal{A}}, a) = q\} \subseteq \mathcal{T}(A)$

- If $t_s, t'_s \in T_s$ then $t_s \sim_L t'_s$.
- If $t_q, t'_q \in T_q$ then $t_q \equiv_L t'_q$.
- $s \mapsto [T_s]_{\sim_L}, q \mapsto [T_q]_{\equiv_L}$ defines a surjective morphism from $\mathcal{A} = \langle S, s_0, Q, \gamma, \lambda, F \rangle$ to $\mathcal{A}_L$
- $s_0$ is mapped to $[0]_{\sim_L}$
- $F$ is mapped to $[L]_{\equiv_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Minimality proof

$T_s = \{t \in \mathcal{F}(A) \mid t^{\mathcal{A}} = s\} \subseteq \mathcal{F}(A)$
$T_q = \{at \in \mathcal{F}(A) \mid \lambda(t^{\mathcal{A}}, a) = q\} \subseteq \mathcal{T}(A)$

- If $t_s, t'_s \in T_s$ then $t_s \sim_L t'_s$.
- If $t_q, t'_q \in T_q$ then $t_q \equiv_L t'_q$.
- $s \mapsto [T_s]_{\sim_L}, q \mapsto [T_q]_{\equiv_L}$ defines a surjective morphism from $\mathcal{A} = \langle S, s_0, Q, \gamma, \lambda, F \rangle$ to $\mathcal{A}_L$
- $s_0$ is mapped to $[0]_{\sim_L}$
- $F$ is mapped to $[L]_{\equiv_L}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $V_a : \gamma_s \mapsto \gamma_{as}, V_{\mathrm{in}_L(t)} : \gamma_s \mapsto \gamma_{t+s}, V_{\mathrm{in}_R(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$, $\mathrm{in}_R(t)$, $\mathrm{in}_L(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $V_a : \gamma_s \mapsto \gamma_{as}, V_{\mathrm{in_L}(t)} : \gamma_s \mapsto \gamma_{t+s}, V_{\mathrm{in_R}(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$,$\mathrm{in_R}(t)$, $\mathrm{in_L}(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}$, $\gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $V_a : \gamma_s \mapsto \gamma_{as}$, $V_{\mathrm{in_L}(t)} : \gamma_s \mapsto \gamma_{t+s}$, $V_{\mathrm{in_R}(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $V_p$ is defined on the decomposition of $p$ using $a(*)$, $\mathrm{in_R}(t)$, $\mathrm{in_L}(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = V_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff V_{p_1} = V_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $v_a : \gamma_s \mapsto \gamma_{as}$, $v_{\mathrm{in_L}(t)} : \gamma_s \mapsto \gamma_{t+s}$, $v_{\mathrm{in_R}(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$, $\mathrm{in_R}(t)$, $\mathrm{in_L}(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L}\rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $V_a : \gamma_s \mapsto \gamma_{as}, V_{\mathrm{in_L}(t)} : \gamma_s \mapsto \gamma_{t+s}, V_{\mathrm{in_R}(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$, $\mathrm{in_R}(t)$, $\mathrm{in_L}(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $v_a : \gamma_s \mapsto \gamma_{as}$, $v_{\mathrm{in}_L(t)} : \gamma_s \mapsto \gamma_{t+s}$, $v_{\mathrm{in}_R(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$, $\mathrm{in}_R(t)$, $\mathrm{in}_L(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/{\sim_L}, \mathcal{T}(A)/{\equiv_L}, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $v_a : \gamma_s \mapsto \gamma_{as}, v_{\mathrm{in}_L(t)} : \gamma_s \mapsto \gamma_{t+s}, v_{\mathrm{in}_R(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$,$\mathrm{in}_R(t)$, $\mathrm{in}_L(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Minimal automaton

$$\mathcal{A}_L = \langle \mathcal{F}(A)/\sim_L, \mathcal{T}(A)/\equiv_L, [0]_{\sim_L}, \gamma, \lambda, [L]_{\sim_L} \rangle$$

$$\lambda([t]_{\sim_L}, a) = [at]_{\equiv_L}, \gamma([t_1]_{\sim_L}, [t_2]_{\equiv_L}) = [t_1 + t_2]_{\sim_L}$$

## Transition forest algebra

- $\gamma_t$ instead of $\gamma_{[t]_{\equiv_L}}$ if $t \in \mathcal{T}(A)$
- $\gamma_{at}$ instead of $\gamma_{\lambda(\gamma_t([0]_{\sim_L}),a)}$
- $\gamma_{t_1} * \gamma_{t_2} = \gamma_{t_1+t_2}$
- $[t_1]_{\equiv_L} = [t_2]_{\equiv_L} \iff \gamma_{t_1} = \gamma_{t_2}$
- $v_a : \gamma_s \mapsto \gamma_{as}, v_{\mathrm{in_L}(t)} : \gamma_s \mapsto \gamma_{t+s}, v_{\mathrm{in_R}(t)} : \gamma_s \mapsto \gamma_{s+t}$
- If $p \in \mathcal{C}(A)$, $v_p$ is defined on the decomposition of $p$ using $a(*)$,$\mathrm{in_R}(t)$, $\mathrm{in_L}(t)$
- $\forall t \in \mathcal{F}(A), p \in \mathcal{C}(A), \gamma_{p \cdot t} = v_p(\gamma_t)$
- $[p_1]_{\equiv_L} = [p_2]_{\equiv_L} \iff v_{p_1} = v_{p_2}$

## Theorem

Every recognizable forest language is accepted by an unique (up to isomorphism) BUDFA whose transition forest algebra is equal to the syntactic forest algebra of the language.

## Corollary

Every forest algebra has a canonical representation in which the vertical monoid is a transformation monoid over the horizontal monoid and the action is function application.

## Remark

If $\mathcal{A}$ is a BUDFA such that $|S| = n$, $|V| \leq n^{2n}(1 + n^{n^2})$
In other words, $|V^L| = \mathcal{O}(|H^L|^{\dim(H^L)})$, a much lower bound than the expected $|H^L|^{|H^L|}$

## Theorem

Every recognizable forest language is accepted by an unique (up to isomorphism) BUDFA whose transition forest algebra is equal to the syntactic forest algebra of the language.

## Corollary

Every forest algebra has a canonical representation in which the vertical monoid is a transformation monoid over the horizontal monoid and the action is function application.

## Remark

If $\mathcal{A}$ is a BUDFA such that $|S| = n$, $|V| \leq n^{2n}(1 + n^{n^2})$
In other words, $|V^L| = \mathcal{O}(|H^L|^{\dim(H^L)})$, a much lower bound than the expected $|H^L|^{|H^L|}$

## Theorem

Every recognizable forest language is accepted by an unique (up to isomorphism) BUDFA whose transition forest algebra is equal to the syntactic forest algebra of the language.

## Corollary

Every forest algebra has a canonical representation in which the vertical monoid is a transformation monoid over the horizontal monoid and the action is function application.

## Remark

If $\mathcal{A}$ is a BUDFA such that $|S| = n$, $|V| \leq n^{2n}(1 + n^{n^2})$

In other words, $|V^L| = \mathcal{O}(|H^L|^{\dim(H^L)})$, a much lower bound than the expected $|H^L|^{|H^L|}$

# Logical characterization

## $FO[N, C]$

$\varphi ::= x = y \mid N(x, y) \mid C(x, y) \mid L_a(x), a \in A \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x \varphi$

- Forest over $A$: $\mathcal{P}os(t) \subseteq \mathbb{N}^* \to A$
- Domain of interpretation $\mathcal{I}$ is $\mathbb{N}^*$
- $\forall p, p' \in \mathbb{N}^*, N^{\mathcal{I}}(p, p') \Leftrightarrow \exists p'' \in \mathbb{N}^*, \exists i \in \mathbb{N}, p = p'' \cdot i \wedge p' = p'' \cdot (i + 1)$
- $\forall p, p' \in \mathbb{N}^*, C^{\mathcal{I}}(p, p') \Leftrightarrow \exists i \in \mathbb{N}, p = p' \cdot i$
- $\mathcal{V}$-forests: forests over $A \times 2^{\mathcal{V}}$ given $\mathcal{V}$ finite set of first-order variables
- $\{(a_i, U_i), i \leq n\}$ set of labels in $t$: $\forall i \neq j, U_i \cap U_j = \emptyset$ and $\bigcup_{i \leq n} U_i = \mathcal{V}$

## Model using $\mathcal{V}$-forests

- $t \models_{\mathcal{I}} L_a(x)$ if and only if $\exists p \in \mathcal{P}\mathrm{os}(t), t(p) = (a, U)$ with $x \in U$.
- If $P$ is a binary predicate, $t \models_{\mathcal{I}} P(x, y)$ if and only if $P^{\mathcal{I}}(p_x, p_y)$, where $p_x$ is the position such that $t(p_x) = (a_i, U_i)$ with $x \in U_i$.
- $t \models_{\mathcal{I}} \varphi_1 \wedge \varphi_2$ if and only if $t \models_{\mathcal{I}} \varphi_1$ and $t \models_{\mathcal{I}} \varphi_2$.
- $t \models_{\mathcal{I}} \neg\varphi$ if and only if $t \not\models_{\mathcal{I}} \varphi$.
- $t \models_{\mathcal{I}} \exists x \varphi$ if and only if $\exists i \leq n, t_i \models_{\mathcal{I}} \varphi$ where $t_i$ is the $\mathcal{V} \cup \{x\}$-forest obtained by replacing the label $(a_i, U_i)$ in $t$ by $(a_i, U_i \cup \{x\})$.

## $MSO[N, C]$

$\Phi ::= \varphi \in FO[N, C] \mid X(x) \mid \exists X \Phi$

- $(\mathcal{V}, \mathcal{W})$-forests over $A \times 2^{\mathcal{V}} \times 2^{\mathcal{W}}$
- Restriction to labels in $A \times 2^{\mathcal{V}}$ is a $\mathcal{V}$-forest
- $\mathcal{W}$ is a finite set of second-order variables
- $t \models_{\mathcal{I}} \varphi \in FO[N, C]$ if and only if the restriction of $t$ to labels in $A \times 2^{\mathcal{V}}$ is a model of $\varphi$.
- $t \models_{\mathcal{I}} X(x) \Leftrightarrow \exists p \in \mathcal{P}\mathrm{os}(t)$ such that $t(p) = (a_i, V_i, W_i)$ with $x \in V_i$ and $X \in W_i$
- $t \models_{\mathcal{I}} \exists X \Phi \Leftrightarrow \exists P \subseteq \mathcal{P}\mathrm{os}(t)$ such that the $(\mathcal{V}, \mathcal{W} \cup \{X\})$-forest obtained by replacing all labels $t(p) = (a_p, V_p, W_p)$ for $p \in P$ by $(a_p, V_p, W_p \cup \{X\})$ satisfies $\Phi$

# From $MSO[N, C]$ to BUNFA

- $\Phi = L_a(x)$: a BUDFA with 2 horizontal and 2 vertical states can easily test if there is a label $(a, V, W)$ with $x \in V$. The intersection with $\mathcal{A}_V$ for $\mathcal{V}$-forests accepts $L_\Phi$
- Cases $\Phi = N(x, y)$, $\Phi = C(x, y)$, $\Phi = (x = y)$ and $\Phi = X(x)$ are easy
- Cases $\Phi = \Phi_1 \wedge \Phi_2$, $\Phi = \neg\Psi$: intersection construction and final state complementation
- Cases $\Phi = \exists x \Psi$ and $\Phi = \exists X, \Psi$ require the construction of a BUNFA, but it can be minimized at each inductive step.

# Implementation of BUDFA

## The `forestalg` GAP package

We implemented support for forest algebras and BUDFA in GAP, a formal computational discrete algebra system.

- Accepts BUDFA, BUNFA, forest algebras as input
- Offers determinization, trimming, minimization, intersection and union, computation of the transition forest algebras
- Can compute and draw Green's relations on the horizontal and vertical monoids and the Cayley graphs
- Can draw automata and their algebraic representation

# Implementation of BUDFA

## The `forestalg` GAP package

We implemented support for forest algebras and BUDFA in GAP, a formal computational discrete algebra system.

- Accepts BUDFA, BUNFA, forest algebras as input
- Offers determinization, trimming, minimization, intersection and union, computation of the transition forest algebras
- Can compute and draw Green's relations on the horizontal and vertical monoids and the Cayley graphs
- Can draw automata and their algebraic representation

# Implementation of BUDFA

## The `forestalg` GAP package

We implemented support for forest algebras and BUDFA in GAP, a formal computational discrete algebra system.

- Accepts BUDFA, BUNFA, forest algebras as input
- Offers determinization, trimming, minimization, intersection and union, computation of the transition forest algebras
- Can compute and draw Green's relations on the horizontal and vertical monoids and the Cayley graphs
- Can draw automata and their algebraic representation

# Implementation of BUDFA

## The `forestalg` GAP package

We implemented support for forest algebras and BUDFA in GAP, a formal computational discrete algebra system.

- Accepts BUDFA, BUNFA, forest algebras as input
- Offers determinization, trimming, minimization, intersection and union, computation of the transition forest algebras
- Can compute and draw Green's relations on the horizontal and vertical monoids and the Cayley graphs
- Can draw automata and their algebraic representation

**Theorem**

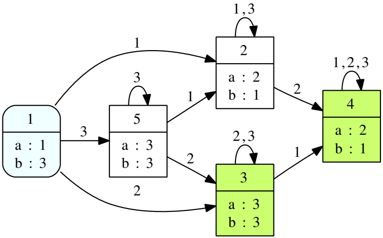
The minimal BUDFA of $\mathcal{A}$ can be computed in $\mathcal{O}(|\mathcal{A}| \log |\mathcal{A}|)$

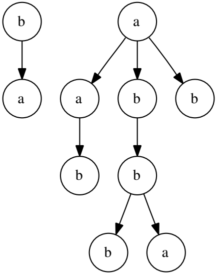
**Generalized Hopcroft algorithm**

1: $\sim := \{F, F^C\} \uplus \{Q\}$
2: **while** $\exists (C, x)$ with $C \in \sim$ and $x \in S \uplus Q \uplus A$ such that $obj(C, x, \sim) \neq \emptyset$ **do**
3: **for all** $B \in obj(C, x, \sim)$ **do**
4: Replace $B$ with $B_{(C,x)}$ and $B^{(C,x)}$ in $\sim$
5: **end for**
6: **end while**

# Drawing automata and forests

```
gap> A := ExistsPathInLanguageAutomaton(RationalExpression("ab*a"),true);
<Forest automaton on {ab}>< deterministic automaton on 5 letters with 8 states >
gap> A := ExistsPathInLanguageAutomaton(RationalExpression("ab*a"),true);
<Forest automaton on {ab}>< deterministic automaton on 3 letters with 5 states >
gap> f := Forest("ba+a(ab+bb(b+a)+b)");;
gap> ForestAutomatonAccepts(A, f);
true
gap> DrawForest(f);; DrawForestAutomaton(A);;
```



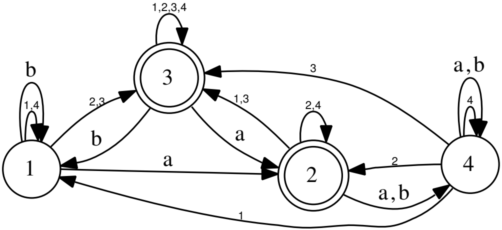






Antoine Delignat-Lavaud

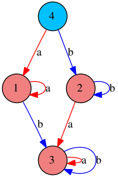
# Drawing the algebraic automaton

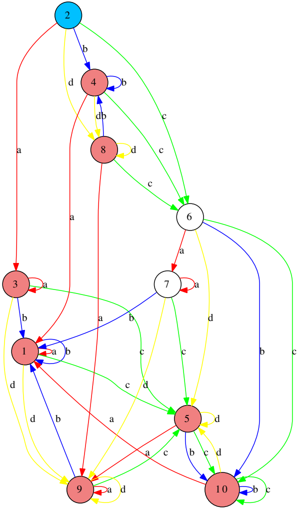
```
gap> DrawForestAutomatonAction(A);;
```



# Drawing Cayley graphs

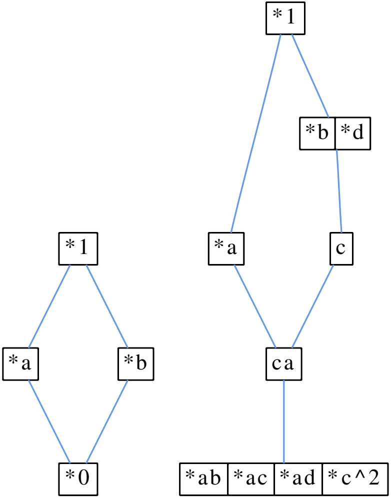
```
gap> HV := SyntacticForestAlgebra(A);
<H has 3 generators and 4 elements, V has 5 generators>
gap> DrawCayleyGraph(ForestHorizontalMonoid(HV));;
gap> DrawCayleyGraph(ForestVerticalMonoid(HV));;
```



# Drawing Green's relations

```
gap> DrawDClasses(ForestHorizontalMonoid(HV));;
gap> DrawDClasses(ForestVerticalMonoid(HV));;
```



# Conclusion and perspectives

## Summary of our results

- A robust and simple automaton model for forest algebras
- An implementation of this model for the study of algebraic properties of forest languages
- A loglinear minimization algorithm for BUDFA and other automaton models on unranked trees
- An afficient way to compute the syntactic forest algebra of a logical sentence

# Conclusion and perspectives

## Summary of our results

- A robust and simple automaton model for forest algebras
- An implementation of this model for the study of algebraic properties of forest languages
- A loglinear minimization algorithm for BUDFA and other automaton models on unranked trees
- An afficient way to compute the syntactic forest algebra of a logical sentence

# Conclusion and perspectives

## Summary of our results

- A robust and simple automaton model for forest algebras
- An implementation of this model for the study of algebraic properties of forest languages
- A loglinear minimization algorithm for BUDFA and other automaton models on unranked trees
- An afficient way to compute the syntactic forest algebra of a logical sentence

# Conclusion and perspectives

## Summary of our results

- A robust and simple automaton model for forest algebras
- An implementation of this model for the study of algebraic properties of forest languages
- A loglinear minimization algorithm for BUDFA and other automaton models on unranked trees
- An afficient way to compute the syntactic forest algebra of a logical sentence

## Further research

- Word languages: $FO[<]$-definability $\Leftrightarrow$ aperiodicity $\Leftrightarrow$ wreath product of $U_2$
- Right generalization of aperiodicity in forest algebras?
- A Krohn-Rhodes theorem for trees: decomposition of forest algebras into wreath products

## Further research

- Word languages: $FO[<]$-definability $\Leftrightarrow$ aperiodicity $\Leftrightarrow$ wreath product of $U_2$
- Right generalization of aperiodicity in forest algebras?
- A Krohn-Rhodes theorem for trees: decomposition of forest algebras into wreath products

## Further research

- Word languages: $FO[<]$-definability $\Leftrightarrow$ aperiodicity $\Leftrightarrow$ wreath product of $U_2$
- Right generalization of aperiodicity in forest algebras?
- A Krohn-Rhodes theorem for trees: decomposition of forest algebras into wreath products