Dynamic Membership for Regular Tree Languages

Antoine Amarilli, Corentin Barloy, Louis Jachiet, Charles Paperman

Incremental maintenance of words

a	b	b	a	b	a	a	b	$\in$ Even

Incremental maintenance of words

a	b	b	a	b	a	b	b	$\notin$ Even

Incremental maintenance of words

a	b	b	a	b	a	b	a	$\in$ Even

Incremental maintenance of words

a	b	a	a	b	a	b	a	$\notin$ Even

Incremental maintenance of words

a	b	a	a	b	a	b	a	$\notin$ Even

$\rightarrow$ use auxiliary data structures

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

→ use auxiliary data structures

  → for Even: flip a bit

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

$\rightarrow$ use auxiliary data structures

  $\rightarrow$ for Even: flip a bit

$\rightarrow$ Here: RAM model with linear preprocessing

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

- $\rightarrow$ use auxiliary data structures
  - $\rightarrow$ for Even: flip a bit
- $\rightarrow$ Here: RAM model with linear preprocessing
  - $\rightarrow$ What is the time needed to recompute membership?

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

$\rightarrow$ use auxiliary data structures

$\rightarrow$ for Even: flip a bit

$\rightarrow$ Here: RAM model with linear preprocessing

$\rightarrow$ What is the time needed to recompute membership?

Complete answer for regular languages:

Theorem (Amarilli, Jachiet, Paperman)

A regular language L can either be maintained in:

- constant time
- $\Theta(\log\log(n))$ time. (Conditional)
- $\Theta(\log(n)/\log\log(n))$ time

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

→ use auxiliary data structures

→ for Even: flip a bit

→ Here: RAM model with linear preprocessing

→ What is the time needed to recompute membership?

Complete answer for regular languages:

Theorem (Amarilli, Jachiet, Paperman)

A regular language L can either be maintained in:

- constant time
- $\Theta(\log\log(n))$ time. (Conditional)
- $\Theta(\log(n)/\log\log(n))$ time

→ based on algebraic properties

Incremental maintenance of words

a	b	a	a	b	a	b	a

$\notin$ Even

→ use auxiliary data structures

→ for Even: flip a bit

→ Here: RAM model with linear preprocessing

→ What is the time needed to recompute membership?

Complete answer for regular languages:

Theorem (Amarilli, Jachiet, Paperman)

A regular language L can either be maintained in:

- constant time
- $\Theta(\log\log(n))$ time. (Conditional)
- $\Theta(\log(n)/\log\log(n))$ time

→ based on algebraic properties

→ Decidable

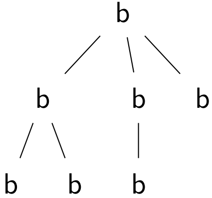
Forest languages

Incremental maintenance of forests

→ Labelled ordered unranked forests

Incremental maintenance of forests

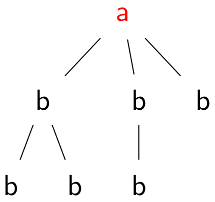
$\rightarrow$ Labelled ordered unranked forests



$\in$ Antichain

Incremental maintenance of forests

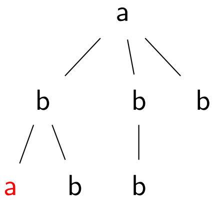
$\rightarrow$ Labelled ordered unranked forests



$\in$ Antichain

Incremental maintenance of forests

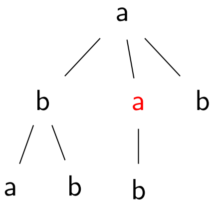
→ Labelled ordered unranked forests



$\notin$ Antichain

Incremental maintenance of forests

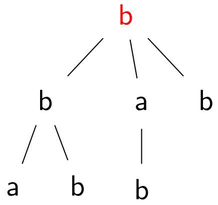
→ Labelled ordered unranked forests



$\notin$ Antichain

Incremental maintenance of forests

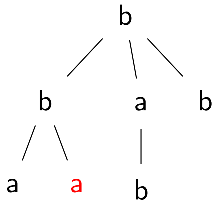
→ Labelled ordered unranked forests



$\in$ Antichain

Incremental maintenance of forests

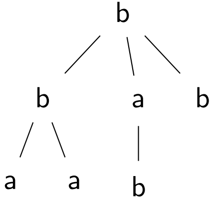
→ Labelled ordered unranked forests



$\in$ Antichain

Incremental maintenance of forests

→ Labelled ordered unranked forests

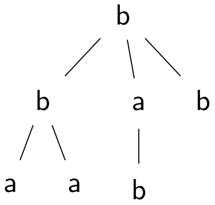


$\in$ Antichain

→ Fixed structure

Incremental maintenance of forests

→ Labelled ordered unranked forests



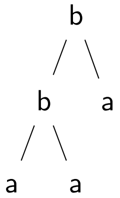
$\in$ Antichain

→ Fixed structure

→ Technicality: presence of neutral letter (here b)

Incremental maintenance of forests

$\rightarrow$ Labelled ordered unranked forests



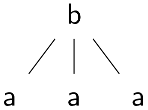
$\in$ Antichain

$\rightarrow$ Fixed structure

$\rightarrow$ Technicality: presence of neutral letter (here b)

Incremental maintenance of forests

→ Labelled ordered unranked forests



$\in$ Antichain

→ Fixed structure

→ Technicality: presence of neutral letter (here b)

Incremental maintenance of forests

$\rightarrow$ Labelled ordered unranked forests

a a a $\in$ Antichain

$\rightarrow$ Fixed structure

$\rightarrow$ Technicality: presence of neutral letter (here b)

Incremental maintenance of forests

→ Labelled ordered unranked forests

a a a $\in$ Antichain

→ Fixed structure
→ Technicality: presence of neutral letter (here b)
→ No proven trichotomy

Incremental maintenance of forests

→ Labelled ordered unranked forests

a a a ∈ Antichain

→ Fixed structure
→ Technicality: presence of neutral letter (here b)
→ No proven trichotomy

Theorem (this talk)

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- There is a decidable characterization of regular languages of forests that can be maintained in constant time

Forest algebras

finite word automaton $\approx$ finite monoid $(M, \cdot)$

Forest algebras

finite set   associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

Forest algebras

finite set   associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$

Forest algebras

finite set — associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

Forest algebras

finite set   associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$

$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

Forest algebras

finite set

associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

finite tree automaton $\approx$ finite ???

Forest algebras

finite set / associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

finite tree automaton $\approx$ finite ???

Forests grows horizontally and vertically: two operations

Forest algebras

finite set　　associative operation

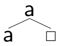
finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

finite tree automaton $\approx$ finite ???

Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

Forest algebras

finite set → associative operation

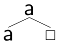
finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a,b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

finite tree automaton $\approx$ finite ???

Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

$\to$ two-sorted algebra

Forest algebras

finite set   associative operation

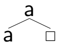
finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$

Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

$\to$ two-sorted algebra

Forest algebras

finite set   associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

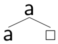
recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

contexts
forests

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$

Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

$\to$ two-sorted algebra

Forest algebras

finite word automaton $\approx$ finite monoid $(M, \cdot)$

finite set

associative operation

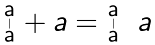
recognize same languages

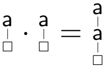
Take $\mu\colon \{a,b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1)\cdots\mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

contexts

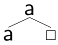
forests

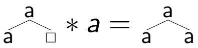
finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$





Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 
$\to$ two-sorted algebra



Forest algebras

finite word automaton $\approx$ finite monoid $(M, \cdot)$

finite set

associative operation

recognize same languages

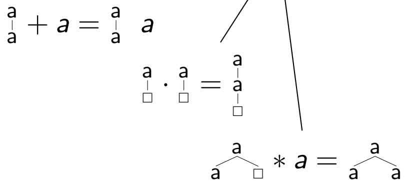
Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

forests

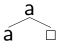
contexts

other stuff

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$



Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

$\to$ two-sorted algebra

Forest algebras

finite set — associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

forests — contexts — other stuff

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$

$\begin{smallmatrix} a \\ | \\ a \end{smallmatrix} + a = \begin{smallmatrix} a \\ | \\ a \end{smallmatrix} \ a$

Forests grows horizontally and vertically: two operations

$\begin{smallmatrix} a \\ | \\ \square \end{smallmatrix} \cdot \begin{smallmatrix} a \\ | \\ \square \end{smallmatrix} = \begin{smallmatrix} a \\ | \\ a \\ | \\ \square \end{smallmatrix}$

How to add two trees vertically? $\to$ use contexts: ie. $\overset{a}{a \quad \square}$
$\to$ two-sorted algebra

$\overset{a}{a \quad \square} * a = \overset{a}{a \quad a}$

Take $\mu\colon \{a, b\} \to H, \{\begin{smallmatrix} a \\ | \\ \square \end{smallmatrix}, \begin{smallmatrix} b \\ | \\ \square \end{smallmatrix}\} \to V$ extended to all forests and contexts

Forest algebras

finite set ... associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

forests ... contexts ... other stuff

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$

$\substack{a \\ | \\ a} + a = \substack{a \\ | \\ a} \;\; a$

$\substack{a \\ | \\ \square} \cdot \substack{a \\ | \\ \square} = \substack{a \\ | \\ a \\ | \\ \square}$

$\substack{a \\ \diagup \;\; \diagdown \\ a \quad\; \square} * a = \substack{a \\ \diagup \;\; \diagdown \\ a \quad\; a}$

Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. $\substack{a \\ \diagup \;\; \diagdown \\ a \quad\; \square}$
$\to$ two-sorted algebra

Take $\mu\colon \{a, b\} \to H, \{\substack{a \\ | \\ \square}, \substack{b \\ | \\ \square}\} \to V$ extended to all forests and contexts
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq H$

Forest algebras

finite set — associative operation

finite word automaton $\approx$ finite monoid $(M, \cdot)$

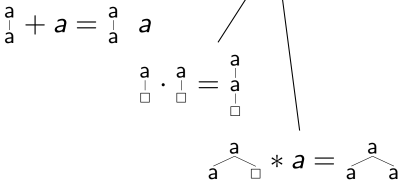
recognize same languages

Take $\mu\colon \{a, b\} \to M$ extended to Σ^* by $\mu(a_1 \cdots a_n) = \mu(a_1) \cdots \mu(a_n)$
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq M$

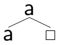
forests — contexts — other stuff

finite tree automaton $\approx$ finite forest algebra $(H, +, V, \cdot, *, \cdots)$

recognize same languages



Forests grows horizontally and vertically: two operations

How to add two trees vertically? $\to$ use contexts: ie. 

$\to$ two-sorted algebra

Take $\mu\colon \{a, b\} \to H, \{\overset{a}{\square}, \overset{b}{\square}\} \to V$ extended to all forests and contexts
$\to$ L recognized by μ: $L = \mu^{-1}(P)$ for $P \subseteq H$

Examples

$\rightarrow$ Even on words is recognized by $(\{0,1\}, +$ mod 2) via $\mu(w) = \#_a(w)$ mod 2

Examples

$\rightarrow$ Even on words is recognized by $(\{0,1\}, + \text{ mod } 2)$ via $\mu(w) = \#_a(w) \text{ mod } 2$

$\rightarrow$ Even on forests is recognized by:

- $H = V = \{0,1\}$
- $+ = \cdot = * = (+ \text{ mod } 2)$
- $\mu = \#_a \text{ mod } 2$

Examples

→ Even on words is recognized by $(\{0,1\}, + \bmod 2)$ via $\mu(w) = \#_a(w) \bmod 2$

→ Even on forests is recognized by:

- $H = V = \{0,1\}$
- $+ = \cdot = * = (+ \bmod 2)$
- $\mu = \#_a \bmod 2$

→ Antichain is recognized by:

- $H = \{\varepsilon, a, \begin{smallmatrix} a \\ | \\ a \end{smallmatrix}\}$, $V = \{\Box, a + \Box, \begin{smallmatrix} a \\ | \\ \Box \end{smallmatrix}, \begin{smallmatrix} a \\ | \\ a \\ | \\ \Box \end{smallmatrix}\}$

Examples

→ Even on words is recognized by $(\{0,1\}, + \bmod 2)$ via $\mu(w) = \#_a(w) \bmod 2$

→ Even on forests is recognized by:

- $H = V = \{0,1\}$
- $+ = \cdot = * = (+ \bmod 2)$
- $\mu = \#_a \bmod 2$

→ Antichain is recognized by:

- $H = \{\varepsilon, a, \begin{smallmatrix} a \\ | \\ a \end{smallmatrix}\}$, $V = \{\square, a + \square, \begin{smallmatrix} a \\ | \\ \square \end{smallmatrix}, \begin{smallmatrix} a \\ | \\ a \\ | \\ \square \end{smallmatrix}\}$

- $\mu(\text{no } a) = \varepsilon$
- $\mu(\text{antichain}) = a$
- $\mu(\text{comparable } a) = \begin{smallmatrix} a \\ | \\ a \end{smallmatrix}$

- $\mu(\text{no } a) = \square$
- $\mu(\text{antichain } + \text{no } a \text{ before } \square) = a + \square$
- $\mu(\text{antichain } + a \text{ before } \square) = \begin{smallmatrix} a \\ | \\ \square \end{smallmatrix}$
- $\mu(\text{comparable } a) = \begin{smallmatrix} a \\ | \\ a \\ | \\ \square \end{smallmatrix}$

Examples

→ Even on words is recognized by $(\{0,1\}, + \bmod 2)$ via $\mu(w) = \#_a(w) \bmod 2$

→ Even on forests is recognized by:

- $H = V = \{0,1\}$
- $+ = \cdot = * = (+ \bmod 2)$
- $\mu = \#_a \bmod 2$

→ Antichain is recognized by:

- $H = \{\varepsilon, a, \substack{a\\|\\a}\}$, $V = \{\Box, a + \Box, \substack{a\\|\\\Box}, \substack{a\\|\\a\\|\\\Box}\}$
- $\mu(\text{no } a) = \varepsilon$
- $\mu(\text{antichain}) = a$
- $\mu(\text{comparable } a) = \substack{a\\|\\a}$
- $\mu(\text{no } a) = \Box$
- $\mu(\text{antichain } + \text{no } a \text{ before } \Box) = a + \Box$
- $\mu(\text{antichain } + a \text{ before } \Box) = \substack{a\\|\\\Box}$
- $\mu(\text{comparable } a) = \substack{a\\|\\a\\|\\\Box}$
- $+, \cdot, *$ defined to match the semantic.

Maintenance in $O(\log(n)/\log\log(n))$ time

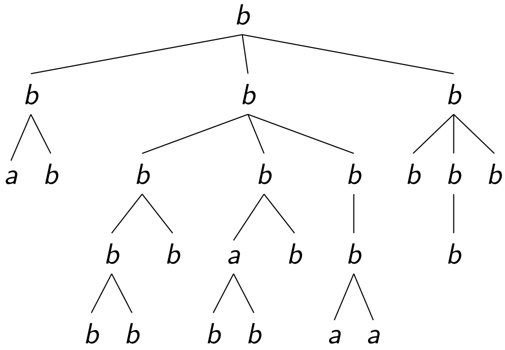
Clustering

Theorem

Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

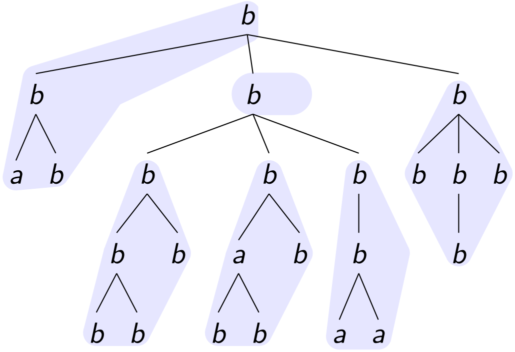


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$

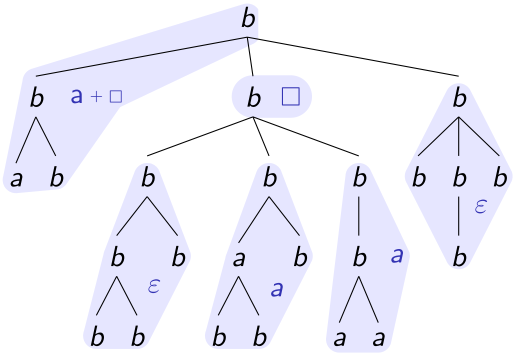


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force

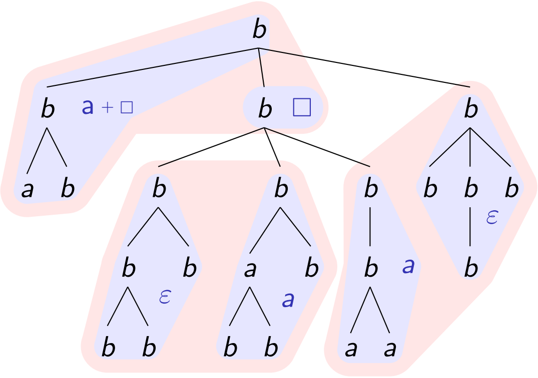


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$

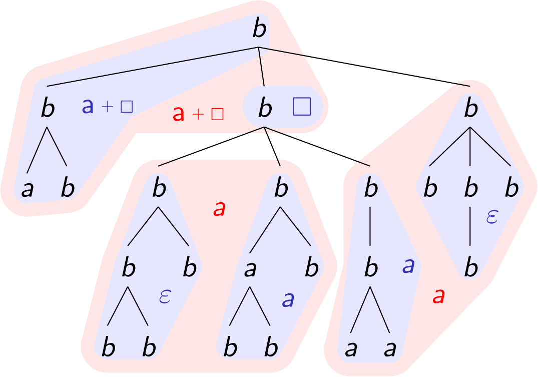


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$

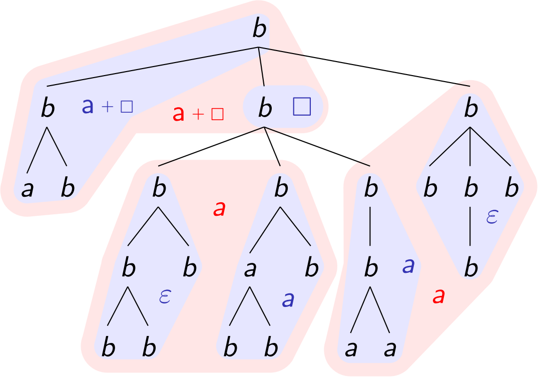


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

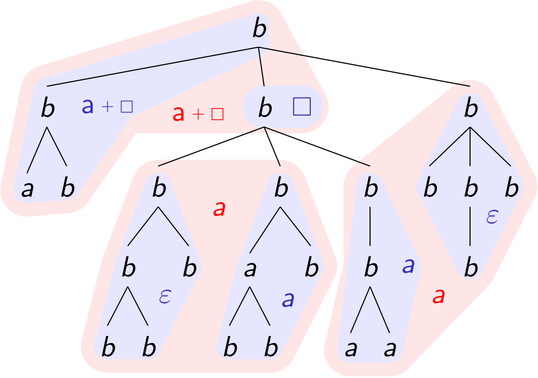


$\in$ Antichain?

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

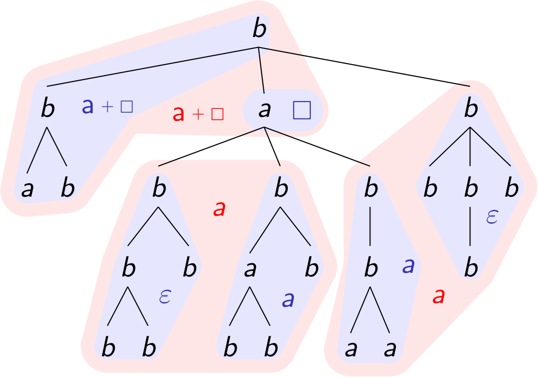


evaluates to a $\in$ Antichain

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

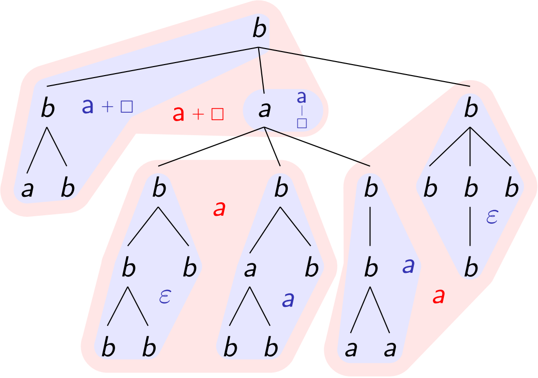


evaluates to a $\in$ Antichain

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

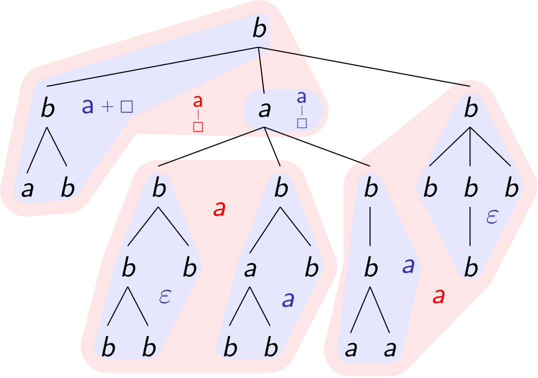


evaluates to a $\in$ Antichain

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

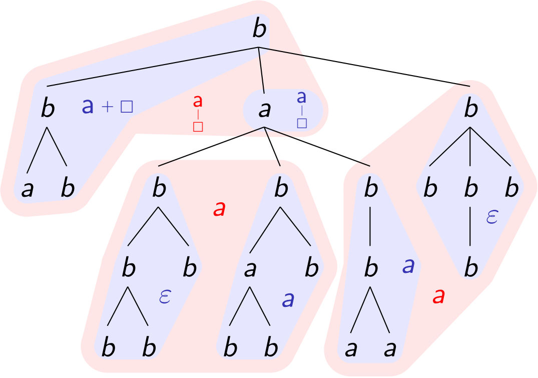


evaluates to a $\in$ Antichain

Clustering

Theorem
Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times

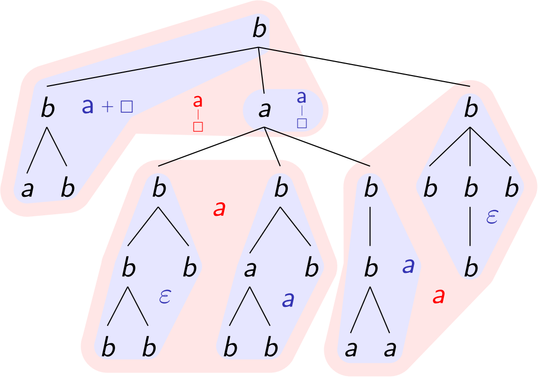


evaluates to $\overset{\mathrm{a}}{\underset{\mathrm{a}}{|}} \notin$ Antichain

Clustering

> **Theorem**
> Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times
- handle updates in constant time per layer

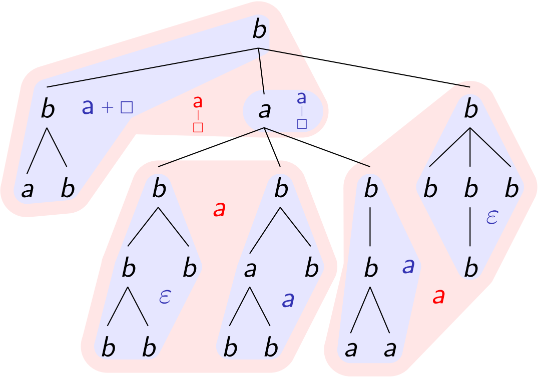


evaluates to $\overset{\text{a}}{\underset{\text{a}}{|}} \notin$ Antichain

Clustering

> **Theorem**
> Every regular language of tree can be maintained in $O(\log(n)/\log\log(n))$ time

- $\leq \frac{n}{\log(n)}$ clusters of size $\leq \log(n)$
- maintain images in clusters by brute force
- $\leq \frac{n}{\log(n)^2}$ clusters of size $\leq \log(n)$
- iterate $\leq \frac{\log(n)}{\log\log(n)}$ times
- handle updates in constant time per layer
- preprocessing in linear time



evaluates to $\genfrac{}{}{0pt}{}{\mathrm{a}}{\mathrm{a}} \notin$ Antichain

Maintenance in constant time

Upper bounds

Commutative languages: membership only depends on the number of each letter

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject
- If not: reconstruct the subtree with a precomputed structure for ancestors.

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject
- If not: reconstruct the subtree with a precomputed structure for ancestors.

Almost-commutative languages: Boolean combination of commutative and singleton languages

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject
- If not: reconstruct the subtree with a precomputed structure for ancestors.

Almost-commutative languages: Boolean combination of commutative and singleton languages
$\rightarrow$ Maintainable in constant time.

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\rightarrow$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject
- If not: reconstruct the subtree with a precomputed structure for ancestors.

Almost-commutative languages: Boolean combination of commutative and singleton languages
$\rightarrow$ Maintainable in constant time.
$\rightarrow$ What is this class?

Upper bounds

Commutative languages: membership only depends on the number of each letter
$\to$ Maintainable in constant time.

Proof

- Maintain the count of the number of each letter in $O(1)$
- regular $\Rightarrow$ ultimately periodic conditions on the counts

Singleton languages: $\{w\}+$ neutral letters
$\to$ Maintainable in constant time.

Proof

- Maintain an unordered doubly-linked list of the non-neutral letters in $O(1)$
- If the size if more than $|w|$: reject
- If not: reconstruct the subtree with a precomputed structure for ancestors.

Almost-commutative languages: Boolean combination of commutative and singleton languages
$\to$ Maintainable in constant time.
$\to$ What is this class? $\to$ algebra

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Almost commutative	$\Leftrightarrow$	The syntactic forest algebra satisfies: $\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$	(ZG)

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v \qquad \text{(ZG)}$$

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L

$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies: $\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied

An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image

An algebraic characterization

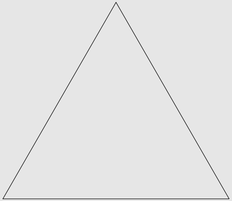
Syntactic forest algebra: smallest forest algebra recognizing L
$\to$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V, \ \forall k \geq \omega, \ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\to$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

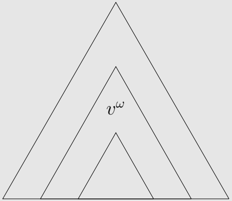
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies: (ZG)
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

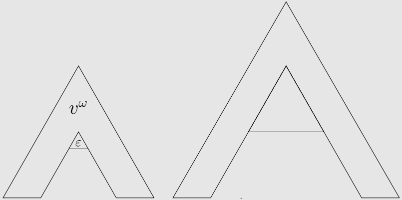
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V, \ \forall k \geq \omega, \ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

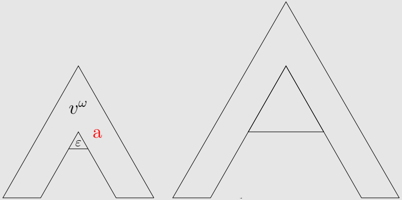
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

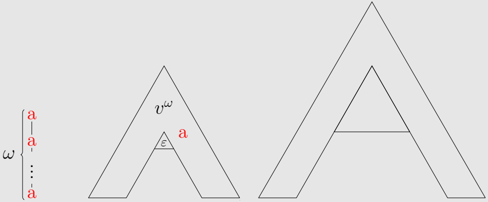
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

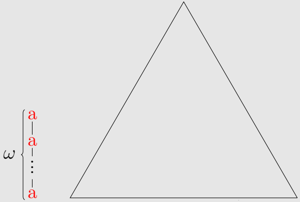
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^{\omega} \cdot v^{\omega} = v^{\omega}$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

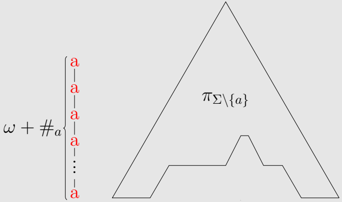
Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L

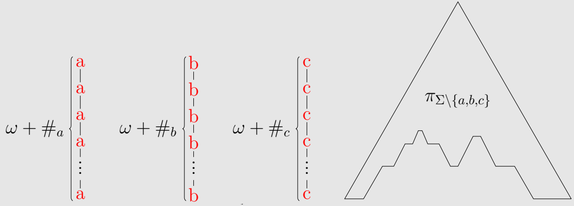
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies:
$\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image



An algebraic characterization

Syntactic forest algebra: smallest forest algebra recognizing L
$\rightarrow$ there is an integer ω, such that $v^\omega \cdot v^\omega = v^\omega$, $\forall v \in V$

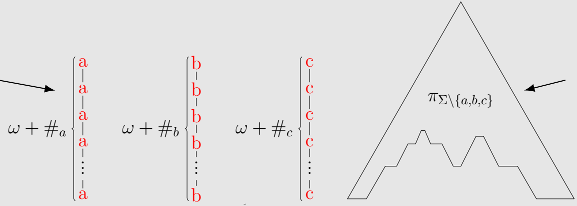
Decidable

Almost commutative $\Leftrightarrow$ The syntactic forest algebra satisfies: $\forall v, w \in V,\ \forall k \geq \omega,\ v \cdot w^k = w^k \cdot v$ (ZG)

Proof

- L commutative $\Rightarrow$ V commutative $\Rightarrow$ (ZG)
- L singleton $\Rightarrow$ w^k behaves like a context with no non-neutral letters or a lot of them $\Rightarrow$ (ZG)
- (ZG) $\Rightarrow$ many other equations are implied
  $\rightarrow$ terms v^k for $k \geq \omega$ can be moved anywhere whithout changing the image

commutative
information



singleton
information

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
Output: is there a 1 before the $\#$?

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.

Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

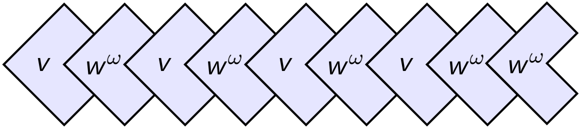
> **Dynamic problem (Prefix-$\vee$)**
>
> Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
>
> Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

Prefix-$\vee$: 

$\downarrow$

L 

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

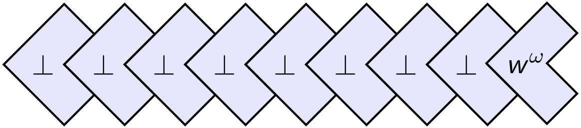
Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)
$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

Prefix-$\vee$:

$\downarrow$

L



Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
Output: is there a 1 before the $\#$?

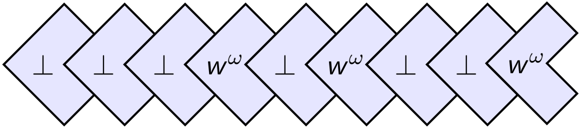
L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)
$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

Prefix-$\vee$:

0	1	1	0

$\downarrow$

L



Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.

Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

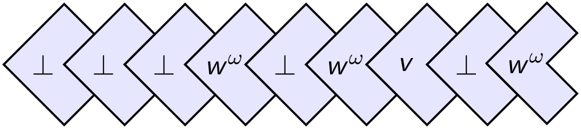
$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

Prefix-$\vee$:

0	1	1	#

$\downarrow$

L



Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.

Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

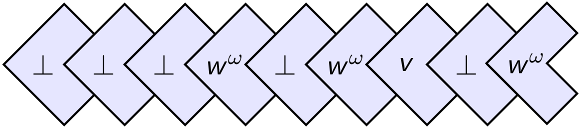
Prefix-$\vee$:

0	1	1	#

$\in$ Prefix-$\vee$

$\downarrow$

L



evaluates to $w^\omega \cdot v \cdot w^\omega$

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.

Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

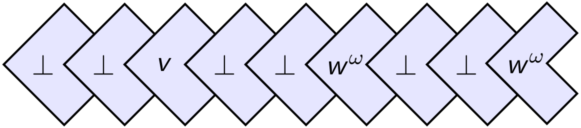
Prefix-$\vee$:

0	#	1	0

$\notin$ Prefix-$\vee$

$\downarrow$

L



evaluates to $v \cdot w^\omega$

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.

Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)

$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

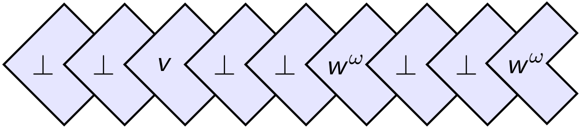
Prefix-$\vee$:

0	#	1	0

$\notin$ Prefix-$\vee$

$\downarrow$ $\Updownarrow$

L



evaluates to $v \cdot w^\omega$

Lower bound

Conjecture (used in Amarilli,Jachiet,Paperman): this is not in $O(1)$

Dynamic problem (Prefix-$\vee$)

Input: a word $w \in \{0, 1, \#\}$ with at most one $\#$.
Output: is there a 1 before the $\#$?

L not almost-commutative $\Rightarrow$ there are $v, w \in V$, $v \cdot w^\omega \neq w^\omega \cdot v$ (simplification)
$\rightarrow$ Wlog. $v \cdot w^\omega \neq w^\omega \cdot v \cdot w^\omega$

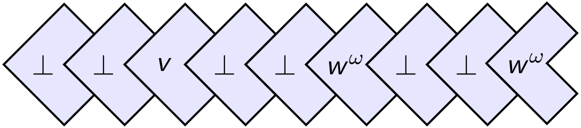
Prefix-$\vee$:

0	#	1	0

$\notin$ Prefix-$\vee$

$\downarrow$ $\Updownarrow$

L



evaluates to $v \cdot w^\omega$

Theorem
Maintainable in $O(1)$ time $\Leftrightarrow$ almost-commutative

Conclusion

Theorem

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- Maintainable in $O(1)$ time $\Leftrightarrow$ Boolean combinations of commutative and singleton languages

Conclusion

Theorem

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- Maintainable in $O(1)$ time $\Leftrightarrow$ Boolean combinations of commutative and singleton languages

What's next:

Conclusion

Theorem

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- Maintainable in $O(1)$ time $\Leftrightarrow$ Boolean combinations of commutative and singleton languages

What's next:

- Obtaining a trichotomy like for words
  $\to$ study the $O(\log\log(n))$ regime
  $\to$ different already for aperiodics

Conclusion

Theorem

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- Maintainable in $O(1)$ time $\Leftrightarrow$ Boolean combinations of commutative and singleton languages

What's next:

- Obtaining a trichotomy like for words
  $\rightarrow$ study the $O(\log\log(n))$ regime
  $\rightarrow$ different already for aperiodics
- Remove the neutral letter assumption
  $\rightarrow$ the algebraic theory of trees is less developed than for words

Conclusion

Theorem

- All regular languages of forests can be maintained in $O(\log(n)/\log\log(n))$ time
- Maintainable in $O(1)$ time $\Leftrightarrow$ Boolean combinations of commutative and singleton languages

What's next:

- Obtaining a trichotomy like for words
  $\to$ study the $O(\log\log(n))$ regime
  $\to$ different already for aperiodics
- Remove the neutral letter assumption
  $\to$ the algebraic theory of trees is less developed than for words
- handle modifications of the shape of the tree