A Complexity Approach to Tree Algebras: The Polynomial Case

Arthur Jaquard

joint work with Thomas Colcombet

Université Paris Cité, CNRS, IRIF

MFCS 2022 | August 25, 2022

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Université de Paris

Algebras are used to characterize classes of languages

Algebras are used to characterize classes of languages

Finite words
Monoids, semigroups

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words

Monoids, semigroups

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words
Monoids, semigroups
Infinite words
Wilke algebras, ω-semigroups, $\circ$-algebras...

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words
Monoids, semigroups

Infinite words
Wilke algebras, ω-semigroups, $\circ$-algebras...

Trees
Deterministic automata, Preclones, Hyperclones, Operads,...

Graphs
HR-algebras, VR-algebras

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words
Monoids, semigroups
Infinite words
Wilke algebras, ω-semigroups, $\circ$-algebras...
Trees
Deterministic automata, Preclones, Hyperclones, Operads,...
Graphs
HR-algebras, VR-algebras

Infinitely sorted algebras

$(A_n)_{n\in\mathbb{N}}$

$(A_X)_{X \text{ finite}}$

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words
Monoids, semigroups
Infinite words
Wilke algebras, ω-semigroups, $\circ$-algebras...
Trees
Deterministic automata, Preclones, Hyperclones, Operads,...
Graphs
HR-algebras, VR-algebras

Infinitely sorted algebras

$(A_n)_{n\in\mathbb{N}}$

$(A_X)_{X \text{ finite}}$

Problem: Hard to derive characterizations with infinitely sorted algebras

Algebras and classes of languages

Algebras are used to characterize classes of languages

Schützenberger, 1965

A regular language L is star-free if and only if its syntactic monoid is aperiodic.

Finite words
Monoids, semigroups
Infinite words
Wilke algebras, ω-semigroups, $\circ$-algebras...
Trees
Deterministic automata, Preclones, Hyperclones, Operads,...
Graphs
HR-algebras, VR-algebras

Infinitely sorted algebras

$(A_n)_{n\in\mathbb{N}}$

$(A_X)_{X \text{ finite}}$

Problem: Hard to derive characterizations with infinitely sorted algebras

We study tree algebras under the angle of asymptotic complexity

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x,y\}} \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

$$\begin{array}{c} a \\ / \ \backslash \\ b \quad c \end{array} \in T_\emptyset \qquad \begin{array}{c} a \\ / \ \backslash \\ x \quad x \end{array} \in T_{\{x\}}$$

$$\begin{array}{c} a \\ / \ \backslash \\ x \quad x \end{array} \in T_{\{x,y\}} \qquad \begin{array}{c} a \\ / \ \backslash \\ x \quad y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{c} a \\ / \ \backslash \\ x \quad y \end{array} \cdot_x \begin{array}{c} a \\ / \ \backslash \\ b \quad c \end{array} = \begin{array}{ccc} & a & \\ & / \ \backslash & \\ a & & y \\ / \ \backslash & & \\ b \quad c & & \end{array}$$

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x,y\}} \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & & y \\ / & & \backslash & & \\ b & & c & & \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \overset{\sigma}{\longmapsto} \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array}$$

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x,y\}} \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & & y \\ / & & \backslash & & \\ b & & c & & \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \overset{\sigma}{\mapsto} \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array}$$

Def. A finite tree algebra $\mathcal{A}$ consists of an infinite series of finite carrier sets A_X indexed by finite sets of variables X, together with operations:

- **Constants.** $a(x_0, \ldots, x_{n-1})^{\mathcal{A}} \in A_{\{x_0,\ldots,x_{n-1}\}}$ for all $a \in \Sigma_n$ and variables x_i,
- **Substitution.** $\cdot_x^{\mathcal{A}} \colon A_X \times A_Y \to A_{X \setminus \{x\} \cup Y}$ for all finite X, Y and variable x,
- **Renaming.** $\sigma^{\mathcal{A}} \colon A_X \to A_Y$ for all maps $\sigma \colon X \to Y$.

Infinitely sorted tree algebras

Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x,y\}} \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & & y \\ / & & \backslash & & \\ b & & c & & \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \overset{\sigma}{\longmapsto} \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array}$$

Def. A finite tree algebra $\mathcal{A}$ consists of an infinite series of finite carrier sets A_X indexed by finite sets of variables X, together with operations:

> **Constants.** $a(x_0, \ldots, x_{n-1})^{\mathcal{A}} \in A_{\{x_0,\ldots,x_{n-1}\}}$ for all $a \in \Sigma_n$ and variables x_i,
> **Substitution.** $\cdot_x^{\mathcal{A}} \colon A_X \times A_Y \to A_{X \setminus \{x\} \cup Y}$ for all finite X, Y and variable x,
> **Renaming.** $\sigma^{\mathcal{A}} \colon A_X \to A_Y$ for all maps $\sigma \colon X \to Y$.

Identities? $\quad a(x, y) \cdot_y b \qquad a(x, z) \cdot_z b$

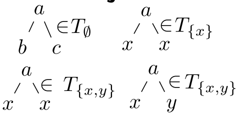
We also define morphisms, congruences...

Infinitely sorted tree algebras

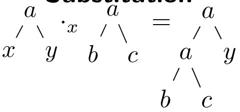
Let Σ be a ranked alphabet and $\mathcal{V}$ be a countably infinite set of variables. The free tree algebra has as carrier sets the $(T_X)_{X \subseteq \mathcal{V} \text{ finite}}$.

$$T_X = \{\text{trees in which all the variables on the leaves are in } X\}$$

Objects

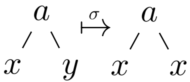


Substitution



Renaming

$$\sigma(x) = \sigma(y) = x$$



Def. A finite tree algebra $\mathcal{A}$ consists of an infinite series of finite carrier sets A_X indexed by finite sets of variables X, together with operations:

- **Constants.** $a(x_0, \ldots, x_{n-1})^{\mathcal{A}} \in A_{\{x_0,\ldots,x_{n-1}\}}$ for all $a \in \Sigma_n$ and variables x_i,
- **Substitution.** $\cdot_x^{\mathcal{A}} \colon A_X \times A_Y \to A_{X \setminus \{x\} \cup Y}$ for all finite X, Y and variable x,
- **Renaming.** $\sigma^{\mathcal{A}} \colon A_X \to A_Y$ for all maps $\sigma \colon X \to Y$.

Identities? $\quad a(x, y) \cdot_y b \qquad a(x, z) \cdot_z b$

We also define morphisms, congruences...

Given a finite tree algebra $\mathcal{A}$, there is a unique morphism from the free algebra to $\mathcal{A}$. It is called the evaluation morphism of $\mathcal{A}$.

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Prop. Finite tree algebras exactly recognize the regular languages.

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Prop. Finite tree algebras exactly recognize the regular languages.

Example $L =$ trees with a b on the leftmost branch

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Prop. Finite tree algebras exactly recognize the regular languages.

Example $L =$ trees with a b on the leftmost branch

$$\begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & b & & & & a & \\ / & & \backslash & & / & & \backslash \\ c & & c & & y & & y \end{array} \quad \overset{\alpha}{\longmapsto} \top \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & c & & y & & y \end{array} \quad \overset{\alpha}{\longmapsto} (\bot, x),$$

$$A_X = \{\top, \bot\} \uplus (\{\top, \bot\} \times X)$$

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Prop. Finite tree algebras exactly recognize the regular languages.

Example $L =$ trees with a b on the leftmost branch

$$
\begin{array}{ccccccc}
 & & & a & & & \\
 & & / & & \backslash & & \\
 & b & & & & a & \\
 / & & \backslash & & / & & \backslash \\
c & & c & & y & & y
\end{array}
\overset{\alpha}{\mapsto} \top
\qquad
\begin{array}{ccccccc}
 & & & a & & & \\
 & & / & & \backslash & & \\
 & a & & & & a & \\
 / & & \backslash & & / & & \backslash \\
x & & c & & y & & y
\end{array}
\overset{\alpha}{\mapsto} (\bot, x),
$$

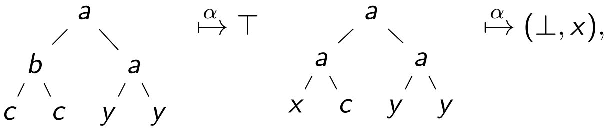
$A_X = \{\top, \bot\} \uplus (\{\top, \bot\} \times X)$ $\qquad$ $|A_X| = 2 + 2|X|$ is linear in $|X|$.

Languages and the size of the algebra

Def. A language L of finite trees over Σ is recognized by a finite algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

Prop. Finite tree algebras exactly recognize the regular languages.

Example $L =$ trees with a b on the leftmost branch



$A_X = \{\top, \perp\} \uplus (\{\top, \perp\} \times X)$ $\qquad$ $|A_X| = 2 + 2|X|$ is linear in $|X|$.

This algebra has linear complexity.

Complexity

Def. The complexity of a finite algebra $\mathcal{A}$ is the asymptotic size of $|A_X|$ as a function of $|X|$.

Complexity

Def. The complexity of a finite algebra $\mathcal{A}$ is the asymptotic size of $|A_X|$ as a function of $|X|$.

Prop. All regular languages are recognized by algebras of doubly-exponential complexity.

Complexity

Def. The complexity of a finite algebra $\mathcal{A}$ is the asymptotic size of $|A_X|$ as a function of $|X|$.

Prop. All regular languages are recognized by algebras of doubly-exponential complexity.

Describe the languages recognized by algebras of bounded / polynomial / exponential complexity.

Complexity

Def. The complexity of a finite algebra $\mathcal{A}$ is the asymptotic size of $|A_X|$ as a function of $|X|$.

Prop. All regular languages are recognized by algebras of doubly-exponential complexity.

Describe the languages recognized by algebras of bounded / polynomial / exponential complexity.

Bounded complexity	[Colcombet, J, 2021]
Polynomial complexity	This talk
Exponential complexity	-
Doubly-exponential complexity	All regular languages

Another example

$L =$ trees whose leftmost branch ends with $a(c, c)$, where
$\Sigma = \{c : 0, d : 0, a : 2\}$

Another example

$L =$ trees whose leftmost branch ends with $a(c, c)$, where $\Sigma = \{c : 0, d : 0, a : 2\}$

$$A_X = \{c, d\} \cup \{a(x, y) \mid x, y \in X \cup \{c, *\}\}$$

Another example

> $L =$ trees whose leftmost branch ends with $a(c, c)$, where
> $\Sigma = \{c : 0, d : 0, a : 2\}$

$$A_X = \{c, d\} \cup \{a(x, y) \mid x, y \in X \cup \{c, *\}\}$$

$$d \overset{\alpha}{\mapsto} d \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & z & & y & & y \end{array} \overset{\alpha}{\mapsto} a(x, z) \qquad \begin{array}{ccccc} & a & & & \\ / & & \backslash & & \\ x & & a & & \\ & / & & \backslash & \\ & y & & y & \end{array} \overset{\alpha}{\mapsto} a(x, *)$$

Another example

$L =$ trees whose leftmost branch ends with $a(c, c)$, where
$\Sigma = \{c : 0, d : 0, a : 2\}$

$$A_X = \{c, d\} \cup \{a(x, y) \mid x, y \in X \cup \{c, *\}\}$$

$$d \overset{\alpha}{\mapsto} d \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & z & & y & & y \end{array} \overset{\alpha}{\mapsto} a(x, z) \qquad \begin{array}{ccccc} & a & & & \\ / & & \backslash & & \\ x & & a & & \\ & / & & \backslash & \\ & y & & y & \end{array} \overset{\alpha}{\mapsto} a(x, *)$$

$$c \overset{\alpha}{\mapsto} c \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & c & & y & & y \end{array} \overset{\alpha}{\mapsto} a(x, c) \qquad \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & y & \\ / & & \backslash & & \\ c & & c & & \end{array} \overset{\alpha}{\mapsto} a(c, c)$$

Another example

$L =$ trees whose leftmost branch ends with $a(c, c)$, where
$\Sigma = \{c : 0, d : 0, a : 2\}$

$$A_X = \{c, d\} \cup \{a(x, y) \mid x, y \in X \cup \{c, *\}\}$$

$$d \overset{\alpha}{\mapsto} d \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & z & & y & & y \end{array} \overset{\alpha}{\mapsto} a(x, z) \qquad \begin{array}{ccccc} & a & & & \\ / & & \backslash & & \\ x & & a & & \\ & / & & \backslash & \\ & y & & y & \end{array} \overset{\alpha}{\mapsto} a(x, *)$$

$$c \overset{\alpha}{\mapsto} c \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & c & & y & & y \end{array} \overset{\alpha}{\mapsto} a(x, c) \qquad \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & y & \\ / & & \backslash & & \\ c & & c & & \end{array} \overset{\alpha}{\mapsto} a(c, c)$$

Orbits: c, d, $a(x, y)$, $a(x, x)$, $a(x, c)$, $a(c, x)$, $a(x, *)$, $a(*, x)$, $a(c, c)$, $a(*, *)$

Another example

L = trees whose leftmost branch ends with $a(c,c)$, where
$\Sigma = \{c : 0, d : 0, a : 2\}$

$$A_X = \{c, d\} \cup \{a(x,y) \mid x, y \in X \cup \{c, *\}\}$$

$$d \overset{\alpha}{\mapsto} d \qquad a(a(x,z),a(y,y)) \overset{\alpha}{\mapsto} a(x,z) \qquad a(x,a(y,y)) \overset{\alpha}{\mapsto} a(x,*)$$

$$c \overset{\alpha}{\mapsto} c \qquad a(a(x,c),a(y,y)) \overset{\alpha}{\mapsto} a(x,c) \qquad a(a(c,c),y) \overset{\alpha}{\mapsto} a(c,c)$$

Orbits: c, d, $a(x,y)$, $a(x,x)$, $a(x,c)$, $a(c,x)$, $a(x,*)$, $a(*,x)$, $a(c,c)$, $a(*,*)$

This algebra has quadratic complexity and bounded orbit complexity.

Orbit complexity

Def. The orbit complexity of a finite algebra $\mathcal{A}$ is the asymptotic number of orbits in A_X under the action of $\mathbf{Sym}(X)$ as a function of $|X|$.

Orbit complexity

Def. The orbit complexity of a finite algebra $\mathcal{A}$ is the asymptotic number of orbits in A_X under the action of $\mathbf{Sym}(X)$ as a function of $|X|$.

Prop. All regular languages are recognized by algebras of doubly-exponential orbit complexity.

Another bounded hierarchy of classes.

What complexity means

Complexity is a measure of the quantity of information the algebra remembers about the variables:

Bounded complexity
The algebra does not remember anything about the variables.
$A_X \rightsquigarrow$ the variables that appear in the tree are in X.

What complexity means

Complexity is a measure of the quantity of information the algebra remembers about the variables:

Bounded complexity
The algebra does not remember anything about the variables.
$A_X \rightsquigarrow$ the variables that appear in the tree are in X.

Polynomial complexity
$A_X = X^k \rightsquigarrow k$ variables (e.g. k branches)

What complexity means

Complexity is a measure of the quantity of information the algebra remembers about the variables:

Bounded complexity
The algebra does not remember anything about the variables.
$A_X \rightsquigarrow$ the variables that appear in the tree are in X.

Polynomial complexity
$A_X = X^k \rightsquigarrow k$ variables (e.g. k branches)

Exponential complexity
$A_X = k^X \rightsquigarrow$ a function from X to k (e.g. a set of variables when $k = 2$, or modulo counting if $k = \mathbb{Z}/q\mathbb{Z}$)

What complexity means

Complexity is a measure of the quantity of information the algebra remembers about the variables:

Bounded complexity
The algebra does not remember anything about the variables.
$A_X \rightsquigarrow$ the variables that appear in the tree are in X.

Polynomial complexity
$A_X = X^k \rightsquigarrow k$ variables (e.g. k branches)

Exponential complexity
$A_X = k^X \rightsquigarrow$ a function from X to k (e.g. a set of variables when $k = 2$, or modulo counting if $k = \mathbb{Z}/q\mathbb{Z}$)

Doubly exponential complexity
All regular languages.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Common property: at all times, these algebras only keep in memory a bounded number of branches.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Common property: at all times, these algebras only keep in memory a bounded number of branches.

Main theorem

For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Common property: at all times, these algebras only keep in memory a bounded number of branches.

Main theorem

For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.

b. Being recognized by a finite tree algebra of bounded orbit complexity.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Common property: at all times, these algebras only keep in memory a bounded number of branches.

Main theorem

For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.

b. Being recognized by a finite tree algebra of bounded orbit complexity.

Equivalence between a. and b. is not obvious.

Polynomial complexity

What are the languages recognized by algebras of polynomial complexity?

- $L =$ trees with a b on the leftmost branch,
- $L =$ trees with some fixed branch in a fixed regular language,
- Boolean combinations of such languages.
- $L =$ trees whose leftmost branch ends with $a(c, c)$.

Common property: at all times, these algebras only keep in memory a bounded number of branches.

Main theorem

For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.

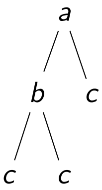
b. Being recognized by a finite tree algebra of bounded orbit complexity.

c. Being described by a coding automaton.

Equivalence between a. and b. is not obvious.

Coding automaton (high level description)

How to build the following tree ?



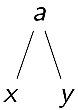
Coding automaton (high level description)

How to build the following tree ?

x $[x]$

Coding automaton (high level description)

How to build the following tree ?

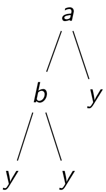


$[x]$

$[\cdot_x a(x, y)]$

Coding automaton (high level description)

How to build the following tree ?



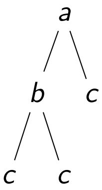
$[x]$

$[\cdot_x a(x, y)]$

$[\cdot_x b(y, y)]$

Coding automaton (high level description)

How to build the following tree ?



$[x]$

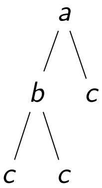
$[\cdot_x a(x, y)]$

$[\cdot_x b(y, y)]$

$[\cdot_y c]$

Coding automaton (high level description)

How to build the following tree ?



$[x]$

$[\cdot_x a(x, y)]$

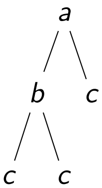
$[\cdot_x b(y, y)]$

$[\cdot_y c]$

Such a word is called a tree coding.

Coding automaton (high level description)

How to build the following tree ?



$[x]$

$[\cdot_x a(x,y)]$

$[\cdot_x b(y,y)]$

$[\cdot_y c]$

Such a word is called a tree coding.

Coding automata are register automata that map

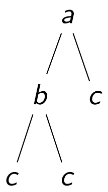
coding $c \longmapsto$ configuration $\in Q \times \mathcal{V}^R$

They are given by

- finite set of states Q
- finite set of registers R
- transitions and accepting states

Coding automaton (high level description)

How to build the following tree ?



$[x]$

$[\cdot_x a(x, y)]$

$[\cdot_x b(y, y)]$

$[\cdot_y c]$

Such a word is called a tree coding.

Coding automata are register automata that map

coding $c \longmapsto$ configuration $\in Q \times \mathcal{V}^R$

They are given by

- finite set of states Q
- finite set of registers R
- transitions and accepting states

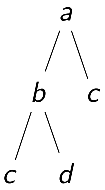
A coding automaton describes a tree language L if either

- it accepts all codings for a tree t (then it accepts t)
- it rejects all codings for a tree t (then it rejects t)

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_0

$r_1 := \square$

$r_2 := \square$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$

x

q_1

$r_1 := x$

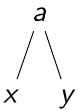
$r_2 := \square$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{12}

$r_1 := x$

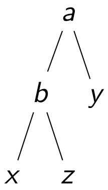
$r_2 := y$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{12}

$r_1 := x$

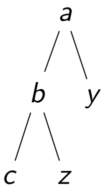
$r_2 := z$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{c2}

$r_1 := \Box$

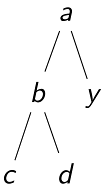
$r_2 := z$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



$$\top$$

$$r_1 := \square$$

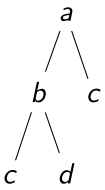
$$r_2 := \square$$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c]{\color{red}[\cdot_z d]}[\cdot_y c]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



$$\top$$

$$r_1 := \square$$

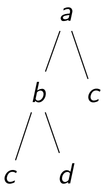
$$r_2 := \square$$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_0

$r_1 := \square$

$r_2 := \square$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)][\cdot_y c][\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$

z

q_1

$r_1 := z$

$r_2 := \square$

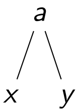
$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)][\cdot_y c][\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{12}

$r_1 := x$

$r_2 := y$

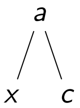
$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)][\cdot_y c][\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



$$q_1$$

$$r_1 := x$$

$$r_2 := \square$$

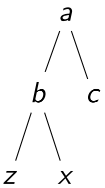
$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)]{\color{red}[\cdot_y c]}[\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{12}

$r_1 := z$

$r_2 := x$

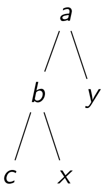
$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)][\cdot_y c][\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

$L =$ trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a:2, b:2, c:0, d:0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



q_{c2}

$r_1 := \square$

$r_2 := x$

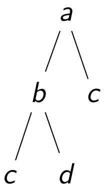
$$[x][\cdot_x a(x,y)][\cdot_x b(x,z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x,y)][\cdot_y c][\cdot_x b(z,x)][\cdot_z c][\cdot_x d]$$

Coding automaton (example)

L = trees in which the leftmost branch ends with $*(c, d)$, where $*$ is any letter. $\Sigma = \{a : 2, b : 2, c : 0, d : 0\}$

$$Q = \{q_0, q_1, q_{12}, q_{1d}, q_{c2}, \top, \bot\}$$
$$R = \{r_1, r_2\}$$



$\top$

$r_1 := \square$

$r_2 := \square$

$$[x][\cdot_x a(x, y)][\cdot_x b(x, z)][\cdot_x c][\cdot_z d][\cdot_y c]$$

$$[z][\cdot_z a(x, y)][\cdot_y c][\cdot_x b(z, x)][\cdot_z c][\cdot_x d]$$

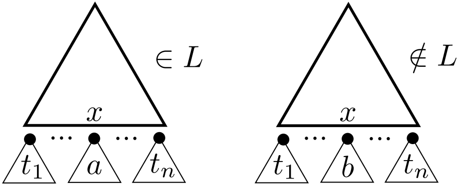
Decidability

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

Decidability

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

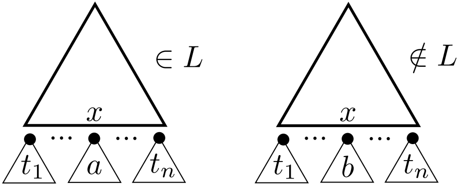
Fix L. A tree $t \in T_{\{\bullet\}}$ is L-sensitive to a leaf x if there exist trees $a, b, t_1, \ldots, t_n$ such that



Decidability

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

Fix L. A tree $t \in T_{\{\bullet\}}$ is L-sensitive to a leaf x if there exist trees $a, b, t_1, \ldots, t_n$ such that

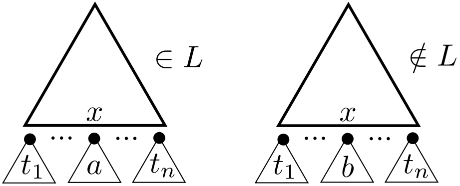


Lem. A regular language of trees L is described by a coding automaton if and only if there is a bound on the number of L-sensitive leaves in trees.

Decidability

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

Fix L. A tree $t \in T_{\{\bullet\}}$ is L-sensitive to a leaf x if there exist trees $a, b, t_1, \ldots, t_n$ such that



Lem. A regular language of trees L is described by a coding automaton if and only if there is a bound on the number of L-sensitive leaves in trees.

The existence of such a bound can be encoded into cost-MSO. Thus, it is decidable.

Summary

Notions: tree algebra, complexity, orbit complexity, coding, coding automaton

Main theorem

For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.

b. Being recognized by a finite tree algebra of bounded orbit complexity.

c. Being described by a coding automaton.

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

Summary

Notions: tree algebra, complexity, orbit complexity, coding, coding automaton

Main theorem

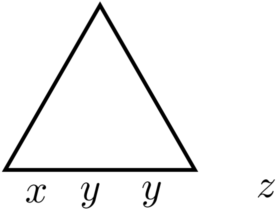
For a regular language of finite trees, the following properties are equivalent:

a. Being recognized by a finite tree algebra of polynomial complexity.
b. Being recognized by a finite tree algebra of bounded orbit complexity.
c. Being described by a coding automaton.

Thm. It is decidable whether a regular tree language is recognizable by a tree algebra of polynomial complexity.

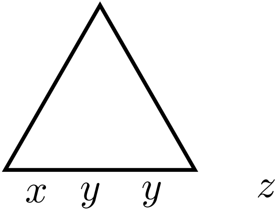
Different types of tree algebras

Unrestrained tree algebras

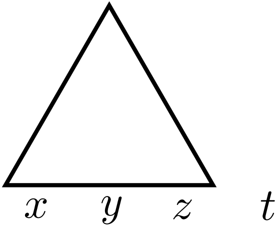


Different types of tree algebras

Unrestrained tree algebras

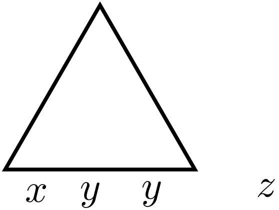


Affine tree algebras

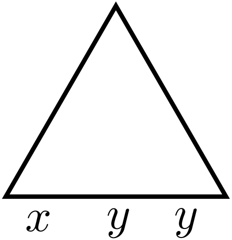


Different types of tree algebras

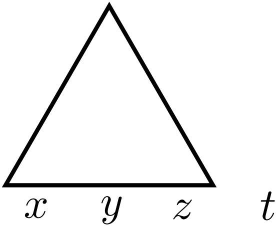
Unrestrained tree algebras



Relevant tree algebras

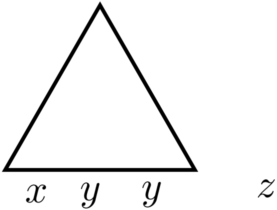


Affine tree algebras

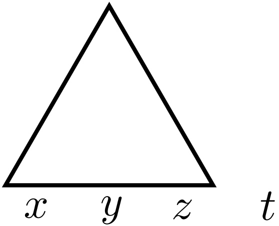


Different types of tree algebras

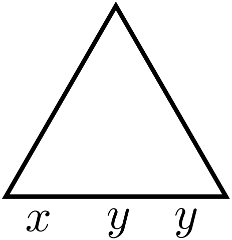
Unrestrained tree algebras



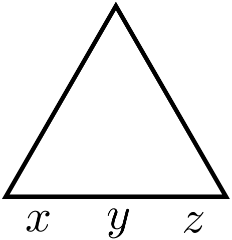
Affine tree algebras



Relevant tree algebras

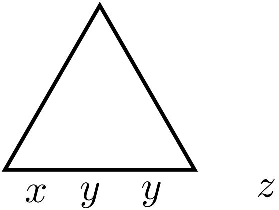


Linear tree algebras

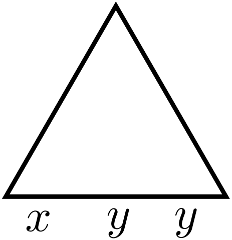


Different types of tree algebras

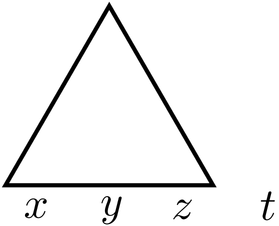
Unrestrained tree algebras



Relevant tree algebras



Affine tree algebras



Linear tree algebras