Analysis of an efficient reduction algorithm for random regular expressions based on universality detection

Pablo Rotondo

LIGM, Université Gustave Eiffel

Joint work with

Florent Koechlin

CSR 2021,

30 June, 2021.

Plan of the talk

1. Introduction: regular expression trees, uniform distribution
2. Semantic reductions: absorbing patterns, universality
3. Main results: expected size, proportion of universals
4. Techniques for the proof
5. Conclusions and further work

Introduction: context

Problem

Automatically test a program taking regular expressions as input

$$(a+b)\cdot b^\star\,, \qquad (b\cdot(a+\varepsilon))^\star\,, \qquad (a\cdot a^\star)+(b+a)^\star\,.$$

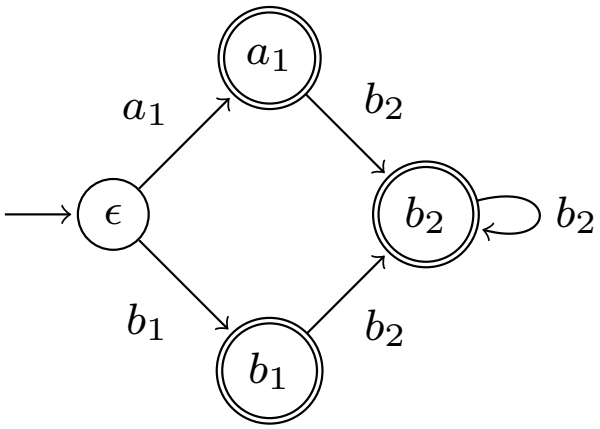
Introduction: context

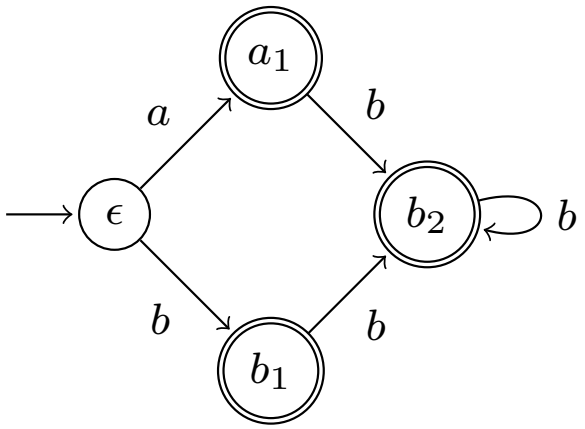
Problem

Automatically test a program taking regular expressions as input

$$(a+b)\cdot b^\star\,, \qquad (b\cdot(a+\varepsilon))^\star\,, \qquad (a\cdot a^\star)+(b+a)^\star\,.$$

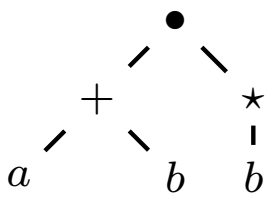
Example: automata constructions



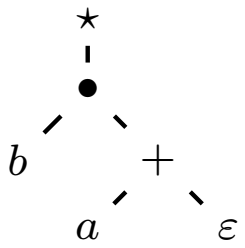


Introduction: random regular expressions

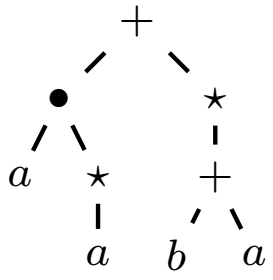
- **Expression trees**



$(a + b) \cdot b^\star$



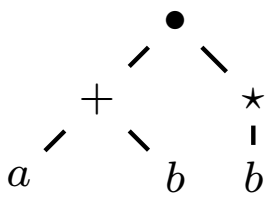
$(b \cdot (a + \varepsilon))^\star$



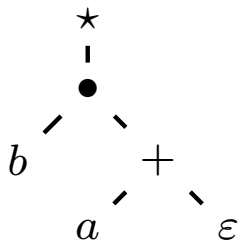
$(a \cdot a^\star) + (b + a)^\star$

Introduction: random regular expressions

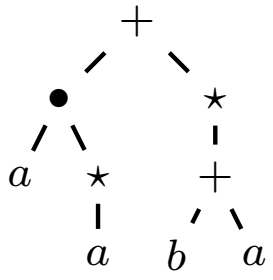
- **Expression trees**



$(a+b)\cdot b^\star$



$(b\cdot(a+\varepsilon))^\star$



$(a\cdot a^\star)+(b+a)^\star$

- **Generate a random expression tree**
  - Realistic distribution
  - Simple implementation, possibility of theoretical analysis.

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \underset{\mathcal{L}}{\overset{\star}{|}} + \underset{\mathcal{L}\ \mathcal{L}}{\overset{\bullet}{\wedge}} + \underset{\mathcal{L}\ \mathcal{L}}{\overset{+}{\wedge}},$$

- size $|T|$ = number of nodes.

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n,

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \underset{\mathcal{L}}{\overset{\star}{|}} + \underset{\mathcal{L}\ \mathcal{L}}{\overset{\bullet}{\wedge}} + \underset{\mathcal{L}\ \mathcal{L}}{\overset{+}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n, pick tree T of size $|T| = n$ *uniformly*

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n, pick tree T of size $|T| = n$ *uniformly*

- natural a priori choice,

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n, pick tree T of size $|T| = n$ *uniformly*

- natural a priori choice,
- efficient sampling

(Boltzmann, Recursive, Devroye's constrainted GW),

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n, pick tree T of size $|T| = n$ *uniformly*

- natural a priori choice,
- efficient sampling
  (Boltzmann, Recursive, Devroye's constrainted GW),
- amenable to theoretical study (Analytic Combinatorics).

Uniform random expression trees

Expression trees:

- trees defined inductively,

$$\mathcal{L} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}},$$

- size $|T|$ = number of nodes.

Idea: Fix target size n, pick tree T of size $|T| = n$ *uniformly*

- natural a priori choice,
- efficient sampling
  (Boltzmann, Recursive, Devroye's constrainted GW),
- amenable to theoretical study (Analytic Combinatorics).

$\Longrightarrow$ Model used in numerous practical and theoretical works

Warning signs: reduction by an absorbing pattern

Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

Warning signs: reduction by an absorbing pattern

Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

- **Universal result**: not only regular expressions,

Warning signs: reduction by an absorbing pattern

Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

- **Universal result**: not only regular expressions,
- Absorbing patterns: only semantic hypothesis, absorbing pattern $\mathcal{P}$,

Warning signs: reduction by an absorbing pattern

Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

- **Universal result**: not only regular expressions,
- Absorbing patterns: only semantic hypothesis, absorbing pattern $\mathcal{P}$,

$$\begin{array}{c} \circledast \\ \mathcal{P} \quad T \end{array} \rightsquigarrow \mathcal{P} \qquad \begin{array}{c} \circledast \\ T \quad \mathcal{P} \end{array} \rightsquigarrow \mathcal{P}$$

simplest case, $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$.

Warning signs: reduction by an absorbing pattern

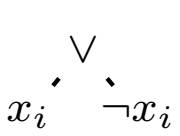
Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

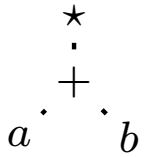
- **Universal result**: not only regular expressions,
- Absorbing patterns: only semantic hypothesis, absorbing pattern $\mathcal{P}$,

simplest case, $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$.

- Wide variety of examples:



operator $\vee$



operator $+$

$x \mapsto 0$

operator $\times$

Warning signs: reduction by an absorbing pattern

Uniform expression trees [Koechlin,Nicaud,R. 2020]

Expected size after (linear) reduction is bounded $O(1)$.

- **Universal result**: not only regular expressions,
- Absorbing patterns: only semantic hypothesis, absorbing pattern $\mathcal{P}$,

$$\begin{array}{c} \circledast \\ \mathcal{P} \quad T \end{array} \rightsquigarrow \mathcal{P} \qquad \begin{array}{c} \circledast \\ T \quad \mathcal{P} \end{array} \rightsquigarrow \mathcal{P}$$

  simplest case, $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$.

- Wide variety of examples:

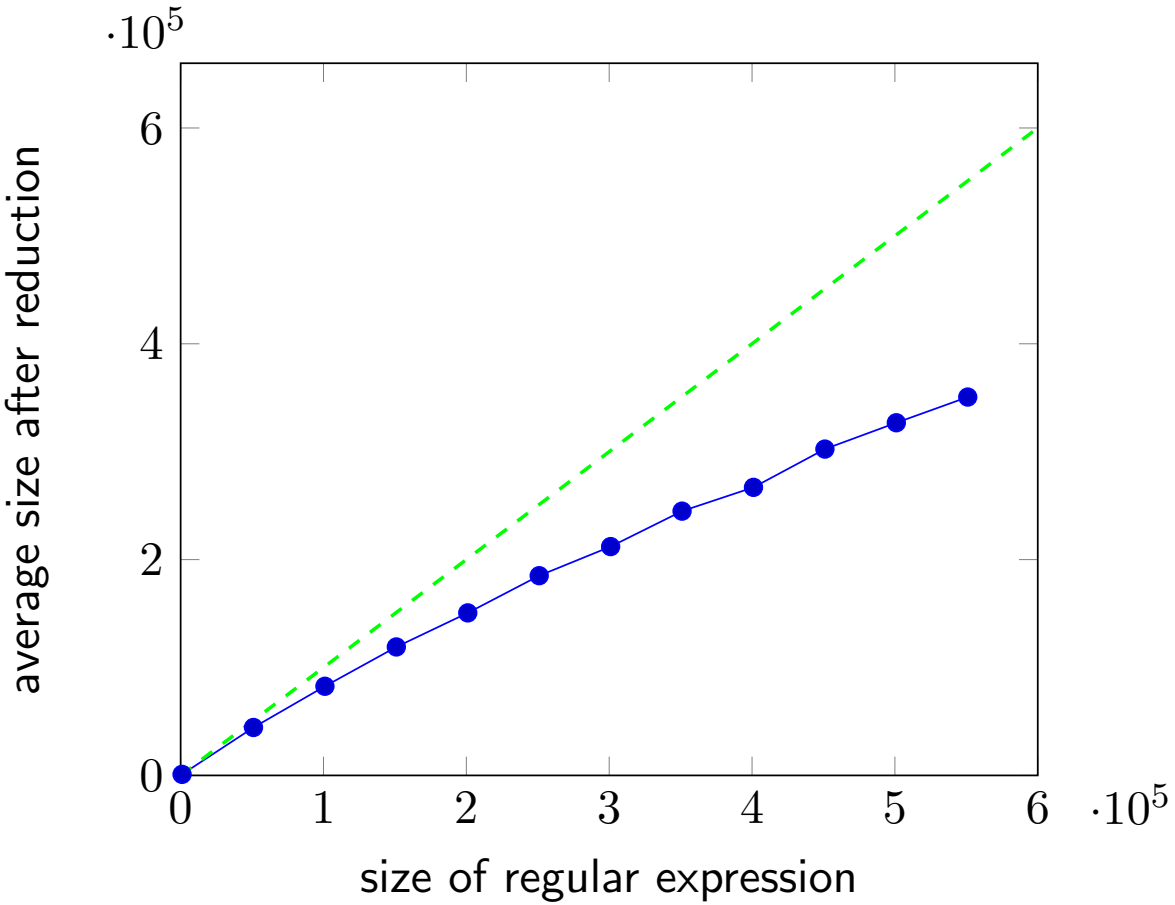
$$\begin{array}{c} \vee \\ x_i \quad \neg x_i \end{array} \qquad \begin{array}{c} \star \\ + \\ a \quad b \end{array} \qquad x \mapsto 0$$

operator $\vee$ | operator $+$ | operator $\times$

What does this say about regular expressions? $O(1)$?

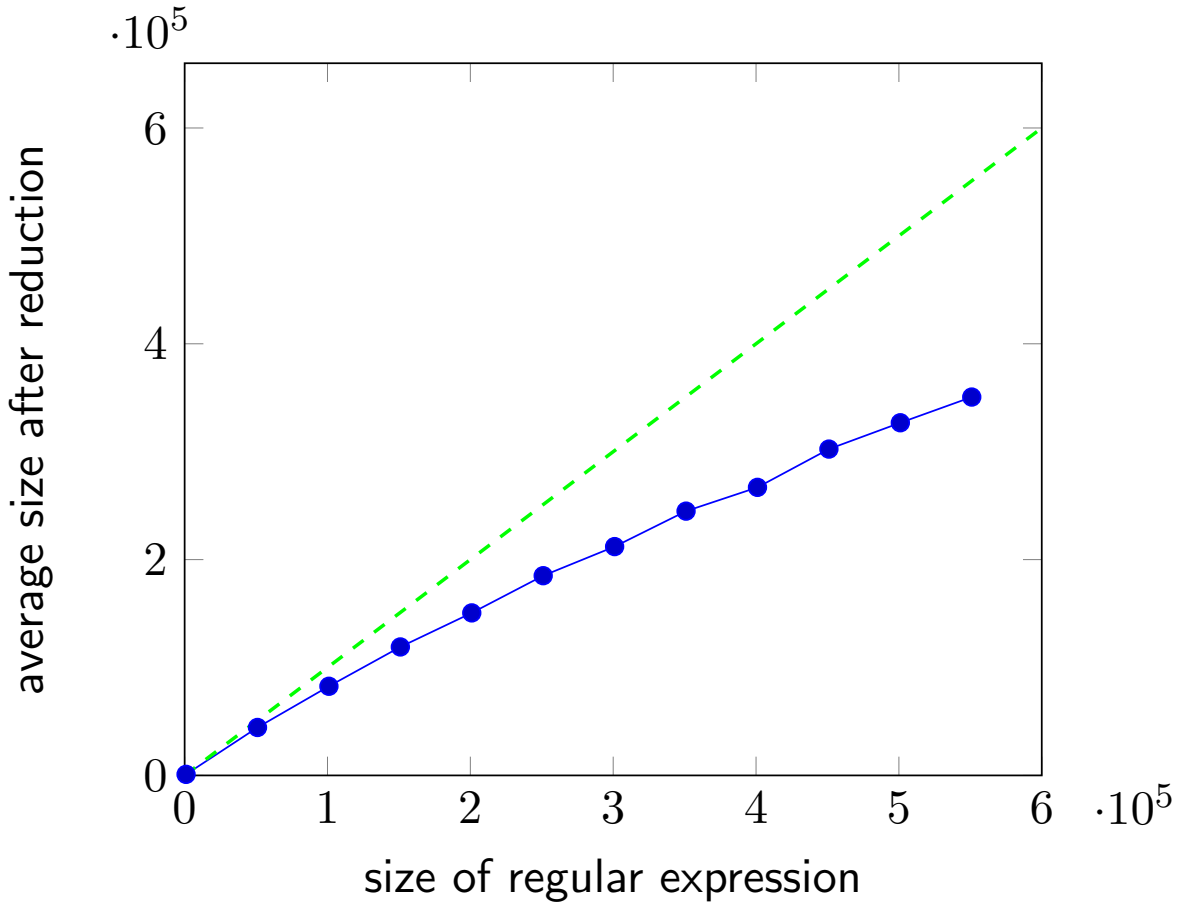
Regular expressions: reduction by absorbing pattern

Hidden constant $O(1)$: for regular expressions on two letters, the limit size after reduction is $3\ 624\ 217$.



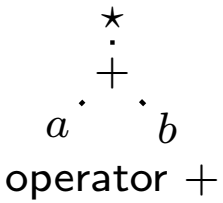
Regular expressions: reduction by absorbing pattern

Hidden constant $O(1)$: for regular expressions on two letters, the limit size after reduction is $3\ 624\ 217$.



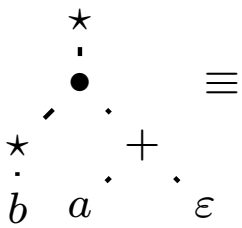
Question. Are uniform regular expressions useful nonetheless?

Reduction based on universality detection

- Reduction from absorbing pattern 

misses fine semantics

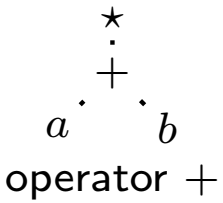
Example: just avoid the pattern



$\equiv \quad \{a, b\}^\star$.

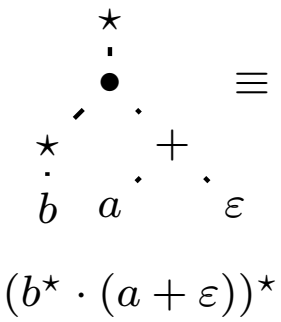
$(b^\star \cdot (a + \varepsilon))^\star$

Reduction based on universality detection

- Reduction from absorbing pattern 

misses fine semantics

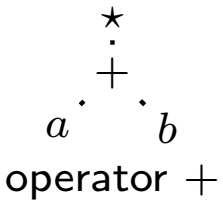
Example: just avoid the pattern



$\equiv \quad \{a, b\}^\star$.

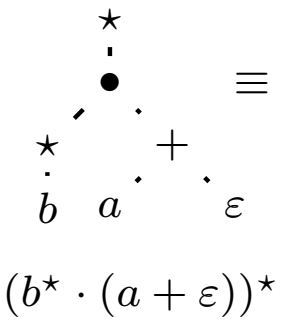
- We consider more specific algorithm based on *universality detection*

Reduction based on universality detection

- Reduction from absorbing pattern 

misses fine semantics

Example: just avoid the pattern

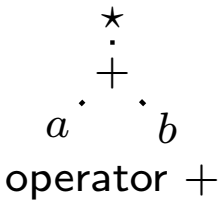


$\equiv \quad \{a, b\}^\star \, .$

- We consider more specific algorithm based on *universality detection*

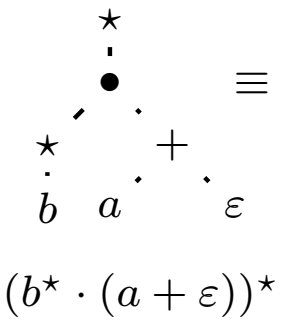
$$\texttt{expression is universal} \Leftrightarrow \texttt{equivalent to } \Sigma^\star \, ,$$

Reduction based on universality detection

- Reduction from absorbing pattern 

misses fine semantics

Example: just avoid the pattern



$\equiv \quad \{a, b\}^\star$.

- We consider more specific algorithm based on *universality detection*

$$\texttt{expression is universal} \Leftrightarrow \texttt{equivalent to } \Sigma^\star ,$$

$\Rightarrow$ substitute universal subtrees by smallest universal tree $\mathcal{U}$.

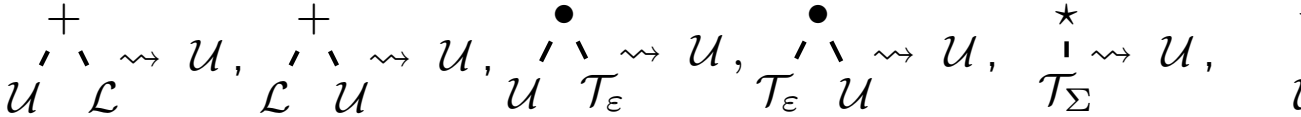
Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

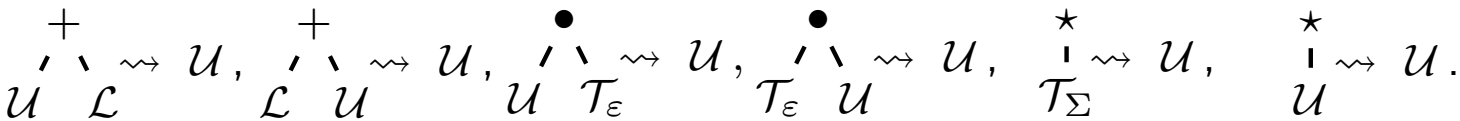
- We define bottom-up propagation rules



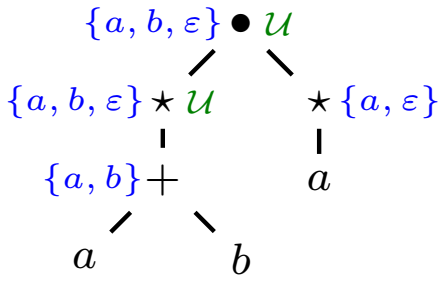
Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

- We define bottom-up propagation rules



- Examples for $\Sigma = \{a, b\}$,

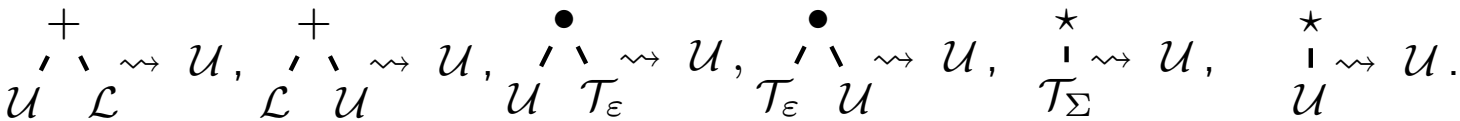


(I) : $(a + b)^\star \cdot a^\star$

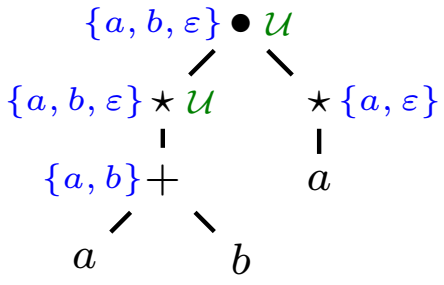
Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

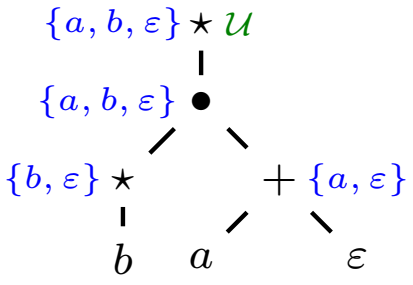
- We define bottom-up propagation rules



- Examples for $\Sigma = \{a, b\}$,



(I) : $(a + b)^\star \cdot a^\star$

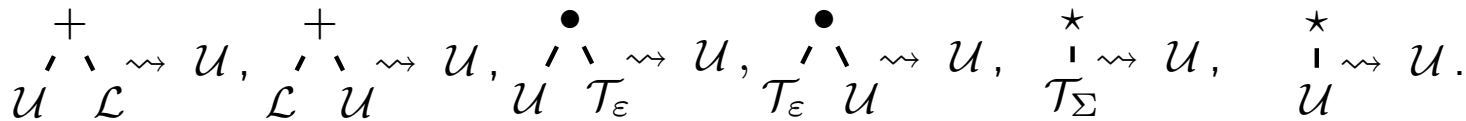


(II) : $(b^\star \cdot (a + \varepsilon))^\star$

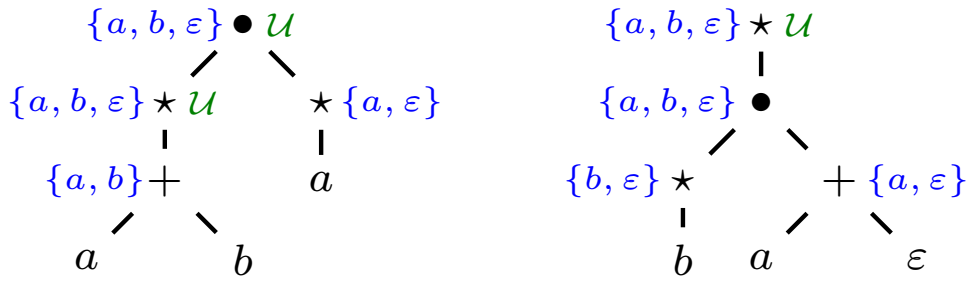
Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

- We define bottom-up propagation rules



- Examples for $\Sigma = \{a, b\}$,



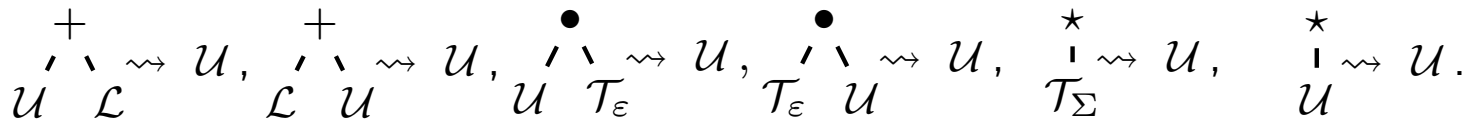
(I) : $(a + b)^\star \cdot a^\star$ **(II)** : $(b^\star \cdot (a + \varepsilon))^\star$

- Detection is only partial: example $\Sigma \cdot \Sigma^\star + \varepsilon$

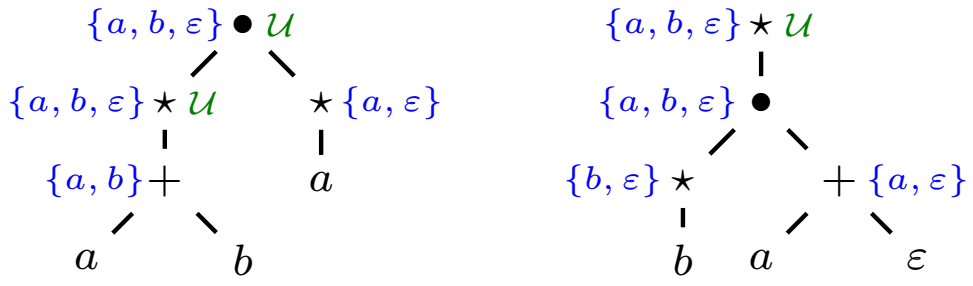
Universality detection: propagation rules

Idea: substitute universal subtrees by smallest universal tree $\mathcal{U}$.

- We define bottom-up propagation rules



- Examples for $\Sigma = \{a, b\}$,



(I) : $(a + b)^\star \cdot a^\star$ **(II)** : $(b^\star \cdot (a + \varepsilon))^\star$

- Detection is only partial: example $\Sigma \cdot \Sigma^\star + \varepsilon$

$\Longrightarrow$ universality problem is PSPACE-complete !

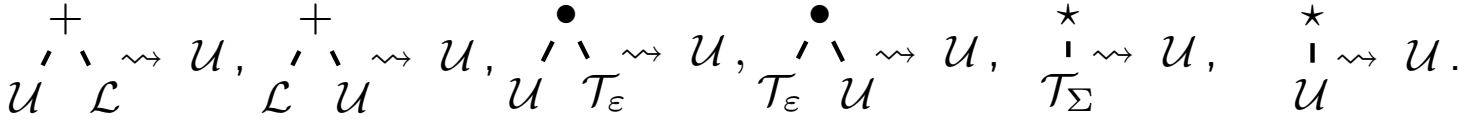
Results I

Rewriting rules:

$$\begin{tikzcd}[column sep=small] & + \arrow[dl, dash] \arrow[dr, dash] & \\ \mathcal{U} & & \mathcal{L} \end{tikzcd} \rightsquigarrow \mathcal{U}, \quad \begin{tikzcd}[column sep=small] & + \arrow[dl, dash] \arrow[dr, dash] & \\ \mathcal{L} & & \mathcal{U} \end{tikzcd} \rightsquigarrow \mathcal{U}, \quad \begin{tikzcd}[column sep=small] & \bullet \arrow[dl, dash] \arrow[dr, dash] & \\ \mathcal{U} & & \mathcal{T}_\varepsilon \end{tikzcd} \rightsquigarrow \mathcal{U}, \quad \begin{tikzcd}[column sep=small] & \bullet \arrow[dl, dash] \arrow[dr, dash] & \\ \mathcal{T}_\varepsilon & & \mathcal{U} \end{tikzcd} \rightsquigarrow \mathcal{U}, \quad \begin{tikzcd}[column sep=small] & \star \arrow[d, dash] & \\ & \mathcal{T}_\Sigma & \end{tikzcd} \rightsquigarrow \mathcal{U}, \quad \begin{tikzcd}[column sep=small] & \star \arrow[d, dash] & \\ & \mathcal{U} & \end{tikzcd} \rightsquigarrow \mathcal{U}.$$

Results I

Rewriting rules:

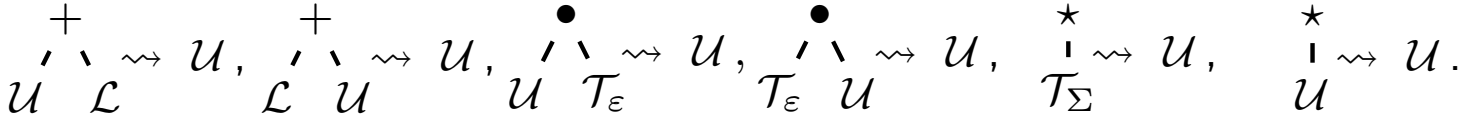


Main result

Consider the regular expression trees over $\Sigma = \{a_1, \ldots, a_k\}$.
Take the bottom-up simplification σ induced by our rewriting rules.

Results I

Rewriting rules:



Main result

Consider the regular expression trees over $\Sigma = \{a_1, \ldots, a_k\}$.
Take the bottom-up simplification σ induced by our rewriting rules.

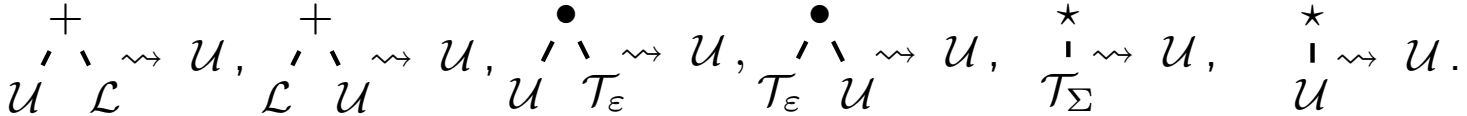
Then the expected size of the simplification of a random uniform tree tends to a **constant** as the size n tends to infinity.

Moreover, the constant can be computed efficiently

$\lvert\Sigma\rvert$	2	3	4	5
$\lim \mathbb{E}_n[\lvert\sigma(T)\rvert]$	$77.79724\ldots$	$495.59151\ldots$	$2\,518.20513\ldots$	$11\,694.43727\ldots$

Results I

Rewriting rules:



Main result

Consider the regular expression trees over $\Sigma = \{a_1, \ldots, a_k\}$.
Take the bottom-up simplification σ induced by our rewriting rules.

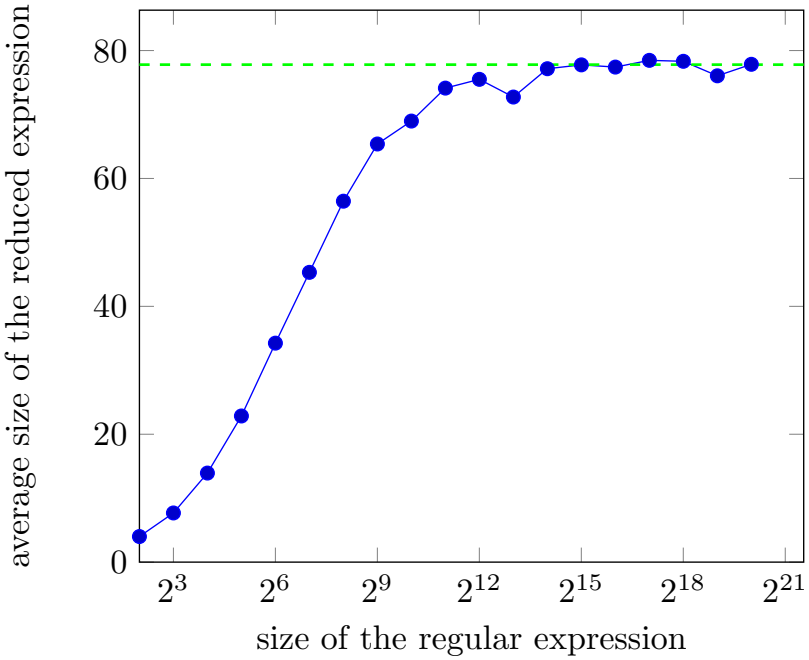
Then the expected size of the simplification of a random uniform tree tends to a **constant** as the size n tends to infinity.

Moreover, the constant can be computed efficiently

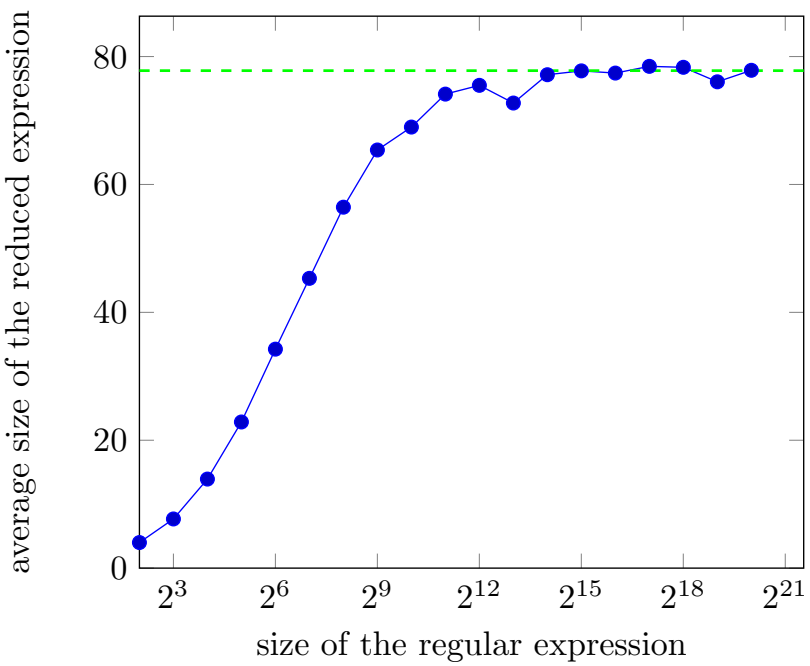
$\lvert\Sigma\rvert$	2	3	4	5
$\lim \mathbb{E}_n[\lvert\sigma(T)\rvert]$	$77.79724\ldots$	$495.59151\ldots$	$2\,518.20513\ldots$	$11\,694.43727\ldots$

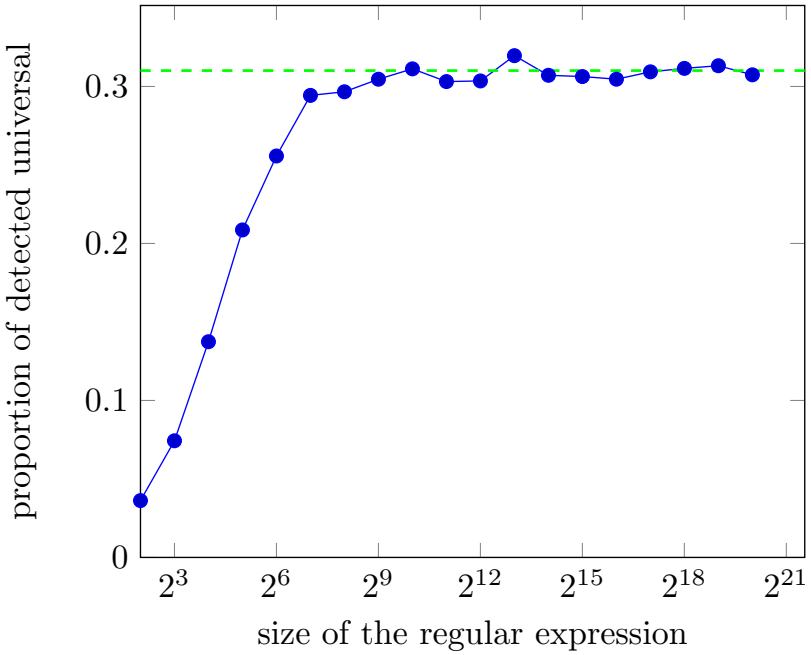
Note. Compare ~ 77.8 against previous $\sim 3.6 \times 10^6$ for two letters.

Results: plots



Results: plots





Results II

Proposition

For n large enough, the proportion $\mathrm{Pr}_n(\texttt{univ.})$ of universal expressions trees belongs to the intervals:

$\lvert\Sigma\rvert$	2	3	4	5
`interval`	$(0.31, 0.46)$	$(0.13, 0.27)$	$(0.062, 0.15)$	$(0.028, 0.077)$

Results II

Proposition

For n large enough, the proportion $\mathrm{Pr}_n(\texttt{univ.})$ of universal expressions trees belongs to the intervals:

$\lvert\Sigma\rvert$	2	3	4	5
`interval`	$(0.31, 0.46)$	$(0.13, 0.27)$	$(0.062, 0.15)$	$(0.028, 0.077)$

- Preponderance of universal expression trees:
  between 31% and 46% for two letters $\{a, b\}$
- Uniform model not adapted to sampling regular languages

Scheme of the proof

We employ Analytic Combinatorics to study the expectation,

- Bivariate generating function

$$L(z,u) := \sum_{T \in \mathcal{L}} u^{|\sigma(T)|} z^{|T|} \implies \mathbb{E}_n[|\sigma(T)|] = \frac{[z^n]\partial_u L(z,u)|_{u=1}}{[z^n]L(z,u)|_{u=1}} ,$$

encodes input and output sizes.

Scheme of the proof

We employ Analytic Combinatorics to study the expectation,

- Bivariate generating function

$$L(z,u) := \sum_{T \in \mathcal{L}} u^{|\sigma(T)|} z^{|T|} \implies \mathbb{E}_n[|\sigma(T)|] = \frac{[z^n]\partial_u L(z,u)|_{u=1}}{[z^n]L(z,u)|_{u=1}} ,$$

  encodes input and output sizes.

- **Symbolic Step.** We find a formal equation describing $L(z,u)$.

  Here this is done from a *combinatorial specificiation*

$$\boldsymbol{y}(z,u) = \boldsymbol{F}(z,u;\boldsymbol{y}(z,u)) .$$

Scheme of the proof

We employ Analytic Combinatorics to study the expectation,

- Bivariate generating function

$$L(z,u) := \sum_{T \in \mathcal{L}} u^{|\sigma(T)|} z^{|T|} \implies \mathbb{E}_n[|\sigma(T)|] = \frac{[z^n]\partial_u L(z,u)|_{u=1}}{[z^n]L(z,u)|_{u=1}} ,$$

  encodes input and output sizes.

- **Symbolic Step.** We find a formal equation describing $L(z,u)$.

  Here this is done from a *combinatorial specificiation*

$$\boldsymbol{y}(z,u) = \boldsymbol{F}(z,u;\boldsymbol{y}(z,u)) .$$

- **Analytic Step.** A *Transfer Theorem* links the behaviour at dominant singularities $\rho \in \mathbb{C}$ to asymptotics of coefficients

$$L(z) \sim_{z \to \rho} \lambda(1 - z/\rho)^{-\alpha} \implies [z^n]L(z) \sim \lambda n^{\alpha-1}/\Gamma(\alpha)\rho^{-n} .$$

Scheme of the proof

We employ Analytic Combinatorics to study the expectation,

- Bivariate generating function

$$L(z,u) := \sum_{T \in \mathcal{L}} u^{|\sigma(T)|} z^{|T|} \implies \mathbb{E}_n[|\sigma(T)|] = \frac{[z^n]\partial_u L(z,u)|_{u=1}}{[z^n]L(z,u)|_{u=1}} ,$$

  encodes input and output sizes.

- **Symbolic Step.** We find a formal equation describing $L(z,u)$.

  Here this is done from a *combinatorial specificiation*

$$\boldsymbol{y}(z,u) = \boldsymbol{F}(z,u;\boldsymbol{y}(z,u)) .$$

- **Analytic Step.** A *Transfer Theorem* links the behaviour at dominant singularities $\rho \in \mathbb{C}$ to asymptotics of coefficients

$$L(z) \sim_{z \to \rho} \lambda(1 - z/\rho)^{-\alpha} \implies [z^n]L(z) \sim \lambda n^{\alpha-1}/\Gamma(\alpha)\rho^{-n} .$$

  $\Rightarrow$ Study asymptotics over $z \in \mathbb{C}$

Combinatorial specification: two letters $\Sigma = \{a, b\}$

For every $X \subseteq \{a, b\}$ introduce:

- $\mathcal{T}_{X,\varepsilon}$ the set of trees recognizing every letter in X and ε, and no letter not in X
- $\mathcal{T}_{X,\overline{\varepsilon}}$ the set of trees recognizing every letter in X, and no letter not in X, nor ε

Combinatorial specification: two letters $\Sigma = \{a, b\}$

For every $X \subseteq \{a, b\}$ introduce:

- $\mathcal{T}_{X,\varepsilon}$ the set of trees recognizing every letter in X and ε, and no letter not in X
- $\mathcal{T}_{X,\overline{\varepsilon}}$ the set of trees recognizing every letter in X, and no letter not in X, nor ε

$$\mathcal{T}_{X,\varepsilon} = \varepsilon \mathbf{1}_{X=\emptyset} + \overset{\star}{\underset{\mathcal{T}_{X,\varepsilon}}{|}} + \overset{\star}{\underset{\mathcal{T}_{X,\overline{\varepsilon}}}{|}} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{S',\varepsilon}$$

$$+ \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{+}{\wedge} \mathcal{T}_{S',\varepsilon} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{+}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\overline{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\varepsilon}$$

$$\mathcal{T}_{X,\overline{\varepsilon}} = X \mathbf{1}_{|X|=1} + \sum_{S\subseteq\Sigma} \mathcal{T}_{X,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S,\varepsilon} + \sum_{S\subseteq\Sigma} \mathcal{T}_{S,\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{X,\overline{\varepsilon}} + \mathbf{1}_{X=\emptyset} \sum_{S,S'\subseteq\Sigma} \mathcal{T}_{S,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}}$$

$$+ \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\overline{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}},$$

Example: combinatorial specification

Trees recognizing the letter a and no other letter, and not recognizing ε

$$\mathcal{T}_{\{a\},\overline{\varepsilon}} = a + \underset{\mathcal{T}_{\{a\},\overline{\varepsilon}}}{\bullet}\!\!\begin{array}{c}\mathcal{T}_{\emptyset,\varepsilon} + \mathcal{T}_{\{a\},\varepsilon} \\ +\mathcal{T}_{\{b\},\varepsilon} + \mathcal{T}_{\{a,b\},\varepsilon}\end{array} + \underset{\begin{array}{c}\mathcal{T}_{\emptyset,\varepsilon} + \mathcal{T}_{\{a\},\varepsilon} \\ +\mathcal{T}_{\{b\},\varepsilon} + \mathcal{T}_{\{a,b\},\varepsilon}\end{array}}{\bullet}\!\!\mathcal{T}_{\{a\},\overline{\varepsilon}}$$

$$+ \underset{\mathcal{T}_{\emptyset,\overline{\varepsilon}}}{+}\!\!\mathcal{T}_{\{a\},\overline{\varepsilon}} + \underset{\mathcal{T}_{\{a\},\overline{\varepsilon}}}{+}\!\!\mathcal{T}_{\emptyset,\overline{\varepsilon}} + \mathcal{T}_{\{a\},\overline{\varepsilon}}$$

Fully reducible trees: a base case for output size

Definition (Fully reducible expressions)

A regular expression tree T is fully reducible when $\sigma(T) = \mathcal{U}$.

In other words, it is recognized as universal by our algorithm.

Fully reducible trees: a base case for output size

Definition (Fully reducible expressions)

A regular expression tree T is fully reducible when $\sigma(T) = \mathcal{U}$.

In other words, it is recognized as universal by our algorithm.

- Dictate the reduction process: leaves of reduced expression.

Fully reducible trees: a base case for output size

Definition (Fully reducible expressions)

A regular expression tree T is fully reducible when $\sigma(T) = \mathcal{U}$.

In other words, it is recognized as universal by our algorithm.

- Dictate the reduction process: leaves of reduced expression.
- Size after reduction $p = |\mathcal{U}|$ for $T \in \mathcal{R}$.

Fully reducible trees: a base case for output size

Definition (Fully reducible expressions)

A regular expression tree T is fully reducible when $\sigma(T) = \mathcal{U}$.

In other words, it is recognized as universal by our algorithm.

- Dictate the reduction process: leaves of reduced expression.
- Size after reduction $p = |\mathcal{U}|$ for $T \in \mathcal{R}$.
- The class of fully reducible trees $\mathcal{R}$ satisfies the equation:

$$\mathcal{R} = \overset{\star}{\underset{\mathcal{T}_{\Sigma,\bar{\varepsilon}}}{|}} + \overset{\star}{\underset{\mathcal{T}_{\Sigma,\varepsilon}}{|}} + \overset{+}{\underset{\mathcal{R}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\backslash\mathcal{R}\ \mathcal{R}}{\wedge}} + \overset{\bullet}{\underset{\mathcal{R}\ \mathcal{T}_{\varepsilon}}{\wedge}} + \overset{\bullet}{\underset{\mathcal{T}_{\varepsilon}\backslash\mathcal{R}\ \mathcal{R}}{\wedge}}.$$

$\Longrightarrow$ completes the combinatorial specification of $L(z, u)$.

Solving efficiently: auxiliary classes

- every tree : $\mathcal{L} = \bigcup_X \mathcal{T}_{X,\varepsilon} \cup \mathcal{T}_{X,\overline{\varepsilon}}$

$$\mathcal{L} = a + b + \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\ \mathcal{L}}{\wedge}}$$

$$L(z) = 3z + zL(z) + 2z(L(z))^2$$

- trees recognizing ε : $\mathcal{T}_\varepsilon = \bigcup_X \mathcal{T}_{X,\varepsilon}$

$$\mathcal{T}_\varepsilon = \varepsilon + \overset{\star}{\underset{\mathcal{L}}{|}} + \overset{\bullet}{\underset{\mathcal{T}_\varepsilon\ \mathcal{T}_\varepsilon}{\wedge}} + \overset{+}{\underset{\mathcal{T}_\varepsilon\ \mathcal{L}}{\wedge}} + \overset{+}{\underset{\mathcal{L}\setminus\mathcal{T}_\varepsilon\ \mathcal{T}_\varepsilon}{\wedge}}\ .$$

$$T_\varepsilon(z) = \frac{z + zL(z)}{1 - 2zL(z)}$$

- trees not recognizing ε : $T_{\overline{\varepsilon}}(z) = L(z) - T_\varepsilon(z)$

The system becomes triangular

$$
\begin{aligned}
T_{\emptyset,\overline{\varepsilon}}(z) &= function(T_{\emptyset,\overline{\varepsilon}}(z)) \\
T_{\{a\},\overline{\varepsilon}}(z) &= function(T_{\{a\},\overline{\varepsilon}}(z), T_{\emptyset,\overline{\varepsilon}}(z)) \\
T_{\{b\},\overline{\varepsilon}}(z) &= function(T_{\{b\},\overline{\varepsilon}}(z), T_{\emptyset,\overline{\varepsilon}}(z)) \\
T_{\{a,b\},\overline{\varepsilon}}(z) &= function(T_{\{a,b\},\overline{\varepsilon}}(z), T_{\{a\},\overline{\varepsilon}}(z), T_{\{b\},\overline{\varepsilon}}(z), T_{\emptyset,\overline{\varepsilon}}(z)) \\
T_{\emptyset,\varepsilon}(z) &= function(T_{\emptyset,\varepsilon}(z), T_{\emptyset,\overline{\varepsilon}}(z)) \\
&\vdots \\
T_{\{a,b\},\varepsilon}(z) &= function(T_{\{a,b\},\varepsilon}(z), \texttt{and everyone above})
\end{aligned}
$$

- Each equation is of degree $2 \Rightarrow$ exactly solvable

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

where $\Delta(z)$ is the determinant of the equation for $L(z)$.

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

More generally

- square-root behaviour generalizes to $T_{X,\varepsilon}$ and $T_{X,\overline{\varepsilon}}$,

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

More generally

- square-root behaviour generalizes to $T_{X,\varepsilon}$ and $T_{X,\overline{\varepsilon}}$,
  and for every $k = |\Sigma| \Rightarrow$ use Drmota's Theorem.

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

More generally

- square-root behaviour generalizes to $T_{X,\varepsilon}$ and $T_{X,\overline{\varepsilon}}$,
  and for every $k = |\Sigma| \Rightarrow$ use Drmota's Theorem.
- then to $\partial_u L(z,u)|_{u=1}$, numerator of the expectation. [Closure]

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

More generally

- square-root behaviour generalizes to $T_{X,\varepsilon}$ and $T_{X,\overline{\varepsilon}}$,
  and for every $k = |\Sigma| \Rightarrow$ use Drmota's Theorem.
- then to $\partial_u L(z,u)|_{u=1}$, numerator of the expectation. [Closure]
- Coefficients A and B determine asymptotics [Transfer Theorem]

Analytic step: square-root singularity

The expression

$$T_{\{a,b\},\overline{\varepsilon}}(z) = \frac{1}{4z}\left(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\right),$$

implies a *square-root behaviour*

$$T_{\{a,b\},\overline{\varepsilon}}(z) \sim A - B\sqrt{1 - z/\rho}$$

for z close to dominant singularity ρ.

More generally

- square-root behaviour generalizes to $T_{X,\varepsilon}$ and $T_{X,\overline{\varepsilon}}$,
  and for every $k = |\Sigma| \Rightarrow$ use Drmota's Theorem.
- then to $\partial_u L(z,u)|_{u=1}$, numerator of the expectation. [Closure]
- Coefficients A and B determine asymptotics [Transfer Theorem]
  $\Longrightarrow$ we show how to compute these efficiently.

Conclusion and further work

- We have shown a simple linear algorithm, reducing uniform regular expressions to small constant size.
- Therefore, uniform random regular expression trees tend to describe very limited languages.

Conclusion and further work

- We have shown a simple linear algorithm, reducing uniform regular expressions to small constant size.
- Therefore, uniform random regular expression trees tend to describe very limited languages.

Future work

- Other distributions seem more appropriate (BST, ...)
- Algorithm (partially) detects universality, improvements ?

Conclusion and further work

- We have shown a simple linear algorithm, reducing uniform regular expressions to small constant size.
- Therefore, uniform random regular expression trees tend to describe very limited languages.

Future work

- Other distributions seem more appropriate (BST, ...)
- Algorithm (partially) detects universality, improvements ?

Thank you!