Uniform Random Expressions Lack Expressivity

Pablo Rotondo

LIGM, Université Paris-Est Marne-la-Vallée

Joint work with

Florent Koechlin and Cyril Nicaud

LIGM, Université Paris-Est Marne-la-Vallée

MFCS 19',

Aachen, 28 August, 2019.

Introduction

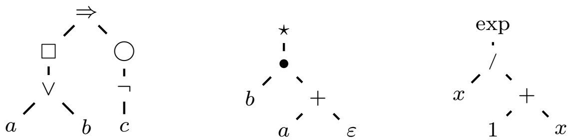
- Uniformly random input
  - Yields diverse values
  - Convenient methods: recursive, Boltzmann samplers.

Introduction

- Uniformly random input
  - Yields diverse values
  - Convenient methods: recursive, Boltzmann samplers.
- Automated testing, benchmark testing
  - Correctness and performance of algorithms

Introduction

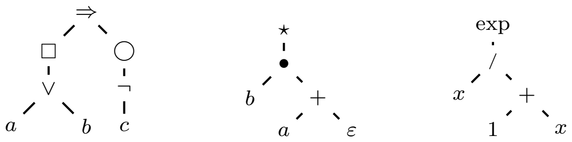
- Uniformly random input
  - Yields diverse values
  - Convenient methods: recursive, Boltzmann samplers.
- Automated testing, benchmark testing
  - Correctness and performance of algorithms
- Expression trees



$\Box(a \vee b) \Rightarrow \bigcirc\neg c$ $\quad$ $(b \cdot (a + \varepsilon))^\star$ $\quad$ $\exp(\frac{x}{1+x})$

Introduction

- Uniformly random input
  - Yields diverse values
  - Convenient methods: recursive, Boltzmann samplers.
- Automated testing, benchmark testing
  - Correctness and performance of algorithms
- Expression trees



$\Box(a \vee b) \Rightarrow \bigcirc \neg c$ $\quad$ $(b \cdot (a + \varepsilon))^{\star}$ $\quad$ $\exp(\frac{x}{1+x})$

Distribution of the resulting objects? $\Rightarrow$ may be bad!

Plan of the talk

1. Expression trees

2. Random expressions and results

3. Toolbox

4. Conclusions

Combinatorial expressions

Let $\mathcal{A} = (\mathcal{A}_i)_i$ be a family of finite sets of labels, indexed on $\mathbb{Z}_{\geq 0}$, with the conditions $\mathcal{A}_0 \neq \emptyset$ and $\mathcal{A}_i \neq \emptyset$ for some $i \geq 2$.

Definition

A combinatorial expression on $\mathcal{A}$ is a rooted tree in which nodes of arity i are labeled exclusively on $\mathcal{A}_i$.

We denote the set of all combinatorial expressions by $\mathcal{L} = \mathcal{L}(\mathcal{A})$.

Combinatorial expressions

Let $\mathcal{A} = (\mathcal{A}_i)_i$ be a family of finite sets of labels, indexed on $\mathbb{Z}_{\geq 0}$, with the conditions $\mathcal{A}_0 \neq \emptyset$ and $\mathcal{A}_i \neq \emptyset$ for some $i \geq 2$.

Definition

A combinatorial expression on $\mathcal{A}$ is a rooted tree in which nodes of arity i are labeled exclusively on $\mathcal{A}_i$.

We denote the set of all combinatorial expressions by $\mathcal{L} = \mathcal{L}(\mathcal{A})$.

Our battle horse

Regular expressions $\mathcal{L}_R$ over the alphabet $\{a, b\}$ are defined by

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c}\star\\ |\\ \mathcal{L}_R\end{array} + \begin{array}{c}\bullet\\ / \ \backslash\\ \mathcal{L}_R \ \ \mathcal{L}_R\end{array} + \begin{array}{c}+\\ / \ \backslash\\ \mathcal{L}_R \ \ \mathcal{L}_R\end{array}.$$

Combinatorial expressions

Let $\mathcal{A} = (\mathcal{A}_i)_i$ be a family of finite sets of labels, indexed on $\mathbb{Z}_{\geq 0}$, with the conditions $\mathcal{A}_0 \neq \emptyset$ and $\mathcal{A}_i \neq \emptyset$ for some $i \geq 2$.

Definition

A combinatorial expression on $\mathcal{A}$ is a rooted tree in which nodes of arity i are labeled exclusively on $\mathcal{A}_i$.

We denote the set of all combinatorial expressions by $\mathcal{L} = \mathcal{L}(\mathcal{A})$.

Our battle horse

Regular expressions $\mathcal{L}_R$ over the alphabet $\{a, b\}$ are defined by

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c}\star\\ | \\ \mathcal{L}_R\end{array} + \begin{array}{c}\bullet\\ \wedge \\ \mathcal{L}_R \;\; \mathcal{L}_R\end{array} + \begin{array}{c}+\\ \wedge \\ \mathcal{L}_R \;\; \mathcal{L}_R\end{array}.$$

Equivalently, *combinatorial expressions* with labels

$$\mathcal{A}_0 = \{a, b, \varepsilon\}, \quad \mathcal{A}_1 = \{\star\}, \quad \mathcal{A}_2 = \{\bullet, +\},$$

and $\mathcal{A}_i = \emptyset$ for $i \geq 3$.

Combinatorial expressions and Analytic Combinatorics

Expressions naturally adapted to *Analytic Combinatorics*

- size $|T|$ of tree expression $T \in \mathcal{L}$ given by number of nodes.

Combinatorial expressions and Analytic Combinatorics

Expressions naturally adapted to *Analytic Combinatorics*

- size $|T|$ of tree expression $T \in \mathcal{L}$ given by number of nodes.
- consider the *ordinary generating function* $L(z) = \sum_{T \in \mathcal{L}} z^{|T|}$

Combinatorial expressions and Analytic Combinatorics

Expressions naturally adapted to *Analytic Combinatorics*

- size $|T|$ of tree expression $T \in \mathcal{L}$ given by number of nodes.
- consider the *ordinary generating function* $L(z) = \sum_{T \in \mathcal{L}} z^{|T|}$
- coefficient $[z^n]L(z)$ counts tree expressions with n nodes.

Combinatorial expressions and Analytic Combinatorics

Expressions naturally adapted to *Analytic Combinatorics*

- size $|T|$ of tree expression $T \in \mathcal{L}$ given by number of nodes.
- consider the *ordinary generating function* $L(z) = \sum_{T \in \mathcal{L}} z^{|T|}$
- coefficient $[z^n]L(z)$ counts tree expressions with n nodes.

$\Longrightarrow$ Specification translates into functional equation

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c} \star \\ | \\ \mathcal{L}_R \end{array} + \begin{array}{c} \bullet \\ / \backslash \\ \mathcal{L}_R \;\; \mathcal{L}_R \end{array} + \begin{array}{c} + \\ / \backslash \\ \mathcal{L}_R \;\; \mathcal{L}_R \end{array} \Rightarrow L(z) = 3z + zL(z) + 2z(L(z))^2\,.$$

Combinatorial expressions and Analytic Combinatorics

Expressions naturally adapted to *Analytic Combinatorics*

- size $|T|$ of tree expression $T \in \mathcal{L}$ given by number of nodes.
- consider the *ordinary generating function* $L(z) = \sum_{T \in \mathcal{L}} z^{|T|}$
- coefficient $[z^n]L(z)$ counts tree expressions with n nodes.

$\Longrightarrow$ Specification translates into functional equation

$$\mathcal{L}_R = a + b + \varepsilon + \underset{\mathcal{L}_R}{\overset{\star}{|}} + \underset{\mathcal{L}_R \quad \mathcal{L}_R}{\overset{\bullet}{/\backslash}} + \underset{\mathcal{L}_R \quad \mathcal{L}_R}{\overset{+}{/\backslash}} \Rightarrow L(z) = 3z + zL(z) + 2z(L(z))^2 \, .$$

More generally, for combinatorial expressions

$$L(z) = z \cdot \phi(L(z)) \, , \qquad \phi(z) := \sum_{i=0}^{\infty} |\mathcal{A}_i| \, z^i \, .$$

Absorbing patterns: simplifying the trees

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c}\star\\ |\\ \mathcal{L}_R\end{array} + \begin{array}{c}\bullet\\ /\backslash\\ \mathcal{L}_R\ \mathcal{L}_R\end{array} + \begin{array}{c}+\\ /\backslash\\ \mathcal{L}_R\ \mathcal{L}_R\end{array}.$$

- Representation of languages not minimal.

Absorbing patterns: simplifying the trees

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c}\star\\ |\\ \mathcal{L}_R\end{array} + \begin{array}{c}\bullet\\ /\ \backslash\\ \mathcal{L}_R\ \ \mathcal{L}_R\end{array} + \begin{array}{c}+\\ /\ \backslash\\ \mathcal{L}_R\ \ \mathcal{L}_R\end{array}.$$

- Representation of languages not minimal.
- Perform simple reductions on trees
  - Let $\mathcal{P} := \begin{array}{c}\star\\ |\\ +\\ /\ \backslash\\ a\quad b\end{array}$, representing language of all words.
  - Make the (quite simple) reductions

$$\begin{array}{c}+\\ /\ \backslash\\ \mathcal{P}\quad \cdot\end{array} \rightsquigarrow \mathcal{P} \qquad \begin{array}{c}+\\ /\ \backslash\\ \cdot\quad \mathcal{P}\end{array} \rightsquigarrow \mathcal{P}$$

Absorbing patterns: simplifying the trees

$$\mathcal{L}_R = a + b + \varepsilon + \begin{array}{c}\star\\ | \\ \mathcal{L}_R\end{array} + \begin{array}{c}\bullet\\ / \backslash \\ \mathcal{L}_R \quad \mathcal{L}_R\end{array} + \begin{array}{c}+\\ / \backslash \\ \mathcal{L}_R \quad \mathcal{L}_R\end{array}.$$

- Representation of languages not minimal.
- Perform simple reductions on trees
  - Let $\mathcal{P} := \begin{array}{c}\star\\ | \\ + \\ / \backslash \\ a \quad b\end{array}$, representing language of all words.
  - Make the (quite simple) reductions

$$\begin{array}{c}+\\ / \backslash \\ \mathcal{P} \quad \cdot\end{array} \rightsquigarrow \mathcal{P} \qquad \begin{array}{c}+\\ / \backslash \\ \cdot \quad \mathcal{P}\end{array} \rightsquigarrow \mathcal{P}$$

This is an absorbing pattern, element $\mathcal{P}$ reduces the operator $+$.

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

Let $\mathcal{L}$ be the family of combinatorial expressions over $\mathcal{A} = (\mathcal{A}_i)$, consider

- an "operation" $\circledast \in \mathcal{A}_a$ with arity $a \geq 2$,
- an expression tree $\mathcal{P} \in \mathcal{L}$.

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

Let $\mathcal{L}$ be the family of combinatorial expressions over $\mathcal{A} = (\mathcal{A}_i)$, consider

- an "operation" $\circledast \in \mathcal{A}_a$ with arity $a \geq 2$,
- an expression tree $\mathcal{P} \in \mathcal{L}$.

We simplify by applying bottom-up the rule:

$$\begin{array}{c} \circledast \\ / \quad \backslash \\ C_1 \cdots C_a \end{array} \quad \rightsquigarrow \mathcal{P}\,, \text{ whenever } C_i = \mathcal{P} \text{ for some } i \in \{1, \ldots, a\}.$$

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

Let $\mathcal{L}$ be the family of combinatorial expressions over $\mathcal{A} = (\mathcal{A}_i)$, consider

- an "operation" $\circledast \in \mathcal{A}_a$ with arity $a \geq 2$,
- an expression tree $\mathcal{P} \in \mathcal{L}$.

We simplify by applying bottom-up the rule:

$$\begin{array}{c} \circledast \\ / \quad \backslash \\ C_1 \cdots C_a \end{array} \rightsquigarrow \mathcal{P}\text{, whenever } C_i = \mathcal{P} \text{ for some } i \in \{1, \ldots, a\}.$$

$\Rightarrow$ We are interested in the *size* of the trees after simplification.

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

Let $\mathcal{L}$ be the family of combinatorial expressions over $\mathcal{A} = (\mathcal{A}_i)$, consider

- an "operation" $\circledast \in \mathcal{A}_a$ with arity $a \geq 2$,
- an expression tree $\mathcal{P} \in \mathcal{L}$.

We simplify by applying bottom-up the rule:

$$\begin{array}{c} \circledast \\ / \quad \backslash \\ C_1 \cdots C_a \end{array} \rightsquigarrow \mathcal{P}\text{, whenever } C_i = \mathcal{P} \text{ for some } i \in \{1, \ldots, a\}.$$

$\Rightarrow$ We are interested in the *size* of the trees after simplification.

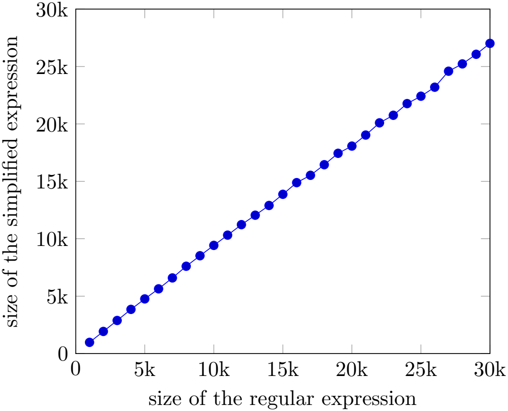
Denote by $\sigma(T) = \sigma(T, \mathcal{P}, \circledast)$ the simplification of $T \in \mathcal{L}$.

Model for random trees

In our work we

- draw an expression tree of size n uniformly at random.
- study expected values and moments of

  sizes of reduced expressions as $n \to \infty$.

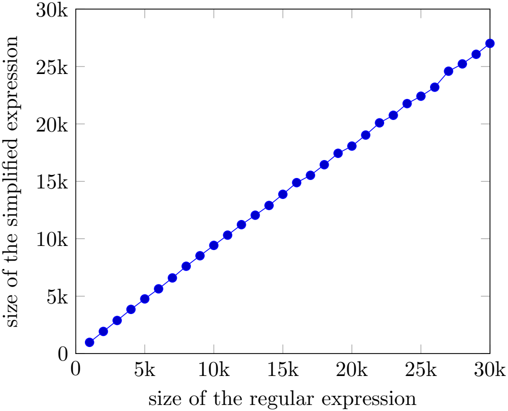


Model for random trees

In our work we

- draw an expression tree of size n uniformly at random.
- study expected values and moments of

  sizes of reduced expressions as $n \to \infty$.



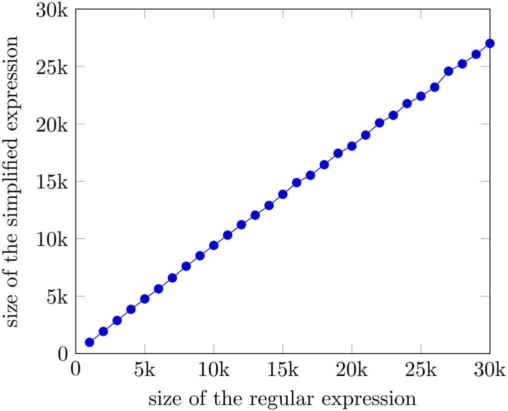
The average size seemingly tends linearly to infinity...

Model for random trees

In our work we

- draw an expression tree of size n uniformly at random.
- study expected values and moments of

  sizes of reduced expressions as $n \to \infty$.



The average size seemingly tends linearly to infinity... yet **it does not!**

Main result

Theorem (Informal version)

Consider a simple variety of expressions with an absorbing pattern $\mathcal{P}$ *for one of the operators* $\circledast$*.*

Take the simplification consisting in inductively changing a $\circledast$*-node by* $\mathcal{P}$ *whenever one of its children simplifies to* $\mathcal{P}$*.*

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

Main result

Theorem (Informal version)

Consider a simple variety of expressions with an absorbing pattern $\mathcal{P}$ for one of the operators $\circledast$.

Take the simplification consisting in inductively changing a $\circledast$-node by $\mathcal{P}$ whenever one of its children simplifies to $\mathcal{P}$.

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

Example

For the regular expressions $\mathcal{L}_R$ on $\{a, b\}$,

$$\delta \approx 3\ 624\ 217\ .$$

Main result

Theorem

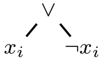
Let $\mathcal{L} = \mathcal{L}(\mathcal{A})$ *a set of combinatorial expressions whose GF* $L(z)$ *belongs to the* $\textit{smooth inverse-function}$ *schema* $L(z) = z \cdot \phi(L(z))$, *with* ϕ $\textit{aperiodic}$. *Let* $\mathcal{P} \in \mathcal{L}$ *and let* $\circledast$ *be an operator of arity* ≥ 2.

Main result

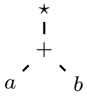
Theorem

Let $\mathcal{L} = \mathcal{L}(\mathcal{A})$ a set of combinatorial expressions whose GF $L(z)$ belongs to the smooth inverse-function *schema $L(z) = z \cdot \phi(L(z))$, with ϕ* aperiodic. *Let $\mathcal{P} \in \mathcal{L}$ and let $\circledast$ be an operator of arity ≥ 2.*

These hypotheses apply to a wide variety of expression families:



For logical formulas (operator $\vee$).



For regular expressions (operator $+$).

$x \mapsto 0$

For functions (operator $\cdot$).

Main result

Theorem

Let $\mathcal{L} = \mathcal{L}(\mathcal{A})$ a set of combinatorial expressions whose GF $L(z)$ belongs to the smooth inverse-function schema $L(z) = z \cdot \phi(L(z))$, with ϕ aperiodic. Let $\mathcal{P} \in \mathcal{L}$ and let $\circledast$ be an operator of arity ≥ 2.

Then, if $\sigma := s(T, \mathcal{P}, \circledast)$, where $|T| = n$ is chosen uniformly at random,

$$\lim_{n\to\infty} \mathbb{E}_n[\sigma] = \delta\,,$$

for some $0 < \delta < \infty$. Furthermore, for $i \in \mathbb{Z}_{\geq 1}$,

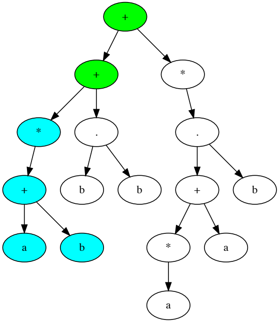
$$\lim_{n\to\infty} \mathbb{E}_n[\sigma^i] = \delta_i$$

for some positive δ_i.

Intuitions

Definition (Completely reducible expressions)

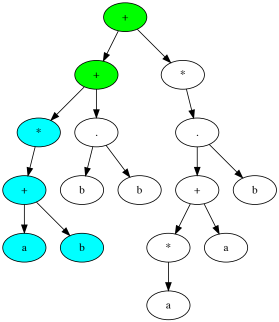
An expression tree T is completely reducible when $s(T, \mathcal{P}, \circledast) = \mathcal{P}$.



Intuitions

Definition (Completely reducible expressions)

An expression tree T is completely reducible when $s(T, \mathcal{P}, \circledast) = \mathcal{P}$.



Completely reducible expressions

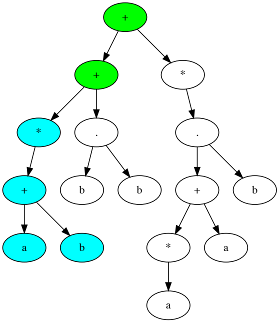
- are not a rarity

$$\lim_{n\to\infty} \mathbb{P}_n\left(T \text{completely reducible}\right) = C > 0\,.$$

Intuitions

Definition (Completely reducible expressions)

An expression tree T is completely reducible when $s(T, \mathcal{P}, \circledast) = \mathcal{P}$.



Completely reducible expressions

- are not a rarity

$$\lim_{n \to \infty} \mathbb{P}_n \left(T \text{completely reducible} \right) = C > 0 \,.$$

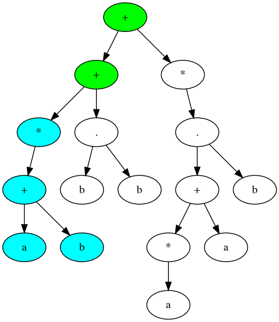
- dictate the reduction process:

  leaves of the reduced expression.

Intuitions

Definition (Completely reducible expressions)

An expression tree T is completely reducible when $s(T, \mathcal{P}, \circledast) = \mathcal{P}$.



Completely reducible expressions

- are not a rarity

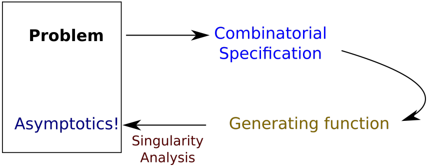
$$\lim_{n \to \infty} \mathbb{P}_n \left(T \text{completely reducible} \right) = C > 0 \,.$$

- dictate the reduction process:
  leaves of the reduced expression.
- can also be specified recursively, e.g.,

$$\mathcal{R} = \mathcal{P} + \underset{\mathcal{R} \quad \mathcal{L}}{\overset{+}{/ \backslash}} + \underset{\mathcal{L} \quad \mathcal{R}}{\overset{+}{/ \backslash}} \,.$$

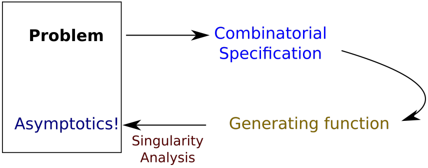
Proof principles: symbolic steps

Proof based on principles of Analytic Combinatorics:



Proof principles: symbolic steps

Proof based on principles of Analytic Combinatorics:

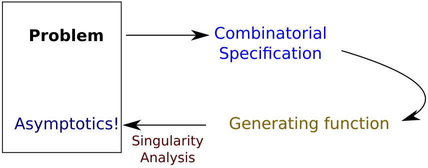


- bivariate generating functions

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

Proof principles: symbolic steps

Proof based on principles of Analytic Combinatorics:



- bivariate generating functions

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

- need appropriate expression for $L(z,u)$, e.g.,

$$\mathcal{L}_R = a + b + \varepsilon + \mathcal{R} \setminus \{\mathcal{P}\} + \begin{array}{c} + \\ / \quad \backslash \\ \mathcal{L}_R\setminus\mathcal{R} \quad \mathcal{L}_R\setminus\mathcal{R} \end{array} + \begin{array}{c} \bullet \\ / \quad \backslash \\ \mathcal{L}_R \quad \mathcal{L}_R \end{array} + \begin{array}{c} \star \\ | \\ \mathcal{L}_R \end{array}$$

$\implies$ functional equation for $L(z,u)$ involving $R(z,u)$.

Proof principles: analytic step

Theorem (Classical, see Flajolet&Sedgewick)

Let $\mathcal{L}$ be a set of combinatorial expressions whose GF $L(z)$ belongs to the smooth inverse-function schema $L(z) = z \cdot \phi(L(z))$.

Let $\tau > 0$ be the solution of $\phi(\tau) - \tau\phi'(\tau) = 0$, and $\rho := \tau/\phi(\tau)$.

Then we have that $L(z) = g(z) - h(z)\sqrt{1 - z/\rho}$ around $z = \rho$.

Transfer Theorem

When ϕ is aperiodic, this implies $[z^n]L(z) \sim C_L \rho^{-n}/n^{3/2}$.

Proof principles: analytic step

Theorem (Classical, see Flajolet&Sedgewick)

Let $\mathcal{L}$ be a set of combinatorial expressions whose GF $L(z)$ belongs to the smooth inverse-function schema $L(z) = z \cdot \phi(L(z))$.

Let $\tau > 0$ be the solution of $\phi(\tau) - \tau\phi'(\tau) = 0$, and $\rho := \tau/\phi(\tau)$.

Then we have that $L(z) = g(z) - h(z)\sqrt{1 - z/\rho}$ around $z = \rho$.

Transfer Theorem

When ϕ is aperiodic, this implies $[z^n]L(z) \sim C_L \rho^{-n}/n^{3/2}$.

For expectations we make use of extensions by Drmota

- $R(z)$, the GF of the completely reducible trees, [Multidim]
- $\partial_u L(z,u)|_{u=1}$, the numerator of the expectation, [Closure]

and then recall $\mathbb{E}_n[\sigma] = [z^n]\partial_u L(z,u)|_{u=1}/[z^n]L(z,u)|_{u=1}$.

Conclusions and further work

Conclusions

- Uniform random expressions often are not a suitable model.

Conclusions and further work

Conclusions

- Uniform random expressions often are not a suitable model.
- Algorithms with polynomial worst case
  $\Rightarrow$ constant on average after simplification (linear).

Conclusions and further work

Conclusions

- Uniform random expressions often are not a suitable model.
- Algorithms with polynomial worst case
  $\Rightarrow$ constant on average after simplification (linear).
- The constant $\delta \approx 3.6 \times 10^6$ may seem humongous
  $\Rightarrow$ adding simplification rules we reduce it to ≈ 75.

Conclusions and further work

Conclusions

- Uniform random expressions often are not a suitable model.
- Algorithms with polynomial worst case
  $\Rightarrow$ constant on average after simplification (linear).
- The constant $\delta \approx 3.6 \times 10^6$ may seem humongous
  $\Rightarrow$ adding simplification rules we reduce it to ≈ 75.

Questions and further work

1. Extend results to multidimensional systems of trees.

Conclusions and further work

Conclusions

- Uniform random expressions often are not a suitable model.
- Algorithms with polynomial worst case
  $\Rightarrow$ constant on average after simplification (linear).
- The constant $\delta \approx 3.6 \times 10^6$ may seem humongous
  $\Rightarrow$ adding simplification rules we reduce it to ≈ 75.

Questions and further work

1. Extend results to multidimensional systems of trees.
2. Experiments suggest that a similar situation holds for BSTs

Conclusions and further work

Conclusions

- ⊛ Uniform random expressions often are not a suitable model.
- ⊛ Algorithms with polynomial worst case
  $\Rightarrow$ constant on average after simplification (linear).
- ⊛ The constant $\delta \approx 3.6 \times 10^6$ may seem humongous
  $\Rightarrow$ adding simplification rules we reduce it to ≈ 75.

Questions and further work

1. Extend results to multidimensional systems of trees.
2. Experiments suggest that a similar situation holds for BSTs
3. Find suitable model!

Thank you!