Absorbing patterns in BST-like expression-trees

Pablo Rotondo

LIGM, Université Gustave Eiffel

Joint work with

Florent Koechlin

LIPN, Sorbonne Paris Nord

Meeting EPA!,
Buenos Aires, 24 October, 2024.

Plan of the talk

1. Random BST-like tree model
2. Semantic simplifications
3. Result for BST-like trees and elements of the proof
4. Conclusions

Section

1. Random BST-like tree model

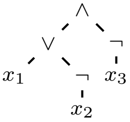
2. Semantic simplifications

3. Result for BST-like trees and elements of the proof

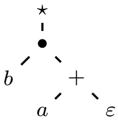
4. Conclusions

Introduction

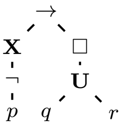
- Expression trees



$(x_1 \vee \neg x_2) \wedge \neg x_3$



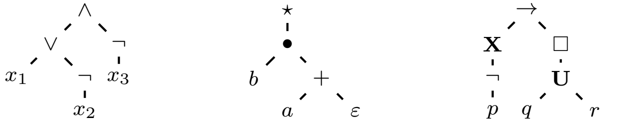
$(b \cdot (a + \varepsilon))^\star$



$\mathbf{X}(\neg p) \rightarrow \Box(q\mathbf{U}r)$

Introduction

- Expression trees

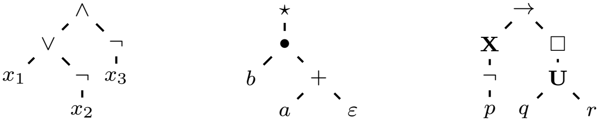


$(x_1 \vee \neg x_2) \wedge \neg x_3$ $\qquad$ $(b \cdot (a + \varepsilon))^\star$ $\qquad$ $\mathbf{X}(\neg p) \rightarrow \Box(q\mathbf{U}r)$

- Automated testing, benchmark testing
  - Correctness and performance of algorithms

Introduction

- Expression trees



$(x_1 \vee \neg x_2) \wedge \neg x_3$ $\quad$ $(b \cdot (a + \varepsilon))^\star$ $\quad$ $\mathbf{X}(\neg p) \to \Box(q\mathbf{U}r)$

- Automated testing, benchmark testing
  - Correctness and performance of algorithms

- Randomly generated input
  - Realistic distribution
  - Simple implementation, possibility of theoretical analysis.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

[1] If $n = 2$, constrained to arity one.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

Idea: fix probability vectors (p_{op}) and (p_a) for operators and leaves.

[1] If $n = 2$, constrained to arity one.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

Idea: fix probability vectors (p_{op}) and (p_a) for operators and leaves.

- If $n > 1$, pick operator[1] following (p_{op}), else pick random leaf (p_a).

[1] If $n = 2$, constrained to arity one.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

Idea: fix probability vectors (p_{op}) and (p_a) for operators and leaves.

- If $n > 1$, pick operator[1] following (p_{op}), else pick random leaf (p_a).
- if operator is binary, choose *size* of branches uniformly,

[1] If $n = 2$, constrained to arity one.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

Idea: fix probability vectors (p_{op}) and (p_a) for operators and leaves.

- If $n > 1$, pick operator[1] following (p_{op}), else pick random leaf (p_a).
- if operator is binary, choose *size* of branches uniformly,

$$\Pr_n(|T_L| = k) = \tfrac{1}{n-2}\,, \qquad k = 1, 2, \ldots, n-1\,.$$

- build sub-trees recursively and independently!

[1] If $n = 2$, constrained to arity one.

BST-like trees: a natural construction algorithm

Problem: sample random tree of size n,

- representing expression with unary and binary operators,
- leaves correspond to constants or variables.

Idea: fix probability vectors (p_{op}) and (p_a) for operators and leaves.

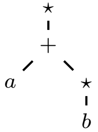
- If $n > 1$, pick operator[1] following (p_{op}), else pick random leaf (p_a).
- if operator is binary, choose *size* of branches uniformly,

$$\Pr_n(|T_L| = k) = \tfrac{1}{n-2}\,, \qquad k = 1, 2, \ldots, n-1\,.$$

- build sub-trees recursively and independently!

Example.

$\Pr_n \Bigg($

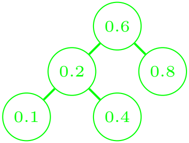


$\Bigg) = p_\star p_+ \frac{1}{2} p_a p_b$

[1] If $n = 2$, constrained to arity one.

Why binary search "like" ?

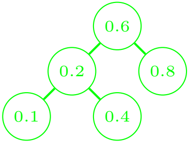
- Build BST from n random numbers $u_i \in [0, 1]$:



- With probability $\frac{1}{n}$, root corresponds to k-th ranked u_i.

Why binary search "like" ?

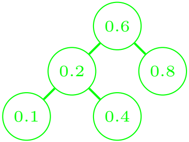
- Build BST from n random numbers $u_i \in [0, 1]$:



- With probability $\frac{1}{n}$, root corresponds to k-th ranked u_i.
- Equivalently, subtrees have sizes $|T_L| = k - 1$ and $|T_R| = n - k$.

Why binary search "like" ?

- Build BST from n random numbers $u_i \in [0, 1]$:



- With probability $\frac{1}{n}$, root corresponds to k-th ranked u_i.
- Equivalently, subtrees have sizes $|T_L| = k - 1$ and $|T_R| = n - k$.

Same construction: force our subtrees to have $|T_L|, |T_R| \geq 1$, as node corresponds to binary operator.

BST-like trees: a natural construction algorithm

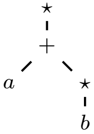
Code used in tool `lbtt` (from `TCS`) to draw an LTL formula:

function RandomFormula(n):
if $n = 1$ **then**
    p := random symbol in $AP \cup \{\top, \bot\}$;
    return p;
else if $n = 2$ **then**
    op := random unary operator in $\{\neg, \mathbf{X}, \Box, \Diamond\}$;
    f := RandomFormula(1);
    return $op\ f$;
else
    op := random operator in $\{\neg, \mathbf{X}, \Box, \Diamond, \wedge, \vee, \rightarrow, \leftrightarrow, \mathbf{U}, \mathbf{R}\}$;
    if op *in* $\{\neg, \mathbf{X}, \Box, \Diamond\}$ **then**
        f := RandomFormula($n - 1$);
        return $op\ f$;
    else
        x := uniform integer in $[1, n-2]$;
        f_1 := RandomFormula(x);
        f_2 := RandomFormula($n - x - 1$);
        return ($f_1\ op\ f_2$);

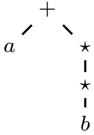
Example: BST-like distribution

Consider the regular expressions $(+, \bullet, \star)$ on two letters a, b

(i)



(ii)



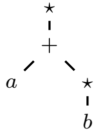
- The expression tree (i) is drawn with probability

$$p_\star p_+ \frac{1}{2} p_a p_b \,.$$

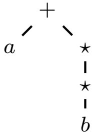
Example: BST-like distribution

Consider the regular expressions $(+, \bullet, \star)$ on two letters a, b

(i)



(ii)



- The expression tree (i) is drawn with probability

$$p_\star p_+ \frac{1}{2} p_a p_b \,.$$

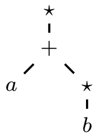
- The expression tree (ii) is drawn with probability

$$p_+ \frac{1}{3} p_a p_\star p_b \,.$$

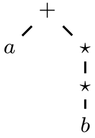
Example: BST-like distribution

Consider the regular expressions $(+, \bullet, \star)$ on two letters a, b

(i)



(ii)



- The expression tree (i) is drawn with probability

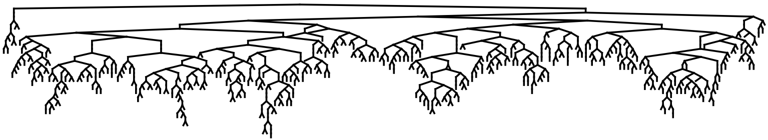
$$p_\star p_+ \frac{1}{2} p_a p_b \,.$$

- The expression tree (ii) is drawn with probability

$$p_+ \frac{1}{3} p_a p_\star p_b \,.$$

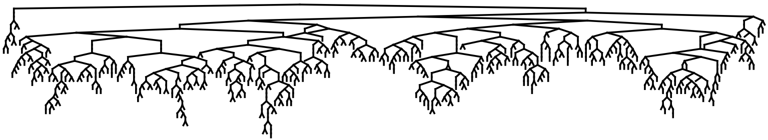
$\Longrightarrow$ Distribution not uniform for any choice of parameters.

BST-like trees: distribution over unary-binary trees



BST-like tree distribution is not uniform[a].

BST-like trees: distribution over unary-binary trees



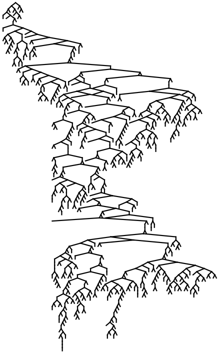
BST-like tree distribution is not uniform[a].

- Binary nodes ≈ balanced $\frac{n}{2}$–$\frac{n}{2}$,
  but for uniform trees

$$\mathbb{E}_n[\min(|T_L|, |T_R|)] \sim c_0\sqrt{n}\,.$$

- Expected height of different order

$$\Theta(\log n) \text{ vs } \Theta(\sqrt{n})\,.$$



[a]Tree T chosen uniformly from $\{T : |T| = n\}$.

Uniform and BST-like distributions

The uniform distribution:

- naturally maximizes entropy.
- can be sampled efficiently with some effort (Recursive method, Boltzmann samplers, Devroye's constrained GW).
- is amenable to theoretical study (Analytic Combinatorics).

Uniform and BST-like distributions

The uniform distribution:

- naturally maximizes entropy.
- can be sampled efficiently with some effort
  (Recursive method, Boltzmann samplers, Devroye's constrained GW).
- is amenable to theoretical study (Analytic Combinatorics).

The BST-like distribution:

- must be parametrized (prob. of operators).
- is easy to implement and very efficient.
- is often used in the automated checking of tools.

Uniform and BST-like distributions

The uniform distribution:

- naturally maximizes entropy.
- can be sampled efficiently with some effort
  (Recursive method, Boltzmann samplers, Devroye's constrained GW).
- is amenable to theoretical study (Analytic Combinatorics).

The BST-like distribution:

- must be parametrized (prob. of operators).
- is easy to implement and very efficient.
- is often used in the automated checking of tools.

We had previously studied *semantically* **uniform expressions**...

Section

Semantic simplification

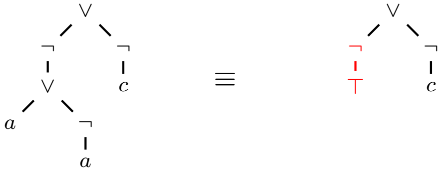
Given tree may be redundant

Semantic simplification

Given tree may be redundant

Semantic simplification

Given tree may be redundant

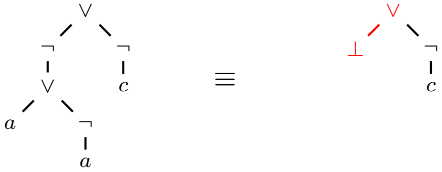


Semantic simplification

Given tree may be redundant

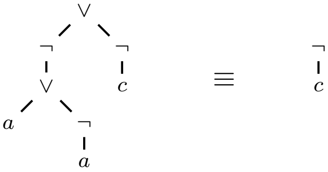
Semantic simplification

Given tree may be redundant

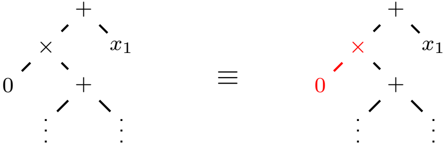


Semantic simplification

Given tree may be redundant

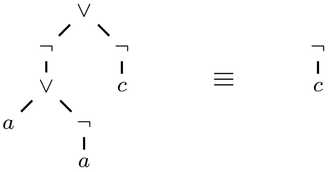


Or even *more:*

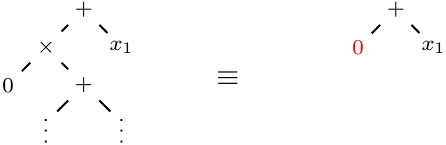


Semantic simplification

Given tree may be redundant

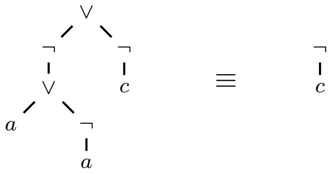


Or even *more:*

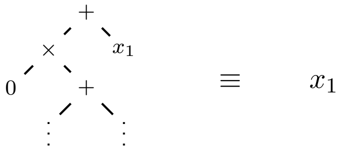


Semantic simplification

Given tree may be redundant



Or even *more:*



Question

Do semantic reductions affect the size of the random expressions?

Semantic simplification

Universal result for uniform tree model:

Theorem (Koechlin,Nicaud,R,'20)

Expected size of reduction of uniform tree bounded, as size $\rightarrow \infty$.

Semantic simplification

Universal result for uniform tree model:

Theorem (Koechlin,Nicaud,R,'20)

Expected size of reduction of uniform tree bounded, as size $\to \infty$.

- Idea based on absorbing pattern $\mathcal{P}$, e.g., $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$,

$$\overset{\circledast}{\mathcal{P} \quad T} \rightsquigarrow \mathcal{P} \qquad \overset{\circledast}{T \quad \mathcal{P}} \rightsquigarrow \mathcal{P}$$

Semantic simplification

Universal result for uniform tree model:

Theorem (Koechlin,Nicaud,R,'20)

Expected size of reduction of uniform tree bounded, as size $\to \infty$.

- Idea based on absorbing pattern $\mathcal{P}$, e.g., $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$,

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

- Works also for systems and different arities,

Semantic simplification

Universal result for uniform tree model:

Theorem (Koechlin,Nicaud,R,'20)

Expected size of reduction of uniform tree bounded, as size $\to \infty$.

- Idea based on absorbing pattern $\mathcal{P}$, e.g., $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$,

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

- Works also for systems and different arities,

$$\texttt{Example} \ : \begin{cases} \mathcal{L}_R = \begin{array}{c}\neg\\ |\\ \mathcal{S}\end{array} + \mathcal{S}, \\ \mathcal{S} = a + b + \begin{array}{c}\vee\\ / \ \backslash\\ \mathcal{L}_R\quad \mathcal{L}_R\end{array} + \begin{array}{c}\wedge\\ / \ \backslash\\ \mathcal{L}_R\quad \mathcal{L}_R\end{array}. \end{cases}$$

Semantic simplification

Universal result for uniform tree model:

Theorem (Koechlin,Nicaud,R,'20)

Expected size of reduction of uniform tree bounded, as size $\to \infty$.

- Idea based on absorbing pattern $\mathcal{P}$, e.g., $\texttt{false} \wedge (\ldots) \equiv \texttt{false}$,

$$\overset{\circledast}{\mathcal{P}\ \ T} \rightsquigarrow \mathcal{P} \qquad \overset{\circledast}{T\ \ \mathcal{P}} \rightsquigarrow \mathcal{P}$$

- Works also for systems and different arities,

$$\texttt{Example}\ :\begin{cases}\mathcal{L}_R = \underset{\mathcal{S}}{\overset{\neg}{|}} + \mathcal{S},\\ \mathcal{S} = a + b + \overset{\vee}{\mathcal{L}_R\ \mathcal{L}_R} + \overset{\wedge}{\mathcal{L}_R\ \mathcal{L}_R}.\end{cases}$$

- For regular expressions on two letters, $\texttt{constant bound} \approx 77.8$.

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

To define an absorbing pattern we fix:

- an "operator" $\circledast \in \mathcal{A}_{ops}$ with arity 2,
- an expression tree $\mathcal{P}$ in the family.

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

To define an absorbing pattern we fix:

- an "operator" $\circledast \in \mathcal{A}_{ops}$ with arity 2,
- an expression tree $\mathcal{P}$ in the family.

Simplify by applying bottom-up the rule:

$$\begin{array}{c} \circledast \\ / \quad \backslash \\ C_1 \qquad C_2 \end{array} \quad \rightsquigarrow \mathcal{P}\,, \text{ whenever } C_i = \mathcal{P} \text{ for some } i \in \{1,2\}.$$

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

To define an absorbing pattern we fix:

- an "operator" $\circledast \in \mathcal{A}_{ops}$ with arity 2,
- an expression tree $\mathcal{P}$ in the family.

Simplify by applying bottom-up the rule:

$$\begin{array}{ccc} & \circledast & \\ / & & \backslash \\ C_1 & & C_2 \end{array} \rightsquigarrow \mathcal{P}\,, \text{ whenever } C_i = \mathcal{P} \text{ for some } i \in \{1, 2\}.$$

- Wide variety of examples:

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

operator $\vee$ — operator $+$ — operator $\times$

- Weak hypothesis: in practice often several coexisting patterns !

Absorbing patters: simplifying the trees

Definition (Simplification, absorbing pattern)

To define an absorbing pattern we fix:

- an "operator" $\circledast \in \mathcal{A}_{ops}$ with arity 2,
- an expression tree $\mathcal{P}$ in the family.

Simplify by applying bottom-up the rule:

$$\begin{array}{ccc} & \circledast & \\ / & & \backslash \\ C_1 & & C_2 \end{array} \rightsquigarrow \mathcal{P}\,, \text{ whenever } C_i = \mathcal{P} \text{ for some } i \in \{1, 2\}.$$

- Wide variety of examples:

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

operator $\vee$ — operator $+$ — operator $\times$

- Weak hypothesis: in practice often several coexisting patterns !

Absorbing patters: simplifying the trees

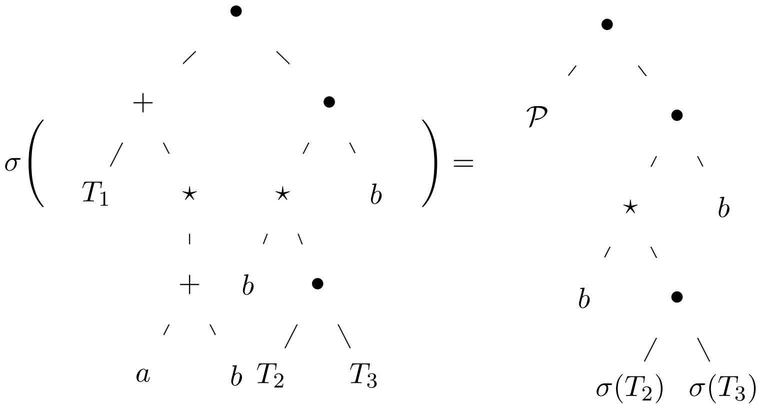
Denote by $\sigma(T) = \sigma(T, \mathcal{P}, \circledast)$ the simplification of T.

Absorbing patters: simplifying the trees

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

Example: regular expressions $(+, \bullet, \star)$ on two letters a, b:

$\mathcal{P} = (a+b)^\star$ absorbing for union $\circledast = +$



If we draw a random BST-like tree expression of size n:

- do we have the same flaw as uniform trees?

[2] Left to right $(p_\star, p_\bullet, p_+) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$,$(p_\star, p_\bullet, p_+) = (\frac{5}{29}, \frac{5}{29}, \frac{19}{29})$,$(p_\star, p_\bullet, p_+) = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$

If we draw a random BST-like tree expression of size n:

- do we have the same flaw as uniform trees?

  $\Longrightarrow$ we characterize 5 regimes

[2] Left to right $(p_\star, p_\bullet, p_+) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$,$(p_\star, p_\bullet, p_+) = (\frac{5}{29}, \frac{5}{29}, \frac{19}{29})$,$(p_\star, p_\bullet, p_+) = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$

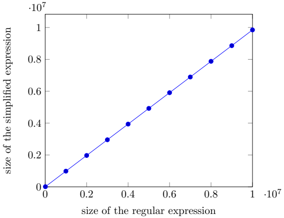
If we draw a random BST-like tree expression of size n:

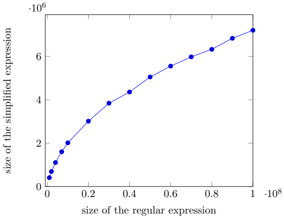
- do we have the same flaw as uniform trees?

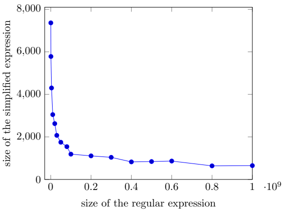
$\Longrightarrow$ we characterize 5 regimes

Experimental expected size (10 000 samples)[2] on regular expressions $(+, \bullet, \star)$ on two letters a, b:

$\mathcal{P} = (a+b)^\star$ absorbing for union $\circledast = +$







[2] Left to right $(p_\star, p_\bullet, p_+) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$,$(p_\star, p_\bullet, p_+) = (\frac{5}{29}, \frac{5}{29}, \frac{19}{29})$,$(p_\star, p_\bullet, p_+) = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$

Section

1. Random BST-like tree model

2. Semantic simplifications

3. Result for BST-like trees and elements of the proof

4. Conclusions

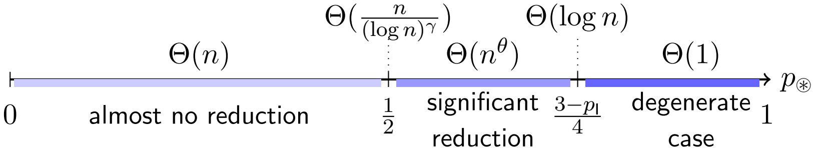
Theorem. *Consider a family of expression trees defined from unary and binary operators with an absorbing pattern* $\mathcal{P}$ *for an operator* $\circledast$ *of arity* 2*.*

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

Theorem. *Consider a family of expression trees defined from unary and binary operators with an absorbing pattern $\mathcal{P}$ for an operator $\circledast$ of arity 2.*

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

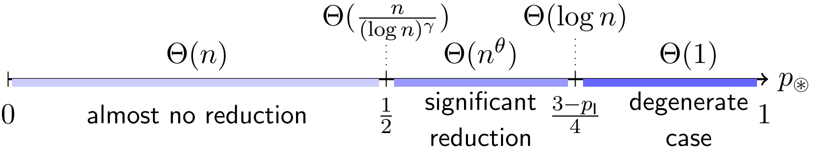
Then the expected size of the simplification of a random BST-like tree has an asymptotic behaviour given by the following cases, depending on the probability $p_\circledast$ of the absorbing operator:



Theorem. *Consider a family of expression trees defined from unary and binary operators with an absorbing pattern $\mathcal{P}$ for an operator $\circledast$ of arity 2.*

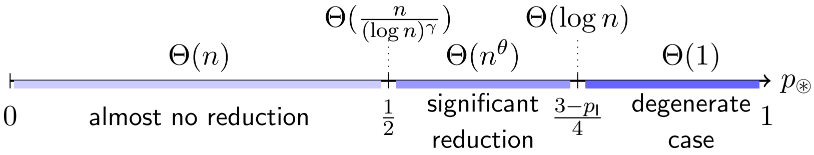
Take the simplification consisting in inductively changing a $\circledast$-node by $\mathcal{P}$ whenever one of its children simplifies to $\mathcal{P}$.

Then the expected size of the simplification of a random BST-like tree has an asymptotic behaviour given by the following cases, depending on the probability $p_\circledast$ of the absorbing operator:

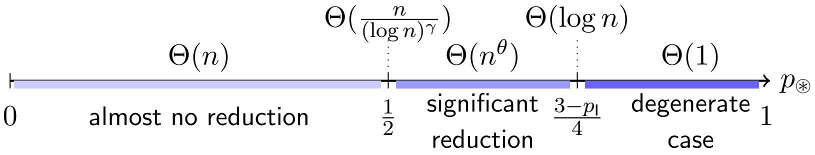


- Probability $p_\circledast$ of $\circledast$, and p_{l} of picking unary operator.
- Two critical points $p_\circledast = 1/2$ and $p_\circledast = (3 - p_{\mathsf{l}})/4$
- Regimes from no reduction $\Theta(n)$ to complete reduction $\Theta(1)$

The main regimes experimentally



The main regimes experimentally



Experimental plots (10 000 samples)for regular expressions on two letters a, b: $\mathcal{P} = (a+b)^\star$ absorbing for union $\circledast = +$

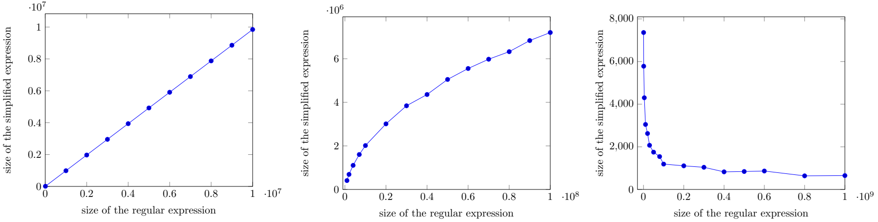


Figure: Left to right: linear ($p_+ = p_\star = p_\cdot = \frac{1}{3}$), sublinear ($p_+ = \frac{19}{29}$, $p_\star = p_\cdot = \frac{5}{29}$) and constant ($p_+ = \frac{8}{10}$, $p_\star = p_\cdot = \frac{1}{10}$).

Scheme of the proof: steps from Analytic combinatorics

We employ Analytic Combinatorics to study the expectation,

- Ordinary generating function

$$E(z) := \sum_{n=0}^{\infty} e_n z^n \,,$$

encodes sequence $e_n := \mathbb{E}_n[\sigma(T)]$.

Scheme of the proof: steps from Analytic combinatorics

We employ Analytic Combinatorics to study the expectation,

- Ordinary generating function

$$E(z) := \sum_{n=0}^{\infty} e_n z^n \,,$$

encodes sequence $e_n := \mathbb{E}_n[\sigma(T)]$.

- **Symbolic Step.** We find a formal equation describing $E(z)$.

  Here this will be an *ordinary differential equation*

$$E'(z) = B(z) + C(z) \cdot E(z) \,.$$

Scheme of the proof: steps from Analytic combinatorics

We employ Analytic Combinatorics to study the expectation,

- Ordinary generating function

$$E(z) := \sum_{n=0}^{\infty} e_n z^n \,,$$

encodes sequence $e_n := \mathbb{E}_n[\sigma(T)]$.

- **Symbolic Step.** We find a formal equation describing $E(z)$.

  Here this will be an *ordinary differential equation*

$$E'(z) = B(z) + C(z) \cdot E(z) \,.$$

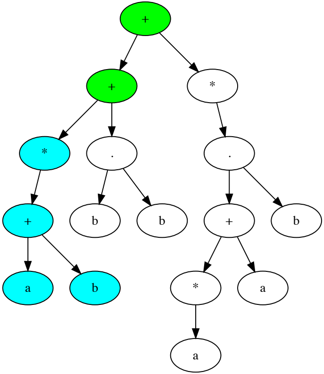
- **Analytic Step.** We look at $E(z)$ over the complex $z \in \mathbb{C}$.

  A *Transfer Theorem* links the behaviour of $E(z)$ at its dominant singularity to asymptotics of $e_n \Rightarrow$ Study singularities

$$E(z) \sim_{z \to 1} \lambda(1-z)^{-\alpha} \Longrightarrow e_n \sim \lambda n^{\alpha-1}/\Gamma(\alpha)$$

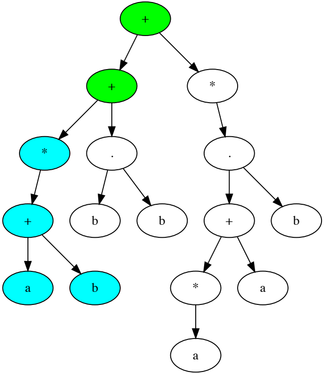
Symbolic step: fully reducible trees

An expression tree T is *fully reducible* when $\sigma(T) = \mathcal{P}$.



Symbolic step: fully reducible trees

An expression tree T is *fully reducible* when $\sigma(T) = \mathcal{P}$.

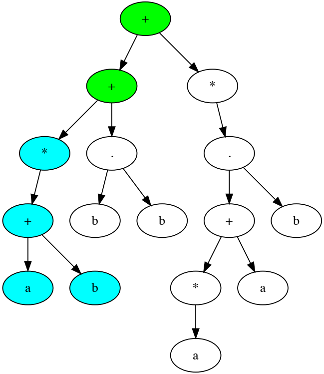


Fully reducible expressions

- dictate the reduction process:
  leaves of the reduced expression.

Symbolic step: fully reducible trees

An expression tree T is *fully reducible* when $\sigma(T) = \mathcal{P}$.



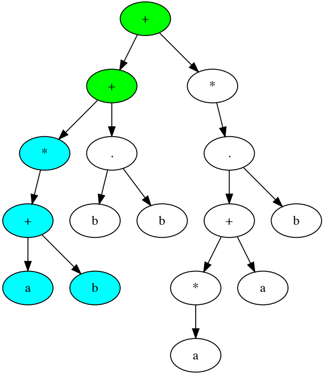
Fully reducible expressions

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

$$\mathcal{P} = \underset{a \qquad b}{\overset{\star}{+}} \;; \quad \mathcal{R} = \mathcal{P} + \underset{\mathcal{R} \quad \mathcal{L}}{\overset{+}{/\ \backslash}} + \underset{\mathcal{L} \quad \mathcal{R}}{\overset{+}{/\ \backslash}} \,.$$

Symbolic step: fully reducible trees

An expression tree T is *fully reducible* when $\sigma(T) = \mathcal{P}$.



Fully reducible expressions

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

$$\mathcal{P} = \begin{array}{c} \star \\ \vdots \\ + \\ a \quad\;\; b \end{array} \; ; \quad \mathcal{R} = \mathcal{P} + \begin{array}{c} + \\ / \; \backslash \\ \mathcal{R} \quad \mathcal{L} \end{array} + \begin{array}{c} + \\ / \; \backslash \\ \mathcal{L} \quad \mathcal{R} \end{array} .$$

We consider a fundamental sequence

$$\gamma_n := \Pr_n \{\sigma(T) = \mathcal{P}\} \, , \qquad A(z) := \sum \gamma_n z^n \, ,$$

of probabilities of *full* reduction and their generating function.

Symbolic step: recurrence for the expected value

We consider a fundamental sequence

$$\gamma_n := \Pr_n \{\sigma(T) = \mathcal{P}\} \ ,$$

of probabilities of *full* reduction.

Symbolic step: recurrence for the expected value

We consider a fundamental sequence

$$\gamma_n := \Pr_n \{\sigma(T) = \mathcal{P}\} ,$$

of probabilities of *full* reduction.

Recurrence for expected values

The recurrence for e_n involves γ_n,

$$e_{n+1} = 1 + (s-1)\gamma_{n+1}\mathbf{1}_{n+1 \neq s} + p_{\mathsf{I}} e_n + \frac{2p_{\mathsf{II}}}{n-1}\sum_{j=1}^{n-1} e_j + \frac{2p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(e_j - s\gamma_j)(1-\gamma_{n-j}) ,$$

here $p_{\mathsf{II}} := 1 - p_{\mathsf{I}} - p_{\circledast}$ and $s = |\mathcal{P}|$.

Symbolic step: recurrence = differential equation

Recurrence yields first order differential equation

$$E'(z) = F(z, A(z)) + \tfrac{1}{1-p_{\mathsf{I}} z}\Big(\tfrac{2}{z} - p_{\mathsf{I}} + 2\,(1-p_{\mathsf{I}})\,\frac{z}{1-z} - 2p_{\circledast} A(z)\Big)\cdot E(z)\,,$$

in terms of $A(z) = \sum_n \gamma_n z^n$.

Symbolic step: recurrence = differential equation

Recurrence yields first order differential equation

$$E'(z) = F(z, A(z)) + \frac{1}{1-p_{\mathsf{I}} z}\left(\frac{2}{z} - p_{\mathsf{I}} + 2\,(1-p_{\mathsf{I}})\,\frac{z}{1-z} - 2p_{\circledast} A(z)\right) \cdot E(z)\,,$$

in terms of $A(z) = \sum_n \gamma_n z^n$.

Proof.

Differentiating we have $E'(z) = \sum (n+1) e_n z^n$. $\square$

Symbolic step: recurrence = differential equation

Recurrence yields first order differential equation

$$E'(z) = F(z, A(z)) + \tfrac{1}{1-p_{\mathsf{l}} z}\left(\tfrac{2}{z} - p_{\mathsf{l}} + 2\,(1-p_{\mathsf{l}})\,\frac{z}{1-z} - 2p_{\circledast} A(z)\right) \cdot E(z)\,,$$

in terms of $A(z) = \sum_n \gamma_n z^n$.

Proof.

Differentiating we have $E'(z) = \sum (n+1) e_n z^n$. □

First order differential equations can be solved explicitly

Proposition

The equation $U'(z) = f(z) + g(z)U(z)$ where f, g are analytic functions on Ω has a unique solution analytic on Ω, satisfying $U(0) = u_0$,

$$U(z) = \exp\left(\int_0^z g(\zeta) d\zeta\right)\left(u_0 + \int_0^z f(\zeta)\exp\left(-\int_0^\zeta g(w)dw\right) d\zeta\right).$$

Symbolic step: recurrence = differential equation

Recurrence yields first order differential equation

$$E'(z) = F(z, A(z)) + \frac{1}{1-p_{\mathsf{l}} z}\left(\frac{2}{z} - p_{\mathsf{l}} + 2\,(1-p_{\mathsf{l}})\,\frac{z}{1-z} - 2p_{\circledast} A(z)\right) \cdot E(z)\,,$$

in terms of $A(z) = \sum_n \gamma_n z^n$.

Proof.

Differentiating we have $E'(z) = \sum (n+1) e_n z^n$. □

First order differential equations can be solved explicitly

Proposition

The equation $U'(z) = f(z) + g(z)U(z)$ where f, g are analytic functions on Ω has a unique solution analytic on Ω, satisfying $U(0) = u_0$,

$$U(z) = \exp\left(\int_0^z g(\zeta) d\zeta\right)\left(u_0 + \int_0^z f(\zeta) \exp\left(-\int_0^\zeta g(w) dw\right) d\zeta\right).$$

Our coefficients depend on z and the unkown generating function $A(z)$.

Analytic step: the singularity $z = 1$

The generating functions $A(z)$ and $E(z)$

- are analytic for $|z| < 1$, as the series converge absolutely.

Analytic step: the singularity $z = 1$

The generating functions $A(z)$ and $E(z)$

- are analytic for $|z| < 1$, as the series converge absolutely.
- have $z = 1$ as a dominant singularity: $\rho = 1$ radius of convergence.

Analytic step: the singularity $z = 1$

The generating functions $A(z)$ and $E(z)$

- are analytic for $|z| < 1$, as the series converge absolutely.
- have $z = 1$ as a dominant singularity: $\rho = 1$ radius of convergence.

Solution of ODE gives asymptotics

$$E(z) \sim \frac{c}{(1-z)^2} \left(2 + \int_0^z F(w, A(w)) I(w) dw \right) (I(z))^{-1}, \quad z \to 1,$$

where $I(z) := \exp\left(2p_{\circledast} \int_0^z \frac{A(w)}{1-p_{|}w} dw \right)$.

Analytic step: the singularity $z = 1$

The generating functions $A(z)$ and $E(z)$

- are analytic for $|z| < 1$, as the series converge absolutely.
- have $z = 1$ as a dominant singularity: $\rho = 1$ radius of convergence.

Solution of ODE gives asymptotics

$$E(z) \sim \frac{c}{(1-z)^2} \left(2 + \int_0^z F(w, A(w)) I(w) dw \right) (I(z))^{-1}, \quad z \to 1,$$

where $I(z) := \exp\left(2p_{\circledast} \int_0^z \frac{A(w)}{1-p_{\mathsf{I}} w} dw \right)$.

To apply the Transfer Theorem and complete the proof:

- we require precise asymptotics for $A(z)$ at $z = 1$,
- we show that $A(z)$ and $E(z)$ are analytic over $\Omega = \mathbb{C} \setminus [1, \infty)$.

Fully reducible trees: probabilities

We study the generating function $A(z) = \sum \gamma_n z^n$

Proposition: recurrence for γ_n

The probabilities $\gamma_n = \Pr_n \{\sigma(T) = \mathcal{P}\}$ satisfy, for $n \geq |\mathcal{P}|$,

$$\gamma_{n+1} = \frac{p_{\circledast}}{n-1} \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k \gamma_{n-k}) \, .$$

Fully reducible trees: probabilities

We study the generating function $A(z) = \sum \gamma_n z^n$

Proposition: recurrence for γ_n

The probabilities $\gamma_n = \mathrm{Pr}_n \{\sigma(T) = \mathcal{P}\}$ satisfy, for $n \geq |\mathcal{P}|$,

$$\gamma_{n+1} = \frac{p_{\circledast}}{n-1} \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k \gamma_{n-k}) \, .$$

Remark. If $\gamma_n \to \gamma_\infty$, by Cesàro means $\gamma_\infty = p_{\circledast} \cdot (2\gamma_\infty - \gamma_\infty^2)$.

Fully reducible trees: probabilities

We study the generating function $A(z) = \sum \gamma_n z^n$

Proposition: recurrence for γ_n

The probabilities $\gamma_n = \Pr_n\{\sigma(T) = \mathcal{P}\}$ satisfy, for $n \geq |\mathcal{P}|$,

$$\gamma_{n+1} = \frac{p_{\circledast}}{n-1} \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k \gamma_{n-k}) \,.$$

Remark. If $\gamma_n \to \gamma_\infty$, by Cesàro means $\gamma_\infty = p_{\circledast} \cdot (2\gamma_\infty - \gamma_\infty^2)$.

Recurrence translates into Riccati differential equation

$$A'(z) = (s-2)\gamma_s z^{s-1} + \left(\frac{2}{z} + 2p_{\circledast} \frac{z}{1-z} \right) A(z) - p_{\circledast} \cdot (A(z))^2 \,,$$

where $s = |\mathcal{P}|$ is the size of the absorbing pattern.

Behaviour of solutions of Riccati ODE

Considering $v(z)$ such that $p_{\circledast} A(z) = v'(z)/v(z)$,

Riccati equation becomes linear

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast} \frac{z}{1-z} \right) v'(z) \,.$$

Behaviour of solutions of Riccati ODE

Considering $v(z)$ such that $p_{\circledast} A(z) = v'(z)/v(z)$,

Riccati equation becomes linear

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast} \frac{z}{1-z} \right) v'(z)\,.$$

For linear ODEs:

- domain of analyticity well-understood.
- Frobenius method characterizes behaviour at singularities.

Behaviour of solutions of Riccati ODE

Considering $v(z)$ such that $p_\circledast A(z) = v'(z)/v(z)$,

Riccati equation becomes linear

$$v''(z) = p_\circledast \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_\circledast \frac{z}{1-z} \right) v'(z)\,.$$

For linear ODEs:

- domain of analyticity well-understood.
- Frobenius method characterizes behaviour at singularities.

Proposition

The generating function $A(z)$ satisfies, $z \to 1$

- For $p_\circledast > \frac{1}{2}$, $A(z) = \frac{\gamma_\infty}{1-z} + O((1-z)^{2p_\circledast - 2})$,
- For $p_\circledast = \frac{1}{2}$, $A(z) = \frac{2}{1-z} \left(\log\left(\frac{1}{1-z}\right)\right)^{-1} \left(1 + O\left(\log\left(\frac{1}{1-z}\right)^{-1}\right)\right)$
- For $p_\circledast < \frac{1}{2}$, $A(z) \sim \frac{D}{(1-z)^{2p_\circledast}}$,

where $\gamma_\infty := (2p_\circledast - 1)/p_\circledast$ and $D > 0$ is a constant.

Probability of full reduction

Theorem

The probability γ_n of being fully reducible tends to the constant $\gamma_\infty := (2p_\circledast - 1)/p_\circledast$ for $p_\circledast > \frac{1}{2}$ and to zero otherwise.

Moreover,

- *for $p_\circledast = \frac{1}{2}$ we have $\gamma_n \sim \frac{2}{\log n}$,*
- *for $p_\circledast < \frac{1}{2}$, $\gamma_n \sim D \cdot n^{2p_\circledast - 1}$.*

Probability of full reduction

Theorem

The probability γ_n of being fully reducible tends to the constant $\gamma_\infty := (2p_\circledast - 1)/p_\circledast$ for $p_\circledast > \frac{1}{2}$ and to zero otherwise.

Moreover,

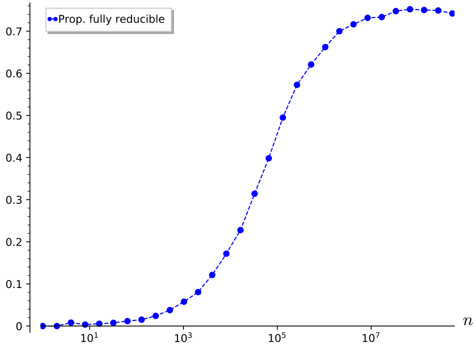
- *for $p_\circledast = \frac{1}{2}$ we have $\gamma_n \sim \frac{2}{\log n}$,*
- *for $p_\circledast < \frac{1}{2}$, $\gamma_n \sim D \cdot n^{2p_\circledast - 1}$.*

Experimental plot:
regular expressions on two letters with

$$(p_+, p_\bullet, p_\star) = \left(\frac{8}{10}, \frac{1}{10}, \frac{1}{10}\right).$$

Then

$$\lim_{n\to\infty} \gamma_n = 3/4\,.$$



Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast} \frac{z}{1-z} \right) v'(z)\,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast} \frac{z}{1-z} \right) v'(z) \,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Frobenius Method [Fuchs' Theorem]

Let $v''(z) = \frac{q(z)}{(1-z)^2} v(z) + \frac{p(z)}{1-z} v'(z)$ with $q(z)$ and $p(z)$ analytic at $z = 1$.

Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z)\,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Frobenius Method [Fuchs' Theorem]

Let $v''(z) = \frac{q(z)}{(1-z)^2} v(z) + \frac{p(z)}{1-z} v'(z)$ with $q(z)$ and $p(z)$ analytic at $z = 1$.

If α is a root of the *indicial equation*

$$X^2 = (p(1) + 1)X + q(1)\,,$$

there is $h_\alpha(z)$ analytic at 1, $h_\alpha(1) = 1$, such that $(1-z)^\alpha h_\alpha(z)$ is a solution of the ODE.

Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z)\,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Frobenius Method [Fuchs' Theorem]

Let $v''(z) = \frac{q(z)}{(1-z)^2} v(z) + \frac{p(z)}{1-z} v'(z)$ with $q(z)$ and $p(z)$ analytic at $z = 1$.

If α is a root of the *indicial equation*

$$X^2 = (p(1)+1)X + q(1)\,,$$

there is $h_\alpha(z)$ analytic at 1, $h_\alpha(1) = 1$, such that $(1-z)^\alpha h_\alpha(z)$ is a solution of the ODE.

Let α_1 and α_2 be the roots of the indicial equation.

Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z)\,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Frobenius Method [Fuchs' Theorem]

Let $v''(z) = \frac{q(z)}{(1-z)^2} v(z) + \frac{p(z)}{1-z} v'(z)$ with $q(z)$ and $p(z)$ analytic at $z = 1$.

If α is a root of the *indicial equation*

$$X^2 = (p(1)+1)X + q(1)\,,$$

there is $h_\alpha(z)$ analytic at 1, $h_\alpha(1) = 1$, such that $(1-z)^\alpha h_\alpha(z)$ is a solution of the ODE.

Let α_1 and α_2 be the roots of the indicial equation.

- If $\alpha_1 - \alpha_2 \notin \mathbb{Z}$, solutions are linearly independent.

Proof principles: analytic step – Frobenius method

Recall. From the Riccati equation, we have a 2nd order linear ODE

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z)\,,$$

and we want the behavior around $z = 1$ (false pole at $z = 0$).

Frobenius Method [Fuchs' Theorem]

Let $v''(z) = \frac{q(z)}{(1-z)^2} v(z) + \frac{p(z)}{1-z} v'(z)$ with $q(z)$ and $p(z)$ analytic at $z = 1$.

If α is a root of the *indicial equation*

$$X^2 = (p(1)+1)X + q(1)\,,$$

there is $h_\alpha(z)$ analytic at 1, $h_\alpha(1) = 1$, such that $(1-z)^\alpha h_\alpha(z)$ is a solution of the ODE.

Let α_1 and α_2 be the roots of the indicial equation.

- If $\alpha_1 - \alpha_2 \notin \mathbb{Z}$, solutions are linearly independent.
- If $\alpha_1 - \alpha_2 \in \mathbb{Z}$, factor $(1-z)^{|\alpha_1 - \alpha_2|}$ is polynomial.
  $\implies$ to obtain independent solution multiply $\times \log(1-z)$.

Section

Conclusions and further work

Conclusions

- BST-like expression trees present a richer range of behaviours than the uniform ones.

Conclusions and further work

Conclusions

- BST-like expression trees present a richer range of behaviours than the uniform ones.
- Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Conclusions and further work

Conclusions

- BST-like expression trees present a richer range of behaviours than the uniform ones.
- Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Conclusions and further work

Conclusions

- ⊛ BST-like expression trees present a richer range of behaviours than the uniform ones.
- ⊛ Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Questions and further work

1. Absorbing operator $\circledast$ of arity $a \geq 3$?

Conclusions and further work

Conclusions

- $\circledast$ BST-like expression trees present a richer range of behaviours than the uniform ones.
- $\circledast$ Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Questions and further work

1. Absorbing operator $\circledast$ of arity $a \geq 3$?
   $\Rightarrow$ expect similar results, threshold $\frac{1}{a}$ instead of $\frac{1}{2}$.

Conclusions and further work

Conclusions

- $\circledast$ BST-like expression trees present a richer range of behaviours than the uniform ones.
- $\circledast$ Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Questions and further work

1. Absorbing operator $\circledast$ of arity $a \geq 3$?
   $\Rightarrow$ expect similar results, threshold $\frac{1}{a}$ instead of $\frac{1}{2}$.
2. Absorbing pattern model is general
   $\Rightarrow$ consider interactions between operators?

Conclusions and further work

Conclusions

- ⊛ BST-like expression trees present a richer range of behaviours than the uniform ones.
- ⊛ Not exempt of degenerate cases however
  $\Rightarrow$ tuning probabilities might be important.

Questions and further work

1. Absorbing operator $\circledast$ of arity $a \geq 3$?
   $\Rightarrow$ expect similar results, threshold $\frac{1}{a}$ instead of $\frac{1}{2}$.
2. Absorbing pattern model is general
   $\Rightarrow$ consider interactions between operators?
3. Take a concrete case: LTL formulas.

References

Philippe Flajolet and Robert Sedgewick.
Analytic Combinatorics.
Cambridge University Press, Cambridge (2009).
https://algo.inria.fr/flajolet/Publications/AnaCombi/anacombi.html

Florent Koechlin, Cyril Nicaud, and Pablo Rotondo.
On the Degeneracy of Random Expressions Specified by Systems of Combinatorial Equations.
Proceedings of DLT 2020, LNCS vol. 12086, pp 164–177.
https://doi.org/10.1007/978-3-030-48516-0_13

Florent Koechlin and Pablo Rotondo.
Analysis of an efficient reduction algorithm for random regular expressions based on universality detection.
Proceedings of CSR 2021. LNCS, vol 12730. Springer.
https://doi.org/10.1007/978-3-030-79416-3_12

Florent Koechlin and Pablo Rotondo.
Absorbing patterns in BST-like expression-trees.
Proceedings of STACS 2021, LIPICs, vol.187, 48:1–48:15
https://doi.org/10.4230/LIPIcs.STACS.2021.48