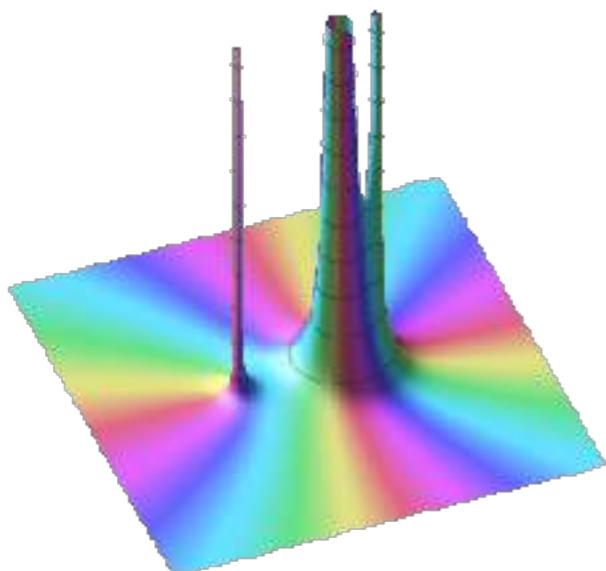
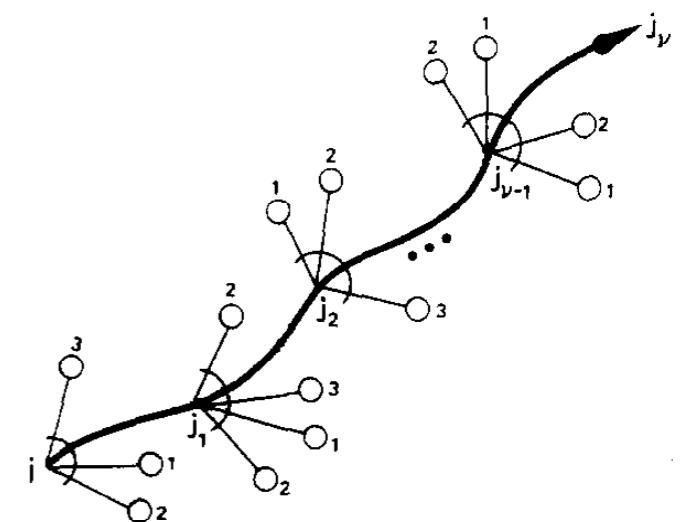


# On (Effective) Analytic Combinatorics

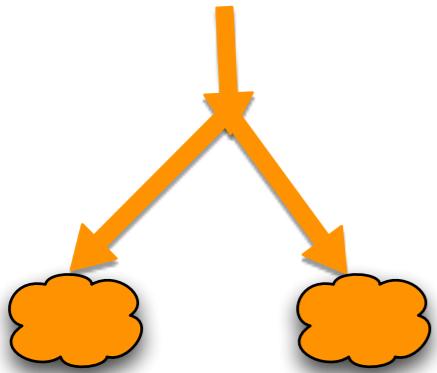


*Bruno Salvy*  
Inria & ENS de Lyon

Séminaire Algorithmes  
Dec. 2017



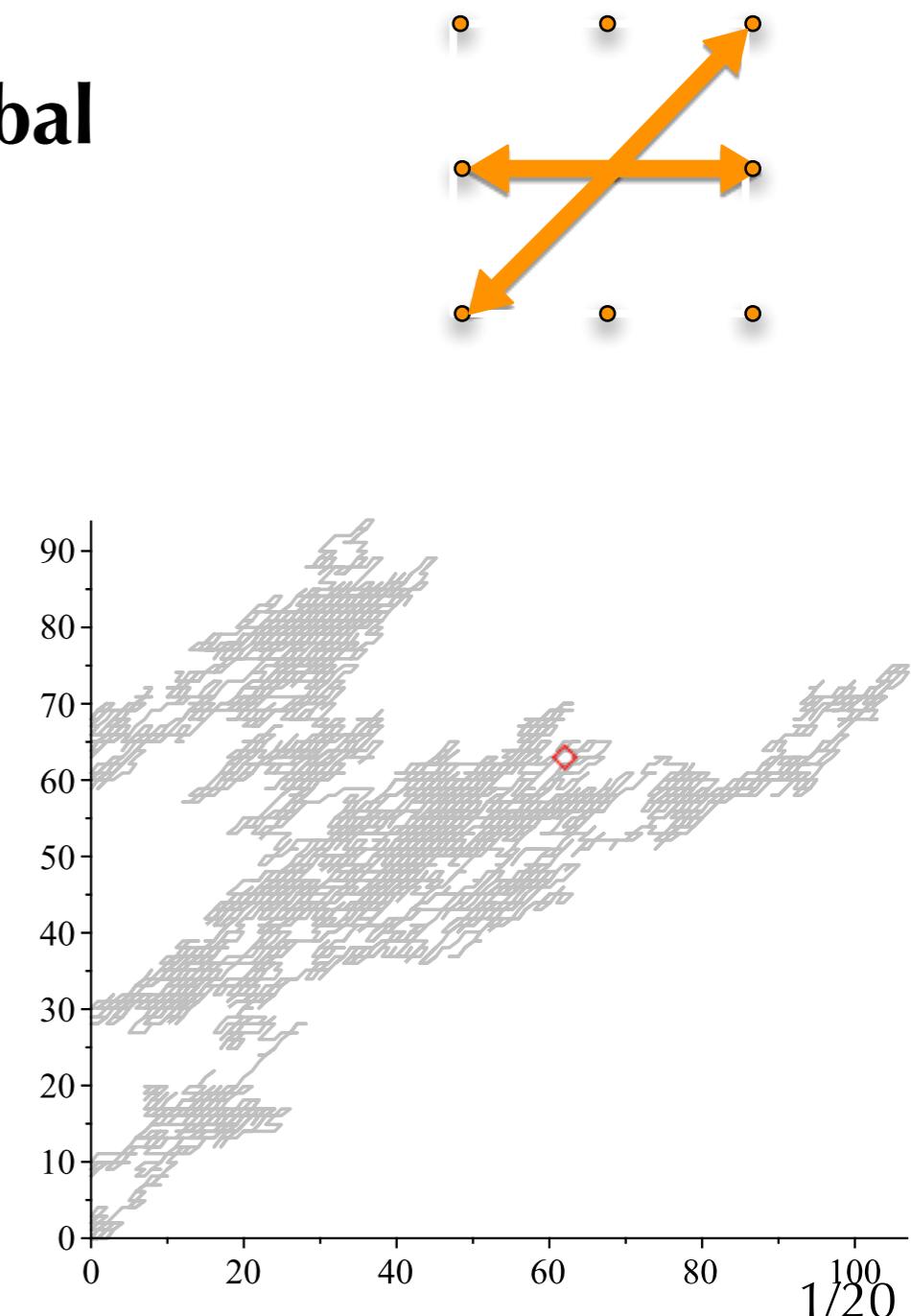
# Combinatorics, Randomness and Analysis



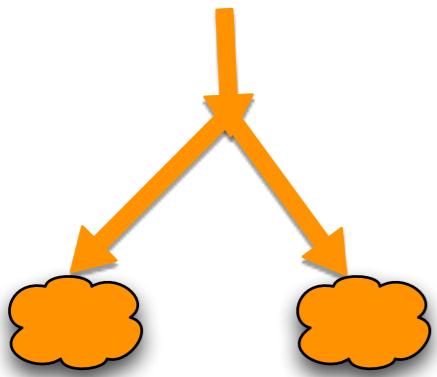
From **simple local** rules, a **global** structure arises.

A quest for **universality** in random discrete structures:  
→ **probabilistic complexity** of structures and algorithms.

Quantitative results using **complex analysis**.



# Combinatorics, Randomness and Analysis

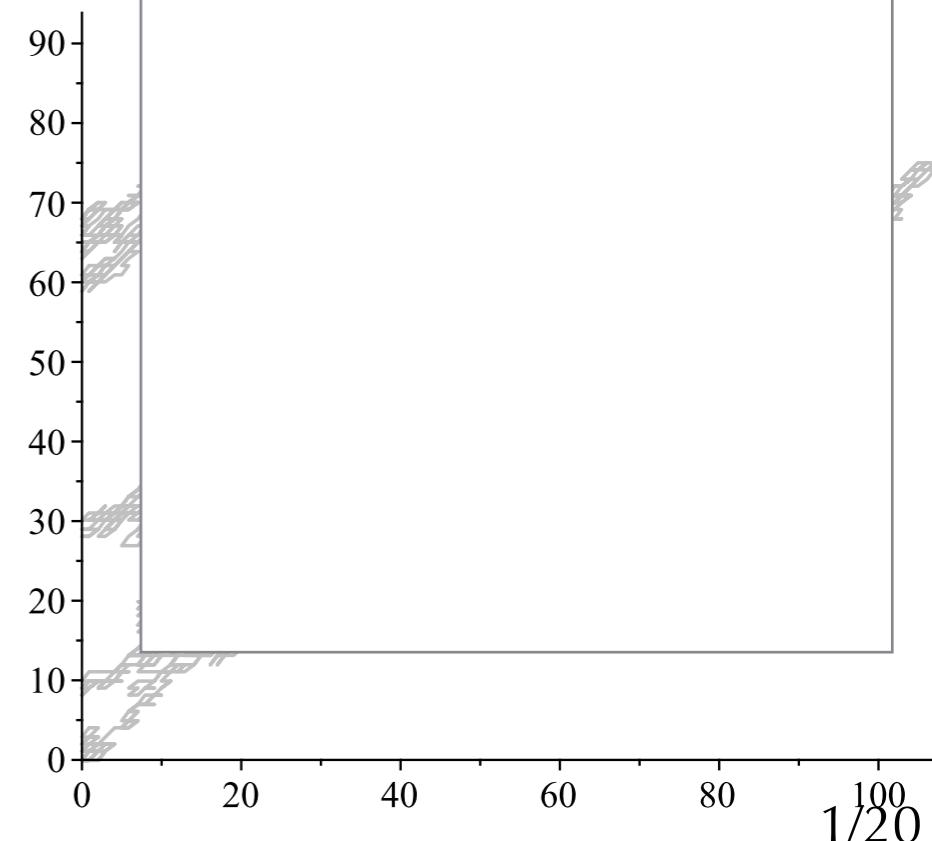


From **simple local** rules, a **global** structure arises.

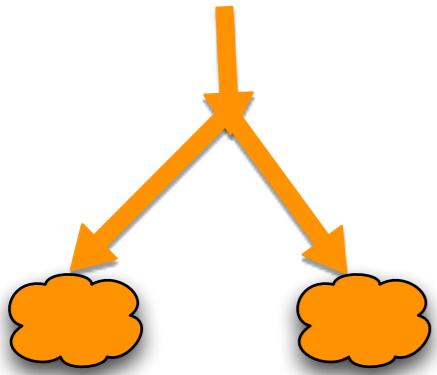
A quest for **universality** in random discrete structures:  
→ **probabilistic complexity** of structures and algorithms.

Quantitative results using **complex analysis**.

Typical questions



# Combinatorics, Randomness and Analysis



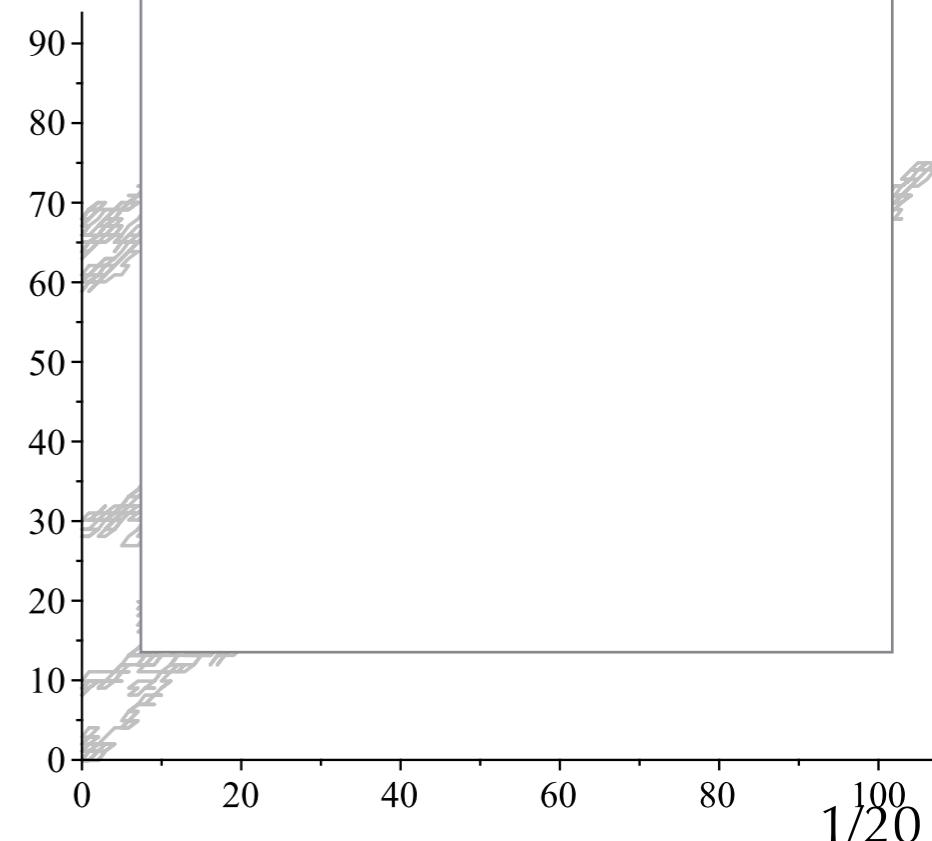
From **simple local** rules, a **global** structure arises.

A quest for **universality** in random discrete structures:  
→ **probabilistic complexity** of structures and algorithms.

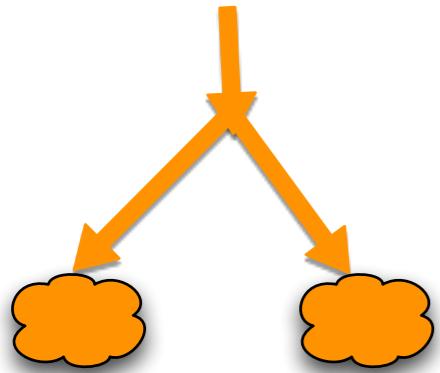
Quantitative results using **complex analysis**.

Typical questions

For a tree of size  $n$ ,  
. probability that a leaf is a child of the root?



# Combinatorics, Randomness and Analysis



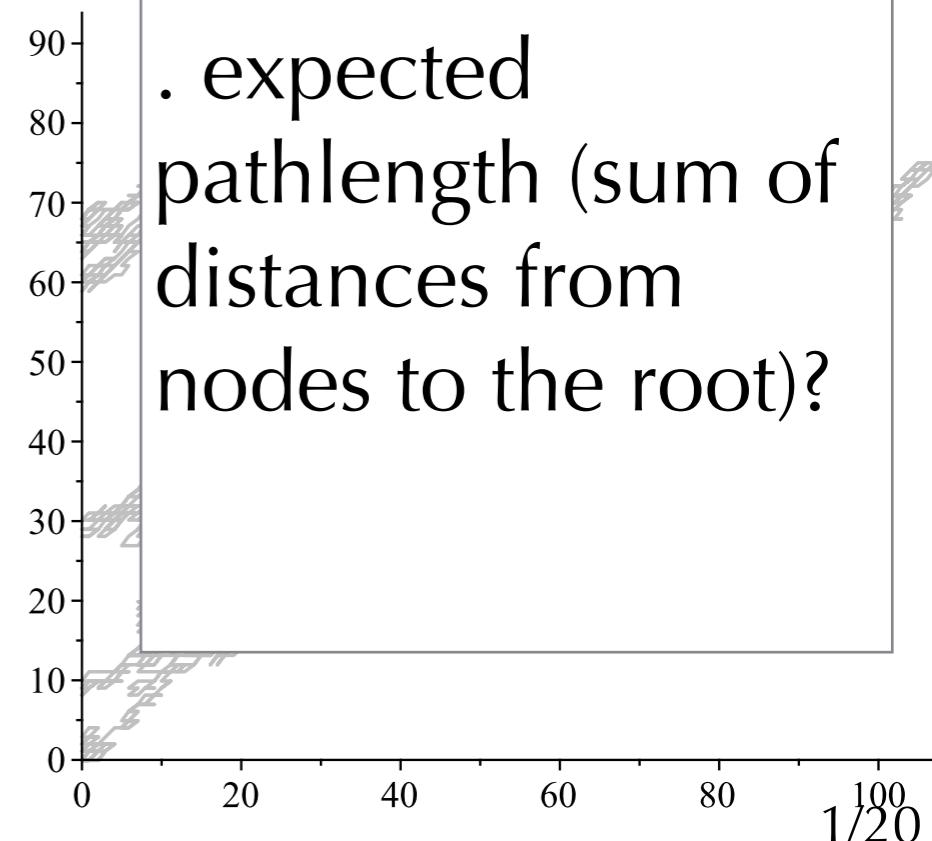
From **simple local** rules, a **global** structure arises.

A quest for **universality** in random discrete structures:  
→ **probabilistic complexity** of structures and algorithms.

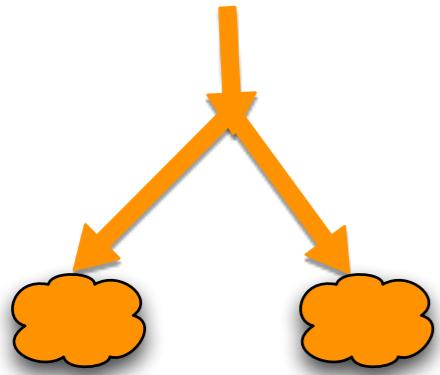
Quantitative results using **complex analysis**.

Typical questions

- For a tree of size  $n$ ,
- . probability that a leaf is a child of the root?
  - . expected pathlength (sum of distances from nodes to the root)?



# Combinatorics, Randomness and Analysis



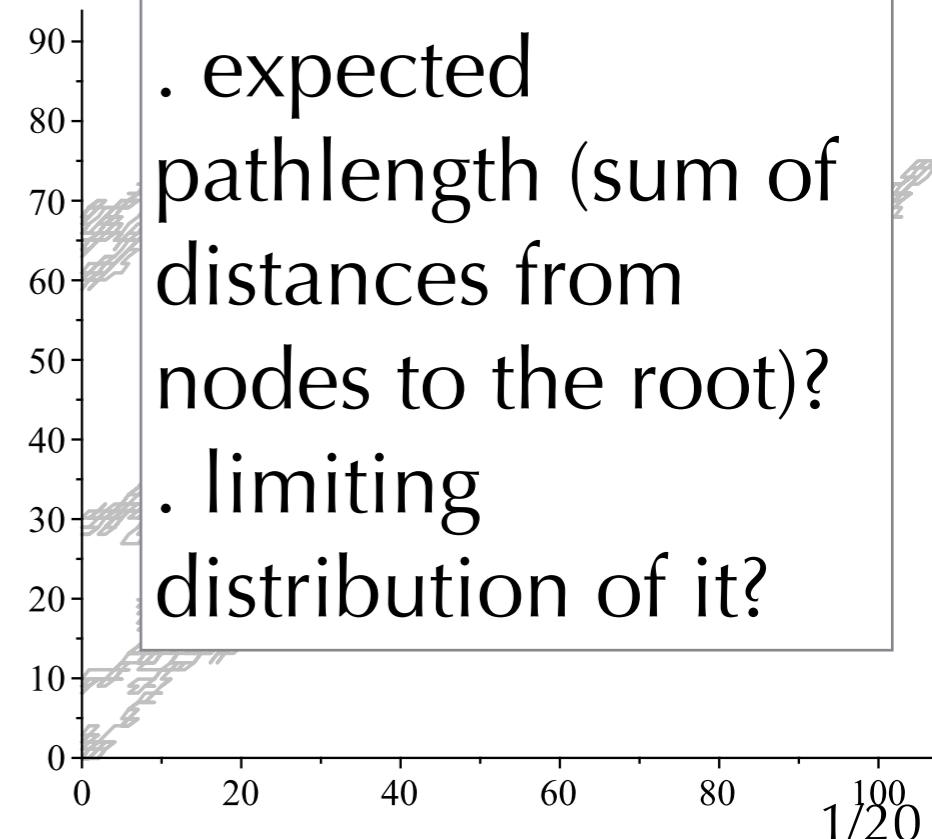
From **simple local** rules, a **global** structure arises.

A quest for **universality** in random discrete structures:  
→ **probabilistic complexity** of structures and algorithms.

Quantitative results using **complex analysis**.

Typical questions

- For a tree of size  $n$ ,
- . probability that a leaf is a child of the root?
  - . expected pathlength (sum of distances from nodes to the root)?
  - . limiting distribution of it?

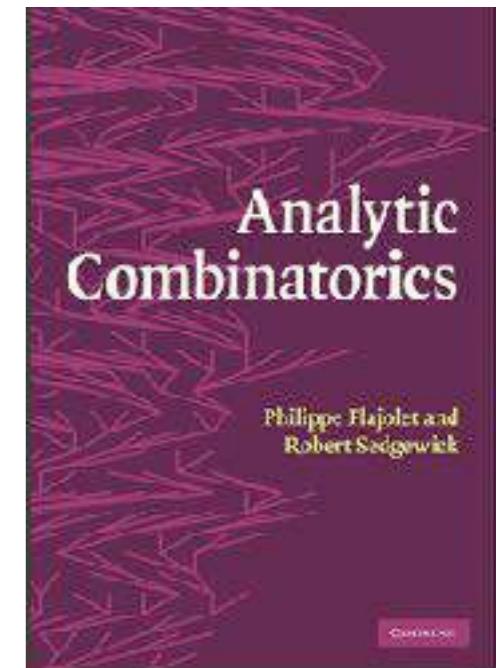


# Philippe Flajolet, the Father of Analytic Combinatorics



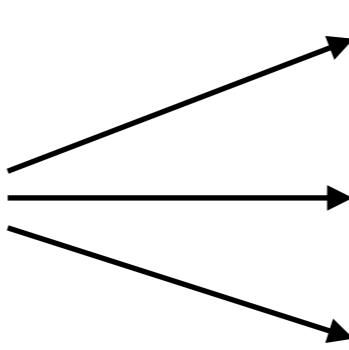
1948-2011

Planned complete works span  
7 volumes of approx 600 pp. each.



2009

Combinatorial  
specification → Generating  
functions



- counting;
- random generation;
- asymptotic analysis.

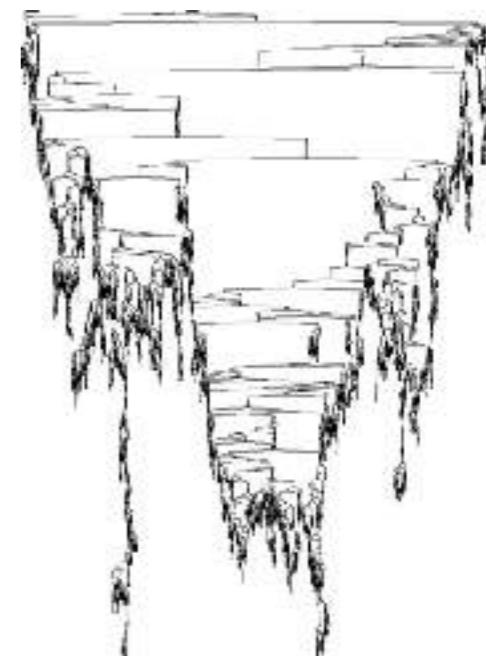
*If you can specify, you can analyze.*

# Constructible Structures

**Language:** 1,  $\mathcal{E}$ , +,  $\times$ , SEQ, SET, CYC  
and recursion.

# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



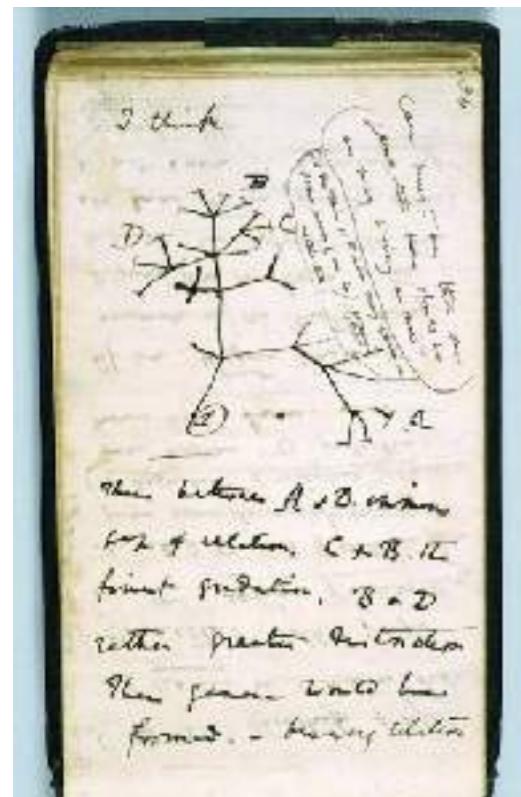
- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$
- Permutations:  $\text{Perm} = \text{SET}(\text{CYC}(\mathcal{Z}))$ ;

# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$
- Permutations:  $\text{Perm} = \text{SET}(\text{CYC}(\mathcal{Z}))$ ;
- ★ Trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ ;

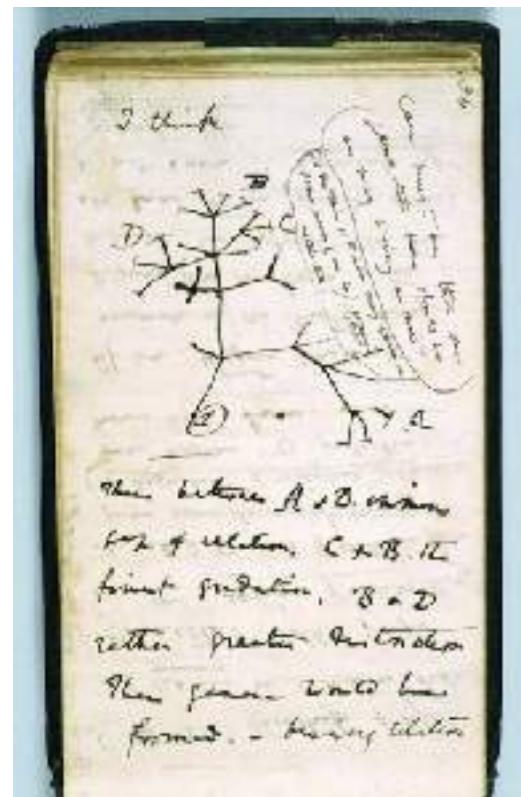


# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$
- Permutations:  $\text{Perm} = \text{SET}(\text{CYC}(\mathcal{Z}))$ ;
- ★ Trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ ;
- Functional graphs:  $\mathcal{F} = \text{SET}(\text{CYC}(\mathcal{T}))$ ;

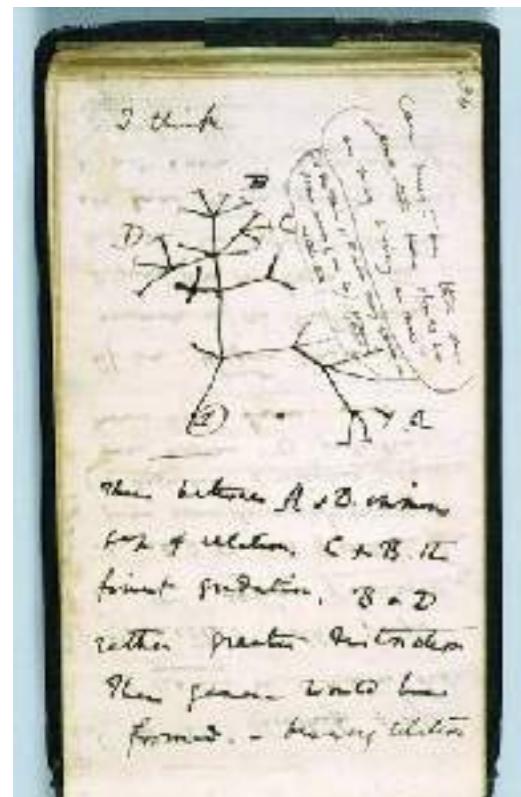
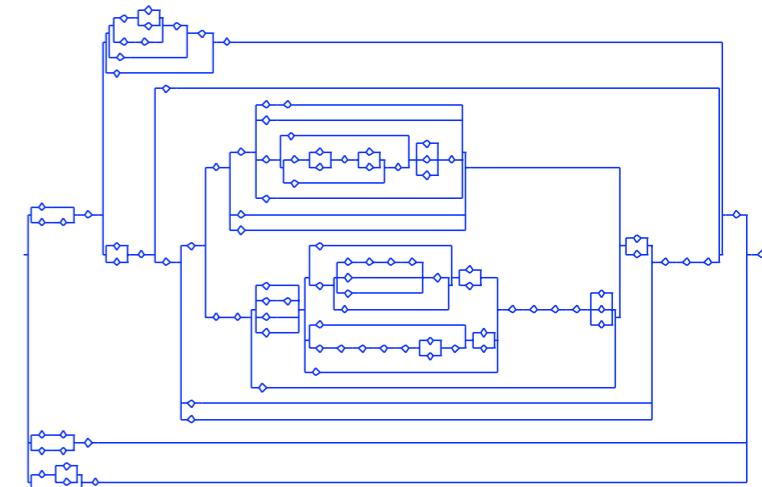


# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$
- Permutations:  $\text{Perm} = \text{SET}(\text{CYC}(\mathcal{Z}))$ ;
- ★ Trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ ;
- Functional graphs:  $\mathcal{F} = \text{SET}(\text{CYC}(\mathcal{T}))$ ;
- ★ Series-parallel graphs:  
 $\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}$ ,  $\mathcal{S} = \text{SEQ}_{>0}(\mathcal{Z} + \mathcal{P})$ ,  $\mathcal{P} = \text{SET}_{>0}(\mathcal{Z} + \mathcal{S})$ ;

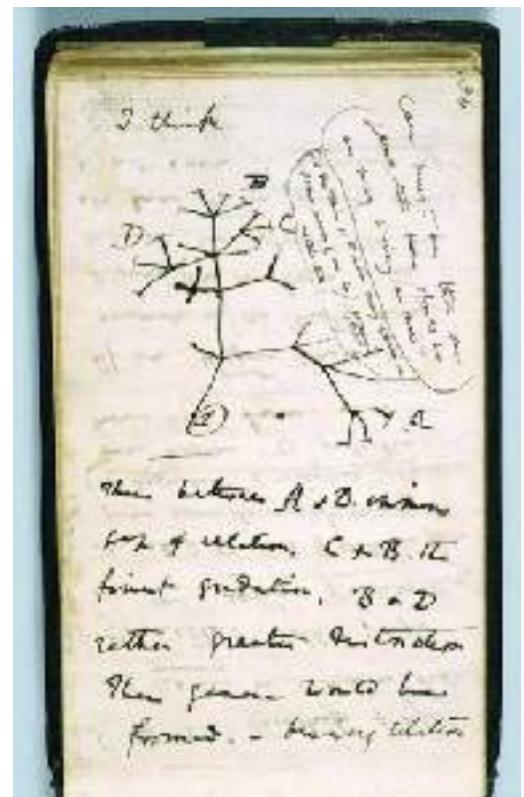
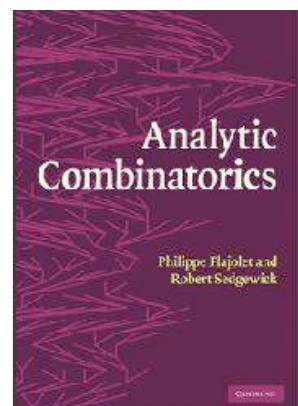
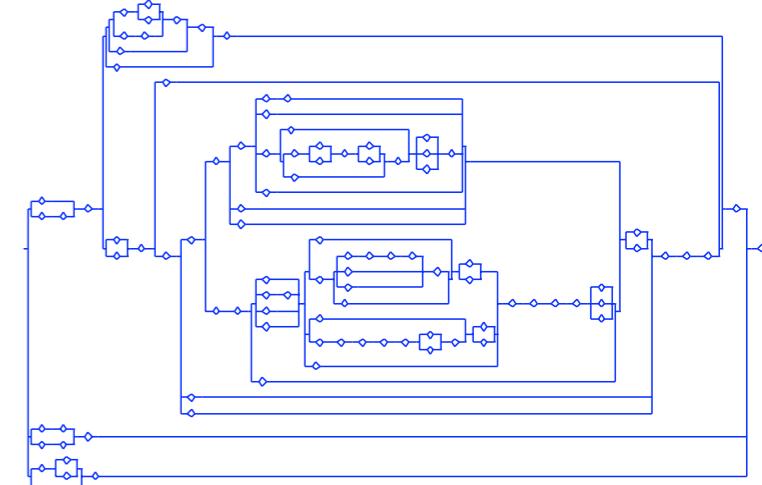


# Constructible Structures

**Language:**  $1, \mathcal{Z}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$   
and recursion.



- ★ Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$
- Permutations:  $\text{Perm} = \text{SET}(\text{CYC}(\mathcal{Z}))$ ;
- ★ Trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ ;
- Functional graphs:  $\mathcal{F} = \text{SET}(\text{CYC}(\mathcal{T}))$ ;
- ★ Series-parallel graphs:  
 $\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}$ ,  $\mathcal{S} = \text{SEQ}_{>0}(\mathcal{Z} + \mathcal{P})$ ,  $\mathcal{P} = \text{SET}_{>0}(\mathcal{Z} + \mathcal{S})$ ;
- ...hundreds of examples in “the purple book”.



# I. Enumeration and Generating Functions

# I. Enumeration and Generating Functions

$$F(z) = \sum_{n=0}^{\infty} f_n z^n$$

# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

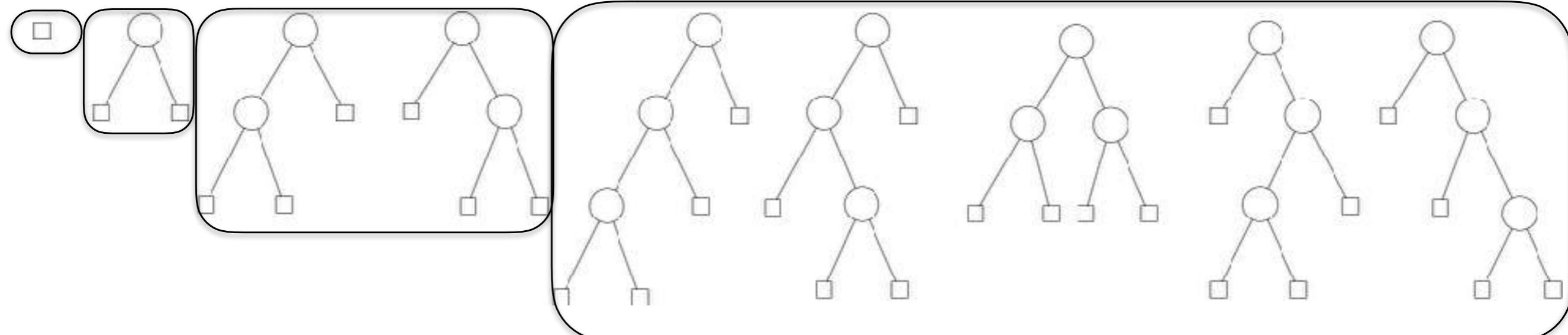
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary  
trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

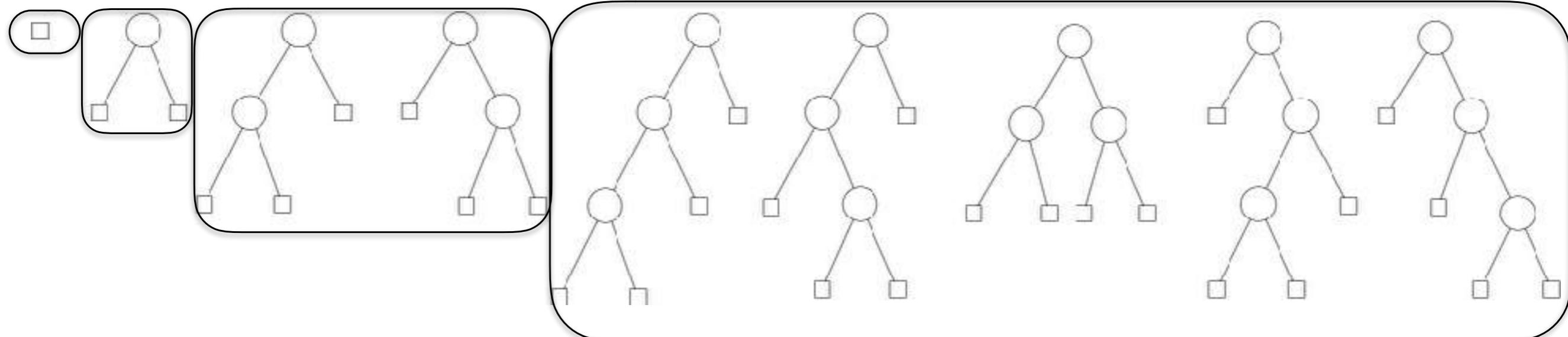
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



$$\text{For } n \geq 1, \quad B_n = \sum_{k=0}^{n-1} B_k B_{n-k-1}$$

# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

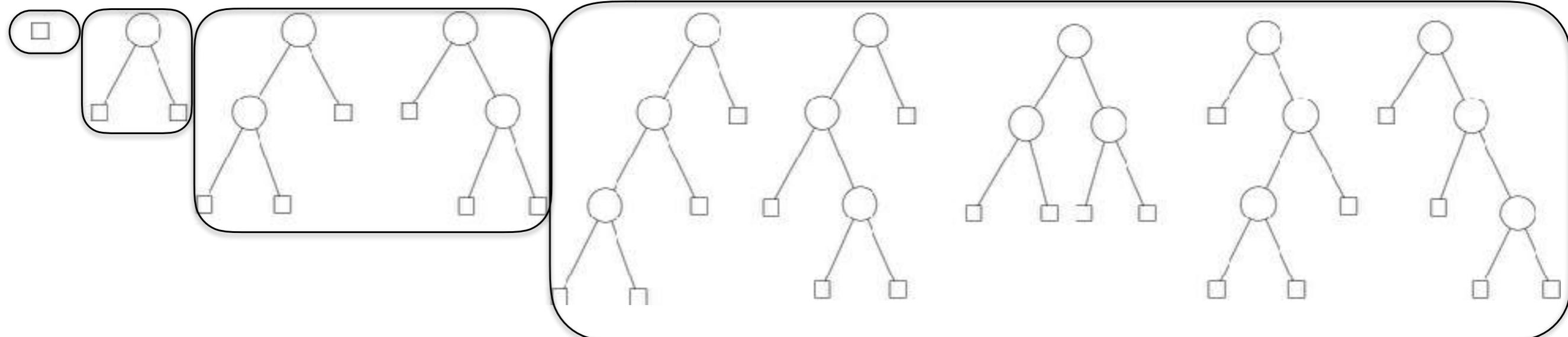
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



$$\text{For } n \geq 1, \quad B_n = \sum_{k=0}^{n-1} B_k B_{n-k-1} \Rightarrow B_n z^n = \sum_{k=0}^{n-1} z(B_k z^k)(B_{n-k-1} z^{n-k-1})$$

# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

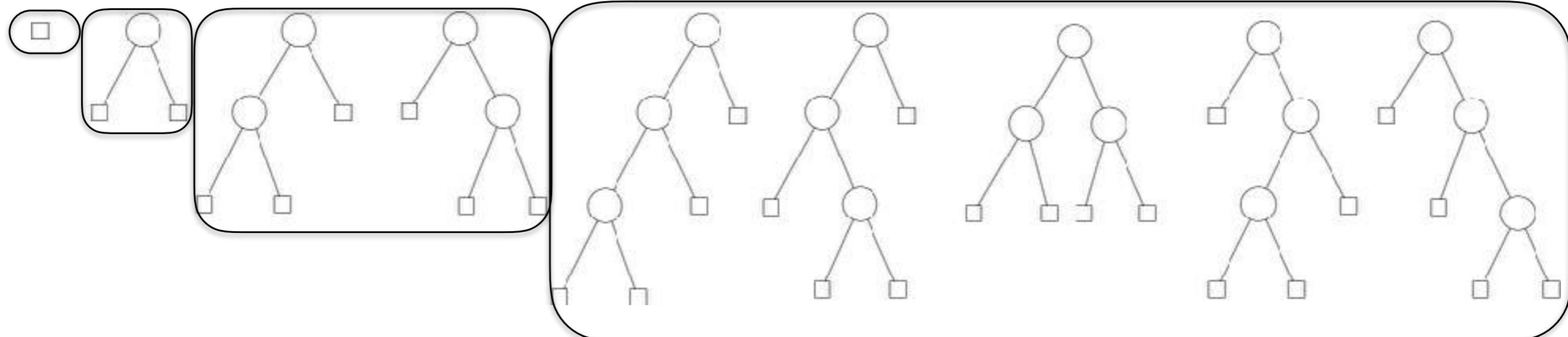
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



$$\text{For } n \geq 1, \quad B_n = \sum_{k=0}^{n-1} B_k B_{n-k-1} \Rightarrow B_n z^n = \sum_{k=0}^{n-1} z(B_k z^k)(B_{n-k-1} z^{n-k-1})$$

$$B(z) = 1 + zB(z)^2$$

# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

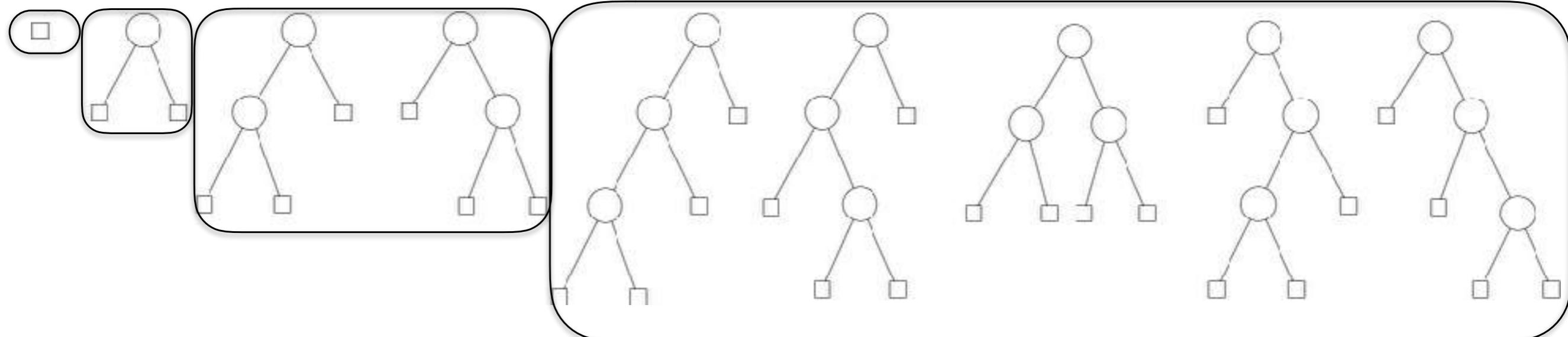
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



$$\text{For } n \geq 1, \quad B_n = \sum_{k=0}^{n-1} B_k B_{n-k-1} \Rightarrow B_n z^n = \sum_{k=0}^{n-1} z(B_k z^k)(B_{n-k-1} z^{n-k-1})$$

$$B(z) = 1 + zB(z)^2 = 1 + z + 2z^2 + 5z^3 + \dots$$

# Example: Binary Trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

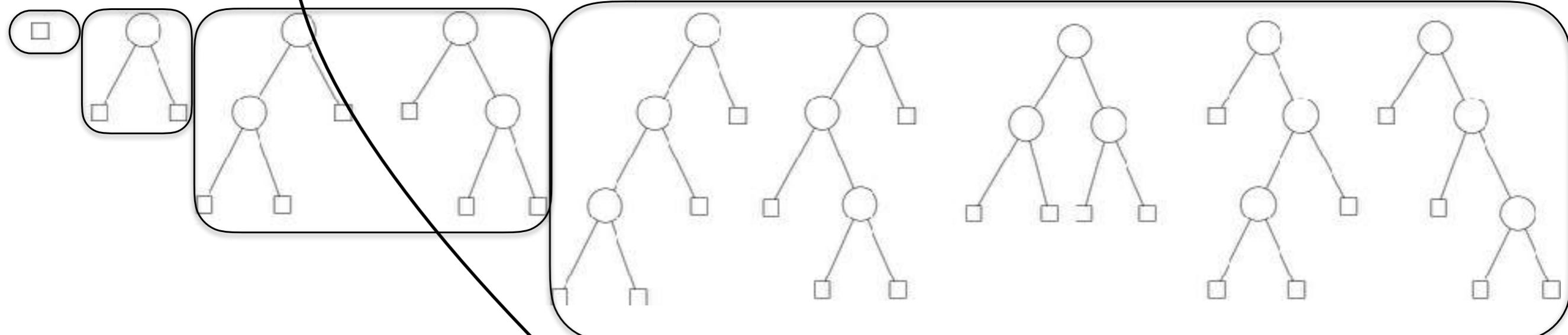
$$B(z) = \sum_{n=0}^{\infty} B_n z^n$$

number of binary trees with  $n$  nodes

(Catalan)

$$B_0 = B_1 = 1 \quad B_2 = 2$$

$$B_3 = 5$$



$$\text{For } n \geq 1, \quad B_n = \sum_{k=0}^{n-1} B_k B_{n-k-1} \Rightarrow B_n z^n = \sum_{k=0}^{n-1} z(B_k z^k)(B_{n-k-1} z^{n-k-1})$$

$$B(z) = 1 + z B(z)^2 = 1 + z + 2z^2 + 5z^3 + \dots$$

# What do we count?

$$\text{Inv}[\{1, 2, 3\}] = \left\{ \begin{array}{c} \text{Diagram 1: } 1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3 \\ \text{Diagram 2: } 1 \rightarrow 2, 2 \rightarrow 1, 3 \rightarrow 3 \\ \text{Diagram 3: } 1 \rightarrow 3, 2 \rightarrow 3, 3 \rightarrow 1 \\ \text{Diagram 4: } 1 \rightarrow 3, 2 \rightarrow 1, 3 \rightarrow 2 \end{array} \right\}$$

4 involutions;  
3 of them permuted by  $\mathfrak{S}_3 \rightarrow$  2 unlabelled structures.

Exponential generating series (EGF):

$$F(z) = \sum_{n=0}^{\infty} f_n \frac{z^n}{n!}, \quad f_n = \text{nb. labelled structs of size } n.$$

$$\text{Inv}_3(z) = \frac{2}{3} z^3$$

Ordinary generating series (OGF):

$$\tilde{F}(z) = \sum_{n=0}^{\infty} \tilde{f}_n z^n, \quad \tilde{f}_n = \text{nb. unlabelled structs of size } n.$$

$$\widetilde{\text{Inv}}_3(z) = 2z^3$$

# A Dictionary for Generating Series

**Language:** 1,  $\mathcal{E}$ , +,  $\times$ , SEQ, SET, CYC and recursion.

describes *constructible structures*

# A Dictionary for Generating Series

**Language:**  $1, \mathcal{X}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$  and recursion.

describes ***constructible structures***

Structure	EGF	OGF
$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$	$\tilde{A}(z) + \tilde{B}(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$	$\tilde{A}(z) \times \tilde{B}(z)$

# A Dictionary for Generating Series

**Language:**  $1, \mathcal{X}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$  and recursion.

describes ***constructible structures***

Structure	EGF	OGF
$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$	$\tilde{A}(z) + \tilde{B}(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$	$\tilde{A}(z) \times \tilde{B}(z)$
$\text{SEQ}(\mathcal{A})$	$\frac{1}{1 - A(z)}$	$\frac{1}{1 - \tilde{A}(z)}$

# A Dictionary for Generating Series

**Language:**  $1, \mathcal{X}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$  and recursion.

describes ***constructible structures***

Structure	EGF	OGF
$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$	$\tilde{A}(z) + \tilde{B}(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$	$\tilde{A}(z) \times \tilde{B}(z)$
$\text{SEQ}(\mathcal{A})$	$\frac{1}{1 - A(z)}$	$\frac{1}{1 - \tilde{A}(z)}$
$\text{SET}(\mathcal{A})$	$\exp(A(z))$	$\exp\left(\sum \tilde{A}(z^i)/i\right)$

# A Dictionary for Generating Series

**Language:**  $1, \mathcal{X}, +, \times, \text{SEQ}, \text{SET}, \text{CYC}$  and recursion.

describes ***constructible structures***

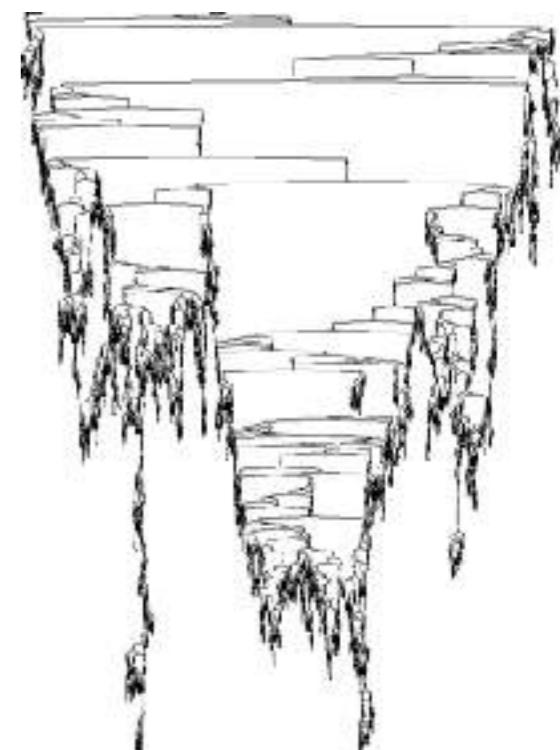
Structure	EGF	OGF
$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$	$\tilde{A}(z) + \tilde{B}(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$	$\tilde{A}(z) \times \tilde{B}(z)$
$\text{SEQ}(\mathcal{A})$	$\frac{1}{1 - A(z)}$	$\frac{1}{1 - \tilde{A}(z)}$
$\text{SET}(\mathcal{A})$	$\exp(A(z))$	$\exp\left(\sum \tilde{A}(z^i)/i\right)$
$\text{CYC}(\mathcal{A})$	$\log \frac{1}{1 - A(z)}$	$\sum \frac{\phi(i)}{i} \log \frac{1}{1 - \tilde{A}(z^i)}$

# Examples

# Examples

Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

$$\rightarrow B(z) = 1 + zB(z)^2 = \tilde{B}(z)$$



# Examples

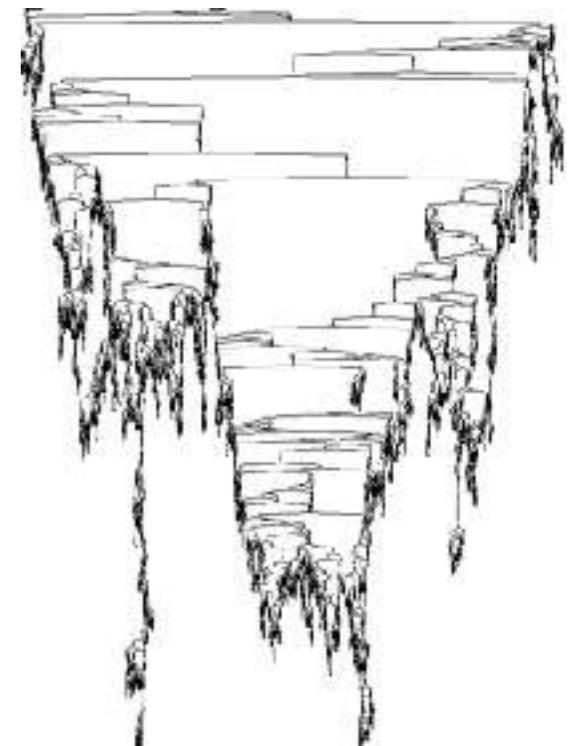
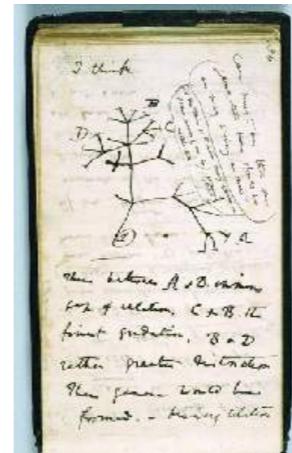
Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

$$\rightarrow B(z) = 1 + zB(z)^2 = \tilde{B}(z)$$

Cayley trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$

$$\rightarrow T(z) = z \exp(T(z));$$

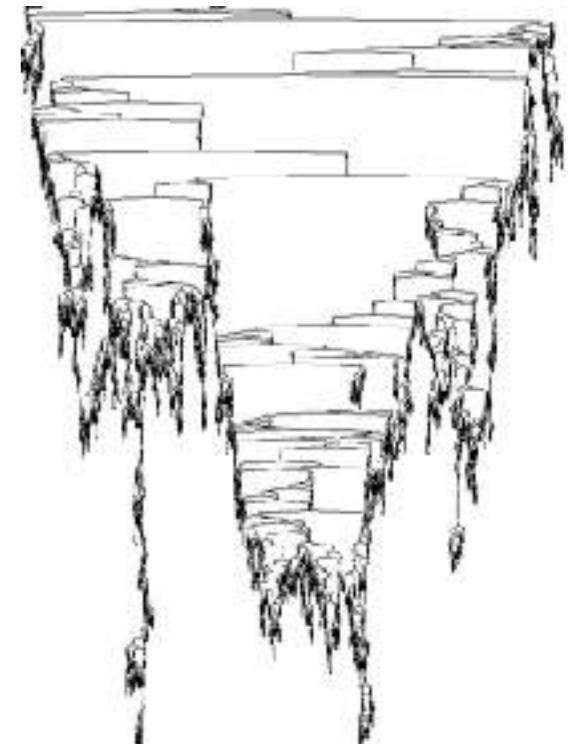
$$\rightarrow \tilde{T}(z) = z \exp\left(\tilde{T}(z) + \frac{1}{2}\tilde{T}(z^2) + \frac{1}{3}\tilde{T}(z^3) + \dots\right)$$



# Examples

Binary trees:  $\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

$$\rightarrow B(z) = 1 + zB(z)^2 = \tilde{B}(z)$$



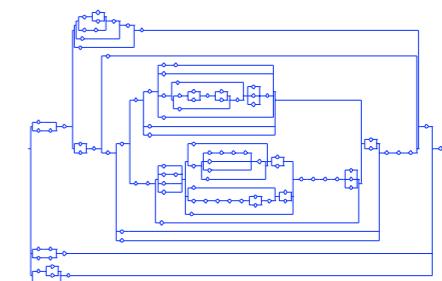
Cayley trees:  $\mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$

$$\rightarrow T(z) = z \exp(T(z));$$

$$\rightarrow \tilde{T}(z) = z \exp\left(\tilde{T}(z) + \frac{1}{2}\tilde{T}(z^2) + \frac{1}{3}\tilde{T}(z^3) + \dots\right)$$



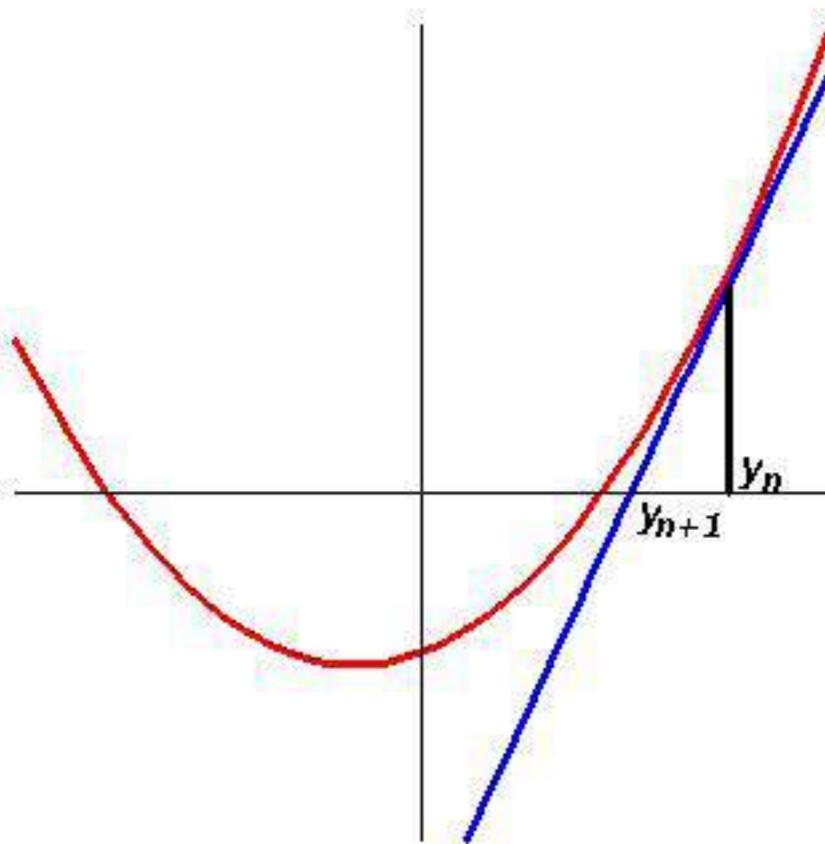
Series-parallel graphs:



$$\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}, \mathcal{S} = \text{SEQ}_{>0}(\mathcal{Z} + \mathcal{P}), \mathcal{P} = \text{SET}_{>0}(\mathcal{Z} + \mathcal{S})$$

$$\rightarrow \left\{ G(z) = z + S(z) + P(z), S(z) = \frac{1}{1 - z - P(z)} - 1, P(z) = e^{z + S(z)} - 1 \right\}$$

## II. Newton Iteration and Fast Enumeration



$$\begin{array}{l}
 y^3 + a^2y - 2a^3 + axy - x^3 = 0, \quad y = a - \frac{x}{4} + \frac{x^2}{64a} + \frac{111x^3}{512a^2} + \frac{509x^4}{16384a^3} \text{ &c.} \\
 \\ 
 \begin{array}{ll|l}
 + a + p = y, & +y^3 & +a^3 + 3a^2p + 3ap^2 + p^3 \\
 + axy & +a^2y & +a^2x + axp \\
 + x^2y & -x^3 & +ax^3 + a^3p \\
 -x^3 & -2a^3 & -x^3 \\
 & & -2a^3
 \end{array} \\
 \\ 
 \begin{array}{ll|l}
 -\frac{1}{4}x + q = p, & +p^3 & -\frac{1}{4}x^3 + \frac{1}{16}x^2q - \frac{1}{4}xq^2 + q^3 \\
 + 3ap^2 & +3ap^2 & +\frac{3}{16}ax^2 - \frac{1}{4}axq + 3aq^2 \\
 + axp & -\frac{1}{4}ax^2 + axq & -ax^3 + a^2q \\
 + 4a^2p & +4a^2p & +a^2x \\
 + a^2x & -a^2x & +a^2x \\
 -x^3 & -x^3 & -x^3
 \end{array} \\
 \\ 
 \begin{array}{ll|l}
 + \frac{x^2}{64a} + r = q, & +q^3 & * \\
 -\frac{1}{4}xq^2 & -\frac{1}{4}xq^2 & * \\
 + 3aq^2 & + 3aq^2 & -\frac{3x^4}{4096a} * + \frac{1}{16}x^2r + 3ar^2 \\
 + \frac{1}{16}x^2q & + \frac{1}{16}x^2q & + \frac{3x^4}{1024a} * + \frac{1}{16}x^2r \\
 -\frac{1}{4}axq & -\frac{1}{4}axq & -\frac{1}{16}x^3 - \frac{1}{4}axr \\
 + 4a^2q & + 4a^2q & + \frac{1}{16}ax^2 + 4a^2r \\
 -\frac{6}{64}x^3 y & -\frac{6}{64}x^3 y & -\frac{6}{64}x^3 \\
 -\frac{1}{16}ax^2 & -\frac{1}{16}ax^2 & -\frac{1}{16}ax^2
 \end{array} \\
 \\ 
 + 4a^2 - \frac{1}{4}ax - \frac{9}{16}x^2) + \frac{111}{128}x^3 - \frac{15x^4}{4096a} \left( + \frac{131x^3}{512a^2} + \frac{509x^4}{16384a^3}
 \end{array}$$

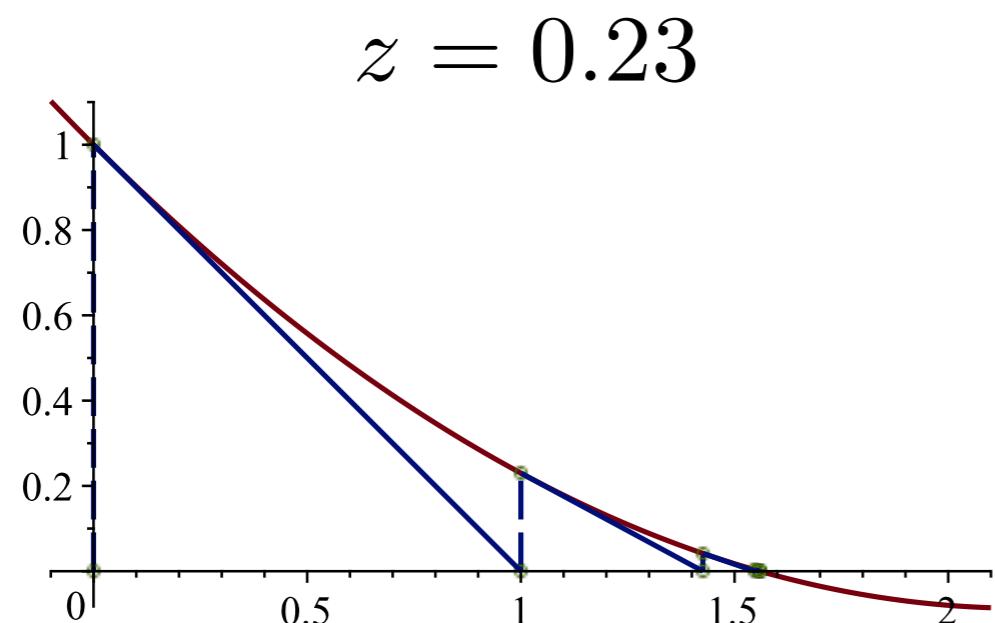
# Numerical Newton Iteration

To solve  $\phi(y) = 0$ , iterate

$$y^{[n+1]} = y^{[n]} + u^{[n]},$$

with  $\phi(y^{[n]}) + \phi'(y^{[n]})u^{[n]} = 0$ .

$$\phi(y) = 1 + zy^2 - y$$



# Numerical Newton Iteration

To solve  $\phi(y) = 0$ , iterate

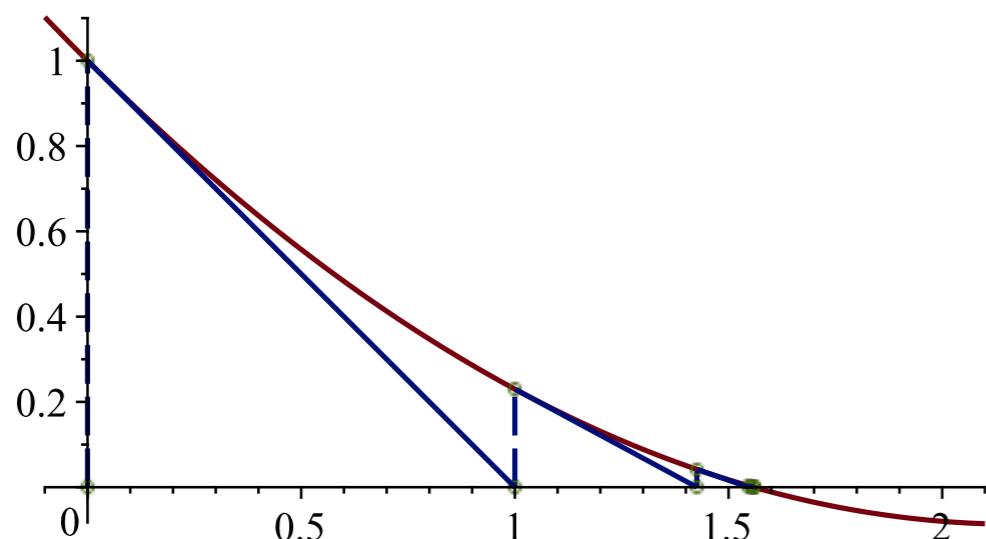
$$y^{[n+1]} = y^{[n]} + u^{[n]},$$

$$\text{with } \phi(y^{[n]}) + \phi'(y^{[n]})u^{[n]} = 0.$$

$$\phi(y) = 1 + zy^2 - y$$

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

$$z = 0.23$$



# Numerical Newton Iteration

To solve  $\phi(y) = 0$ , iterate

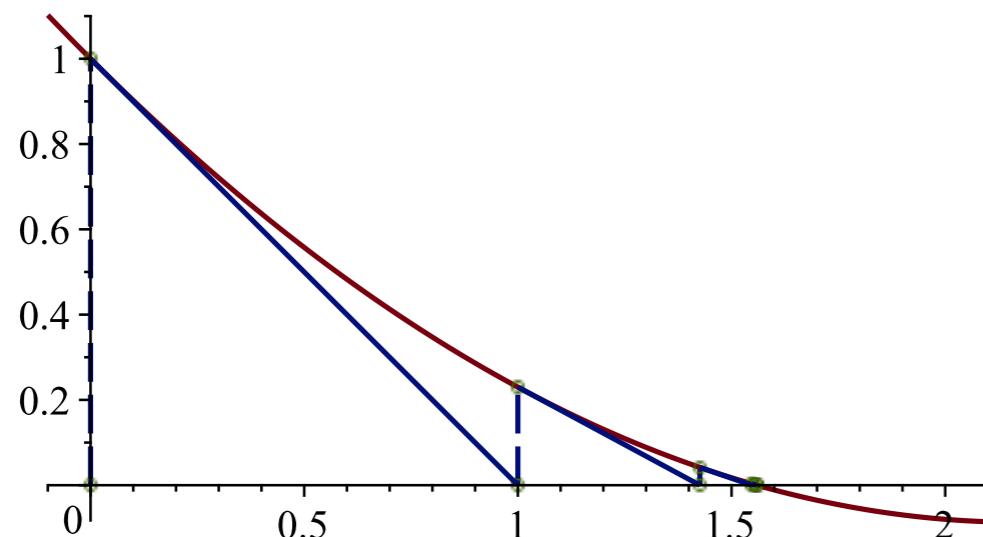
$$y^{[n+1]} = y^{[n]} + u^{[n]},$$

with  $\phi(y^{[n]}) + \phi'(y^{[n]})u^{[n]} = 0$ .

$$\phi(y) = 1 + zy^2 - y$$

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

$$z = 0.23$$



Quadratic convergence

$$y^{[0]} = 0,$$

$$y^{[1]} = 1.000000000000000,$$

$$y^{[2]} \simeq 1.4259259259259259,$$

$$y^{[3]} \simeq 1.5471933181836303,$$

$$y^{[4]} \simeq 1.5589256602748822,$$

$$y^{[5]} \simeq 1.5590375713926592,$$

$$y^{[6]} \simeq 1.5590375815769151$$

# Newton Iteration for Power Series

Same  
Newton  
Iteration

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

# Newton Iteration for Power Series

Same  
Newton  
Iteration

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$

# Newton Iteration for Power Series

Same  
Newton  
Iteration

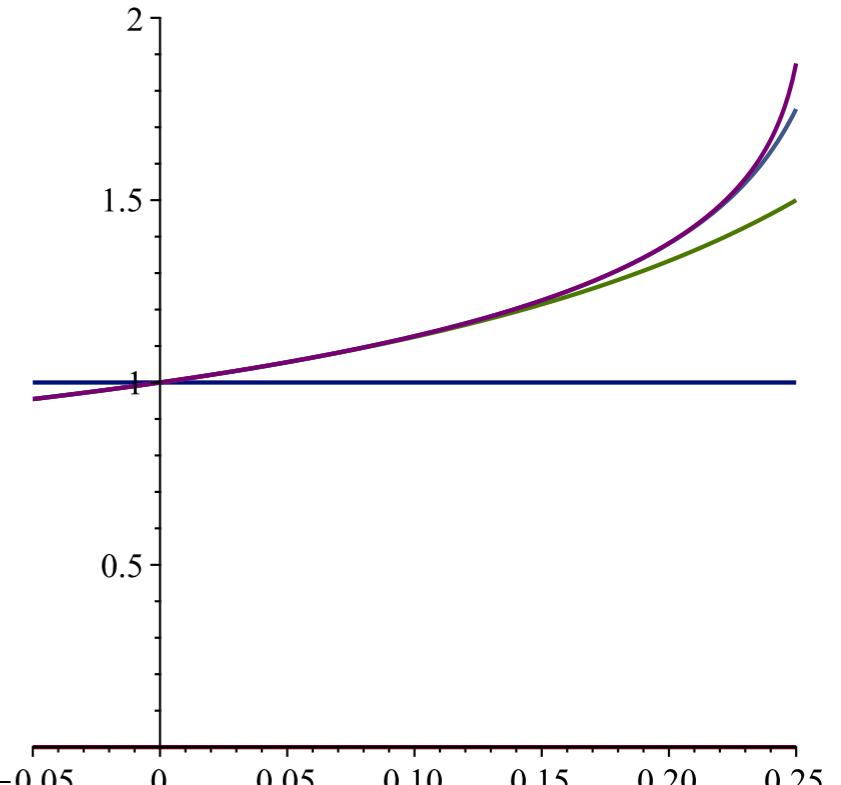
$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$



# Newton Iteration for Power Series

Same  
Newton  
Iteration

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

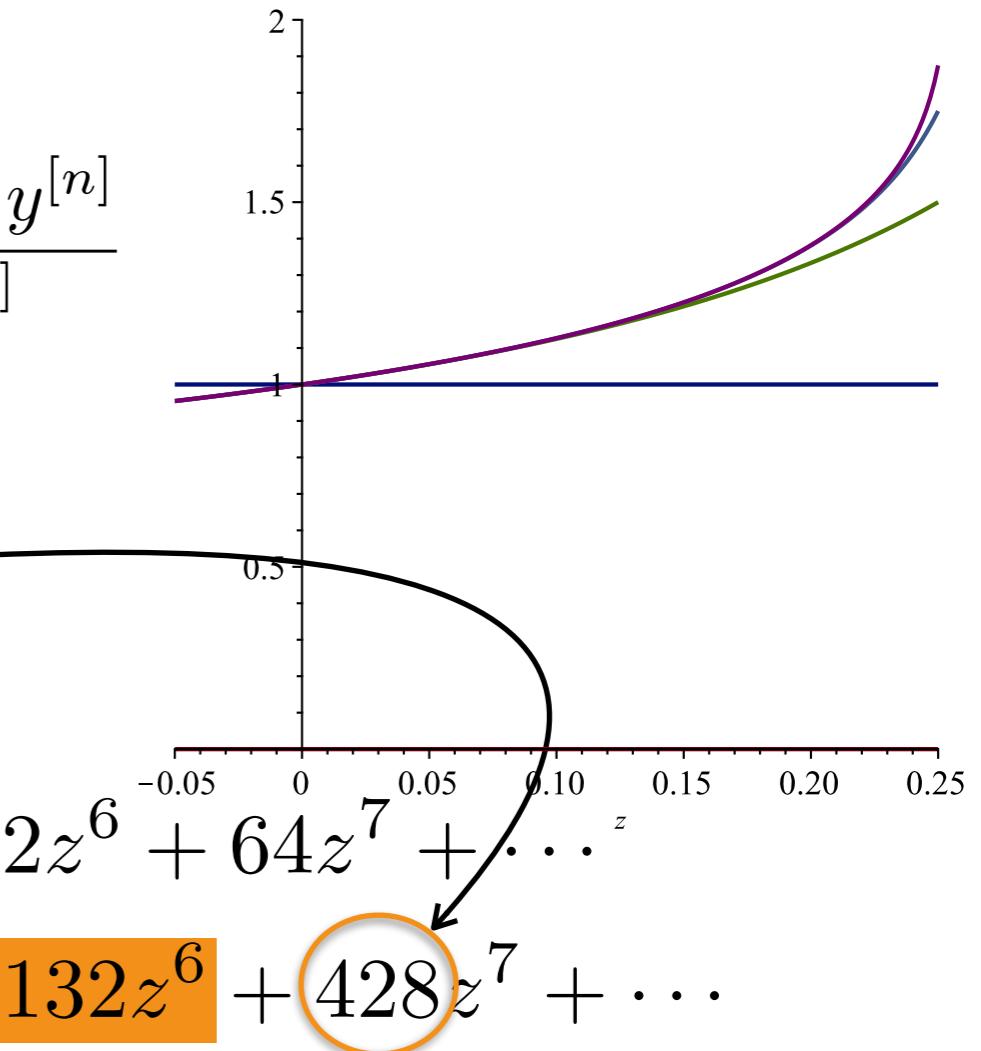
$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$

Proving numerical convergence requires control over the tails



# Newton Iteration for Power Series

Same  
Newton  
Iteration

$$y^{[n+1]} = \mathcal{N}(y^{[n]}) = y^{[n]} + \frac{1 + zy^{[n]} - y^{[n]}}{1 - 2zy^{[n]}}$$

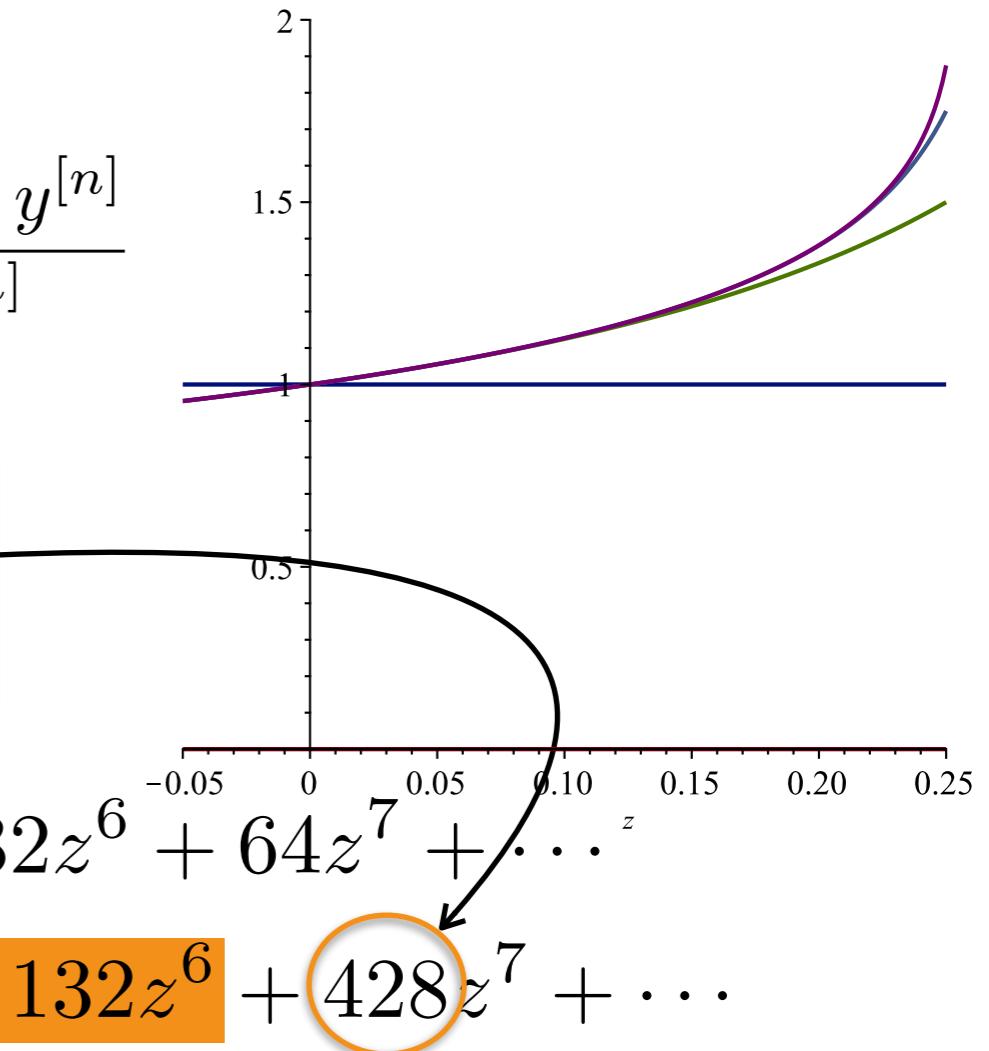
$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$

Proving numerical convergence requires control over the tails



On power series:  $y - y^{[\infty]} = O(z^m) \Rightarrow \mathcal{N}(y) - y^{[\infty]} = O(z^{2m(+1)})$

# Newton Iteration for Power Series

Same  
Newton  
Iteration

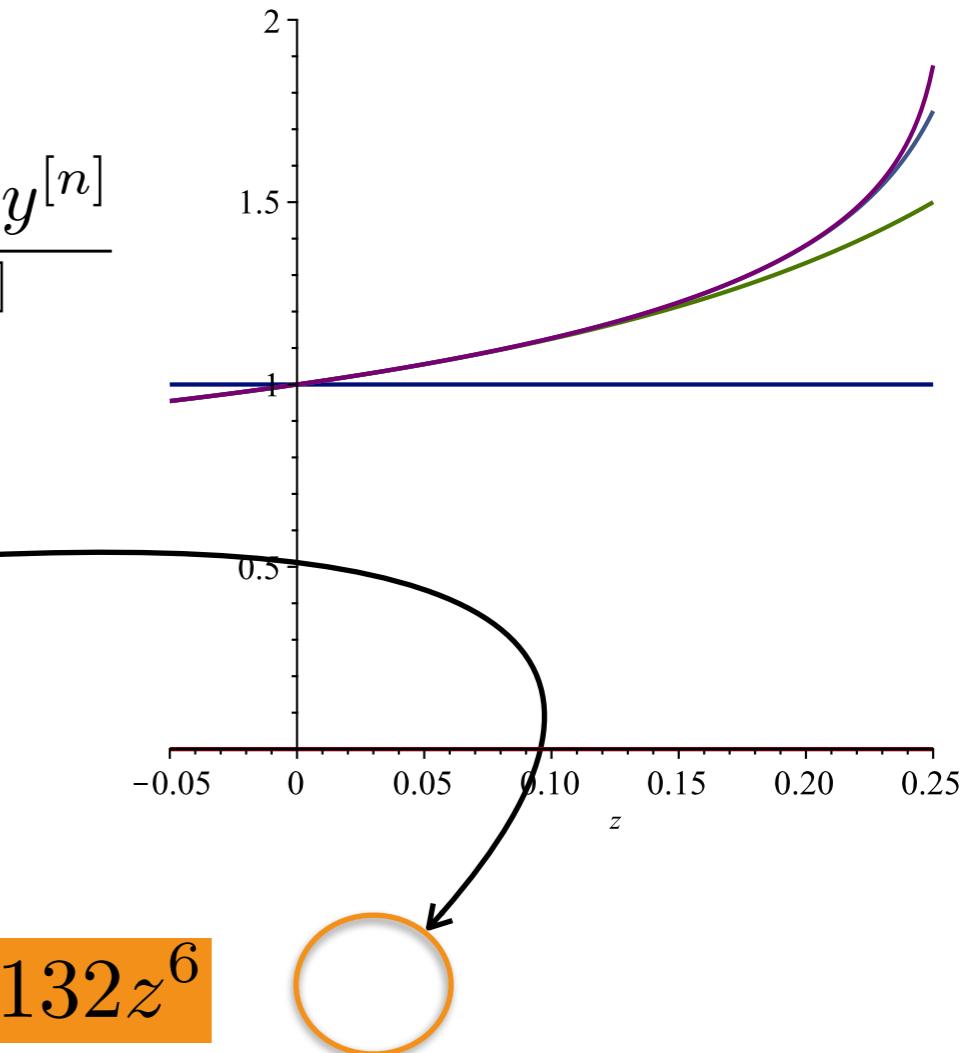
$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6$$

Proving numerical  
convergence requires  
control over the tails



On power series:  $y - y^{[\infty]} = O(z^m) \Rightarrow \mathcal{N}(y) - y^{[\infty]} = O(z^{2m(+1)})$

# Newton Iteration for Power Series

Same  
Newton  
Iteration

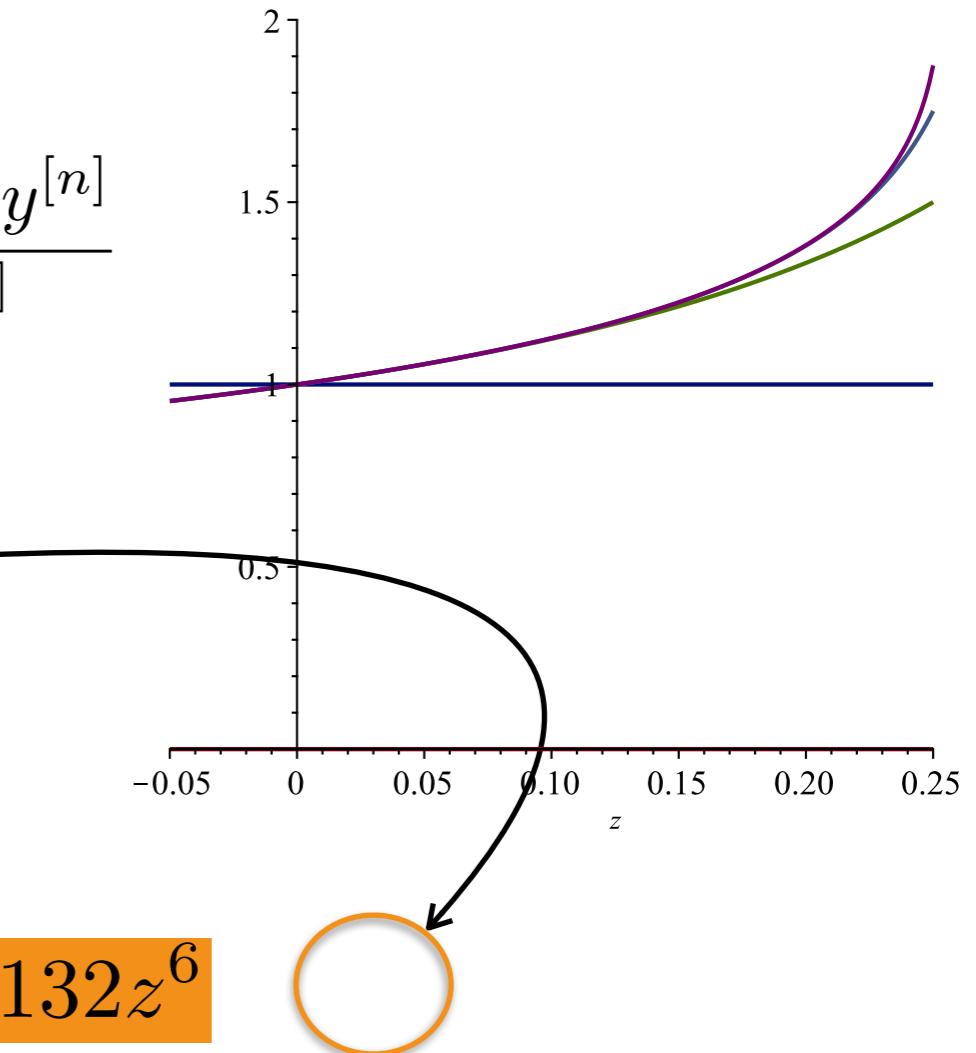
$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6$$

Proving numerical convergence requires control over the tails



On power series:  $y - y^{[\infty]} = O(z^m) \Rightarrow \mathcal{N}(y) - y^{[\infty]} = O(z^{2m(+1)})$

```
Expand(N) = {
    res=Expand(N/2);
    a=phi(res); b=phi'(res);
    u=Solve(a+bx,x);
    return res+u; }
```

# Newton Iteration for Power Series

Same  
Newton  
Iteration

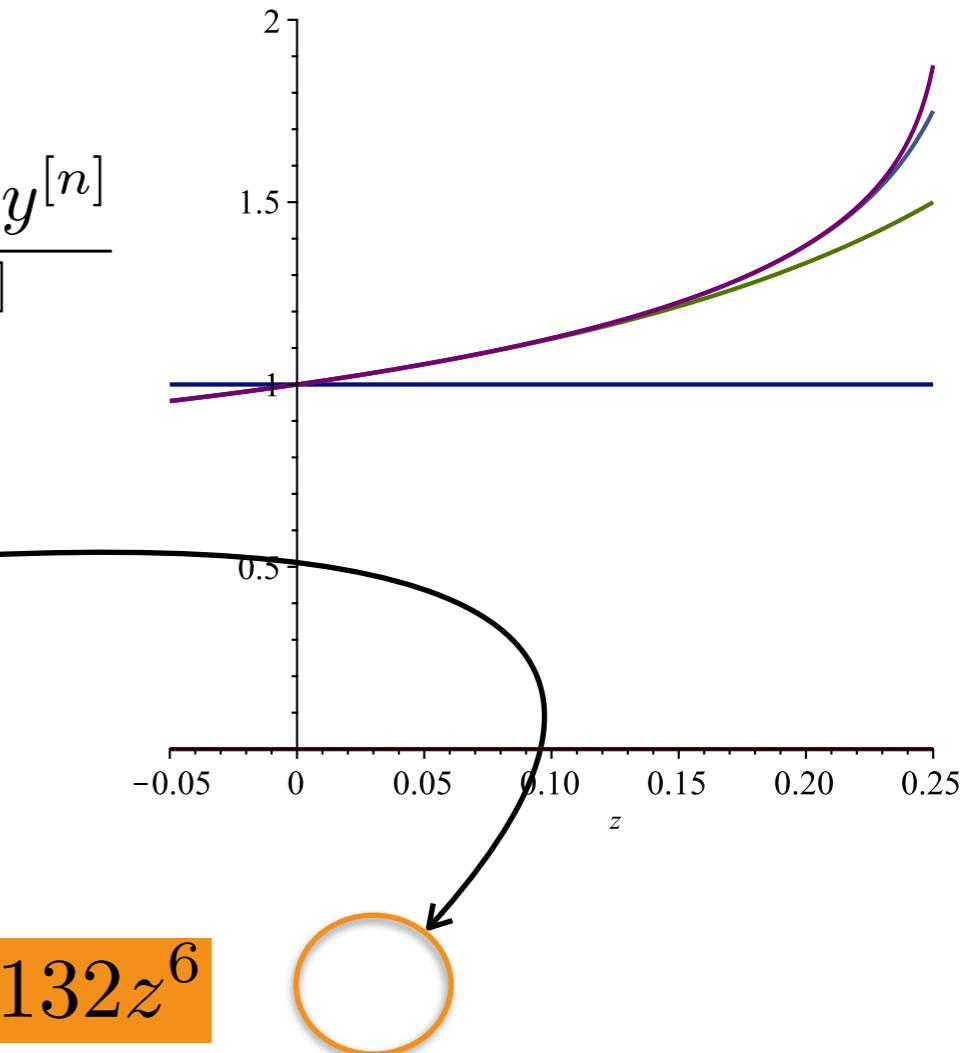
$$y^{[0]} = 0$$

$$y^{[1]} = 1$$

$$y^{[2]} = 1 + z + 2z^2$$

$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6$$

Proving numerical convergence requires control over the tails



On power series:  $y - y^{[\infty]} = O(z^m) \Rightarrow \mathcal{N}(y) - y^{[\infty]} = O(z^{2m+1})$

```
Expand(N) = {
    res=Expand(N/2);
    a=f(res); b=f'(res);
    u=Solve(a+bx,x);
    return res+u; }
```

$\text{Cost}(N) \leq ct \times \text{Cost}(\text{last step})$

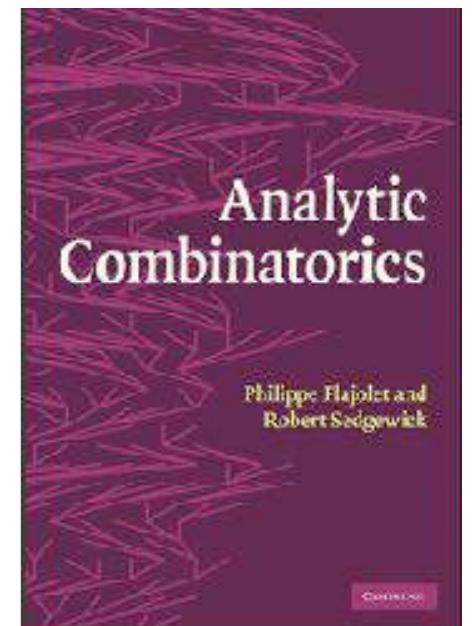
# Fast Enumeration

**Thm.** First  $N$  coefficients of GFs of all *constructible* structures:

1. arithmetic complexity  $O(N \log N)$  (both ogf & egf);
2. bit complexity

$$O(N^2 \log^2 N \log \log N) \text{ (ogf)}; \quad O(N^2 \log^3 N \log \log N) \text{ (egf)}.$$

Ingredients: Newton iteration & FFT.



# Fast Enumeration

**Thm.** First  $N$  coefficients of GFs of all *constructible* structures:

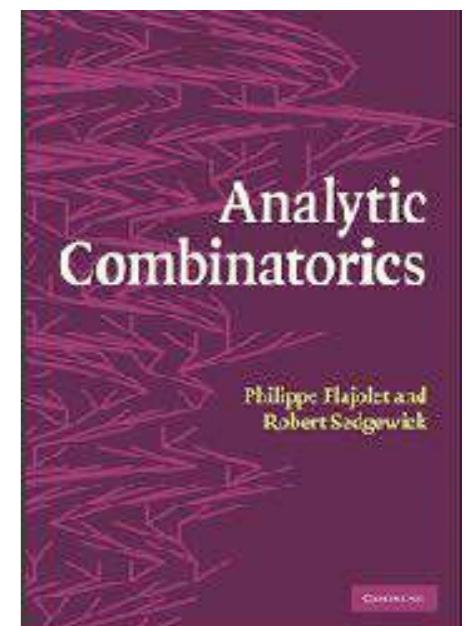
1. arithmetic complexity  $O(N \log N)$  (both ogf & egf);

2. bit complexity

$O(N^2 \log^2 N \log \log N)$  (ogf);     $O(N^2 \log^3 N \log \log N)$  (egf).

Ingredients: Newton iteration & FFT.

```
> with(NewtonGF):
> BinTrees:={B=Union(Epsilon,Prod(z,B,B))}:
> GFSeries(BinTrees,labelled,z,21)[2];
B = 1 + z + 2 z2 + 5 z3 + 14 z4 + 42 z5 + 132 z6 + 429 z7 + 1430 z8
+ 4862 z9 + 16796 z10 + 58786 z11 + 208012 z12 + 742900 z13
+ 2674440 z14 + 9694845 z15 + 35357670 z16 + 129644790 z17
+ 477638700 z18 + 1767263190 z19 + 6564120420 z20 + O(z21)
```

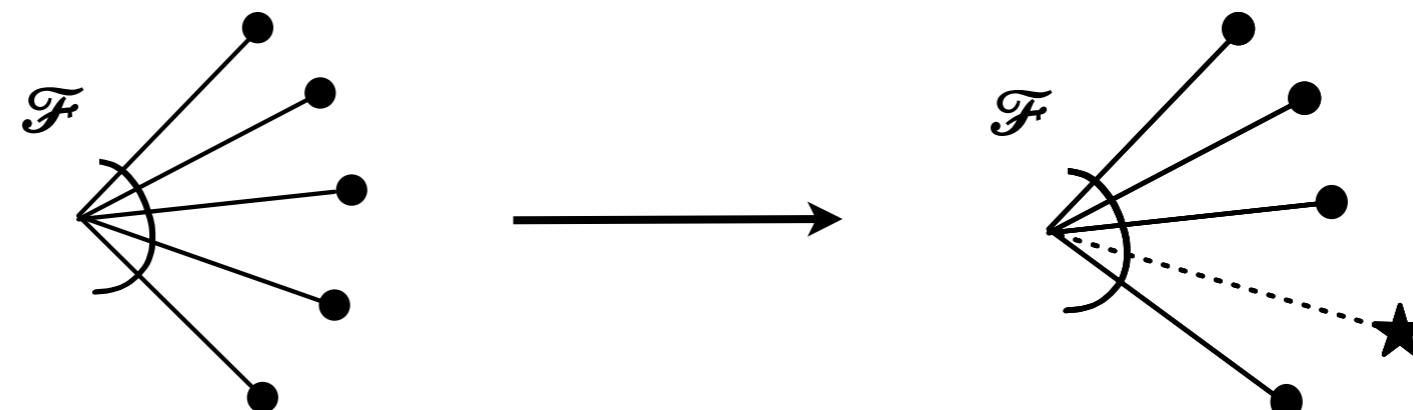
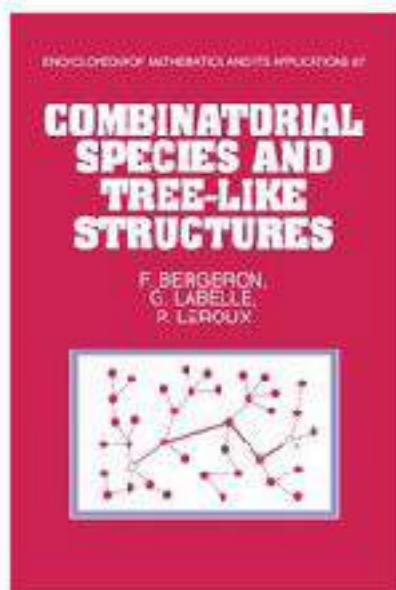


Demo

NewtonGF

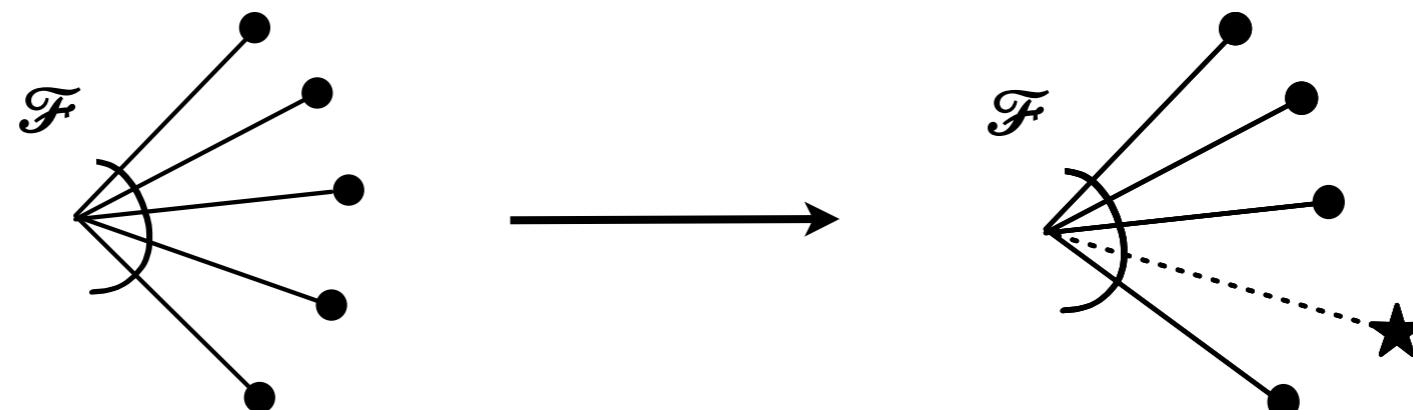
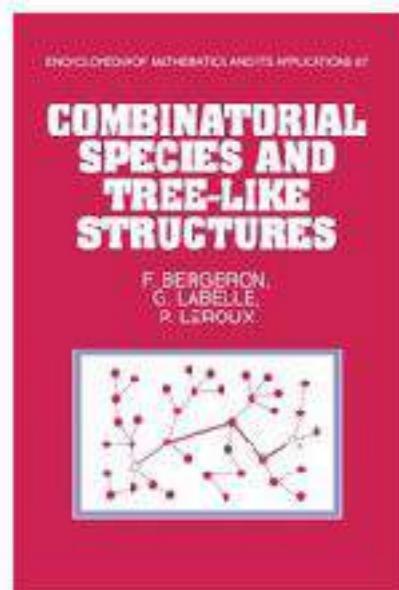
# Derivative of Combinatorial Structures

(origin: species theory)



# Derivative of Combinatorial Structures

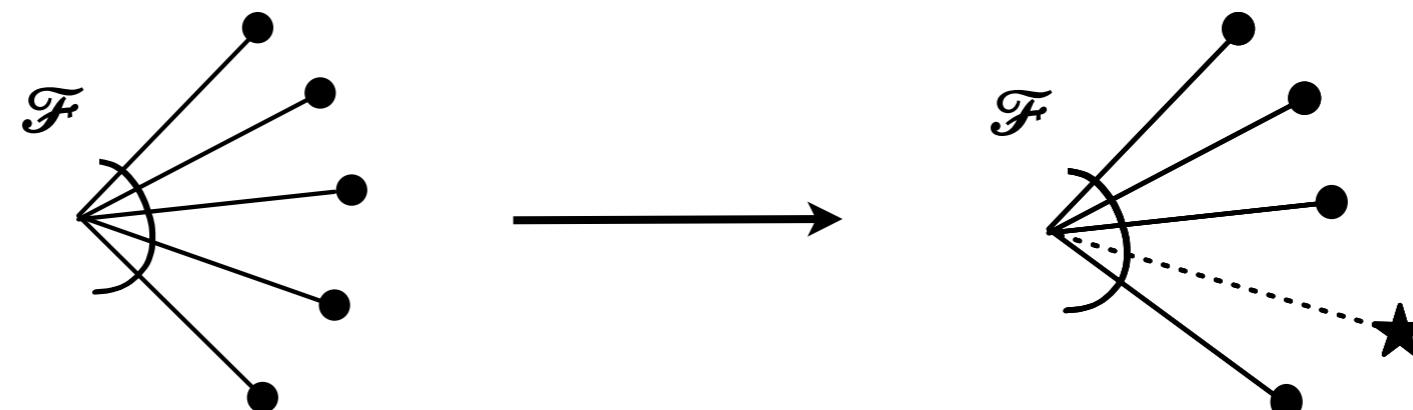
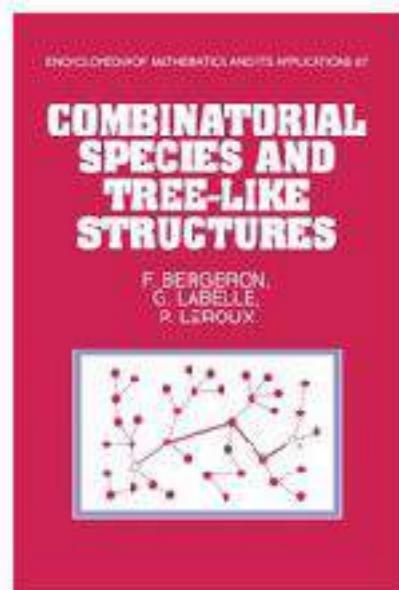
(origin: species theory)



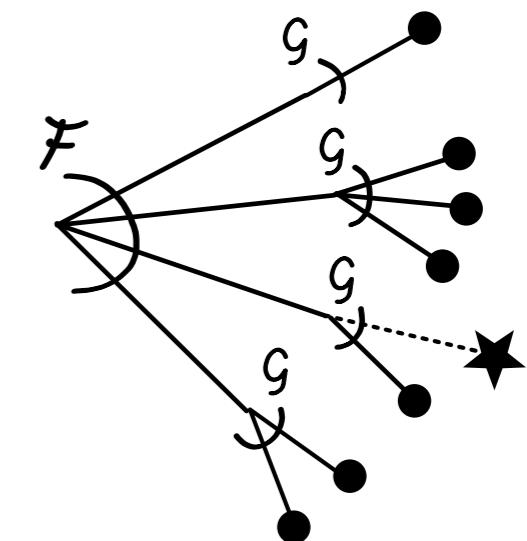
- $(\mathcal{F} + \mathcal{G})' = \mathcal{F}' + \mathcal{G}'; (\mathcal{F} \times \mathcal{G})' = \mathcal{F}' \times \mathcal{G} + \mathcal{F} \times \mathcal{G}';$

# Derivative of Combinatorial Structures

(origin: species theory)

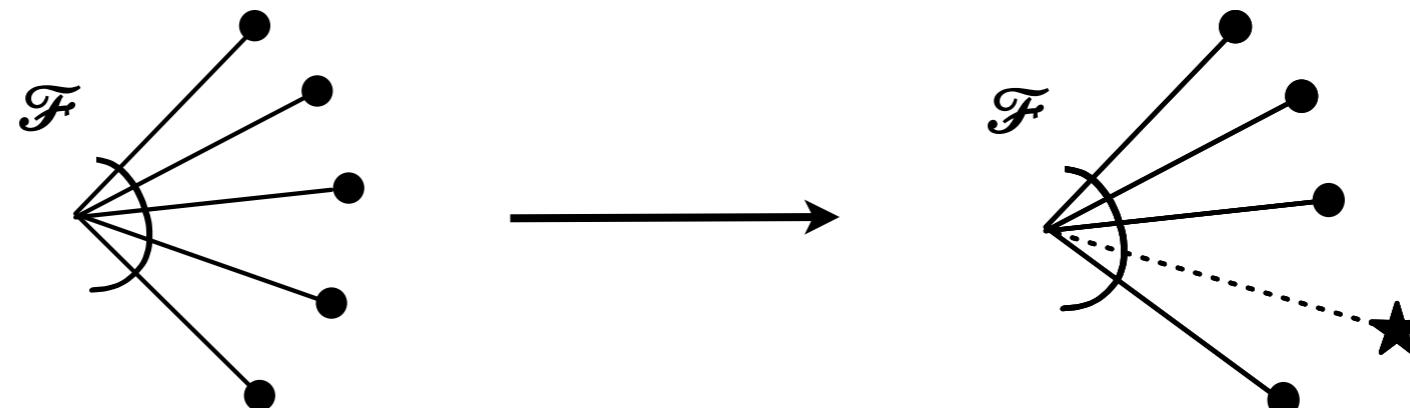
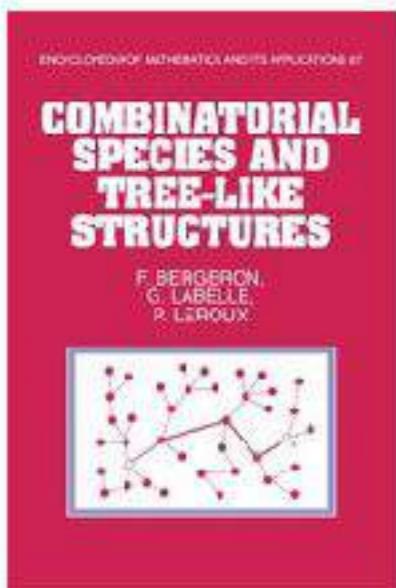


- $(\mathcal{F} + \mathcal{G})' = \mathcal{F}' + \mathcal{G}'; (\mathcal{F} \times \mathcal{G})' = \mathcal{F}' \times \mathcal{G} + \mathcal{F} \times \mathcal{G}';$
- $\mathcal{F}(\mathcal{G})' = \mathcal{F}'(\mathcal{G}) \times \mathcal{G}';$

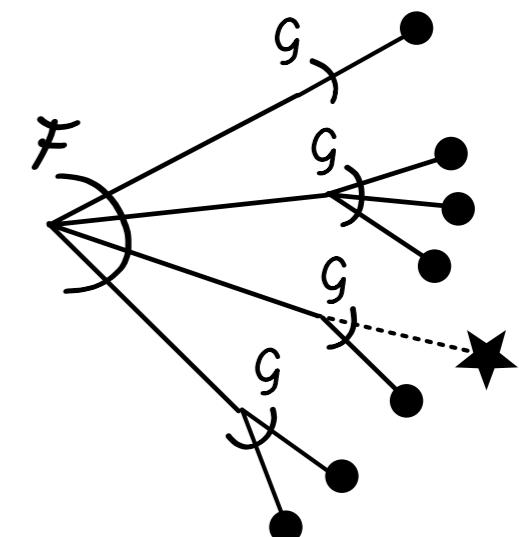


# Derivative of Combinatorial Structures

(origin: species theory)



- $(\mathcal{F} + \mathcal{G})' = \mathcal{F}' + \mathcal{G}'; (\mathcal{F} \times \mathcal{G})' = \mathcal{F}' \times \mathcal{G} + \mathcal{F} \times \mathcal{G}';$
- $\mathcal{F}(\mathcal{G})' = \mathcal{F}'(\mathcal{G}) \times \mathcal{G}';$
- $0' = 1' = 0; \mathcal{Z}' = 1;$
- $\text{SET}' = \text{SET}; \text{CYC}' = \text{SEQ}; \text{SEQ}' = \text{SEQ} \times \text{SEQ}.$



# Combinatorial Newton Iteration

$$\mathcal{Y} = 1 + \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y} =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$$

$$\mathcal{Y}_0 = \emptyset \quad \mathcal{Y}_1 = \circ$$

$$\mathcal{Y}_2 = \boxed{\circ \quad + \quad \begin{array}{c} \text{graph} \\ \text{graph} \end{array}} \quad + \quad \begin{array}{c} \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \end{array} \quad + \dots + \begin{array}{c} \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \end{array} \quad + \dots$$

$$\mathcal{Y}_3 = \boxed{\mathcal{Y}_2 + \begin{array}{c} \text{graph} \\ \text{graph} \end{array} + \dots + \begin{array}{c} \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \end{array} + \dots + \begin{array}{c} \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \\ \text{graph} \end{array}} \quad + \dots$$

# Combinatorial Newton Iteration

$$\mathcal{Y} = 1 + \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y} =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$$

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \text{SEQ}(\mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \star + \mathcal{Z} \cdot \star \cdot \mathcal{Y}^{[n]}) \cdot ((1 + \mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]}).$$

$$\mathcal{Y}_0 = \emptyset \quad \mathcal{Y}_1 = \circ$$

$$\mathcal{Y}_2 = \boxed{\circ \quad + \quad \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet - \circ \end{array} \quad + \quad \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \circ - \bullet \end{array}} + \dots + \dots$$

The diagram shows the second iteration of the combinatorial Newton iteration. It consists of a sum of terms. The first term is a single open circle. The second and third terms are rooted trees with two children, where the root is either a solid black dot or an open circle. Red lines connect the roots of these three terms to a dashed red line, which then connects to the fourth term and beyond.

$$\mathcal{Y}_3 = \boxed{\mathcal{Y}_2 + \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet - \bullet \end{array} + \dots + \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet - \bullet \end{array} + \dots + \dots}$$

The diagram shows the third iteration of the combinatorial Newton iteration. It consists of a sum of terms. The first term is  $\mathcal{Y}_2$ . The second term is a rooted tree with three children, where the root is a solid black dot. This is followed by a sequence of terms, each consisting of a solid black dot connected by a red line to a dashed red line, which then connects to the next term. The last term shown has four children at the root.

# Combinatorial Newton Iteration

$$\mathcal{Y} = 1 + \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y} =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$$

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \text{SEQ}(\mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \star + \mathcal{Z} \cdot \star \cdot \mathcal{Y}^{[n]}) \cdot ((1 + \mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]}).$$

$$\mathcal{Y}_0 = \emptyset \quad \mathcal{Y}_1 = \circ$$

$$\mathcal{Y}_2 = \boxed{\circ + \circ} + \dots + \dots + \dots + \dots$$

The diagram shows the construction of  $\mathcal{Y}_2$  from  $\mathcal{Y}_1$ . It starts with two empty nodes ( $\circ$ ). A red line connects them to form a single node with two children. This is followed by a plus sign, then a sequence of three nodes connected by red lines, each with two children. Another plus sign follows, then a sequence of four nodes connected by red lines, each with two children. This pattern continues with ellipses, followed by a plus sign, then a sequence of five nodes connected by red lines, each with two children.

$$\mathcal{Y}_3 = \boxed{\mathcal{Y}_2 + \circ + \dots + \circ} + \dots + \dots + \dots + \dots$$

The diagram shows the construction of  $\mathcal{Y}_3$  from  $\mathcal{Y}_2$ . It starts with  $\mathcal{Y}_2$ , followed by a plus sign, then a sequence of three nodes connected by red lines, each with two children. This is followed by a plus sign, then a sequence of four nodes connected by red lines, each with two children. Another plus sign follows, then a sequence of five nodes connected by red lines, each with two children. This pattern continues with ellipses, followed by a plus sign, then a sequence of six nodes connected by red lines, each with two children.

# Combinatorial Newton Iteration

$$\mathcal{Y} = 1 + \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y} =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$$

$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \text{SEQ}(\mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \star + \mathcal{Z} \cdot \star \cdot \mathcal{Y}^{[n]}) \cdot ((1 + \mathcal{Z} \cdot \mathcal{Y}^{[n]} \cdot \mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]}).$$

$$\mathcal{Y}_0 = \emptyset \quad \mathcal{Y}_1 = \circ$$

$$\mathcal{Y}_2 = \boxed{\circ + \circ} + \dots + \dots + \dots + \dots$$

Diagram illustrating the construction of  $\mathcal{Y}_2$  from  $\mathcal{Y}_1$ . The first term is a single node (circle). The second term is a node connected to two other nodes. Subsequent terms show more complex structures, such as nodes connected to three or four other nodes, and some with red edges. A blue box labeled 2 encloses the first two terms.

$$\mathcal{Y}_3 = \boxed{\mathcal{Y}_2 + \circ + \dots + \circ} + \dots + \dots + \dots + \dots$$

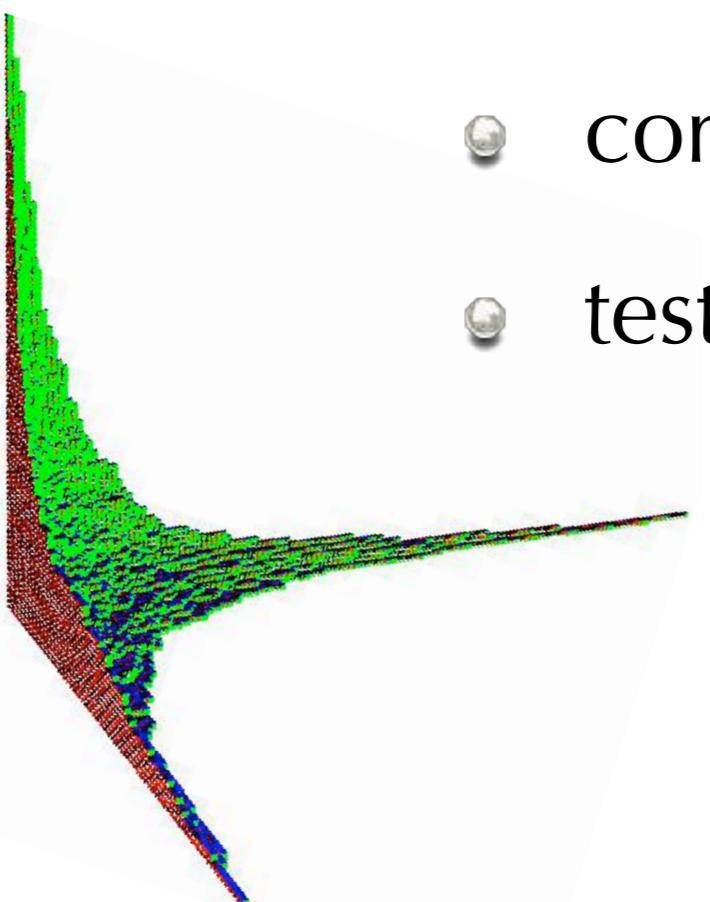
Diagram illustrating the construction of  $\mathcal{Y}_3$  from  $\mathcal{Y}_2$ . The first term is  $\mathcal{Y}_2$  plus a single node. Subsequent terms show more complex structures, such as nodes connected to three or four other nodes, and some with red edges. A blue box labeled 6 encloses the first six terms.

**Ccl:** numerical convergence of the  
Newton iteration starting from 0

### **III. Random Generation**

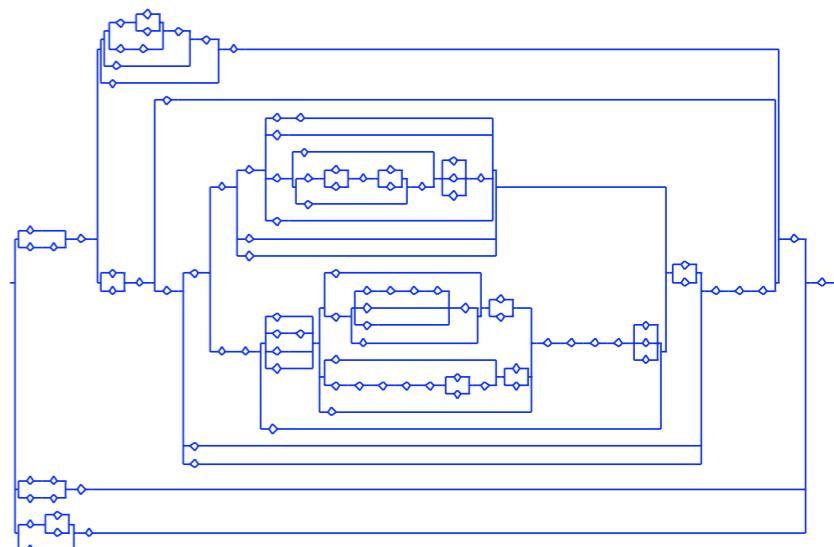


# Why Random Generation?



Simulation in the discrete world; helps

- evaluate parameters;
- compare/refine models;
- test software.



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  
$$\frac{xT'(x)}{T(x)}$$

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H} = 1$  return 1;
    if  $\mathcal{H} = \mathcal{E}$  return  $\mathcal{E}$ ;
    if  $\mathcal{H} = \mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H} = \mathcal{F} + \mathcal{G}$  {
        b := Bernoulli( $F(x)/H(x)$ );
        if  $b=1$  return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H} = \text{Set}(\mathcal{F})$  {
        ...
    }
}
```

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:

$$\frac{xT'(x)}{T(x)}$$

```
Generate(h∈H) = {  
    if H=1 return 1;  
    if H=F return F;  
    if H=F×G {  
        Generate(f∈F);  
        Generate(g∈G);  
        return (f,g); }  
    if H=F+G {  
        b:=Bernoulli(F(x)/H(x));  
        if b=1 return Generate(f∈F);  
        else return Generate(g∈G);  
    if H=Set(F) {  
        ...
```

$F(x), H(x)$  by Newton iteration

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

$$\text{with } 1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$$

Coin: 0,0,1,0,1,1,1,1,0,1,...

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...

# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, ...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, ...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

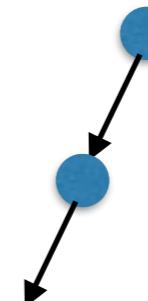
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, ...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

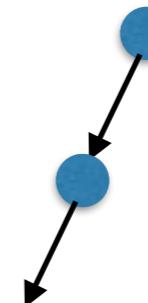
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,**1**,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

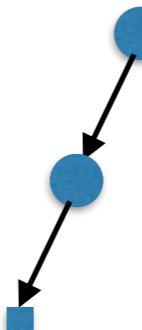
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,**1**,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

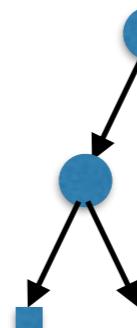
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

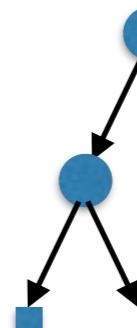
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

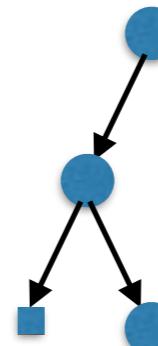
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

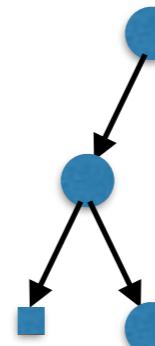
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,**1**,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

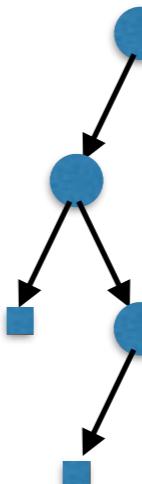
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,**1**,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

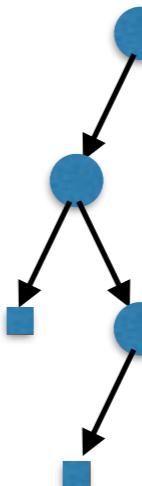
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

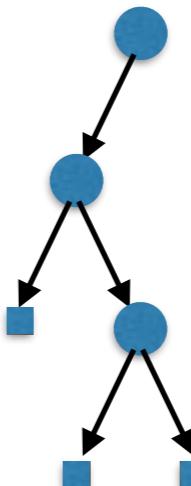
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{F} \times \mathcal{B} \times \mathcal{B}$$

with  $1/B(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

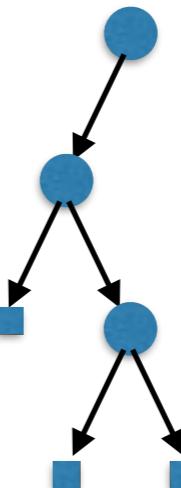
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

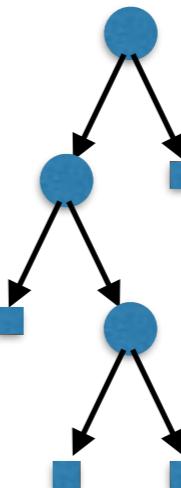
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B}=1+\mathcal{E} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...



# Boltzmann Sampling

$$\text{Proba}(t) = \frac{x^{|t|}}{T(x)} \quad \text{with} \quad T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_n T_n x^n$$

*x is a parameter of the algorithm*

Expected size:  

$$\frac{xT'(x)}{T(x)}$$

```
Generate( $h \in \mathcal{H}$ ) = {
    if  $\mathcal{H}=1$  return 1;
    if  $\mathcal{H}=\mathcal{F}$  return  $\mathcal{F}$ ;
    if  $\mathcal{H}=\mathcal{F} \times \mathcal{G}$  {
        Generate( $f \in \mathcal{F}$ );
        Generate( $g \in \mathcal{G}$ );
        return ( $f, g$ );
    }
    if  $\mathcal{H}=\mathcal{F} + \mathcal{G}$  {
        b:=Bernoulli( $F(x)/H(x)$ );
        if b=1 return Generate( $f \in \mathcal{F}$ );
        else return Generate( $g \in \mathcal{G}$ );
    }
    if  $\mathcal{H}=\text{Set}(\mathcal{F})$  {
        ...
    }
}
```

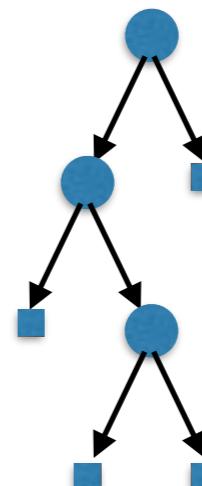
$F(x), H(x)$  by Newton iteration

**Example:** binary trees

$$\mathcal{B} = 1 + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$$

with  $1/\mathcal{B}(.23) \approx 1/1.559 \approx .6414$

Coin: 0,0,1,0,1,1,1,1,0,1,...

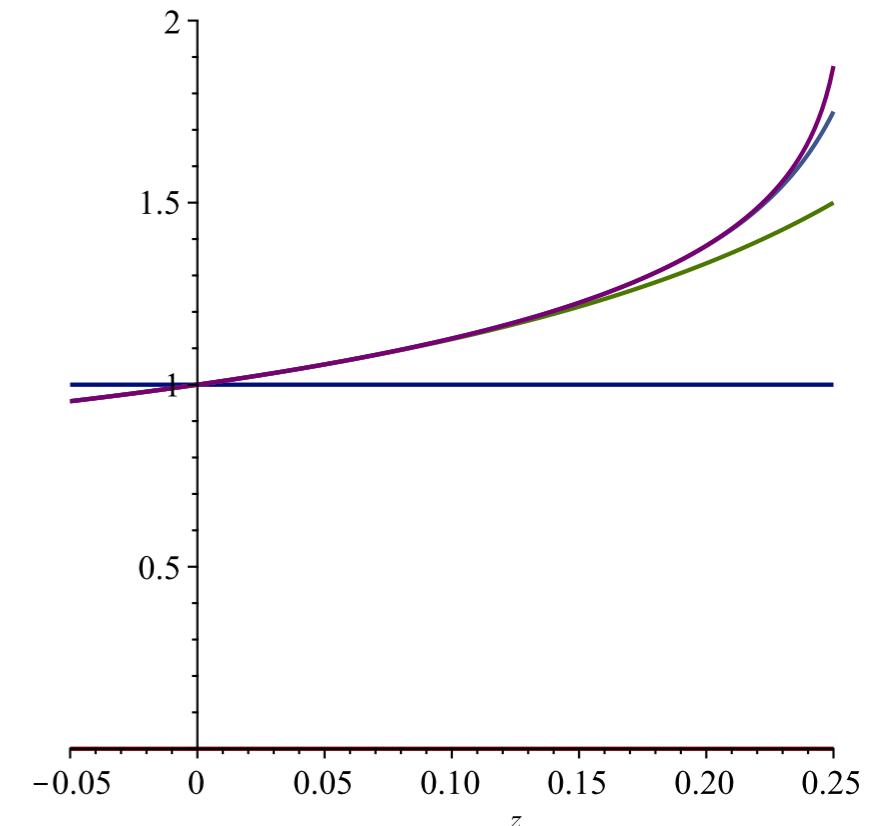


Exp. size  

$$\sim \frac{1}{2\sqrt{1 - 4x}}$$

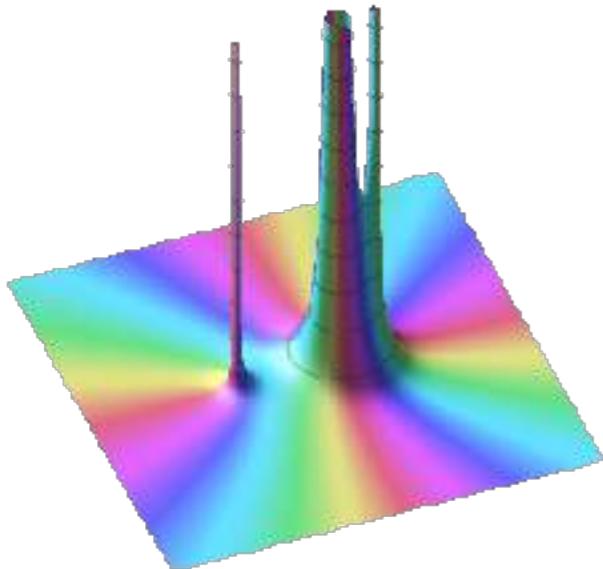
# Example: XML-trees

Grammar	File size	#eq.	Newton
<b>rss</b>	9.5k	16	0.02s.
<b>Relax NG</b>	124k	114	0.10s.
<b>XSLT</b>	168k	122	0.12s.
<b>XHTML Basic</b>	284k	96	0.32s.
<b>SVG</b>	6.3M	232	0.23s.
<b>OpenDocument</b>	2.8M	814	0.34s.
<b>DocBook</b>	11M	977	23s.



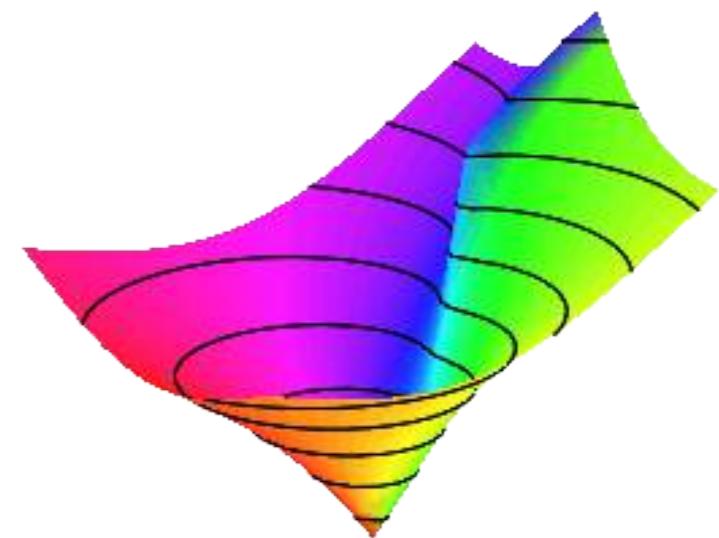
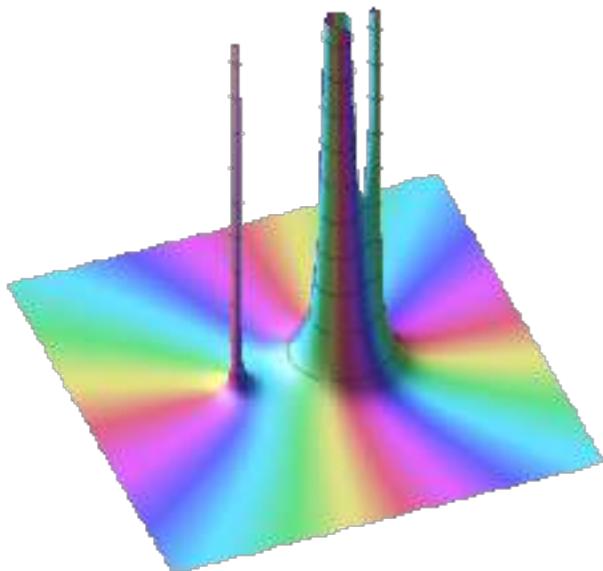
Time for  $x$  s.t.  $E(\text{size})=10,000$

## IV. Asymptotic Analysis



$$F(z) = \sum_{n=0}^{\infty} f_n z^n \longrightarrow f_n \sim \dots, \quad n \rightarrow \infty$$

## IV. Asymptotic Analysis



$$F(z) = \sum_{n=0}^{\infty} f_n z^n \longrightarrow f_n \sim \dots, \quad n \rightarrow \infty$$

# Singularity Analysis

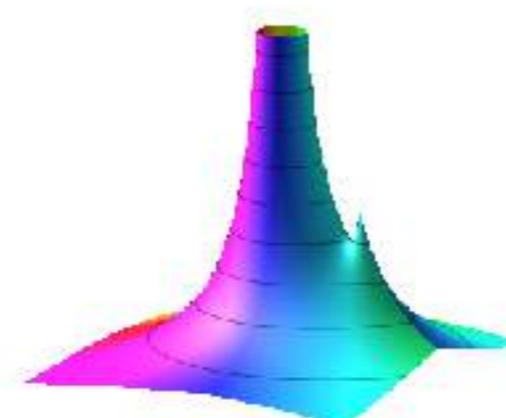
counts the number  
of objects of size  $n$

$$(a_n) \mapsto A(z) :=$$

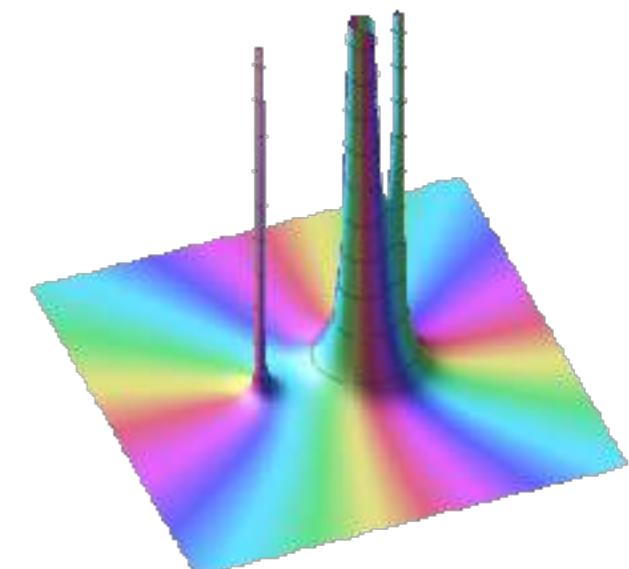
$$\sum_{n \geq 0} a_n z^n$$

captures some  
structure

A 3-Step Method:



$$a_n = \frac{1}{2\pi i} \oint \frac{A(z)}{z^{n+1}} dz$$



# Singularity Analysis

counts the number  
of objects of size  $n$

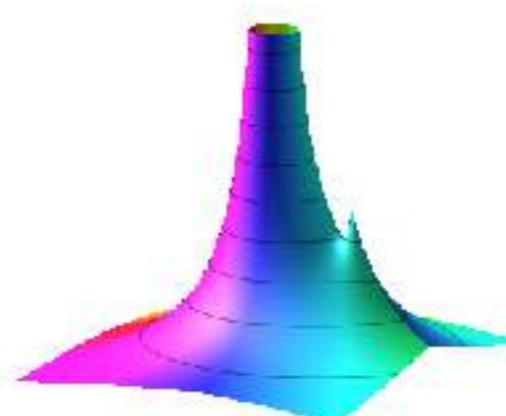
$$(a_n) \mapsto A(z) :=$$

$$\sum_{n \geq 0} a_n z^n$$

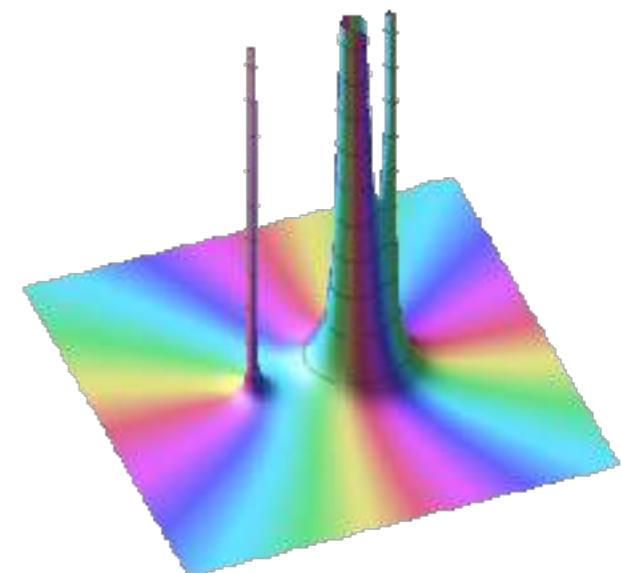
captures some  
structure

## A 3-Step Method:

1. Locate dominant singularities
2. Compute local behaviour
3. Translate into asymptotics



$$a_n = \frac{1}{2\pi i} \oint \frac{A(z)}{z^{n+1}} dz$$



# Singularity Analysis

counts the number  
of objects of size  $n$

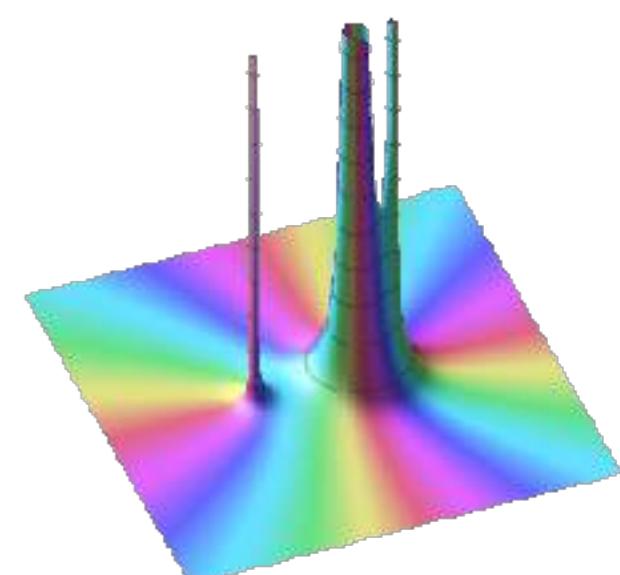
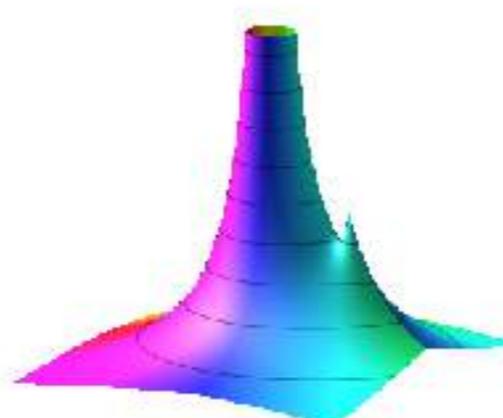
$$(a_n) \mapsto A(z) :=$$

$$\sum_{n \geq 0} a_n z^n$$

captures some  
structure

A 3-Step Method:

1. Locate dominant singularities
2. Compute local behaviour
3. Translate into asymptotics



$$A(z) \underset{z \rightarrow \rho}{\sim} c \left(1 - \frac{z}{\rho}\right)^{\alpha} \log^m \frac{1}{1 - \frac{z}{\rho}}$$

$$a_n \underset{n \rightarrow \infty}{\sim} c \rho^{-n} \frac{n^{-\alpha-1}}{\Gamma(-\alpha)} \log^m n \quad (\alpha \notin \mathbb{N})$$

# Singularity Analysis

counts the number  
of objects of size  $n$

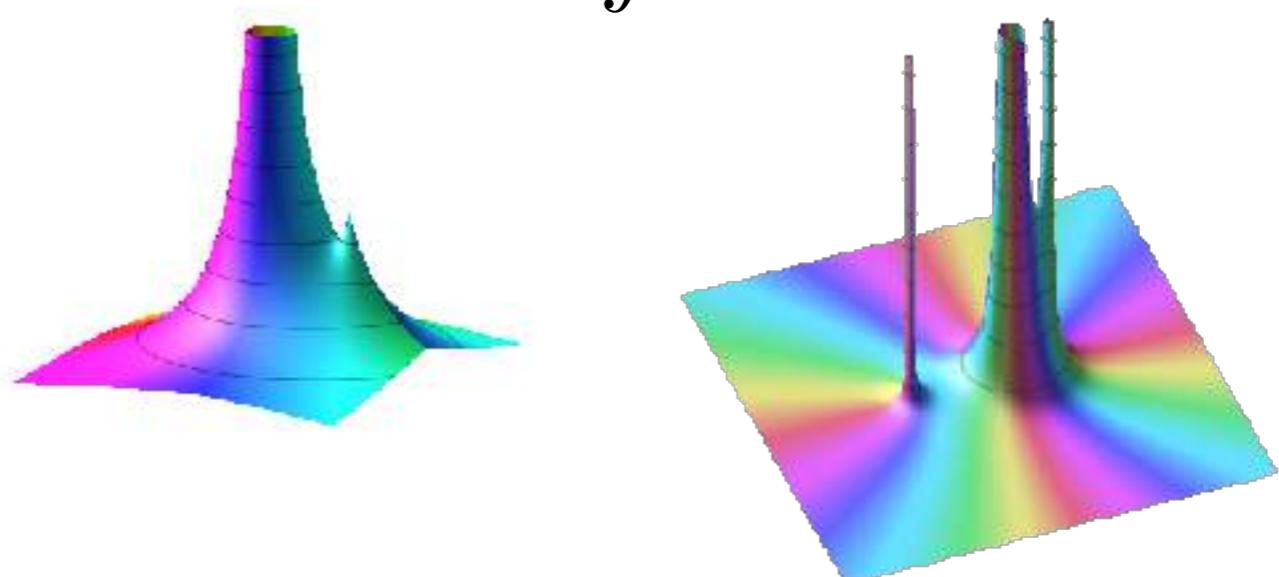
$$(a_n) \mapsto A(z) :=$$

$$\sum_{n \geq 0} a_n z^n$$

captures some  
structure

## A 3-Step Method:

1. Locate dominant singularities
2. Compute local behaviour
3. Translate into asymptotics

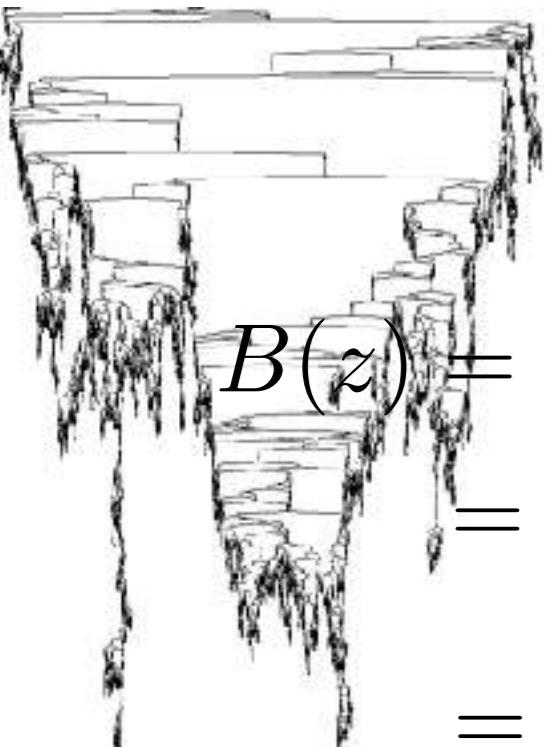


$$A(z) \underset{z \rightarrow \rho}{\sim} c \left(1 - \frac{z}{\rho}\right)^{\alpha} \log^m \frac{1}{1 - \frac{z}{\rho}}$$

$$a_n \underset{n \rightarrow \infty}{\sim} c \rho^{-n} \frac{n^{-\alpha-1}}{\Gamma(-\alpha)} \log^m n \quad (\alpha \notin \mathbb{N})$$

full asymptotic expansion available

# Example: Binary Trees (Again!)


$$\begin{aligned}B(z) &= 1 + zB(z)^2 \\&= 1 + z + 2z^2 + 5z^3 + \dots \\&= \frac{1 - \sqrt{1 - 4z}}{2z}\end{aligned}$$

A 3-Step Method:

1. Locate dominant singularities;
2. Compute local behaviour;
3. Translate into asymptotics.

1. sing. en  $z = 1/4$

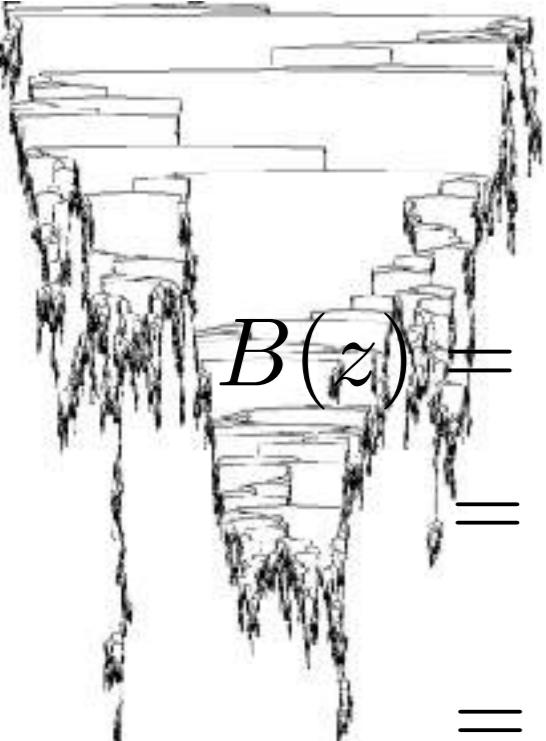
2. local behaviour:

$$B(z) \underset{z \rightarrow 1/4}{=} 2 - 2(1 - 4z)^{1/2} + \dots$$

3. translation:

$$B_n \sim \frac{4^n n^{-3/2}}{\sqrt{\pi}} \left( 1 - \frac{9}{8n} + \dots \right)$$

# Example: Binary Trees (Again!)



$$\begin{aligned}
 B(z) &= 1 + zB(z)^2 \\
 &= 1 + z + 2z^2 + 5z^3 + \dots \\
 &= \frac{1 - \sqrt{1 - 4z}}{2z}
 \end{aligned}$$

1. sing. en  $z = 1/4$
2. local behaviour:

$$B(z) \underset{z \rightarrow 1/4}{=} 2 - 2(1 - 4z)^{1/2} + \dots$$

3. translation:

$$B_n \sim \frac{4^n n^{-3/2}}{\sqrt{\pi}} \left( 1 - \frac{9}{8n} + \dots \right)$$

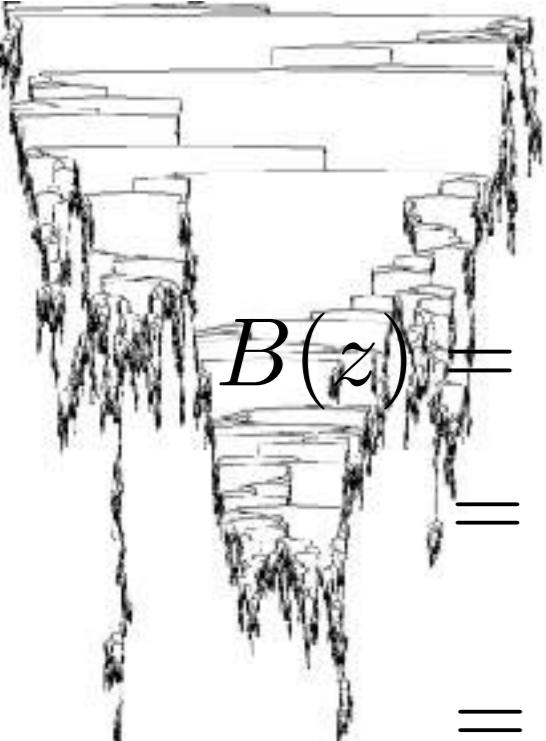
A 3-Step Method:

1. Locate dominant singularities;
2. Compute local behaviour;
3. Translate into asymptotics.

Probability that a leaf  
is a child of the root?

$$\begin{aligned}
 &\frac{[z^n]2zB(z)}{B_n} \\
 &= \frac{1}{2} + \frac{3}{4n} + O\left(\frac{1}{n^2}\right)
 \end{aligned}$$

# Example: Binary Trees (Again!)



$$\begin{aligned}
 B(z) &= 1 + zB(z)^2 \\
 &= 1 + z + 2z^2 + 5z^3 + \dots \\
 &= \frac{1 - \sqrt{1 - 4z}}{2z}
 \end{aligned}$$

1. sing. en  $z = 1/4$
2. local behaviour:

$$B(z) \underset{z \rightarrow 1/4}{=} 2 - 2(1 - 4z)^{1/2} + \dots$$

3. translation:

$$B_n \sim \frac{4^n n^{-3/2}}{\sqrt{\pi}} \left( 1 - \frac{9}{8n} + \dots \right)$$

A 3-Step Method:

1. Locate dominant singularities;
2. Compute local behaviour;
3. Translate into asymptotics.

Probability that a leaf  
is a child of the root?

$$\begin{aligned}
 &\frac{[z^n]2zB(z)}{B_n} \\
 &= \frac{1}{2} + \frac{3}{4n} + O\left(\frac{1}{n^2}\right)
 \end{aligned}$$

Demo.

# Parameters



# Parameters

- Equations over combinatorial structures + **parameters**

*Ex.: path length in  
binary trees*



# Parameters

- Equations over combinatorial structures + **parameters**
- **Multivariate** generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

*Ex.: path length in  
binary trees*

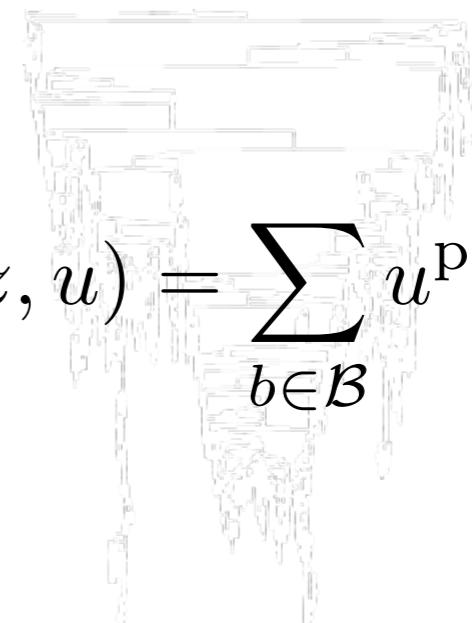


# Parameters

- Equations over combinatorial structures + **parameters**
- **Multivariate** generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

*Ex.: path length in binary trees*



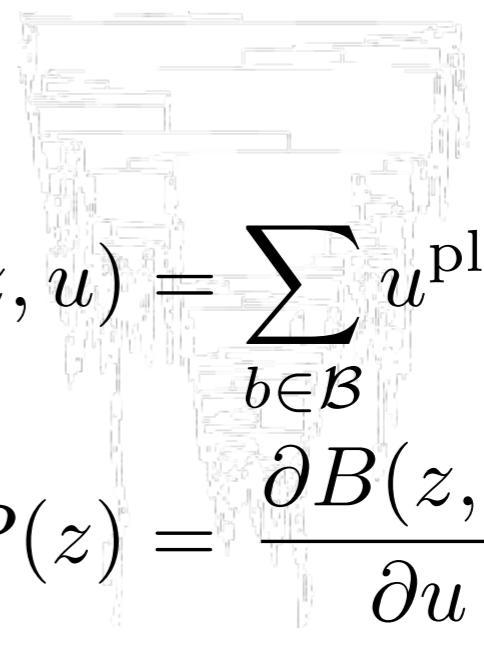
$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$

# Parameters

- Equations over combinatorial structures + **parameters**
- **Multivariate** generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

*Ex.: path length in binary trees*


$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$
$$P(z) = \left. \frac{\partial B(z, u)}{\partial u} \right|_{u=1}$$

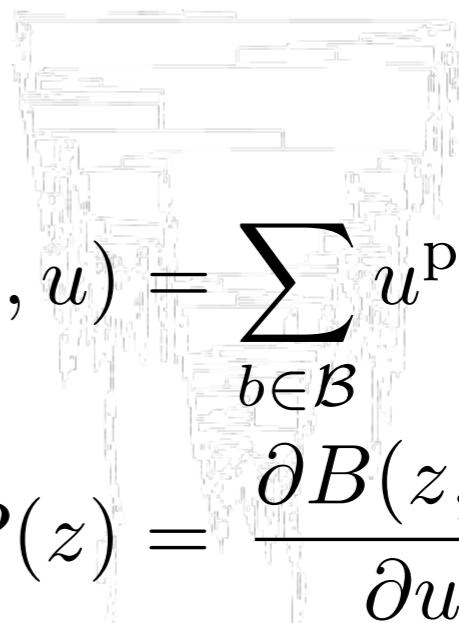
# Parameters

- Equations over combinatorial structures + **parameters**
- **Multivariate** generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

- Complex analysis  
 $f_{n,k} \sim \dots, n \rightarrow \infty$

*Ex.: path length in binary trees*



$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$
$$P(z) = \left. \frac{\partial B(z, u)}{\partial u} \right|_{u=1}$$

# Parameters

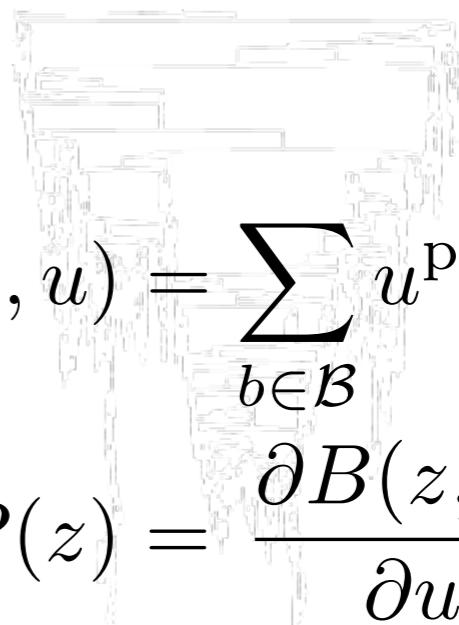
- Equations over combinatorial structures + **parameters**
- **Multivariate** generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

- Complex analysis

$$f_{n,k} \sim \dots, n \rightarrow \infty$$

*Ex.: path length in binary trees*



$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$
$$P(z) = \left. \frac{\partial B(z, u)}{\partial u} \right|_{u=1}$$

$$\frac{P_n}{n B_n} \sim \sqrt{\pi n}$$

# Parameters

- Equations over combinatorial structures + **parameters**

- **Multivariate generating series**

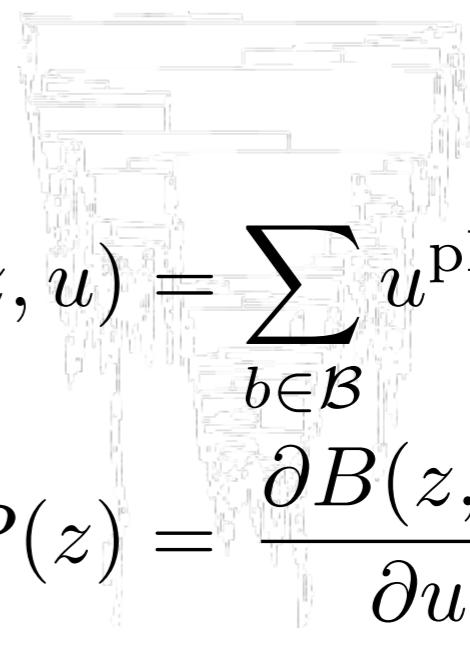
$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

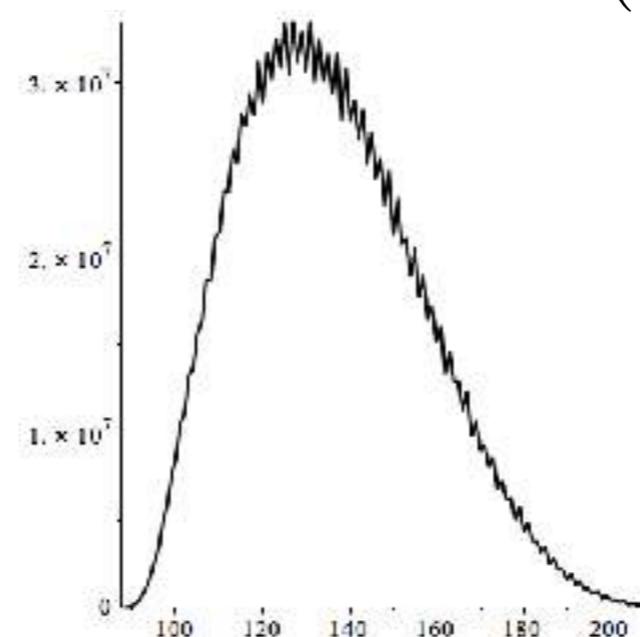
- Complex analysis

$$f_{n,k} \sim \dots, n \rightarrow \infty$$

- Limiting distribution

*Ex.: path length in binary trees*


$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$
$$P(z) = \left. \frac{\partial B(z, u)}{\partial u} \right|_{u=1}$$



$$\frac{P_n}{n B_n} \sim \sqrt{\pi n}$$

# Parameters

- Equations over combinatorial structures + parameters

- Multivariate generating series

$$F(z, u) = \sum_{n,k} f_{n,k} u^k z^n$$

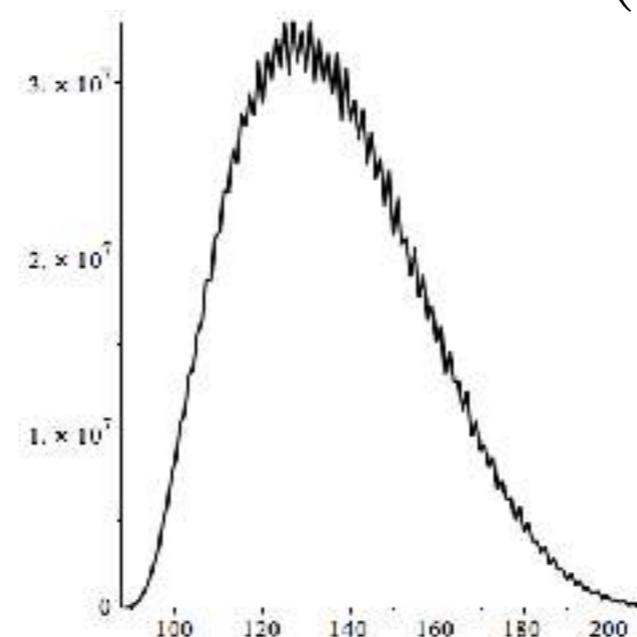
- Complex analysis

$$f_{n,k} \sim \dots, n \rightarrow \infty$$

- Limiting distribution

*Ex.: path length in binary trees*

$$B(z, u) = \sum_{b \in \mathcal{B}} u^{\text{pl}(b)} z^{|b|}$$
$$P(z) = \left. \frac{\partial B(z, u)}{\partial u} \right|_{u=1}$$



$$\frac{P_n}{n B_n} \sim \sqrt{\pi n}$$

Demo.

# If you can specify it, you can analyze it!

Permutations

Mappings

Words

Strings

Urns

Trees

Languages

Integers

Compositions

Partitions

...

+ *parameters*

Universality phenomena:

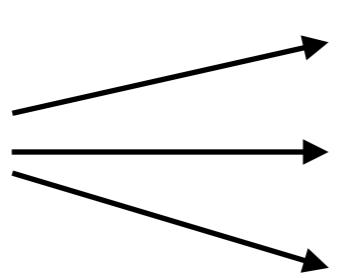
Ex.: # trees of various types  $\longrightarrow K \rho^n n^{-3/2} \longrightarrow$  pathlength in  $n^{1/2}$

# Summary & Conclusion

# Summary & Conclusion

*What we have:*

Combinatorial specification



- counting;
- random generation;
- asymptotic analysis.

Automated?

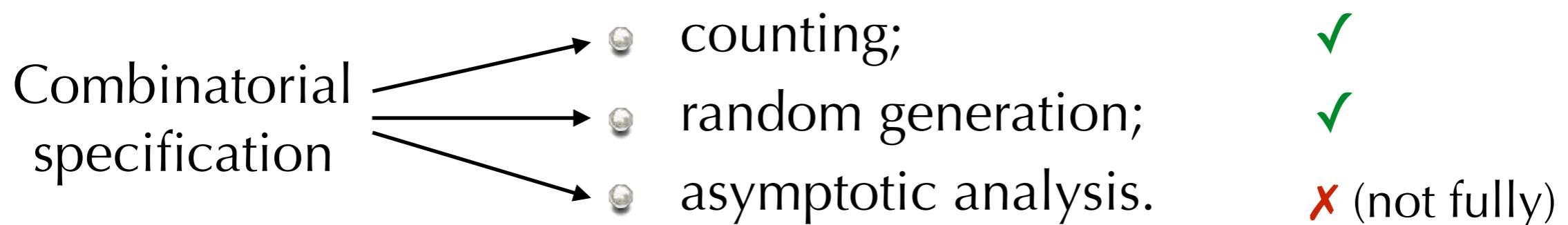
✓

✓

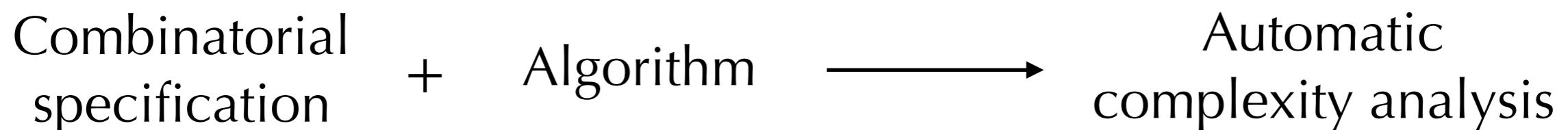
✗ (not fully)

# Summary & Conclusion

*What we have:*

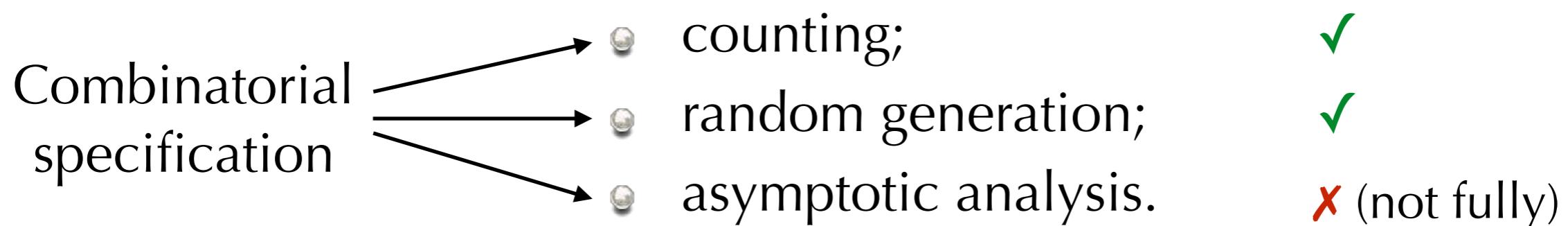


*Ultimate (dream) goal:*

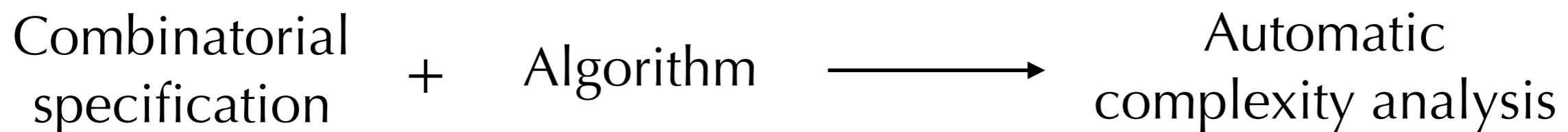


# Summary & Conclusion

*What we have:*

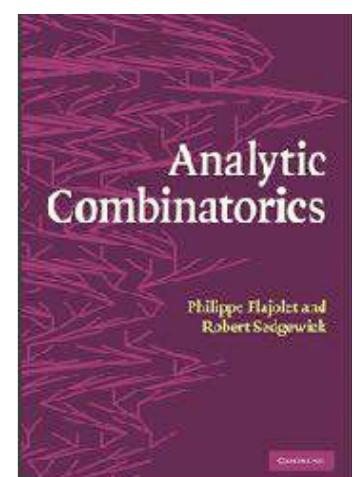


*Ultimate (dream) goal:*



*Where to learn more:*

On-line course by R. Sedgewick at  
<http://ac.cs.princeton.edu/online/>



# Summary & Conclusion

*What we have:*

Combinatorial specification	Automated?
counting;	✓
random generation;	✓
asymptotic analysis.	✗ (not fully)



*Where to learn more:*

On-line course by R. Sedgewick at  
<http://ac.cs.princeton.edu/online/>

