Newton Iteration, Computer Algebra and Combinatorics

Bruno Salvy
@ Inria & ENS-Lyon

1. Introduction

Philippe Flajolet's many contributions

Planned complete works span 7 volumes of approx 600 pp. each (M. Ward editor)

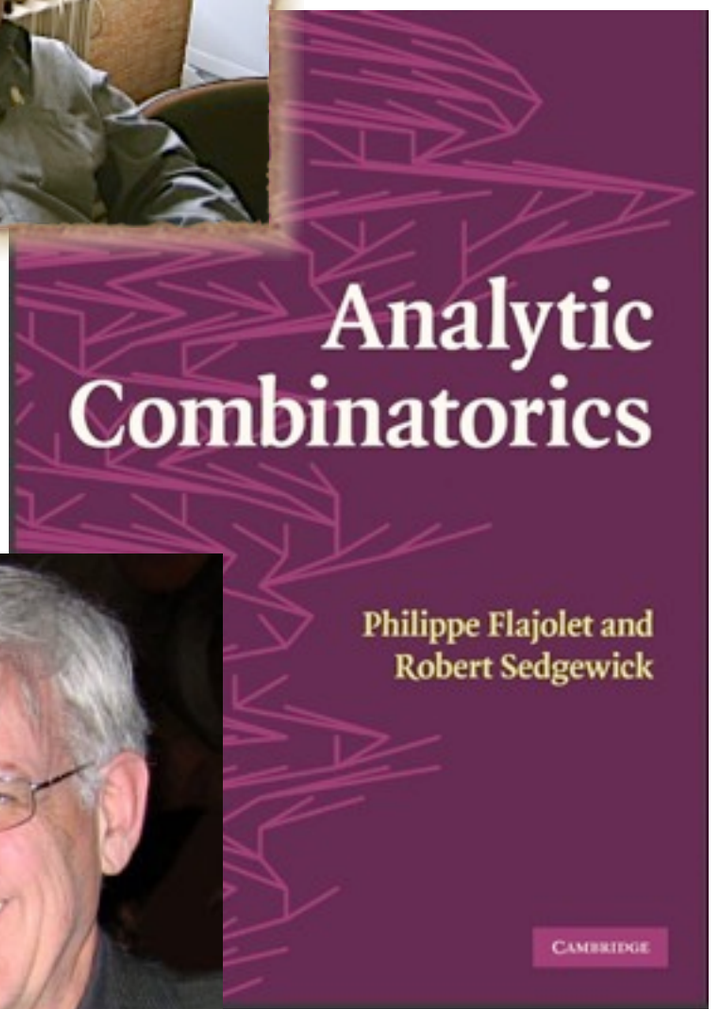
1. Analytic combinatorics
2. Limit laws and dynamical systems
3. Text, information theory and the Mellin transform
4. Trees & graphs
5. Combinatorial Structures
6. Effective methods
7. French texts

Philippe Flajolet's many contributions

Planned complete works span 7 volumes of approx 600 pp. each (M. Ward editor)

1. Analytic combinatorics
2. Limit laws and dynamical systems
3. Text, information theory and the Mellin transform
4. Trees & graphs
5. Combinatorial Structures
6. Effective methods
7. French texts

Analytic Combinatorics



From a ***specification*** to

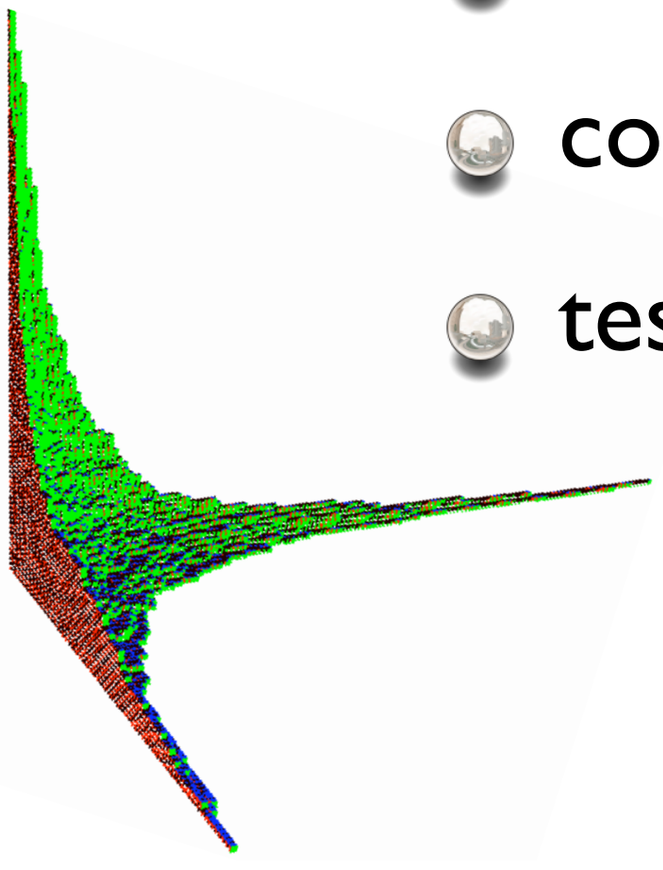
- an analysis (➥ next talk);
- enumeration;
- random generation.

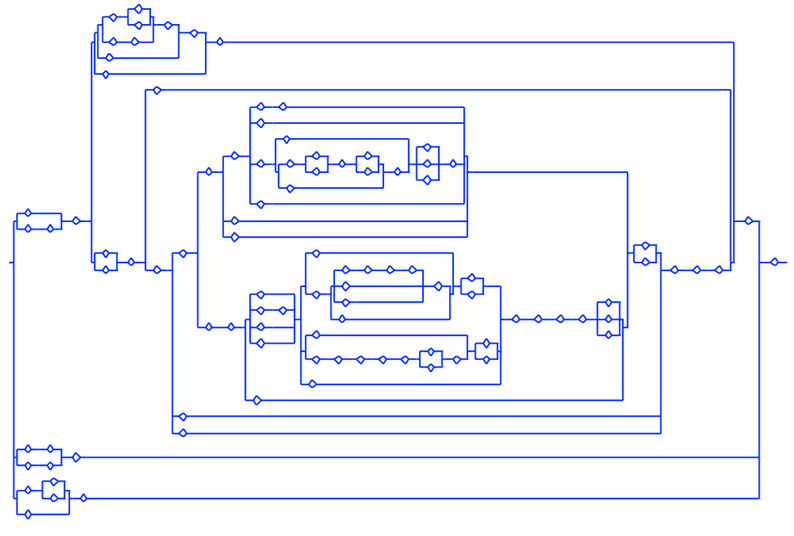
2009

Random Generation

Simulation in the discrete world; helps

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





Combinatorial specifications

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

Combinatorial specifications

Language: 1, $\mathscr{Z}$, +, ×, SEQ, SET, CYC and recursion.

- Binary trees: $\mathscr{B} = \mathscr{Z} + \mathscr{Z} \times \mathscr{B} \times \mathscr{B}$

Combinatorial specifications

Language: 1, $\mathscr{Z}$, +, ×, SEQ, SET, CYC and recursion.

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

Combinatorial specifications

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

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

- Permutations: Perm=SET(CYC($\mathcal{Z}$));
- Trees: $\mathcal{T}=\mathcal{Z}\cdot$SET($\mathcal{T}(\mathcal{Z})$);

Combinatorial specifications

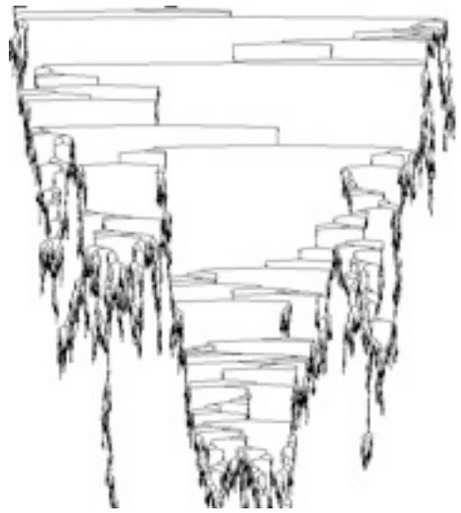
Language: 1, $\mathcal{Z}$, +, ×, SEQ, SET, CYC and recursion.

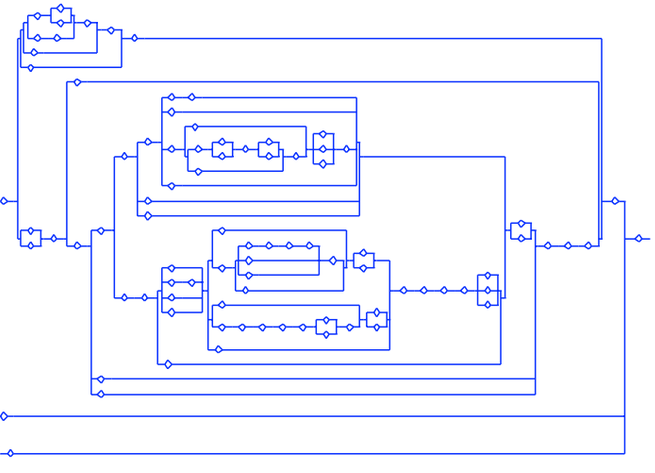
- Binary trees: $\mathcal{B}=\mathcal{Z}+\mathcal{Z}\times\mathcal{B}\times\mathcal{B}$

- Permutations: Perm=SET(CYC($\mathcal{Z}$));
- Trees: $\mathcal{T}=\mathcal{Z}\cdot$SET($\mathcal{T}(\mathcal{Z})$);
- Functional graphs: $\mathcal{F}$=SET(CYC($\mathcal{T}(\mathcal{Z})$));

Combinatorial specifications

Language: 1, $\mathscr{Z}$, +, ×, SEQ, SET, CYC and recursion.



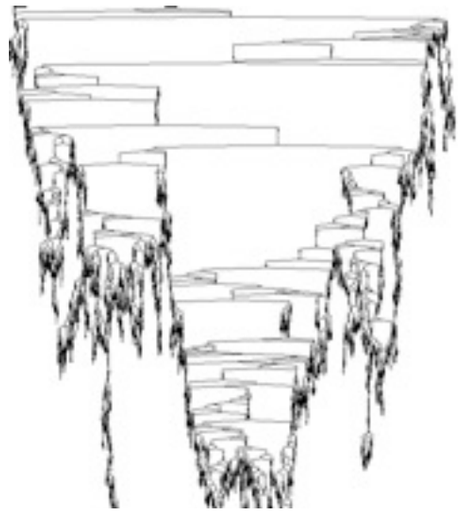


- Binary trees: $\mathscr{B}=\mathscr{Z}+\mathscr{Z}\times\mathscr{B}\times\mathscr{B}$

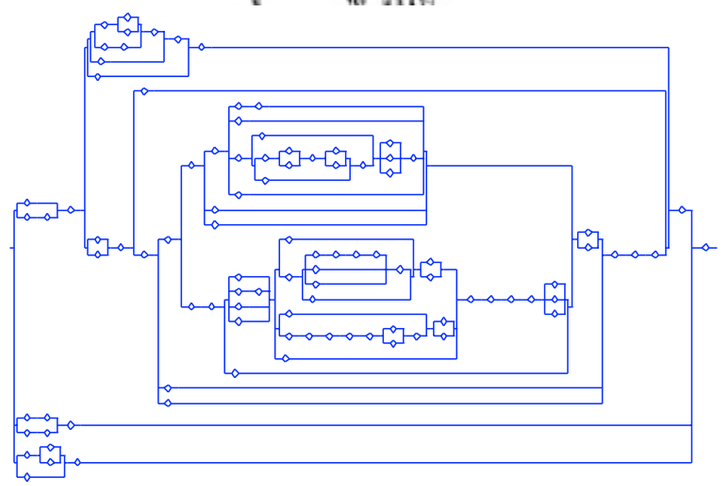
- Permutations: Perm=SET(CYC($\mathscr{Z}$));
- Trees: $\mathscr{T}=\mathscr{Z}\cdot$SET($\mathscr{T}(\mathscr{Z})$);
- Functional graphs: $\mathscr{F}$=SET(CYC($\mathscr{T}(\mathscr{Z})$));
- Series-parallel graphs:
  $\mathscr{G}=\mathscr{Z}+\mathscr{S}+\mathscr{P}$, $\mathscr{S}$=SEQ$_{>0}$($\mathscr{Z}+\mathscr{P}$),$\mathscr{P}$=SET$_{>0}$($\mathscr{Z}+\mathscr{S}$);

Combinatorial specifications

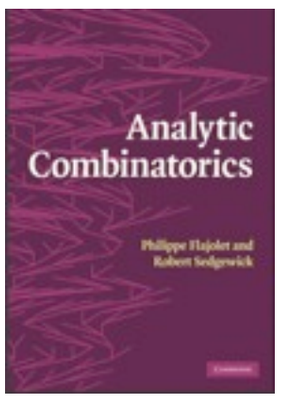
Language: 1, $\mathcal{Z}$, +, ×, SEQ, SET, CYC and recursion.



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



- Permutations: Perm=SET(CYC($\mathcal{Z}$));
- Trees: $\mathcal{T}=\mathcal{Z}\cdot$SET($\mathcal{T}(\mathcal{Z})$);
- Functional graphs: $\mathcal{F}$=SET(CYC($\mathcal{T}(\mathcal{Z})$));
- Series-parallel graphs:
  $\mathcal{G}=\mathcal{Z}+\mathcal{S}+\mathcal{P}$, $\mathcal{S}$=SEQ$_{>0}$($\mathcal{Z}+\mathcal{P}$),$\mathcal{P}$=SET$_{>0}$($\mathcal{Z}+\mathcal{S}$);
- ...hundreds of examples in "the purple book".



Recursive Method

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

```
DrawBinTree(n) = {
if n=1 return Z;
U:=Uniform([0,1]); k:=0; S:=0;
while (S<U) {k:=k+1;S:=S+b_k b_{n-k-1}/b_n;}
return ZxDrawBinTree(k)xDrawBinTree(n-k-1); }
```

b_k: nb binary trees with k nodes (Catalan)

Requires $b_0, b_1, \ldots, b_n$.

[Nijenhuis & Wilf; Flajolet, Zimmermann & Van Cutsem]

Boltzmann Sampling

Principle: Generate each object with a probability depending only on its size.

- The size is not a parameter of the algo.
- Same size → same proba. (as before).

[Duchon, Flajolet, Louchard, Schaeffer]

Boltzmann Sampling

Principle: Generate each object with a probability depending only on its size.

- The size is not a parameter of the algo.
- Same size → same proba. (as before).

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

[Duchon, Flajolet, Louchard, Schaeffer]

Boltzmann Sampling

Principle: Generate each object with a probability depending only on its size.

- The size is not a parameter of the algo.
- Same size → same proba. (as before).

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

Def. *Ordinary generating series* of $\mathcal{T}$:

$$T(x) = \sum_{t \in \mathcal{T}} x^{|t|} = \sum_{n \in \mathbb{N}} t_n x^n$$

[Duchon, Flajolet, Louchard, Schaeffer]

Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.

Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product:

$\mathscr{H} = \mathscr{F} \times \mathscr{G}$

```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H} = \mathscr{F} \times \mathscr{G}$

```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H} = \mathscr{F} + \mathscr{G}$

Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

```
Generate f∈𝓕;
Generate g∈𝓖;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈𝓕
else Generate g∈𝓖;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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

Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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

Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

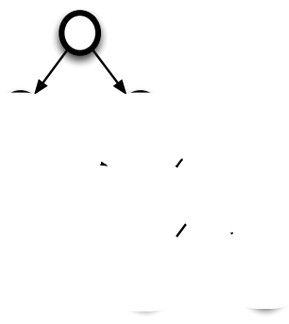
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

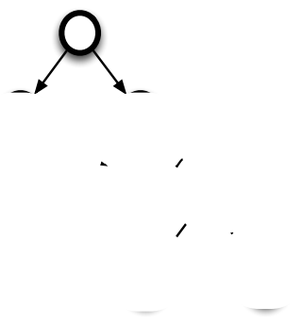
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

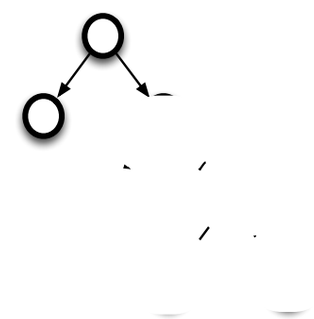
```
Generate f∈𝓕;
Generate g∈𝓖;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈𝓕
else Generate g∈𝓖;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x)=\sum_{t\in\mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

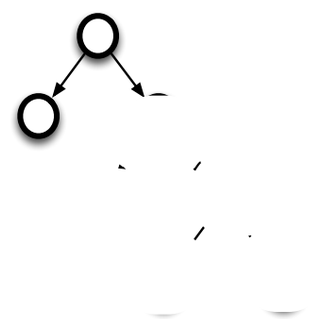
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B}=\mathcal{Z}+\mathcal{Z}\times\mathcal{B}\times\mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H} = \mathscr{F} \times \mathscr{G}$

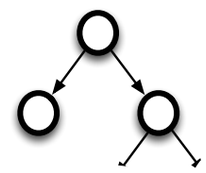
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H} = \mathscr{F} + \mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x)=\sum_{t\in\mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

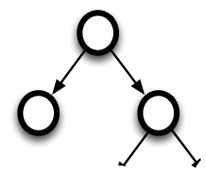
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B}=\mathcal{Z}+\mathcal{Z}\times\mathcal{B}\times\mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x)=\sum_{t\in\mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

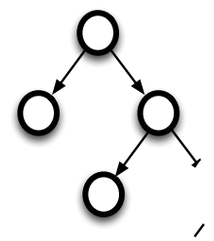
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B}=\mathcal{Z}+\mathcal{Z}\times\mathcal{B}\times\mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

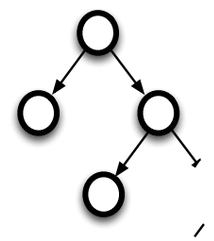
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

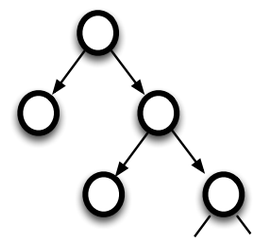
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x)=\sum_{t\in\mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

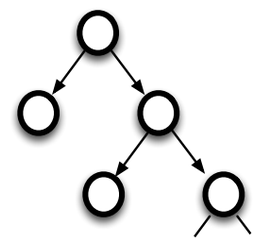
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B}=\mathcal{Z}+\mathcal{Z}\times\mathcal{B}\times\mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H} = \mathscr{F} \times \mathscr{G}$

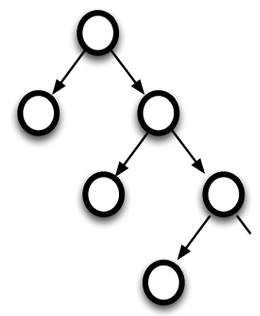
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H} = \mathscr{F} + \mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

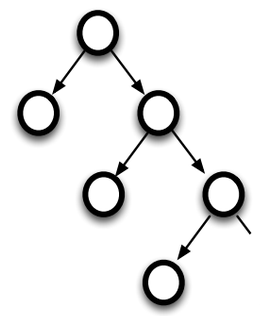
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\frac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

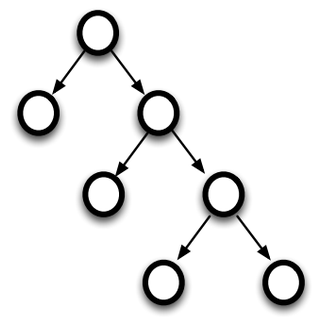
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Boltzmann: Algorithm

Proba : $\dfrac{x^{|t|}}{T(x)}$ with $T(x) = \sum_{t \in \mathcal{T}} x^{|t|}$

- Singleton: easy.
- Cartesian product: $\mathscr{H}=\mathscr{F}\times\mathscr{G}$

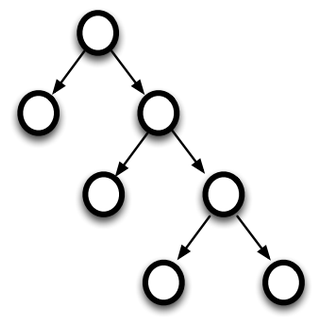
```
Generate f∈ℱ;
Generate g∈𝒢;
return (f,g);
```

- Disjoint union $\mathscr{H}=\mathscr{F}+\mathscr{G}$

```
b=Bernoulli(F(x)/H(x));
if b=1 then Generate f∈ℱ
else Generate g∈𝒢;
```

Example: binary tree with F(.49)/H(.49)≃.5995 $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

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



Complexity linear in |t|.

Generalization

Both methods extend to all "specifiable" structures.

They require

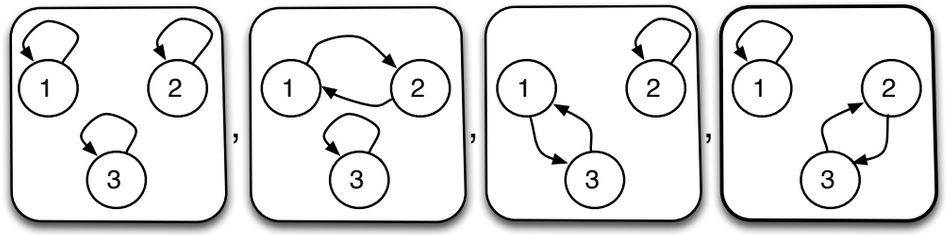
- *enumeration sequences* (recursive method);
- *numerical evaluation* of the g.f. (Boltzmann).

Plan:

2. Generating series
3. Newton iteration in computer algebra
4. Newton iteration in combinatorics

II. Generating Series

Two kinds of Generating Series

$$\mathrm{Inv}[\{1,2,3\}] = \Big\{$$  $$\Big\}$$

4 involutions;

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

Exponential generating series:

$$F(z) = \sum_{n=0}^{\infty} f_n \frac{z^n}{n!}, \quad f_n = \text{nb. labelled structs of size } n. \qquad \mathrm{Inv}_3(z) = \frac{2}{3} z^3$$

Ordinary generating series:

$$\tilde{F}(z) = \sum_{n=0}^{\infty} \tilde{f}_n z^n, \quad \tilde{f}_n = \text{nb. unlabelled structs of size } n. \qquad \widetilde{\mathrm{Inv}}_3(z) = 2z^3$$

A Dictionary for Generating Series

Language: 1, $\mathscr{Z}$, +, ×, SEQ, SET, CYC and recursion.

Structure	EGF	OGF
$\mathscr{A}+\mathscr{B}$	$A(z)+B(z)$	$\tilde{A}(z)+\tilde{B}(z)$
$\mathscr{A}\cdot\mathscr{B}$	$A(z)\times B(z)$	$\tilde{A}(z)\times\tilde{B}(z)$
SEQ($\mathscr{A}$)	$\dfrac{1}{1-A(z)}$	$\dfrac{1}{1-\tilde{A}(z)}$
SET($\mathscr{A}$)	$\exp(A(z))$	$\exp(\sum \tilde{A}(z^i)/i)$
CYC($\mathscr{A}$)	$\log\dfrac{1}{1-A(z)}$	$\sum\dfrac{\phi(i)}{i}\log\dfrac{1}{1-\tilde{A}(z^i)}$

➥ Needed: fast inverse, exp, log and recursion.

Newton iteration solves everything!

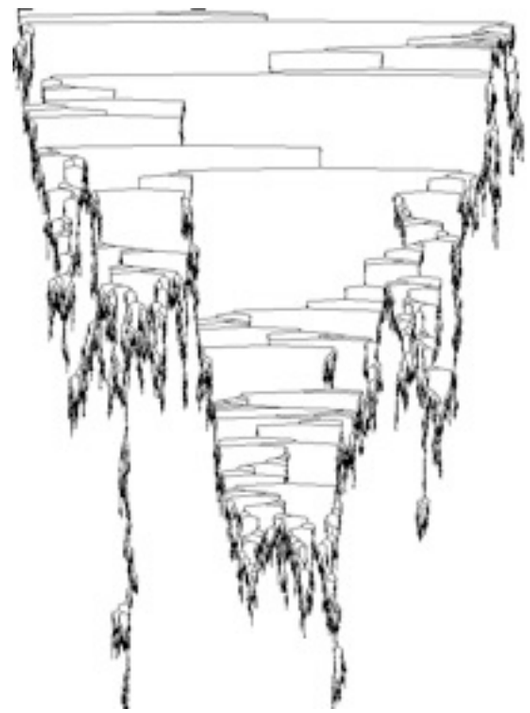
Examples

Examples

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

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

Examples



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

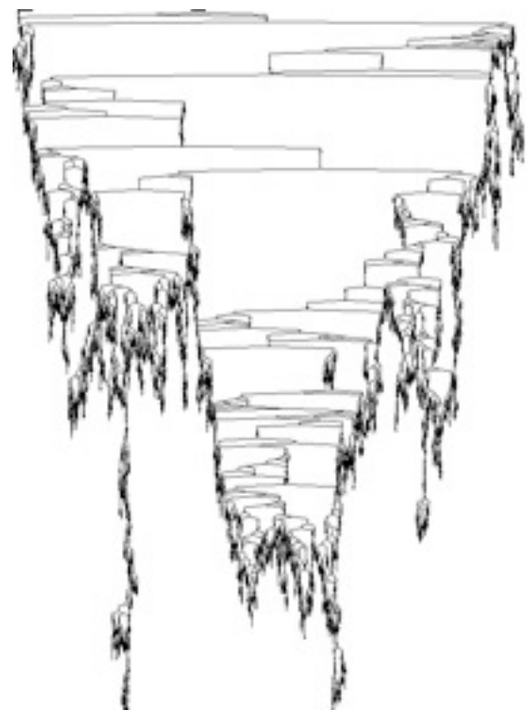
$$\longrightarrow B(z) = z + zB(z)^2 = \tilde{B}(z)$$

Cayley trees: $\mathcal{T} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{T})$

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

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

Examples



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

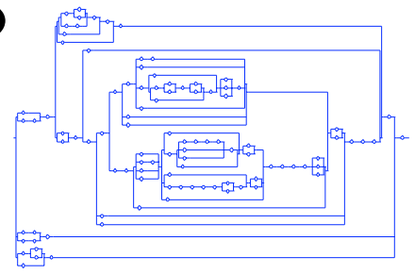
$$\longrightarrow B(z) = z + zB(z)^2 = \tilde{B}(z)$$

Cayley trees: $\mathcal{T} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{T})$

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

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

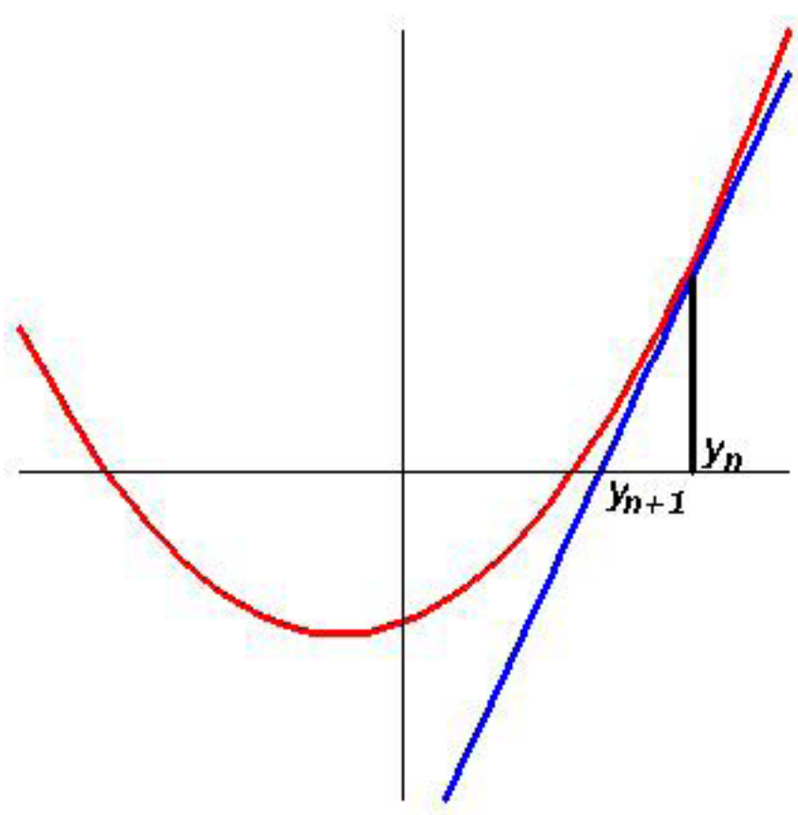
Series-parallel graphs:



$$\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}, \mathcal{S} = \textsc{Seq}_{>0}(\mathcal{Z} + \mathcal{P}), \mathcal{P} = \textsc{Set}_{>0}(\mathcal{Z} + \mathcal{S})$$

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

III. Newton Iteration in Computer Algebra



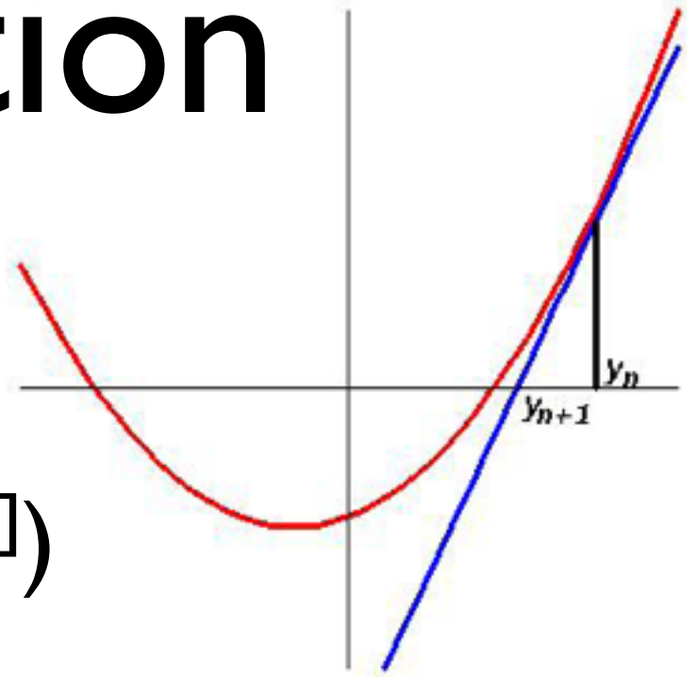
$y^3 + a^2y - 2a^3 + axy - x^3 = 0.$ $y = a - \frac{x}{4} + \frac{x^2}{64a} + \frac{131x^3}{512a^2} + \frac{509x^4}{16384a^3}$ &c.

$+a+p=y.$	$+y^3$	$+a^3 +3a^2p+3ap^2+p^3$
	$+axy$	$+a^2x+axp$
	$+x^2y$	$+x^3 +a^2p$
	$-x^3$	$-x^3$
	$-2a^3$	$-2a^3$
$-\frac{1}{4}x+q=p.$	$+p^3$	$-\frac{1}{64}x^3 +\frac{1}{16}x^2q -\frac{1}{4}xq^2+q^3$
	$+3ap^2$	$+\frac{3}{16}ax^2 -\frac{1}{2}axq +3aq^2$
	$+axp$	$-\frac{1}{4}ax^2 +axq$
	$+4a^2p$	$-a^2x +4a^2q$
	$+a^2x$	$+a^2x$
	$-x^3$	$-x3$
$+\frac{x^2}{64a}+r=q.$	$+q^3$	*
	$-\frac{3}{4}xq^2$	*
	$+3aq^2$	$+\frac{3x^4}{4096a}$ * $+\frac{1}{32}x^2r+3ar^2$
	$+\frac{1}{16}x^2q$	$+\frac{3x^4}{1024a}$ * $+\frac{3}{16}x^2r$
	$-\frac{1}{2}axq$	$-\frac{1}{128}x^3$ $-\frac{1}{2}axr$
	$+4a^2q$	$+\frac{1}{4}ax^2$ $+4a^2r$
	$-\frac{61}{64}x^3$	$-\frac{61}{64}x^3$
	$-\frac{1}{16}ax^2$	$-\frac{1}{16}ax^2$

$+4a^2 - \frac{1}{2}ax + \frac{9}{256}x^2) + \frac{131}{128}x^3 - \frac{15x^4}{4096a}$ $(+\frac{131x^3}{512a^2} + \frac{509x^4}{16384a^3}$

1671

Newton Iteration



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

$y^{[n+1]}=y^{[n]}+u^{[n]}$, with $\phi'(y^{[n]})u^{[n]}=-\phi(y^{[n]})$

Quadratic convergence ⟷ Divide-and-Conquer

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

Cost(N) = ct × Cost(last step)

Useful with fast multiplication (e.g., FFT)

→ O(*N* log *N*) ops.

Catalan numbers (binary trees)

Generating function satisfies $y = 1 + zy^2$

Newton iteration

$$y^{[n+1]} = y^{[n]} + \frac{1 + z{y^{[n]}}^2 - y^{[n]}}{1 - 2zy^{[n]}}$$

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

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

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

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

Inverse

$\phi(y)=a-1/y \rightarrow$ $y^{[n+1]}=y^{[n]}-y^{[n]}(ay^{[n]}-1).$

Cost: a small number of multiplications

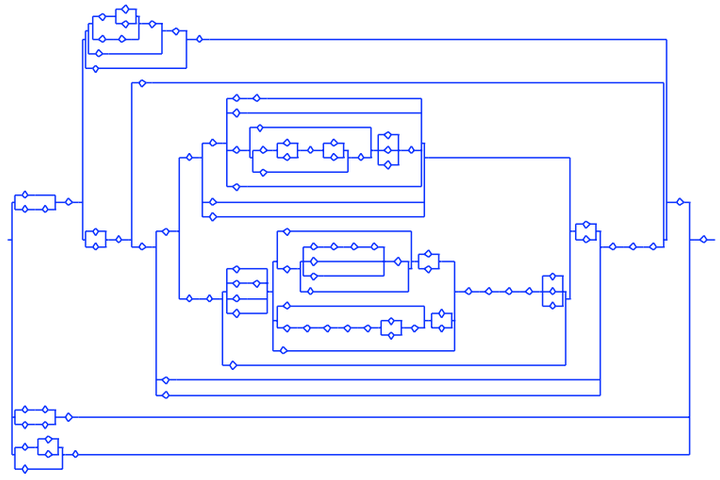
Works for

- numerical inversion;
- reciprocal of power series; SEQ
- inversion of matrices.

[Schulz 1933; Cook 1966; Sieveking 1972; Kung 1974]

Series-Parallel Graphs

$$(G, S, P) = \boldsymbol{H}(G, S, P) \qquad \begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1} - 1, \\ P &= e^{z+S} - 1 - z - S. \end{cases}$$

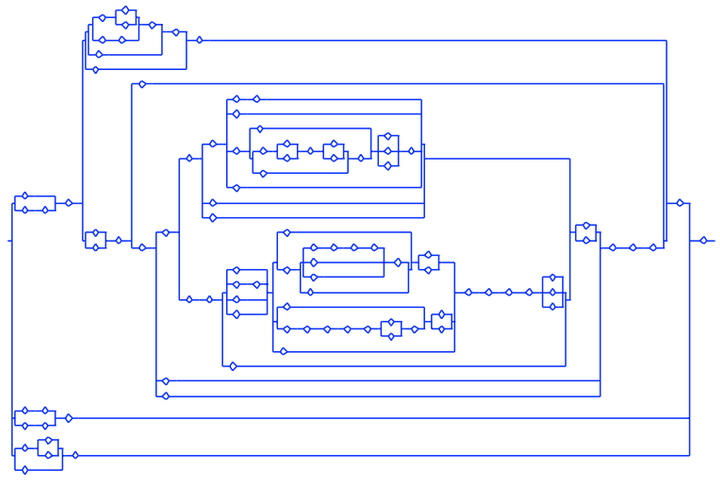


Series-Parallel Graphs

$$(G, S, P) = \boldsymbol{H}(G, S, P) \qquad \begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1} - 1, \\ P &= e^{z+S} - 1 - z - S. \end{cases}$$

Newton iteration:

$$\boldsymbol{Y}^{[n+1]} = \boldsymbol{Y}^{[n]} + (\mathrm{Id} - \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}))^{-1} \cdot (\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]})$$



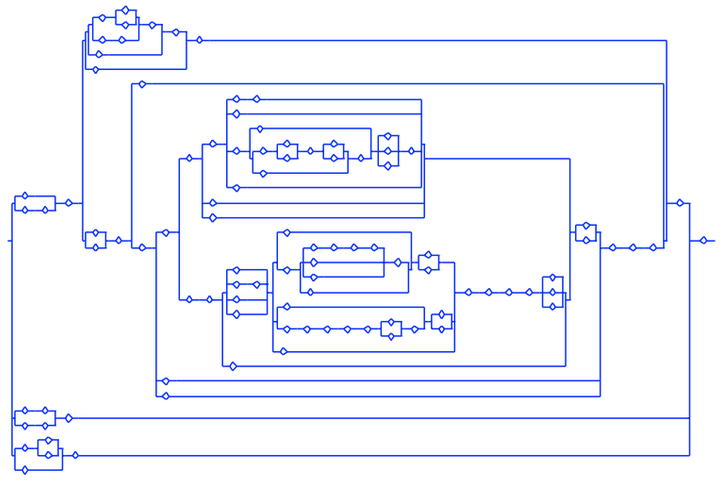
Series-Parallel Graphs

$$(G, S, P) = \boldsymbol{H}(G, S, P) \qquad \begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1} - 1, \\ P &= e^{z+S} - 1 - z - S. \end{cases}$$

Newton iteration:

$$\boldsymbol{Y}^{[n+1]} = \boldsymbol{Y}^{[n]} + (\mathrm{Id} - \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}))^{-1} \cdot (\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]})$$

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & e^{z+S} - 1 & 0 \end{pmatrix}$$

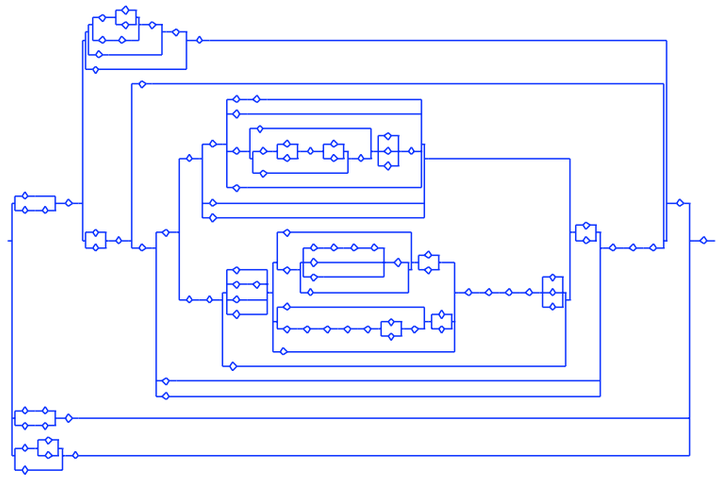


Series-Parallel Graphs

$$(G,S,P) = \boldsymbol{H}(G,S,P) \qquad \begin{cases} G &= S+P, \\ S &= (1-z-P)^{-1}-1, \\ P &= e^{z+S}-1-z-S. \end{cases}$$

Newton iteration:

$$\boldsymbol{Y}^{[n+1]} = \boldsymbol{Y}^{[n]} + (\mathrm{Id} - \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}))^{-1} \cdot (\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]})$$



$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1-z-P)^{-2} \\ 0 & e^{z+S}-1 & 0 \end{pmatrix}$$

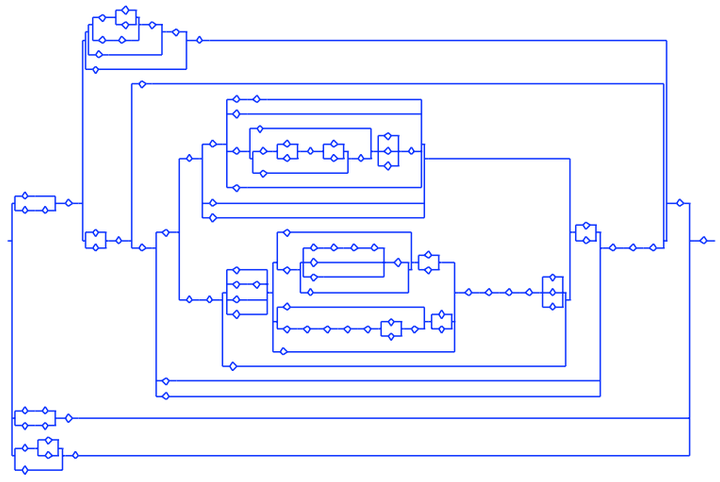
$$\begin{cases} \boldsymbol{Y}^{[n+1]} &= \boldsymbol{Y}^{[n]} + U^{[n+1]} \cdot \left(\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]} \right) \bmod z^{2^{n+1}}, \\ U^{[n+1]} &= U^{[n]} + U^{[n]} \cdot \left(\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}) \cdot U^{[n]} + \mathrm{Id} - U^{[n]} \right) \bmod z^{2^n}. \end{cases}$$

Series-Parallel Graphs

$$(G,S,P) = \boldsymbol{H}(G,S,P) \qquad \begin{cases} G &= S+P, \\ S &= (1-z-P)^{-1}-1, \\ P &= e^{z+S}-1-z-S. \end{cases}$$

Newton iteration:

$$\boldsymbol{Y}^{[n+1]} = \boldsymbol{Y}^{[n]} + (\mathrm{Id} - \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}))^{-1} \cdot (\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]})$$



$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1-z-P)^{-2} \\ 0 & e^{z+S}-1 & 0 \end{pmatrix}$$

$$\begin{cases} \boldsymbol{Y}^{[n+1]} &= \boldsymbol{Y}^{[n]} + U^{[n+1]} \cdot \left(\boldsymbol{H}(\boldsymbol{Y}^{[n]}) - \boldsymbol{Y}^{[n]} \right) \bmod z^{2^{n+1}}, \\ U^{[n+1]} &= U^{[n]} + U^{[n]} \cdot \left(\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{Y}}(\boldsymbol{Y}^{[n]}) \cdot U^{[n]} + \mathrm{Id} - U^{[n]} \right) \bmod z^{2^n}. \end{cases}$$

Still missing: efficient exp.

Logarithm & exponential

(sets and cycles)

- Logarithm of power series: $\log f = \int f'/f$
- Exponential: Newton with $\phi(y) = a - \log y$

$$e^{[n+1]} = e^{[n]} + e^{[n]} \left(a - \int e^{[n]\prime} / e^{[n]} \right) \bmod z^{2^{n+1}}.$$

and $1/e^{[n]}$... by Newton iteration!

[Brent 1975; Hanrot-Zimmermann 2002]

Applications/Extensions

- Newton sums;
- linear differential systems;
- non-linear differential systems;
- ...

Conclusion for Series-Parallel Graphs

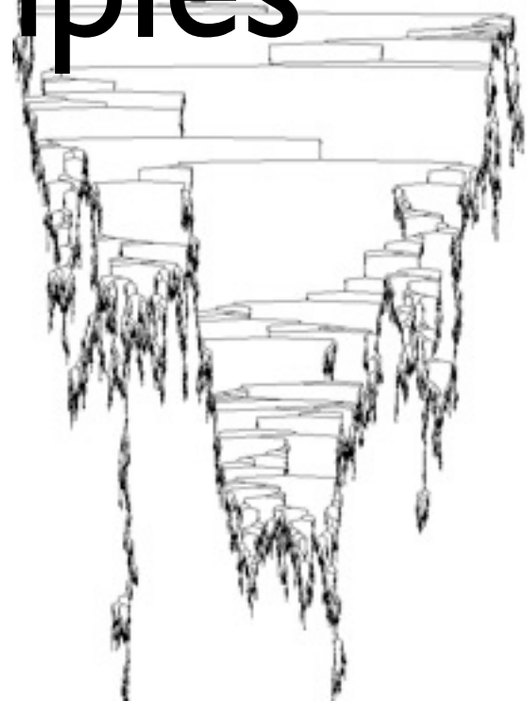
$$\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}, \mathcal{S} = \textsc{Seq}_{>0}(\mathcal{Z} + \mathcal{P}), \mathcal{P} = \textsc{Set}_{>0}(\mathcal{Z} + \mathcal{S})$$

compiles into the Newton iteration

$$\begin{cases} i^{[n+1]} = i^{[n]} - i^{[n]}(e^{[n]}i^{[n]} - 1), & (\to e^{-z-S}) \\ e^{[n+1]} = e^{[n]} - e^{[n]}\left(1 + \frac{d}{dz}S^{[n]} - \int(\frac{d}{dz}e^{[n]})i^{[n]}\right), & (\to e^{z+S}) \\ v^{[n+1]} = v^{[n]} - v^{[n]}((1 - z - P^{[n]})v^{[n]} - 1), & (\to (1-z-P)^{-1}) \\ U^{[n+1]} = U^{[n]} + U^{[n]} \cdot \left(\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & {v^{[n+1]}}^2 \\ 0 & e^{[n+1]} - 1 & 0 \end{pmatrix} \cdot U^{[n]} + \mathrm{Id} - U^{[n]} \right), & (\to (\mathrm{Id} - \partial H/\partial Y)^{-1}) \\ \begin{pmatrix} G^{[n+1]} \\ S^{[n+1]} \\ P^{[n+1]} \end{pmatrix} = \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix} + U^{[n+1]} \cdot \begin{pmatrix} S^{[n]} + P^{[n]} - G^{[n]} \\ v^{[n+1]} - S^{[n]} \\ e^{[n+1]} - P^{[n]} \end{pmatrix} \bmod z^{2^{n+1}}. & (\to \text{solution}) \end{cases}$$

Computation reduced to products and linear operations.

Conclusion for the Examples



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

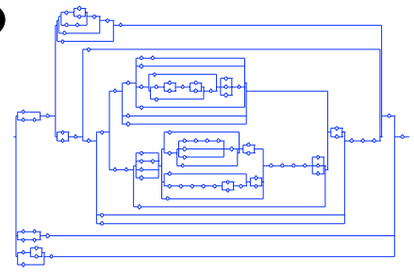
✔ $\longrightarrow B(z) = z + zB(z)^2 = \tilde{B}(z)$

Cayley trees: $\mathcal{T} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{T})$

✔ $\longrightarrow T(z) = z \exp(T(z));$

➥ $\longrightarrow \tilde{T}(z) = z \exp\left(\tilde{T}(z) + \frac{1}{2}\tilde{T}(z^2) + \frac{1}{3}\tilde{T}(z^3) + \cdots\right)$

Series-parallel graphs:

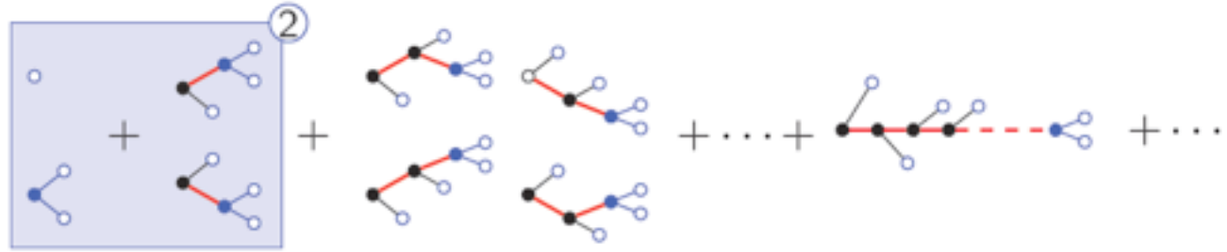


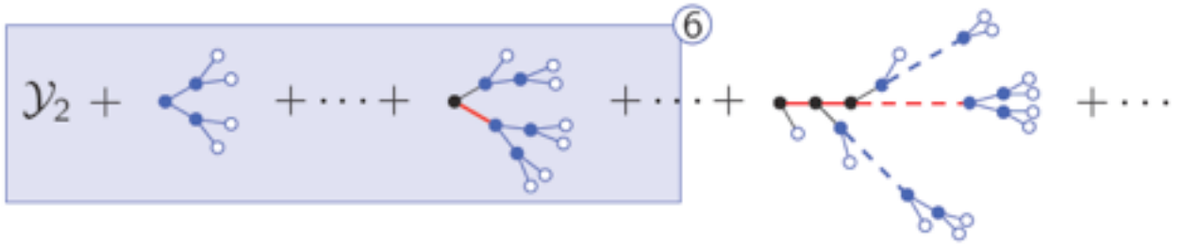
$$\mathcal{G} = \mathcal{Z} + \mathcal{S} + \mathcal{P}, \mathcal{S} = \textsc{Seq}_{>0}(\mathcal{Z} + \mathcal{P}), \mathcal{P} = \textsc{Set}_{>0}(\mathcal{Z} + \mathcal{S})$$

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

IV. Newton Iteration in Combinatorics

$\mathcal{Y}_0 = \varnothing \qquad \mathcal{Y}_1 = \circ$

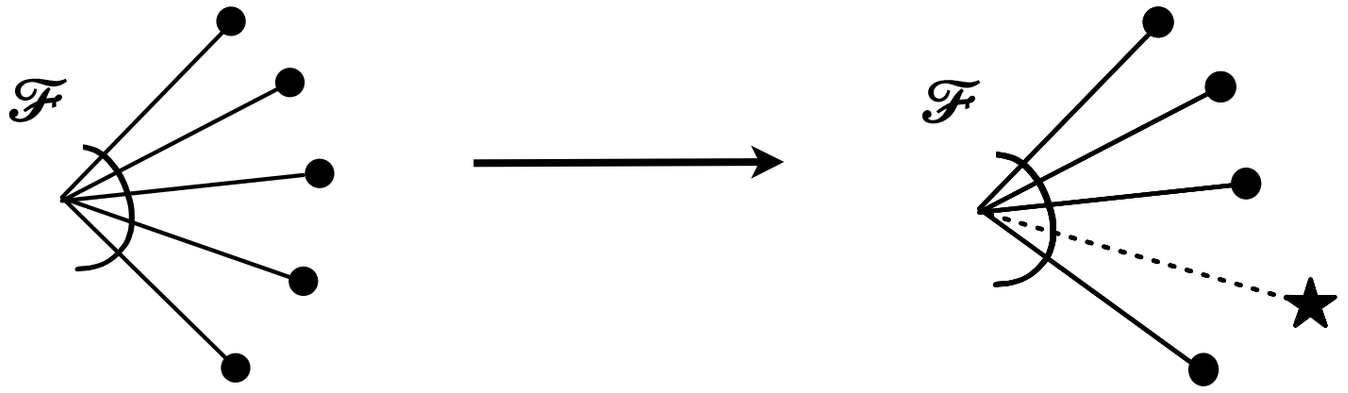
$\mathcal{Y}_2 =$ 

$\mathcal{Y}_3 =$ 

Derivative

(context: species theory)

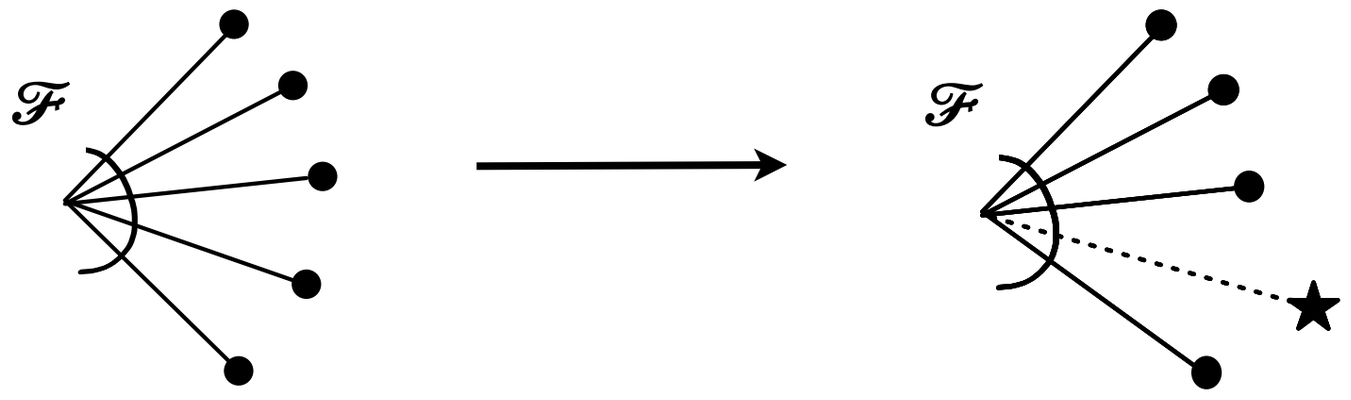
Huet's zipper



Derivative

(context: species theory)

Huet's zipper

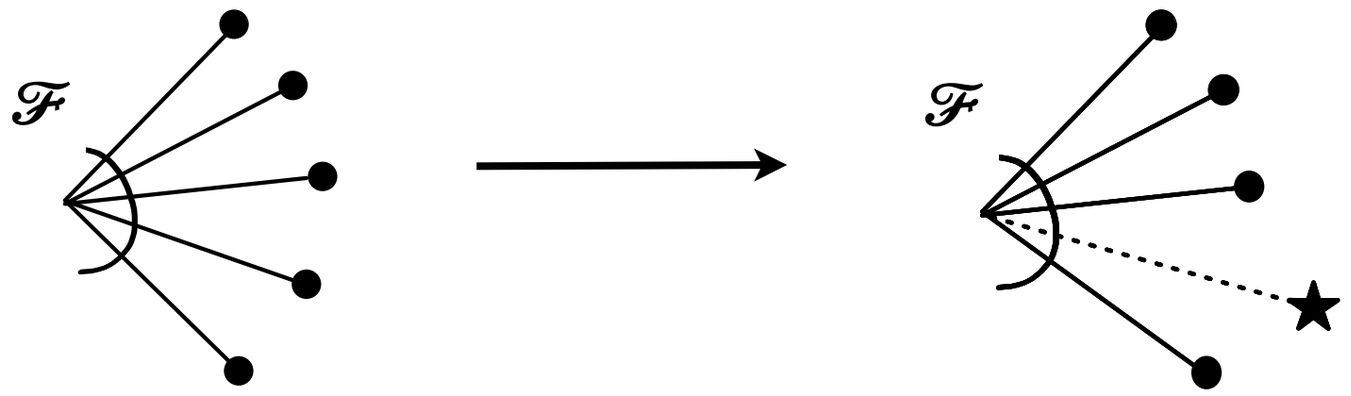


- 0'=1'=0; Z'=1; SET'=SET; CYC'=SEQ; SEQ'=SEQ·SEQ;

Huet's zipper

Derivative

(context: species theory)

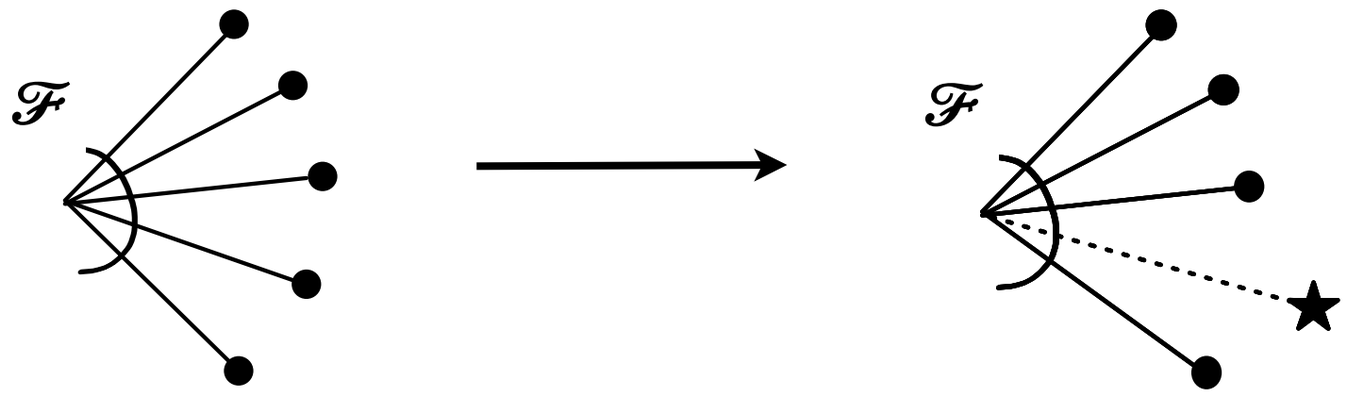


- $0'=1'=0$; $Z'=1$; $\mathrm{SET}'=\mathrm{SET}$; $\mathrm{CYC}'=\mathrm{SEQ}$; $\mathrm{SEQ}'=\mathrm{SEQ}\cdot\mathrm{SEQ}$;
- $(\mathscr{F}+\mathscr{G})'=\mathscr{F}'+\mathscr{G}'$;$(\mathscr{F}\cdot\mathscr{G})'=\mathscr{F}'\cdot\mathscr{G}+\mathscr{F}\cdot\mathscr{G}'$;

Huet's zipper

Derivative

(context: species theory)

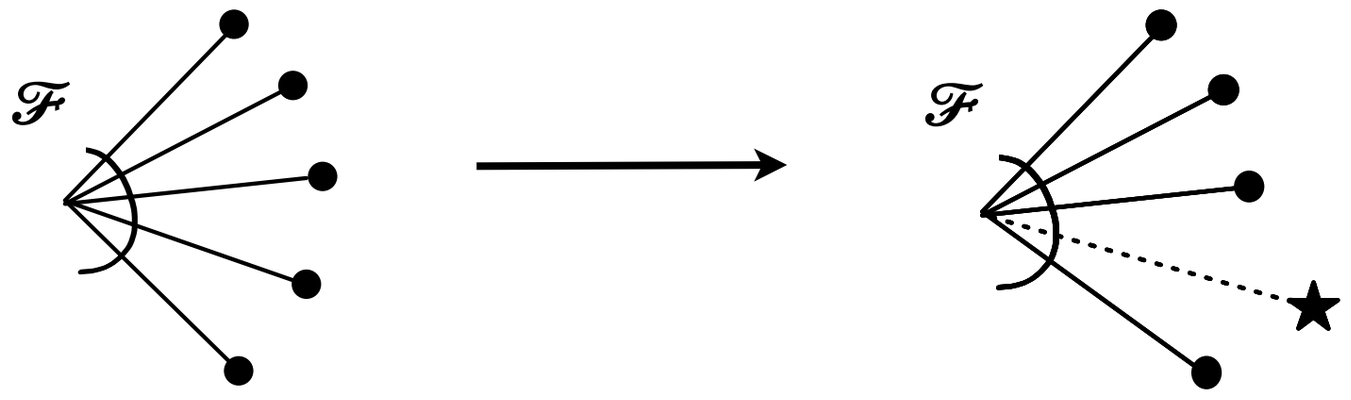


- $0'=1'=0$; $Z'=1$; $\mathrm{SET}'=\mathrm{SET}$; $\mathrm{CYC}'=\mathrm{SEQ}$; $\mathrm{SEQ}'=\mathrm{SEQ}\cdot\mathrm{SEQ}$;
- $(\mathscr{F}+\mathscr{G})'=\mathscr{F}'+\mathscr{G}'$; $(\mathscr{F}\cdot\mathscr{G})'=\mathscr{F}'\cdot\mathscr{G}+\mathscr{F}\cdot\mathscr{G}'$;
- $\mathscr{F}(\mathscr{G})'=\mathscr{F}'(\mathscr{G})\cdot\mathscr{G}'$.

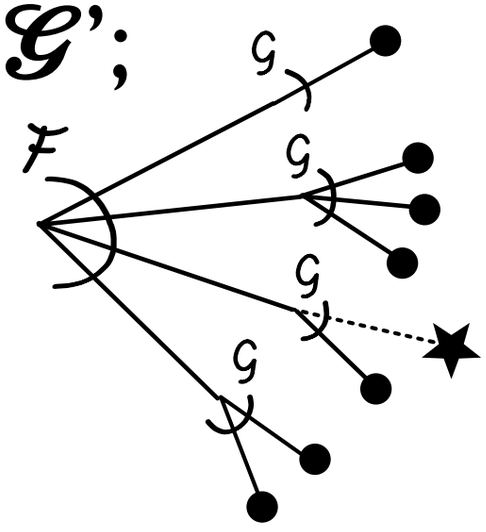
Derivative

(context: species theory)

Huet's zipper



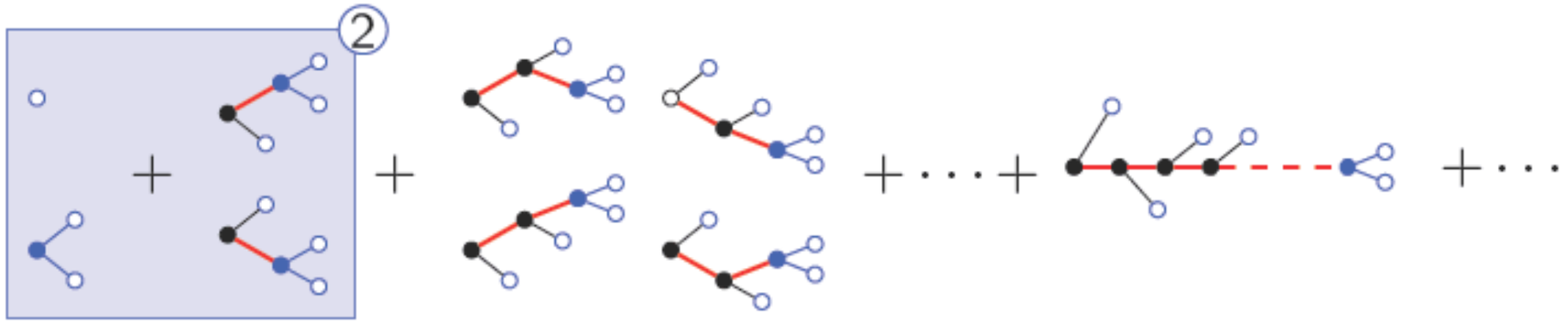
- $0'=1'=0$; $Z'=1$; $\mathrm{SET}'=\mathrm{SET}$; $\mathrm{CYC}'=\mathrm{SEQ}$; $\mathrm{SEQ}'=\mathrm{SEQ}\cdot\mathrm{SEQ}$;
- $(\mathscr{F}+\mathscr{G})'=\mathscr{F}'+\mathscr{G}'$; $(\mathscr{F}\cdot\mathscr{G})'=\mathscr{F}'\cdot\mathscr{G}+\mathscr{F}\cdot\mathscr{G}'$;
- $\mathscr{F}(\mathscr{G})'=\mathscr{F}'(\mathscr{G})\cdot\mathscr{G}'$.

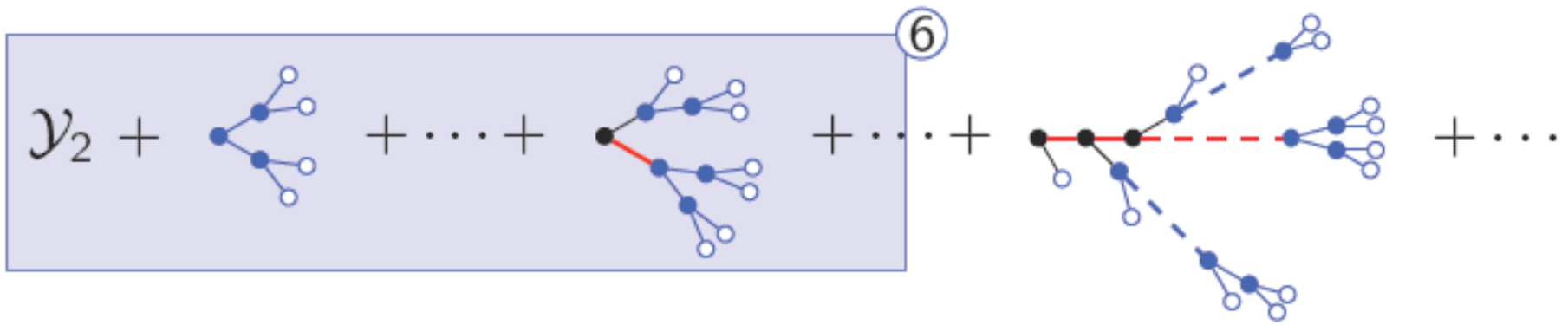


Newton Iteration for binary trees

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

$\mathcal{Y}_0 = \varnothing$ $\mathcal{Y}_1 = \circ$

$\mathcal{Y}_2 =$ 

$\mathcal{Y}_3 =$ 

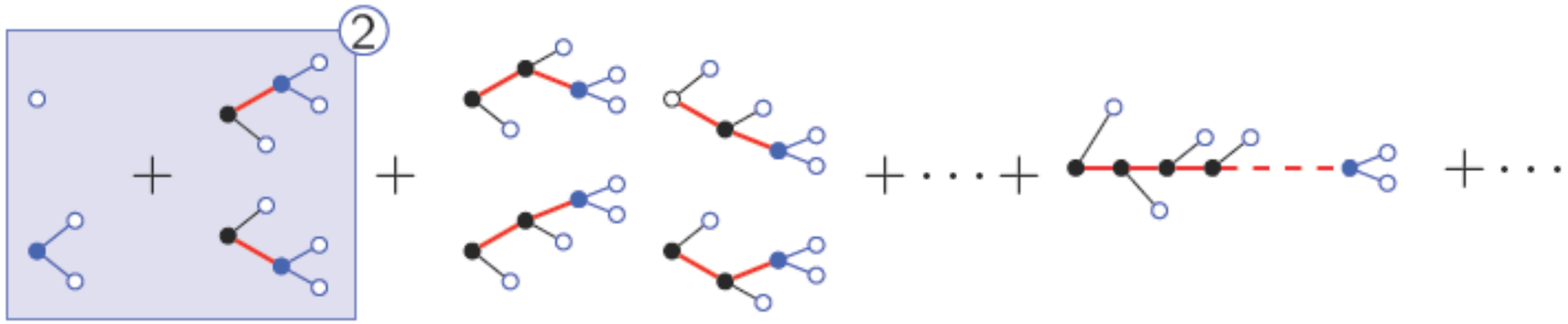
[Décoste, Labelle, Leroux 1982]

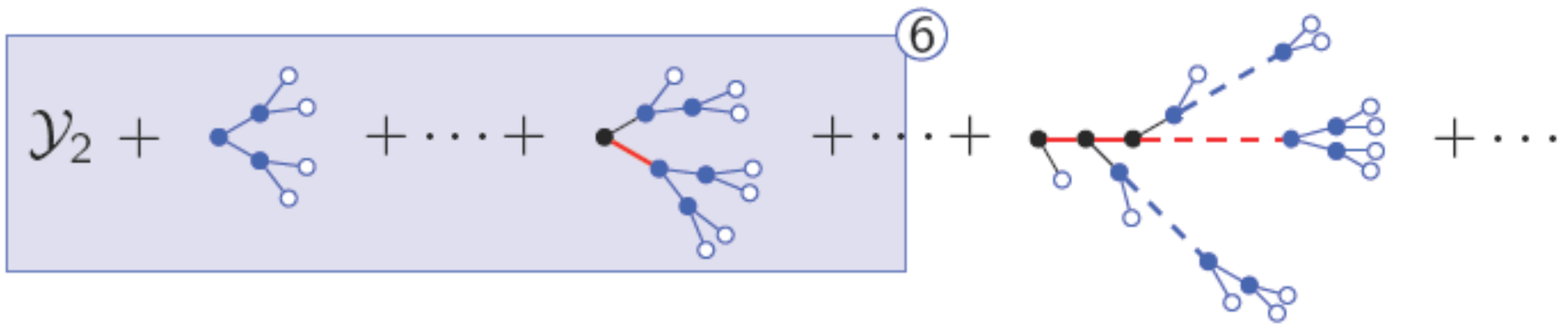
Newton Iteration for binary trees

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

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

$\mathcal{Y}_0 = \varnothing$ $\mathcal{Y}_1 = \circ$

$\mathcal{Y}_2 =$ 

$\mathcal{Y}_3 =$ 

[Décoste, Labelle, Leroux 1982]

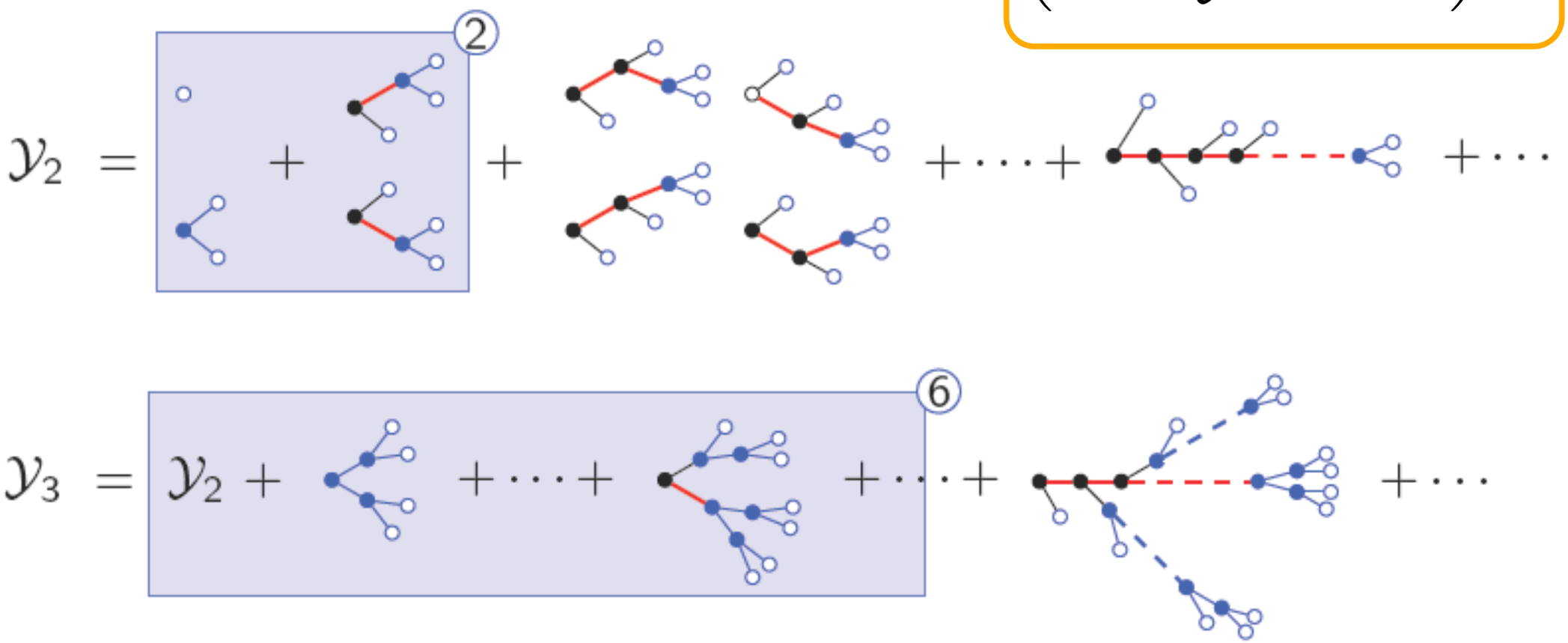
Newton Iteration for binary trees

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

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

$$\left(\mathrm{Id} - \frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{Y}^{[n]})\right)^{-1}$$

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



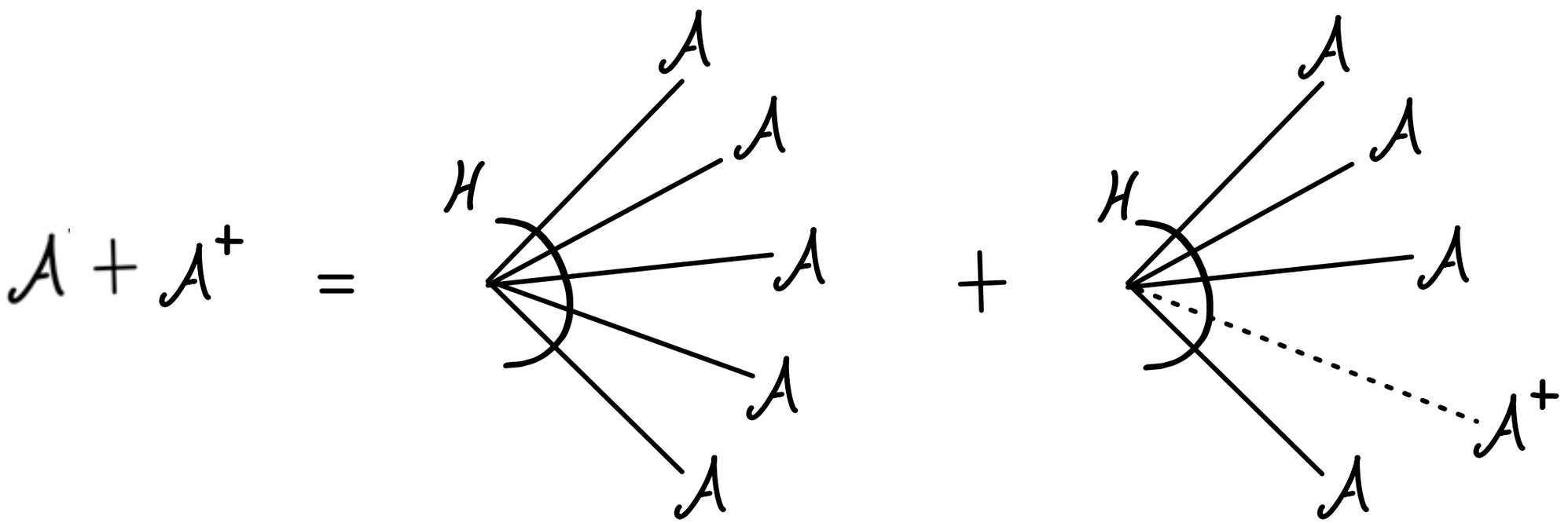
[Décoste, Labelle, Leroux 1982]

Combinatorial Newton Iteration

Thm. [essentially Labelle] For any well-founded system $\mathcal{Y}=\mathcal{H}(\mathcal{Z},\mathcal{Y})$, if $\mathcal{A}$ coincides with the solution up to size k and $\mathcal{A}\subset\mathcal{H}(\mathcal{Z},\mathcal{A})$, then

$$\mathcal{A}+\sum_{i\geq 0}(\partial\mathcal{H}/\partial\mathcal{Y}(\mathcal{Z},\mathcal{A}))^i\cdot(\mathcal{H}(\mathcal{Z},\mathcal{A})\backslash\mathcal{A})$$

coincides with the solution up to size $2k+1$.



Application: Unlabelled Rooted Trees

Combinatorial equation: $\mathcal{T} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{T}) =: \mathcal{H}(\mathcal{Z}, \mathcal{T})$

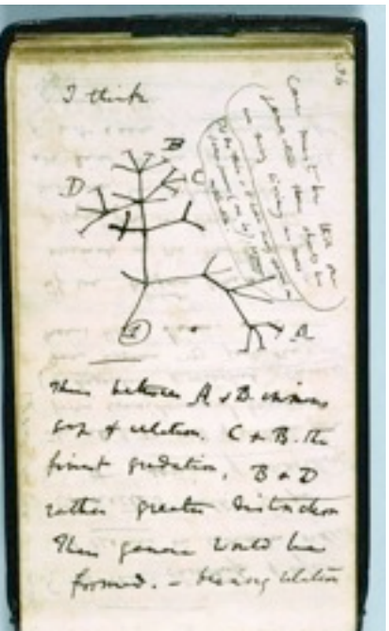
Combinatorial Newton iteration:

$$\mathcal{T}^{[n+1]} = \mathcal{T}^{[n]} + \textsc{Seq}(\mathcal{H}(\mathcal{T}^{[n]})) \cdot (\mathcal{H}(\mathcal{T}^{[n]}) \setminus \mathcal{T}^{[n]})$$

OGF equation: $\tilde{T}(z) = H(z, \tilde{T}(z))$

$$\tilde{T}(z) = z \exp(\tilde{T}(z) + \frac{1}{2}\tilde{T}(z^2) + \frac{1}{3}\tilde{T}(z^3) + \cdots)$$

Newton for OGF

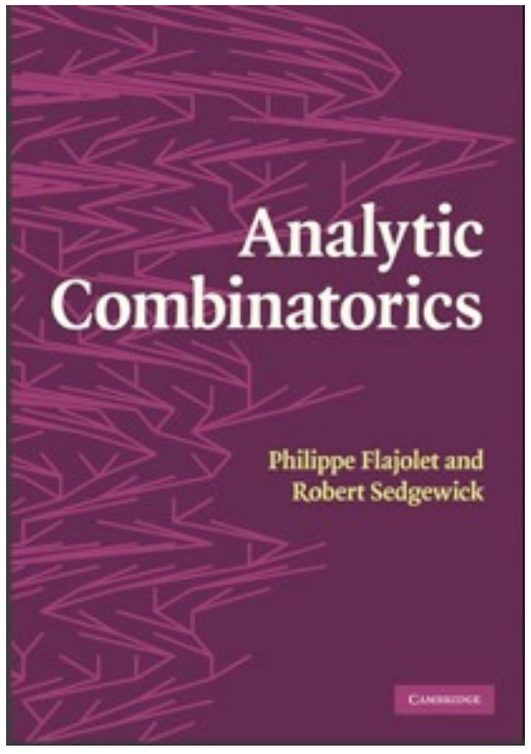


$$\tilde{T}^{[n+1]} = \tilde{T}^{[n]} + \frac{H(z, \tilde{T}^{[n]}) - \tilde{T}^{[n]}}{1 - H(z, \tilde{T}^{[n]})}$$

$0,$

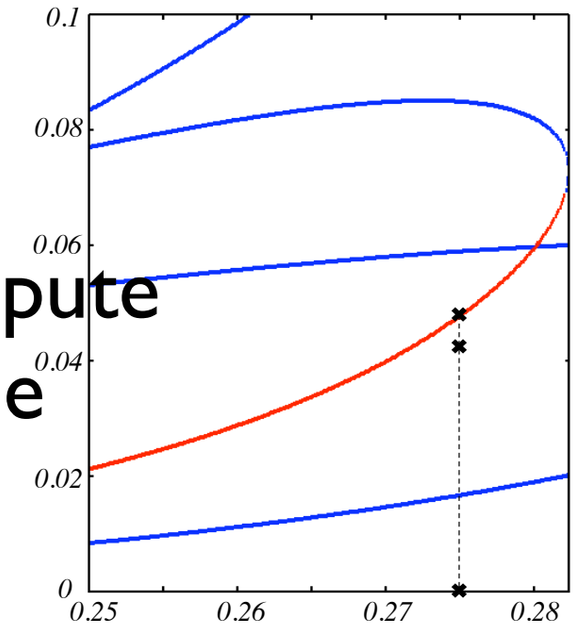
$z + z^2 + z^3 + z^4 + \cdots,$

$z + z^2 + 2z^3 + 4z^4 + 9z^5 + 20z^6 + \cdots$

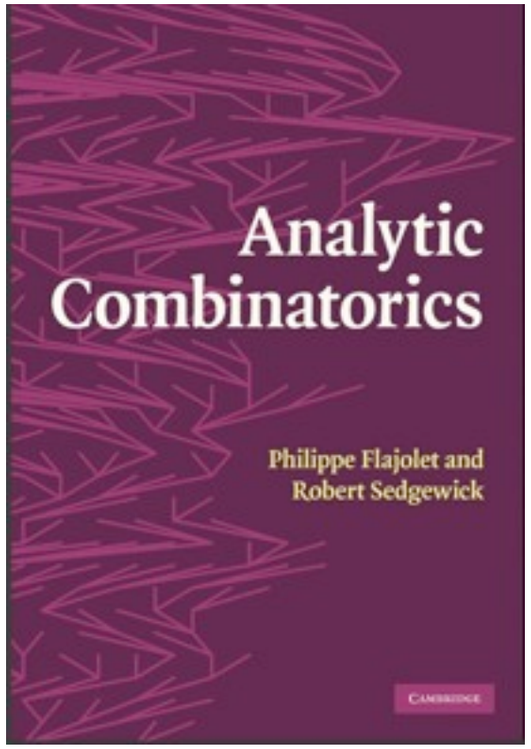


Our Result

Also, those same Newton iterations let one compute the *numerical values* of the generating series inside their disk of convergence.



➥ efficient random generation for the whole class.



Our Result

Thm. [Pivoteau-S-Soria 2012] First N coefficients of GFs of all *constructible* species in

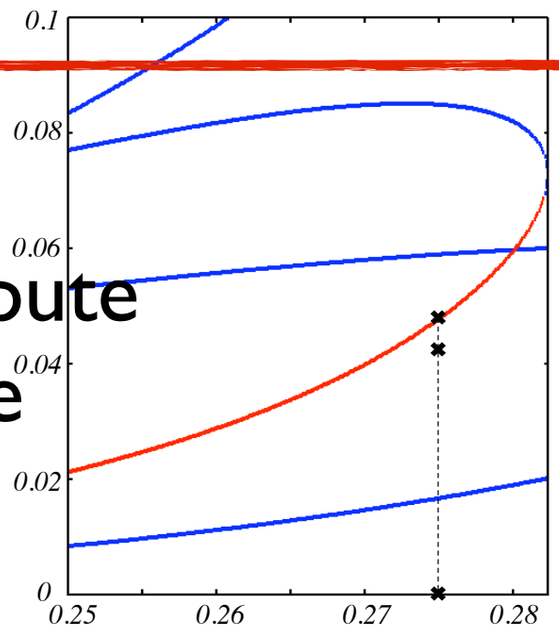
1. arithmetic complexity

$O(N \log N)$ (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).

Also, those same Newton iterations let one compute the *numerical values* of the generating series inside their disk of convergence.



➥ efficient random generation for the whole class.

Random generation for XML grammars

Grammar	nb eqs	max deg	nb sols	oracle (s.)	FGb (s.)
rss	10	5	2	0.02	0.03
PNML	22	4	4	0.05	0.1
xslt	40	3	10	0.4	1.5
relaxng	34	4	32	0.4	3.3
xhtml-basic	53	3	13	1.2	18
mathml2	182	2	18	3.7	882
xhtml	93	6	56	3.4	1124
xhtml-strict	80	6	32	3.0	1590
xmlschema	59	10	24	0.5	6592
SVG	117	10		5.8	>1.5Go
docbook	407	11		67.7	>1.5Go
OpenDoc	500			3.9	

[Darrasse 2008]

Conclusion

Newton iteration everywhere

Newton iteration everywhere

- Quadratic convergence ➙ divide-and-conquer ➙ efficient algorithms in computer algebra

Newton iteration everywhere

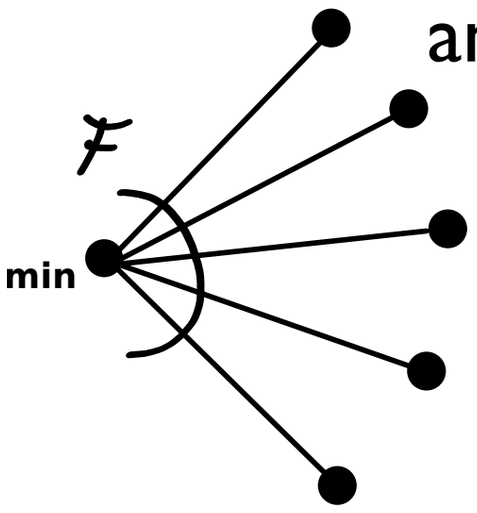
- Quadratic convergence ➙ divide-and-conquer ➙ efficient algorithms in computer algebra
- Combinatorial lifting ➙ efficient iterations for functional equations ➙ random generation

Newton iteration everywhere

- Quadratic convergence ➙ divide-and-conquer ➙ efficient algorithms in computer algebra

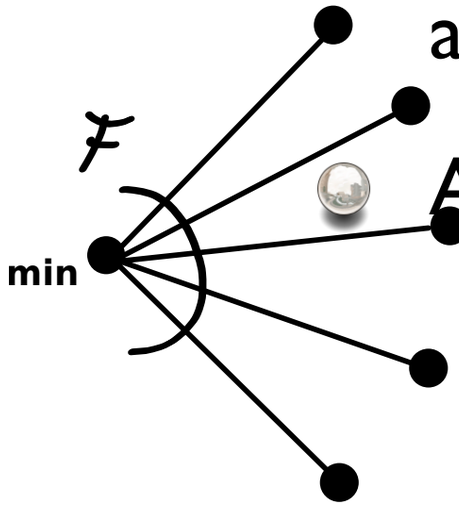
- Combinatorial lifting ➙ efficient iterations for functional equations ➙ random generation

- More is true! Differential combinatorial equations and ordered structures...



Newton iteration everywhere

- Quadratic convergence ➙ divide-and-conquer ➙ efficient algorithms in computer algebra
- Combinatorial lifting ➙ efficient iterations for functional equations ➙ random generation
- More is true! Differential combinatorial equations and ordered structures...
- Analytic combinatorics goes on...



Newton iteration everywhere

- Quadratic convergence ➙ divide-and-conquer ➙ efficient algorithms in computer algebra
- Combinatorial lifting ➙ efficient iterations for functional equations ➙ random generation
- More is true! Differential combinatorial equations and ordered structures...
- Analytic combinatorics goes on...