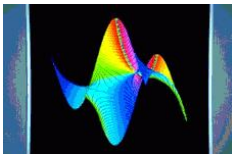


# Newton's Iteration for Combinatorial Systems and Applications

Bruno Salvy

`Bruno.Salvy@inria.fr`

Algorithms Project, Inria

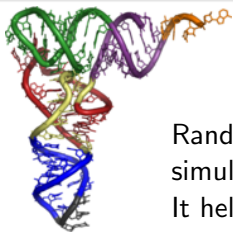


Joint work with Carine Pivoteau and Michèle Soria

Vancouver, May 17, 2011

# I Introduction

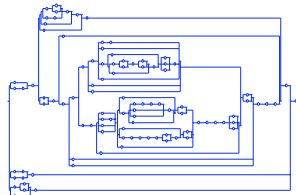
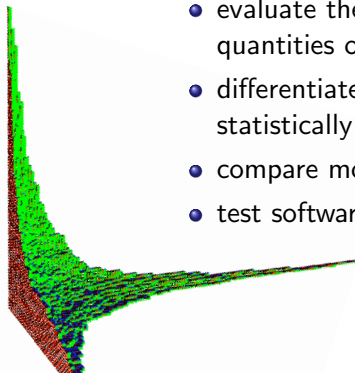
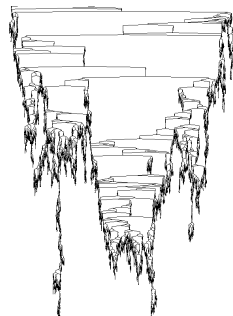
# Motivation: Random Generation



Random generation of large objects = simulation in the discrete world.

It helps

- evaluate the order of magnitude of quantities of interest;
- differentiate exceptional values from statistically expected ones;
- compare models;
- test software.



# Framework: Constructible Species

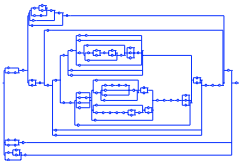
A small set of species

$1, \mathcal{Z}, \times, +, \text{SEQ}, \text{SET}, \text{CYC},$

cardinality constraints that are finite unions of intervals,  
used recursively.

Examples:

- Regular languages
- Unambiguous context-free languages
- Trees ( $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B}^2, \mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ )
- Mappings, ...



# Framework: Constructible Species

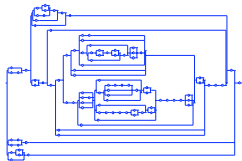
A small set of species

$1, \mathcal{Z}, \times, +, \text{SEQ}, \text{SET}, \text{CYC},$

cardinality constraints that are finite unions of intervals,  
used recursively.

Examples:

- Regular languages
- Unambiguous context-free languages
- Trees ( $\mathcal{B} = \mathcal{Z} + \mathcal{Z} \times \mathcal{B}^2, \mathcal{T} = \mathcal{Z} \times \text{SET}(\mathcal{T})$ )
- Mappings, ...



Two related problems:

- 1 **Enumeration**: number of objects of size  $n$  for  $n = 0, 1, 2, \dots$
- 2 **Random generation**: all objects of size  $n$  with the same proba.

Two contexts: labelled/unlabelled.

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with probability  $x^{|t|}/T(x)$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}$  = generating series of  $\mathcal{T}$ ;  $|t|$  = size.

Same size, same probability  
Expected size  $xT'(x)/T(x)$  increases with  $x$ .

Complexity **linear** in  $|t|$  **when the values  $T(x)$  are available.**

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with **probability**  $x^{|t|}/T(x)$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}$  = generating series of  $\mathcal{T}$ ;  $|t|$  = size.

Same size, same probability

Expected size  $xT'(x)/T(x)$  increases with  $x$ .

Singleton

Easy.

Complexity **linear** in  $|t|$  **when the values**  $T(x)$  **are available.**

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with **probability**  $x^{|t|}/T(x)$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}$  = generating series of  $\mathcal{T}$ ;  $|t|$  = size.

Same size, same probability

Expected size  $xT'(x)/T(x)$  increases with  $x$ .

## Singleton

Easy.

## Cartesian Product $\mathcal{C} = \mathcal{A} \times \mathcal{B}$

- Generate  $a \in \mathcal{A}$ ;  $b \in \mathcal{B}$ ;
- Return  $(a, b)$ .

Proof.  $C(x) = \sum_{(a,b)} x^{|a|+|b|} = A(x)B(x)$ ;  $\frac{x^{|a|+|b|}}{C(x)} = \frac{x^{|a|}}{A(x)} \frac{x^{|b|}}{B(x)}$ .

Complexity **linear** in  $|t|$  **when the values  $T(x)$  are available.**

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with **probability**  $x^{|t|}/T(x)$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}$  = generating series of  $\mathcal{T}$ ;  $|t|$  = size.

Same size, same probability

Expected size  $xT'(x)/T(x)$  increases with  $x$ .

## Singleton

Easy.

## Cartesian Product $\mathcal{C} = \mathcal{A} \times \mathcal{B}$

- Generate  $a \in \mathcal{A}$ ;  $b \in \mathcal{B}$ ;
- Return  $(a, b)$ .

## Disjoint Union $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$

- Draw  $b = \text{Bernoulli}(A(x)/C(x))$ ;
- If  $b = 1$  then generate  $a \in \mathcal{A}$  else generate  $b \in \mathcal{B}$ .

Proof.  $\frac{x^{|a|}}{C(x)} = \frac{x^{|a|}}{A(x)} \frac{A(x)}{C(x)}$ .

Complexity **linear** in  $|t|$  **when the values  $T(x)$  are available.**

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with **probability**  $x^{|t|}/T(x)$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}$  = generating series of  $\mathcal{T}$ ;  $|t|$  = size.

Same size, same probability

Expected size  $xT'(x)/T(x)$  increases with  $x$ .

Singleton

Easy.

Disjoint Union  $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$

- Draw  $b = \text{Bernoulli}(A(x)/C(x))$ ;
- If  $b = 1$  then generate  $a \in \mathcal{A}$  else generate  $b \in \mathcal{B}$ .

Cartesian Product  $\mathcal{C} = \mathcal{A} \times \mathcal{B}$

- Generate  $a \in \mathcal{A}$ ;  $b \in \mathcal{B}$ ;
- Return  $(a, b)$ .

Use recursively (e.g., binary trees  $\mathcal{B} = \mathcal{Z} \cup \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$ )

Also: sets, cycles, ...;

Complexity **linear** in  $|t|$  **when the values  $T(x)$  are available.**

# Boltzmann Samplers

Principle (Duchon, Flajolet, Louchard, Schaeffer 2004)

Generate each  $t \in \mathcal{T}$  with probability  $x^{|t|}/T(x)/|t|!$ , where:  $x > 0$  fixed;  $T(z) := \sum_{t \in \mathcal{T}} z^{|t|}/|t|! =$  generating series of  $\mathcal{T}$ ;  $|t| =$  size.

Same size, same probability

Expected size  $xT'(x)/T(x)$  increases with  $x$ .

Singleton

Easy.

Disjoint Union  $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$

- Draw  $b = \text{Bernoulli}(A(x)/C(x))$ ;
- If  $b = 1$  then generate  $a \in \mathcal{A}$  else generate  $b \in \mathcal{B}$ .

Cartesian Product  $\mathcal{C} = \mathcal{A} \times \mathcal{B}$

- Generate  $a \in \mathcal{A}$ ;  $b \in \mathcal{B}$ ;
- Return  $(a, b)$ .

Use recursively (e.g., binary trees  $\mathcal{B} = \mathcal{Z} \cup \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$ )

Also: sets, cycles, ...; **labelled case**

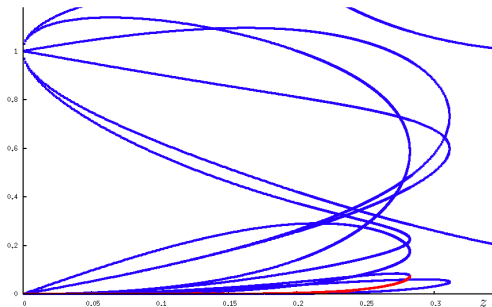
Complexity **linear** in  $|t|$  **when the values  $T(x)$  are available.**

# Oracle: Large Systems that are Interesting to Solve

The generating series are given by systems of equations.

We need:

- only one solution;
- the right one;
- only numerically.



In the worst case, these requirements would make no difference.

**But** these systems inherit **structure** from combinatorics.

# Results (1/2): Fast Enumeration

## Theorem (Enumeration in Quasi-Optimal Complexity)

*First  $N$  coefficients of gfs of constructible species in*

- ① *arithmetic complexity:*
  - $O(N \log N)$  (both ogf and egf);
- ② *binary complexity:*
  - $O(N^2 \log^2 N \log \log N)$  (ogf);
  - $O(N^2 \log^3 N \log \log N)$  (egf).

# Results (2/2): Oracle

- 1 The egfs and the ogfs of constructible species are convergent in the neighborhood of 0;
- 2 A numerical iteration converging to  $\mathbf{Y}(\alpha)$  in the labelled case (inside the disk);
- 3 A numerical iteration converging to the sequence  $\mathbf{Y}(\alpha), \mathbf{Y}(\alpha^2), \mathbf{Y}(\alpha^3), \dots$  for  $\|\cdot\|_\infty$  in the unlabelled case (inside the disk).



## Examples (I): Polynomial Systems

Random generation following given XML grammars

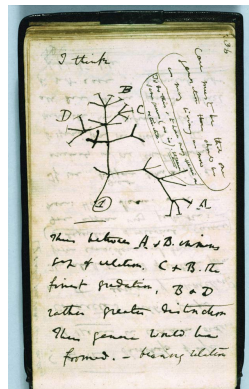
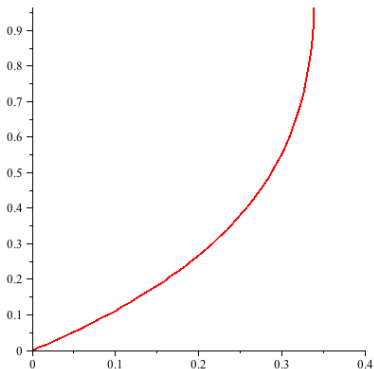
Grammar	nb eqs	max deg	nb sols	oracle (s.)	FGb (s.)
rss	10	5	2	0.02	0.03
PNML	22	4	4	0.05	0.1
xslt	40	3	10	0.4	1.5
relaxng	34	4	32	0.4	3.3
xhtml-basic	53	3	13	1.2	18
mathml2	182	2	18	3.7	882
xhtml	93	6	56	3.4	1124
xhtml-strict	80	6	32	3.0	1590
xmlschema	59	10	24	0.5	6592
SVG	117	10		5.8	>1.5Go
docbook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

[Darrasse 2008]

# Example (II): A Non-Polynomial "System"

Unlabelled rooted trees:

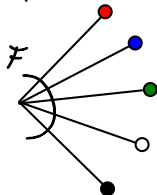
$$f(x) = x \exp(f(x) + \frac{1}{2}f(x^2) + \frac{1}{3}f(x^3) + \dots)$$



## II Combinatorics

# Mini-Introduction to Species

- Species  $\mathcal{F}$ :

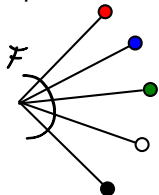


## Examples:

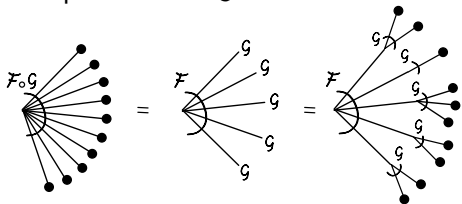
- $0, 1, \mathcal{Z}$ ;
- SET;
- SEQ, CYC.

# Mini-Introduction to Species

- Species  $\mathcal{F}$ :



- Composition  $\mathcal{F} \circ \mathcal{G}$ :

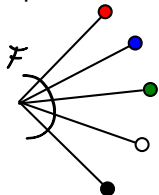


## Examples:

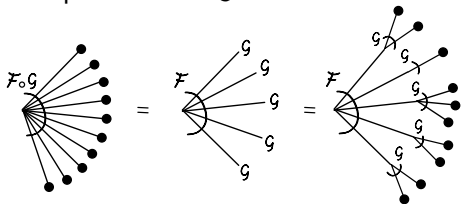
- $0, 1, \mathcal{Z}$ ;
- SET;
- SEQ, CYC.

# Mini-Introduction to Species

- Species  $\mathcal{F}$ :



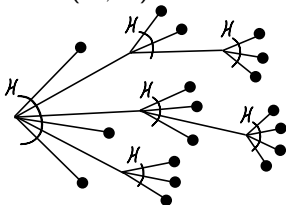
- Composition  $\mathcal{F} \circ \mathcal{G}$ :



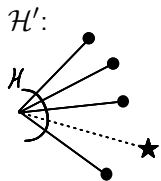
## Examples:

- 0, 1,  $\mathcal{Z}$ ;
- SET;
- SEQ, CYC.

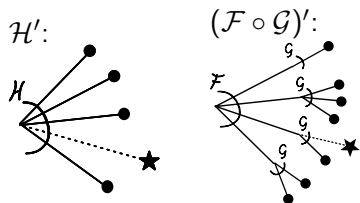
- $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$



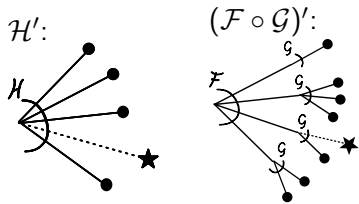
# Derivative



# Derivative

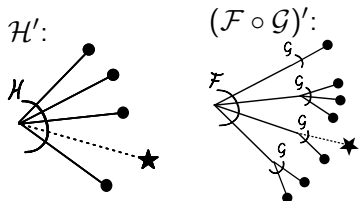


## Derivative



species	derivative
$\mathcal{A} + \mathcal{B}$	$\mathcal{A}' + \mathcal{B}'$
$\mathcal{A} \cdot \mathcal{B}$	$\mathcal{A}' \cdot \mathcal{B} + \mathcal{A} \cdot \mathcal{B}'$
$\text{SEQ}(\mathcal{B})$	$\text{SEQ}(\mathcal{B}) \cdot \mathcal{B}' \cdot \text{SEQ}(\mathcal{B})$
$\text{CYC}(\mathcal{B})$	$\text{SEQ}(\mathcal{B}) \cdot \mathcal{B}'$
$\text{SET}(\mathcal{B})$	$\text{SET}(\mathcal{B}) \cdot \mathcal{B}'$

## Derivative



Example:

species	derivative
$A + B$	$A' + B'$
$A \cdot B$	$A' \cdot B + A \cdot B'$
$\text{SEQ}(B)$	$\text{SEQ}(B) \cdot B' \cdot \text{SEQ}(B)$
$\text{CYC}(B)$	$\text{SEQ}(B) \cdot B'$
$\text{SET}(B)$	$\text{SET}(B) \cdot B'$

$$\mathcal{H}(\mathcal{G}, \mathcal{S}, \mathcal{P}) := (\mathcal{S} + \mathcal{P}, \text{Seq}(\mathcal{Z} + \mathcal{P}), \text{Set}(\mathcal{Z} + \mathcal{S})).$$

$$\frac{\partial \mathcal{H}}{\partial \mathcal{Y}} = \begin{pmatrix} \emptyset & \mathcal{E} & \mathcal{E} \\ \emptyset & \emptyset & \text{Seq}(\mathcal{Z} + \mathcal{P}) \cdot \mathcal{E} \cdot \text{Seq}(\mathcal{Z} + \mathcal{P}) \\ \emptyset & \text{Set}(\mathcal{Z} + \mathcal{S}) \cdot \mathcal{E} & \emptyset \end{pmatrix}$$

# Joyal's Implicit Species Theorem

## Theorem

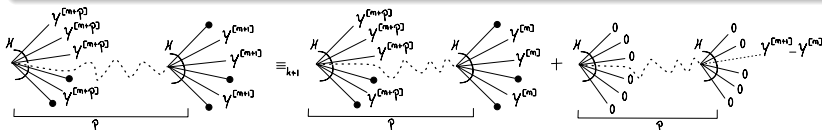
If  $\mathcal{H}(0, 0) = 0$  and  $\partial\mathcal{H}/\partial\mathcal{Y}(0, 0)$  is nilpotent, then  $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$  has a unique solution, limit of

$$\mathcal{Y}^{[0]} = 0, \quad \mathcal{Y}^{[n+1]} = \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) \quad (n \geq 0).$$

**Def.**  $\mathcal{A} =_k \mathcal{B}$  if they coincide up to size  $k$  (contact  $k$ ).

## Key Lemma

If  $\mathcal{Y}^{[n+1]} =_k \mathcal{Y}^{[n]}$ , then  $\mathcal{Y}^{[n+p+1]} =_{k+1} \mathcal{Y}^{[n+p]}$ , ( $p = \text{dimension}$ ).



# Joyal's Implicit Species Theorem

## Theorem

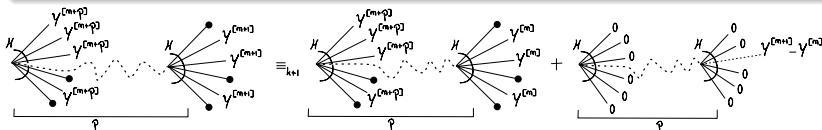
If  $\mathcal{H}(0, 0) = 0$  and  $\partial\mathcal{H}/\partial\mathcal{Y}(0, 0)$  is nilpotent, then  $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$  has a unique solution, limit of

$$\mathcal{Y}^{[0]} = 0, \quad \mathcal{Y}^{[n+1]} = \mathcal{H}(\mathcal{Z}, \mathcal{Y}^{[n]}) \quad (n \geq 0).$$

**Def.**  $\mathcal{A} =_k \mathcal{B}$  if they coincide up to size  $k$  (contact  $k$ ).

## Key Lemma

If  $\mathcal{Y}^{[n+1]} =_k \mathcal{Y}^{[n]}$ , then  $\mathcal{Y}^{[n+p+1]} =_{k+1} \mathcal{Y}^{[n+p]}$ , ( $p = \text{dimension}$ ).



# Newton Iteration for Binary Trees

$$\mathcal{Y} = \mathcal{E} \cup \mathcal{Z} \times \mathcal{Y}^2$$



# Newton Iteration for Binary Trees

$$\mathcal{Y} = \mathcal{E} \cup \mathcal{Z} \times \mathcal{Y}^2$$

$$\mathcal{Y}_{n+1} = \mathcal{Y}_n \cup \text{SEQ}(\mathcal{Z} \times \mathcal{Y}_n \times \square \cup \mathcal{Z} \times \square \times \mathcal{Y}_n) \times (\mathcal{E} \cup \mathcal{Z} \times \mathcal{Y}_n^2 \setminus \mathcal{Y}_n).$$

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

$$\mathcal{Y}_2 = \begin{array}{c} \circ \\ \circ \end{array} + \begin{array}{c} \circ \\ \circ \end{array} + \begin{array}{c} \circ \\ \circ \end{array} + \begin{array}{c} \circ \\ \circ \end{array} + \dots + \begin{array}{c} \circ \\ \circ \end{array} + \dots$$

②

$$\mathcal{Y}_3 = \mathcal{Y}_2 + \begin{array}{c} \circ \\ \circ \end{array} + \dots + \begin{array}{c} \circ \\ \circ \end{array} + \dots + \begin{array}{c} \circ \\ \circ \end{array} + \dots$$

⑥

[Décoste, Labelle, Leroux 1982]

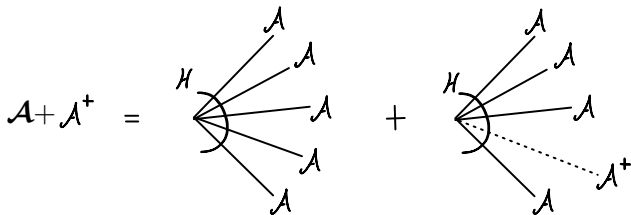
# Combinatorial Newton Iteration

## Theorem (essentially Labelle)

For any well-founded system  $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ , if  $\mathcal{A}$  has contact  $k$  with the solution and  $\mathcal{A} \subset \mathcal{H}(\mathcal{Z}, \mathcal{A})$ , then

$$\mathcal{A} + \sum_{i \geq 0} \left( \frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{A}) \right)^i \cdot (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A})$$

has contact  $2k + 1$  with it.



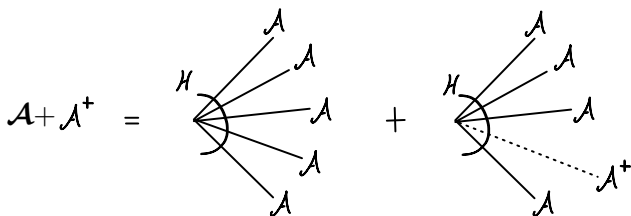
# Combinatorial Newton Iteration

## Theorem (essentially Labelle)

For any well-founded system  $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ , if  $\mathcal{A}$  has contact  $k$  with the solution and  $\mathcal{A} \subset \mathcal{H}(\mathcal{Z}, \mathcal{A})$ , then

$$\mathcal{A} + \sum_{i \geq 0} \left( \frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{A}) \right)^i \cdot (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A})$$

has contact  $2k + 1$  with it.



Generation by increasing Strahler numbers.

## III Newton Iteration for Power Series

# Newton Did It in 1671!

$$y^3 + a^2y - 2a^3 + axy - x^3 = 0, y = a - \frac{x}{4} + \frac{x^2}{64a} + \frac{111x^3}{112a^2} + \frac{509x^4}{16384a^3} \&c.$$

$+ a + p = y.$	$+ y^3$ $- axy$ $+ a^2y$ $- x^3$ $- 2a^3$	$+ a^3 + 3a^2p + 3ap^2 + p^3$ $+ a^2x + axp$ $+ a^3 + a^2p$ $- x^3$ $- 2a^3$
$- \frac{1}{2}x + q = p.$	$+ p^3$ $+ 3ap^2$ $- axp$ $+ 4a^2p$ $+ a^2x$ $- x^3$	$- \frac{1}{8}x^3 + \frac{3}{16}x^2q - \frac{1}{2}xq^2 + q^3$ $+ \frac{3}{16}ax^2 - \frac{1}{2}axq + 3aq^2$ $- \frac{1}{2}ax^2 + axq$ $- a^2x + 4a^2q$ $+ a^2x$ $- x^3$
$+ \frac{x^3}{64a} + r = q.$	$+ q^3$ $- \frac{1}{2}xq^2$ $+ 3aq^2$ $+ \frac{1}{16}x^2q$ $- \frac{1}{2}axq$ $+ 4a^2q$ $- \frac{3}{64}x^3$ $- \frac{3}{16}ax^2$	$*$ $*$ $+ \frac{3x^4}{4096a} * + \frac{1}{16}x^3r + 3ar^2$ $+ \frac{3x^4}{1024a} * + \frac{1}{16}x^2r$ $- \frac{1}{16}axr$ $+ \frac{1}{16}ax^2 + 4a^2r$ $- \frac{3}{64}x^3$ $- \frac{3}{16}ax^2$
$+ 4a^2 - \frac{1}{2}ax + \frac{1}{16}x^2 + \frac{111}{112}x^3 - \frac{509x^4}{16384a^3}$	$+ \frac{111x^3}{112a^2} + \frac{509x^4}{16384a^3}$	



# Generating Series: a Simple Dictionary

$$\text{ogf} := \sum_{t \in \mathcal{T}} z^{|t|}, \quad \text{egf} := \sum_{t \in \mathcal{T}} \frac{z^{|t|}}{|t|!}.$$

## Language and Gen. Fcns (labelled)

$\mathcal{A} \cup \mathcal{B}$	$A(z) + B(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$
$\text{SEQ}(\mathcal{C})$	$\frac{1}{1-C(z)}$
$\mathcal{A}'$	$A'(z)$
$\text{CYC}(\mathcal{C})$	$\log \frac{1}{1-C(z)}$
$\text{SET}(\mathcal{C})$	$\exp(C(z))$

Consequences:

- 1 Newton iteration for EGFs easy;

# Generating Series: a Simple Dictionary

$$\text{ogf} := \sum_{t \in \mathcal{T}} z^{|t|}, \quad \text{egf} := \sum_{t \in \mathcal{T}} \frac{z^{|t|}}{|t|!}.$$

Language	Gen. Fcns (labelled)	(unlabelled)
$\mathcal{A} \cup \mathcal{B}$	$A(z) + B(z)$	$A(z) + B(z)$
$\mathcal{A} \times \mathcal{B}$	$A(z) \times B(z)$	$A(z) \times B(z)$
$\text{SEQ}(\mathcal{C})$	$\frac{1}{1-C(z)}$	$\frac{1}{1-C(z)}$
$\mathcal{A}'$	$A'(z)$	—
$\text{CYC}(\mathcal{C})$	$\log \frac{1}{1-C(z)}$	$\sum_{k \geq 1} \frac{\phi(k)}{k} \log \frac{1}{1-C(z^k)}$
$\text{SET}(\mathcal{C})$	$\exp(C(z))$	$\exp(\sum C(z^i)/i)$

Consequences:

- 1 Newton iteration for EGFs easy;
- 2 Pólya operators for ogfs.

# Newton Iteration for Power Series has Good Complexity

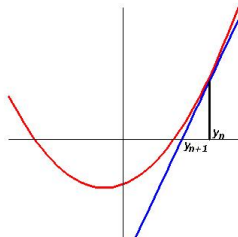
To solve  $\phi(y) = 0$ , iterate

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

Quadratic convergence



Divide-and-Conquer



# Newton Iteration for Power Series has Good Complexity

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

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

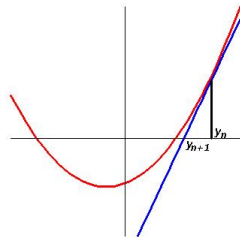
Quadratic convergence



Divide-and-Conquer

To solve at precision  $N$

- 1 Solve at precision  $N/2$ ;
- 2 Compute  $\phi$  and  $\phi'$  there;
- 3 Solve for  $u^{[n+1]}$ .



$$\text{Cost}(y^{[n]}) = \text{constant} \times \text{Cost}(\text{last step}).$$

# Newton Iteration for Power Series has Good Complexity

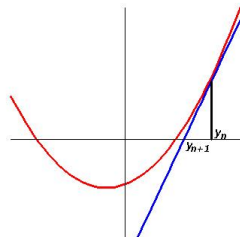
To solve  $\phi(y) = 0$ , iterate

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

Quadratic convergence



Divide-and-Conquer



To solve at precision  $N$

- 1 Solve at precision  $N/2$ ;
- 2 Compute  $\phi$  and  $\phi'$  there;
- 3 Solve for  $u^{[n+1]}$ .

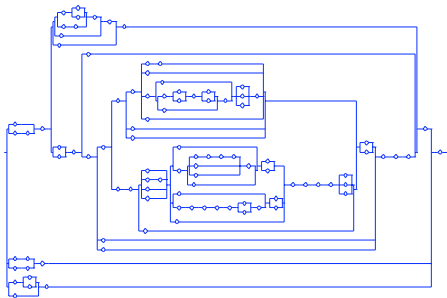
$$\text{Cost}(y^{[n]}) = \text{constant} \times \text{Cost}(\text{last step}).$$

Useful in conjunction with **fast multiplication** (quasi-linear):

- power series at order  $N$ :  $O(N \log N)$  ops on the coefficients;
- $N$ -bit integers:  $O(N \log N \log \log N)$  bit ops.

# Example: Series-Parallel Graphs

$$\begin{cases} \mathcal{G} &= \mathcal{S} + \mathcal{P}, \\ \mathcal{S} &= \text{Seq}(\mathcal{Z} + \mathcal{P}), \\ \mathcal{P} &= \text{Set}_{>0}(\mathcal{Z} + \mathcal{S}). \end{cases} \quad \frac{\partial \mathcal{H}}{\partial \mathbf{y}} = \begin{pmatrix} \emptyset & \mathcal{E} & \mathcal{E} \\ \emptyset & \emptyset & \text{Seq}^2(\mathcal{Z} + \mathcal{P}) \\ \emptyset & \text{Set}(\mathcal{Z} + \mathcal{S}) & \emptyset \end{pmatrix}$$



# Example: Series-Parallel Graphs (labelled case)

$$\begin{cases} G &= S + P, \\ S &= \text{Seq}(Z + P), \\ P &= \text{Set}_{>0}(Z + S). \end{cases} \quad \frac{\partial \mathcal{H}}{\partial \mathbf{Y}} = \begin{pmatrix} \emptyset & \mathcal{E} & \mathcal{E} \\ \emptyset & \emptyset & \text{Seq}^2(Z + P) \\ \emptyset & \text{Set}(Z + S) & \emptyset \end{pmatrix}$$

$$\begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1}, \\ P &= \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

Newton iteration:  $\mathbf{Y}^{[n]} := \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix},$

$$\mathbf{Y}^{[n+1]} = \mathbf{Y}^{[n]} + \left( \text{Id} - \frac{\partial \mathbf{H}}{\partial \mathbf{Y}}(\mathbf{Y}^{[n]}) \right)^{-1} \cdot \left( \mathbf{H}(\mathbf{Y}^{[n]}) - \mathbf{Y}^{[n]} \right).$$

# Example: Series-Parallel Graphs (labelled case)

$$\begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1}, \\ P &= \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

Newton iteration:  $\mathbf{Y}^{[n]} := \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix},$

$$\mathbf{Y}^{[n+1]} = \mathbf{Y}^{[n]} + \left( \text{Id} - \frac{\partial \mathbf{H}}{\partial \mathbf{Y}}(\mathbf{Y}^{[n]}) \right)^{-1} \cdot \left( \mathbf{H}(\mathbf{Y}^{[n]}) - \mathbf{Y}^{[n]} \right).$$

# Example: Series-Parallel Graphs (labelled case)

$$\begin{cases} G = S + P, \\ S = (1 - z - P)^{-1}, \\ P = \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

Newton iteration:  $\mathbf{Y}^{[n]} := \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix},$

$$\mathbf{Y}^{[n+1]} = \mathbf{Y}^{[n]} + \left( \text{Id} - \frac{\partial \mathbf{H}}{\partial \mathbf{Y}}(\mathbf{Y}^{[n]}) \right)^{-1} \cdot \left( \mathbf{H}(\mathbf{Y}^{[n]}) - \mathbf{Y}^{[n]} \right) \bmod z^{2^{n+1}}.$$

# Example: Series-Parallel Graphs (labelled case)

$$\begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1}, \\ P &= \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

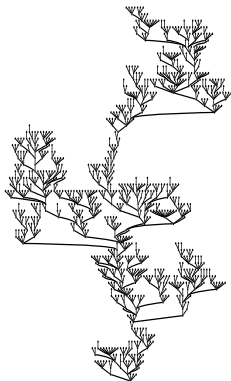
Newton iteration:  $\mathbf{Y}^{[n]} := \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix},$

$$\mathbf{Y}^{[n+1]} = \mathbf{Y}^{[n]} + \left( \text{Id} - \frac{\partial \mathbf{H}}{\partial \mathbf{Y}}(\mathbf{Y}^{[n]}) \right)^{-1} \cdot \left( \mathbf{H}(\mathbf{Y}^{[n]}) - \mathbf{Y}^{[n]} \right) \bmod z^{2^{n+1}}.$$

$\Rightarrow$  **Wanted:** efficient matrix inverse, efficient exp.

# Example: Unlabelled Rooted Trees

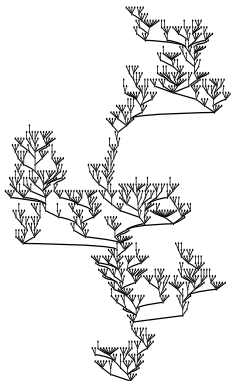
- ① Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;



# Example: Unlabelled Rooted Trees

- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;
- 2 Combinatorial Newton iteration:

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



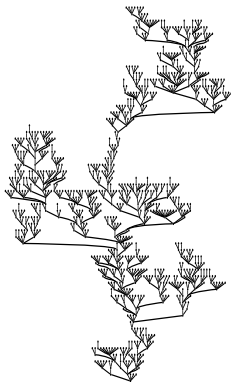
# Example: Unlabelled Rooted Trees

- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;
- 2 Combinatorial Newton iteration:

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

- 3 OGF equation:  $\tilde{Y}(z) = H(z, \tilde{Y}(z))$

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



# Example: Unlabelled Rooted Trees

- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;
- 2 Combinatorial Newton iteration:

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

- 3 OGF equation:  $\tilde{Y}(z) = H(z, \tilde{Y}(z))$

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

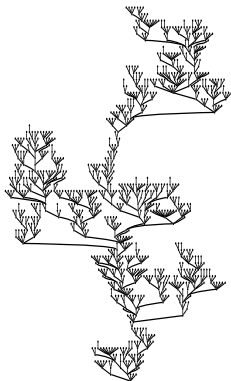
- 4 Newton for OGF:

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

0,

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

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



# Newton Iteration for Inverses

$$\phi(y) = a - 1/y \Rightarrow 1/\phi'(y) = y^2 \Rightarrow \boxed{y^{[n+1]} = y^{[n]} - y^{[n]}(ay^{[n]} - 1)}.$$

Cost: a small number of multiplications

Works for:

- ① Numerical inversion;
- ② Reciprocal of power series;
- ③ Inversion of matrices.

Applications:

- SEQ
- $(I - \frac{\partial H}{\partial Y})^{-1}$

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

# Inverses for Series-Parallel Graphs

$$\begin{cases} G = S + P, \\ S = (1 - z - P)^{-1}, \\ P = \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

Newton iteration:

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

Can be lifted combinatorially.  
Also a numerical iteration!

# Inverses for Series-Parallel Graphs

$$\begin{cases} G &= S + P, \\ S &= (1 - z - P)^{-1}, \\ P &= \exp(z + S) - 1. \end{cases} \quad \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & (1 - z - P)^{-2} \\ 0 & \exp(z + S) & 0 \end{pmatrix}$$

Newton iteration:

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

Can be lifted combinatorially.

Also a numerical iteration!

⇒ **Wanted:** efficient exp.

# From the Inverse to the Exponential

- ① Logarithm of power series:  $\log f = \int (f'/f)$ ;
- ② exponential of power series:  $\phi(y) = a - \log y$ .

$$\begin{aligned}
 e^{[n+1]} &= e^{[n]} + \frac{a - \log e^{[n]}}{1/e^{[n]}} \bmod z^{2^{n+1}}, \\
 &= e^{[n]} + e^{[n]} \left( a - \int e^{[n]}'/e^{[n]} \right) \bmod z^{2^{n+1}}.
 \end{aligned}$$

And  $1/e^{[n]}$  is computed by Newton iteration too!

[Brent 1975]

# Application: Power Sums

$$F = t^N + a_{N-1}t^{N-1} + \cdots + a_0 \leftrightarrow S_i = \sum_{F(\alpha)=0} \alpha^i, \quad i = 0, \dots, N.$$

# Application: Power Sums

$$F = t^N + a_{N-1}t^{N-1} + \cdots + a_0 \leftrightarrow S_i = \sum_{F(\alpha)=0} \alpha^i, \quad i = 0, \dots, N.$$

Fast conversion using the generating series:

$$\frac{\text{rev}(F)'}{\text{rev}(F)} = - \sum_{i \geq 0} S_{i+1} t^i \leftrightarrow \text{rev}(F) = \exp \left( - \sum \frac{S_i}{i} t^i \right).$$

# Application: Power Sums

$$F = t^N + a_{N-1}t^{N-1} + \cdots + a_0 \leftrightarrow S_i = \sum_{F(\alpha)=0} \alpha^i, \quad i = 0, \dots, N.$$

Fast conversion using the generating series:

$$\frac{\text{rev}(F)'}{\text{rev}(F)} = - \sum_{i \geq 0} S_{i+1} t^i \leftrightarrow \text{rev}(F) = \exp \left( - \sum \frac{S_i}{i} t^i \right).$$

Application: composed product and sums

$$(F, G) \mapsto \prod_{F(\alpha)=0, G(\beta)=0} (t - \alpha\beta) \quad \text{or} \quad \prod_{F(\alpha)=0, G(\beta)=0} (t - (\alpha + \beta)).$$

# Application: Power Sums

$$F = t^N + a_{N-1}t^{N-1} + \cdots + a_0 \leftrightarrow S_i = \sum_{F(\alpha)=0} \alpha^i, \quad i = 0, \dots, N.$$

Fast conversion using the generating series:

$$\frac{\text{rev}(F)'}{\text{rev}(F)} = - \sum_{i \geq 0} S_{i+1} t^i \leftrightarrow \text{rev}(F) = \exp \left( - \sum \frac{S_i}{i} t^i \right).$$

Application: composed product and sums

$$(F, G) \mapsto \prod_{F(\alpha)=0, G(\beta)=0} (t - \alpha\beta) \quad \text{or} \quad \prod_{F(\alpha)=0, G(\beta)=0} (t - (\alpha + \beta)).$$

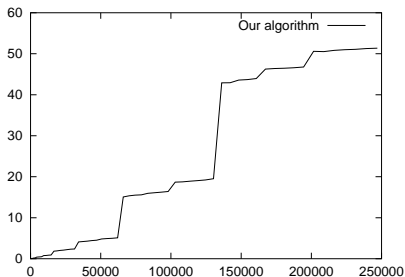
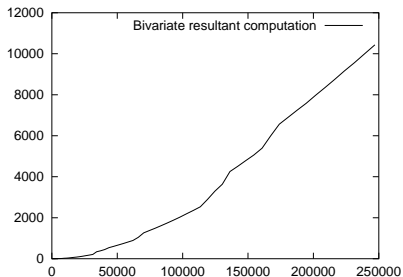
Easy in Newton representation:  $\sum \alpha^s \sum \beta^s = \sum (\alpha\beta)^s$  and

$$\sum \frac{\sum (\alpha + \beta)^s}{s!} t^s = \left( \sum \frac{\sum \alpha^s}{s!} t^s \right) \left( \sum \frac{\sum \beta^s}{s!} t^s \right).$$

[Schönhage 1982; Bostan, Flajolet, Salvy, Schost 2006]

# Timings

Applications (crypto): over finite fields, degree  $> 200000$  expected.



Timings in seconds vs. output degree  $N$ , over  $\mathbb{F}_p$ , 26 bits prime  $p$

# Exponential for Series-Parallel Graphs

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

**compiles** into the Newton iteration:

$$\left\{ \begin{array}{l} i^{[n+1]} = i^{[n]} - i^{[n]}(e^{[n]}i^{[n]} - 1), \\ e^{[n+1]} = e^{[n]} - e^{[n]} \left( 1 + \frac{d}{dz} S^{[n]} - \int \left( \frac{d}{dz} e^{[n]} \right) i^{[n]} \right), \\ v^{[n+1]} = v^{[n]} - v^{[n]}((1 - z - P^{[n]})v^{[n]} - 1), \\ U^{[n+1]} = U^{[n]} + U^{[n]} \cdot \left( \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & v^{[n+1]^2} \\ 0 & e^{[n+1]} & 0 \end{pmatrix} \cdot U^{[n]} + \text{Id} - U^{[n]} \right), \\ \begin{pmatrix} G^{[n+1]} \\ S^{[n+1]} \\ P^{[n+1]} \end{pmatrix} = \begin{pmatrix} G^{[n]} \\ S^{[n]} \\ P^{[n]} \end{pmatrix} + U^{[n+1]} \cdot \begin{pmatrix} S^{[n]} + P^{[n]} - G^{[n]} \\ v^{[n+1]} - S^{[n]} \\ e^{[n+1]} - P^{[n]} \end{pmatrix} \pmod{z^{2^{n+1}}}. \end{array} \right.$$

Computation reduced to products and linear ops.

# Linear Differential Equations of Arbitrary Order

Given a linear differential equation with power series coefficients,

$$a_r(t)y^{(r)}(t) + \cdots + a_0(t)y(t) = 0,$$

compute the first  $N$  terms of a basis of power series solutions.

# Linear Differential Equations of Arbitrary Order

Given a linear differential equation with power series coefficients,

$$a_r(t)y^{(r)}(t) + \cdots + a_0(t)y(t) = 0,$$

compute the first  $N$  terms of a basis of power series solutions.

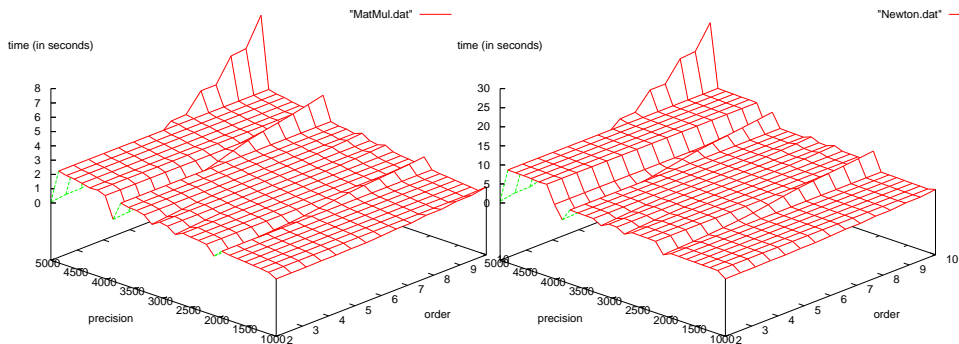
## Algorithm

- 1 Convert into a system  $\Phi : Y \mapsto Y' - A(t)Y$  ( $D\Phi = \Phi$ );
- 2  $D\Phi|_Y(U) = \Phi(Y)$  rewrites  $U' - AU = Y' - AY$ ;
- 3 Variation of constants:  $U = Y \int Y^{-1}(Y' - AY)$ ;
- 4  $Y^{-1}$  by Newton iteration too.

Special case: recover good exponential.

[Bostan, Chyzak, Ollivier, Salvy, Schost, Sedoglavic 2007]

# Timings

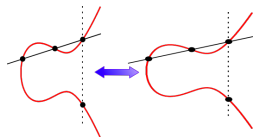


Polynomial matrix multiplication vs. solving  $Y' = AY$ .

# Non-Linear Differential Equations

Example from cryptography:

$$\phi : y \mapsto (x^3 + Ax + B)y'^2 - (y^3 + \tilde{A}y + \tilde{B}).$$



# Non-Linear Differential Equations

Example from cryptography:

$$\phi : y \mapsto (x^3 + Ax + B)y'^2 - (y^3 + \tilde{A}y + \tilde{B}).$$

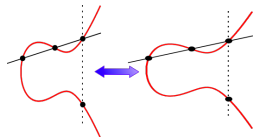
Differential:

$$D\phi|_y : u \mapsto 2(x^3 + Ax + B)y'u' - (3y^2 + \tilde{A})u.$$

Solve the **linear** differential equation

$$D\phi|_y u = \phi(y)$$

at each iteration.



Again, **quasi-linear** complexity.

[Bostan, Morain, Salvy, Schost 2008]

## IV Oracle

# Exponential Generating Series

Arbitrary Combinatorial Specification



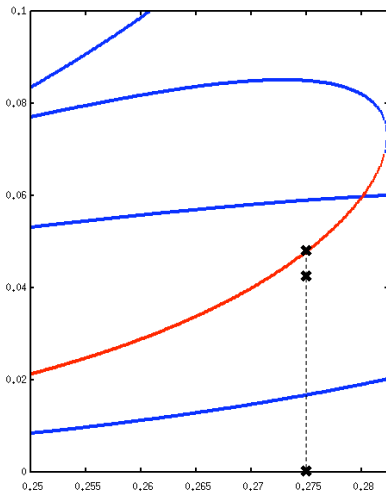
Combinatorial Newton iteration for  $\mathcal{Y}$



Newton iteration for the gf  $Y(z)$   
 (( $y_0, \dots, y_N$ ) fast)



Numerical Newton iteration  
**starting from 0** converges to the  
 value of  $Y(x)$ .



# Exponential Generating Series

Arbitrary Combinatorial Specification



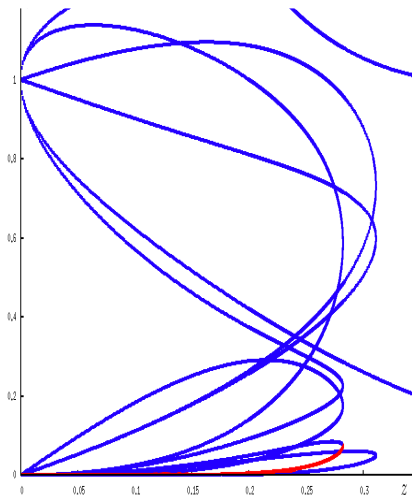
Combinatorial Newton iteration for  $\mathcal{Y}$



Newton iteration for the gf  $Y(z)$   
 $((y_0, \dots, y_N)$  fast)

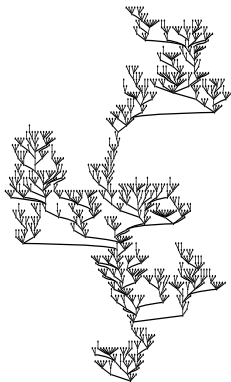


Numerical Newton iteration  
**starting from 0** converges to the  
 value of  $Y(x)$ .



# Ordinary Generating Function on an Example

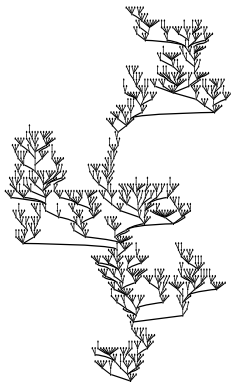
- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;



# Ordinary Generating Function on an Example

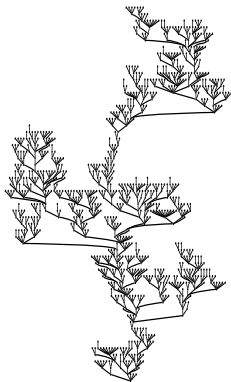
- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y});$
- 2 Combinatorial Newton iteration:  

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



# Ordinary Generating Function on an Example

- 1 Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;
- 2 Combinatorial Newton iteration:  
 $\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \text{SEQ}(\mathcal{H}(\mathcal{Y}^{[n]})) \cdot (\mathcal{H}(\mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]})$
- 3 OGF equation:  $\tilde{Y}(z) = H(z, \tilde{Y}(z))$   
 $= z \exp(\tilde{Y}(z) + \frac{1}{2} \tilde{Y}(z^2) + \frac{1}{3} \tilde{Y}(z^3) + \dots)$



# Ordinary Generating Function on an Example

① Well-founded system:  $\mathcal{Y} = \mathcal{Z} \cdot \text{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$ ;

② Combinatorial Newton iteration:

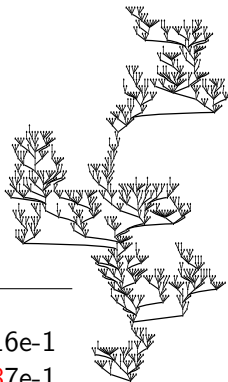
$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \text{SEQ}(\mathcal{H}(\mathcal{Y}^{[n]})) \cdot (\mathcal{H}(\mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]})$$

③ OGF equation:  $\tilde{Y}(z) = H(z, \tilde{Y}(z))$

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

④ Numerical iteration:

$n$	$\tilde{Y}^{[n]}(0.3)$	$\tilde{Y}^{[n]}(0.3^2)$	$\tilde{Y}^{[n]}(0.3^3)$
0	0	0	0
1	.43021322639	0.99370806338e-1	0.27759817516e-1
2	.54875612912	0.99887132154e-1	0.27770629187e-1
3	.55709557053	0.99887147197e-1	0.27770629189e-1
4	.55713907945	0.99887147198e-1	0.27770629189e-1
5	.55713908064	0.99887147198e-1	0.27770629189e-1



# V Conclusion

# Conclusion

- Summary:
  - Newton iteration has good complexity;
  - Oracle: numerical Newton iteration that gives the values of ... power series that are the gfs of ... combinatorial iterates.

# Conclusion

- Summary:
  - Newton iteration has good complexity;
  - Oracle: numerical Newton iteration that gives the values of ... power series that are the gfs of ... combinatorial iterates.
- Read the paper for:
  - Polynomial species and systems with  $\mathcal{E}$ ;
  - Majorant species;
  - Combinatorial differential systems and linear species  

$$\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y}) + \int \mathcal{G}(\mathcal{Z}, \mathcal{Y})$$
 (Leroux-Viennot, Labelle; combinatorial lifting of Bostan et alii);
  - PowerSet (it is not a species).

# Conclusion

- Summary:
  - Newton iteration has good complexity;
  - Oracle: numerical Newton iteration that gives the values of ... power series that are the gfs of ... combinatorial iterates.
- Read the paper for:
  - Polynomial species and systems with  $\mathcal{E}$ ;
  - Majorant species;
  - Combinatorial differential systems and linear species  
 $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y}) + \int \mathcal{G}(\mathcal{Z}, \mathcal{Y})$   
 (Leroux-Viennot, Labelle; combinatorial lifting of Bostan et alii);
  - PowerSet (it is not a species).

**THE END**

