Newton Iteration in Computer Algebra and Combinatorics

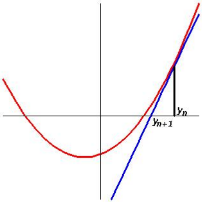
Bruno Salvy
Bruno.Salvy@inria.fr

Inria AriC Project, LIP ENS Lyon

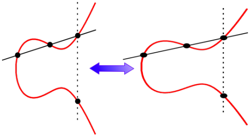
Joint work with Carine Pivoteau and Michèle Soria,
Journal of Combinatorial Theory, Series A **119** (2012), 1711–1773.

Université Paris Diderot, April 2013

I Introduction

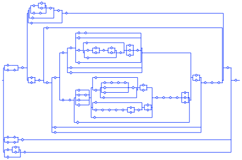


Analysis



Computer Algebra

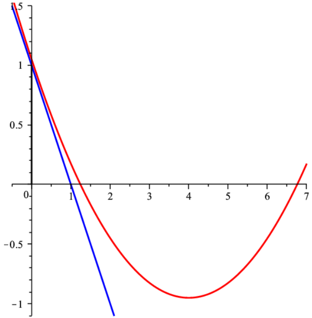
Combinatorics



Numerical Iteration

$$\phi(y) = 1 + \frac{1}{8}y^2 - y$$

$$y^{[n+1]} = y^{[n]} - \frac{\phi(y^{[n]})}{\phi'(y^{[n]})} = y^{[n]} + \frac{1 + {y^{[n]}}^2/8 - y^{[n]}}{1 - y^{[n]}/4}$$



$$y^{[0]} = 0.$$
$$y^{[1]} = 1.0000000000000000000000000$$
$$y^{[2]} = 1.1666666666666666666666666$$
$$y^{[3]} = 1.1715686274509803921568627$$
$$y^{[4]} = 1.1715728752506201787474088$$
$$y^{[5]} = 1.1715728752538099023966207$$

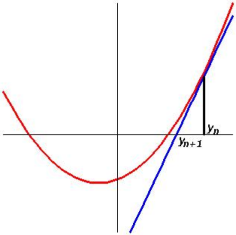
Quadratic Convergence

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

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

Good case: quadratic convergence if

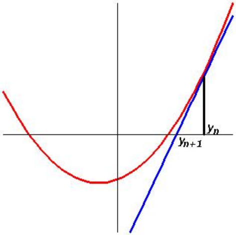
- the initial point is close enough;
- the root is simple.



Quadratic Convergence

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

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



Good case: quadratic convergence if

- the initial point is close enough;
- the root is simple.

Proof: simple root at $\zeta \Rightarrow \phi'(\zeta) \neq 0$,

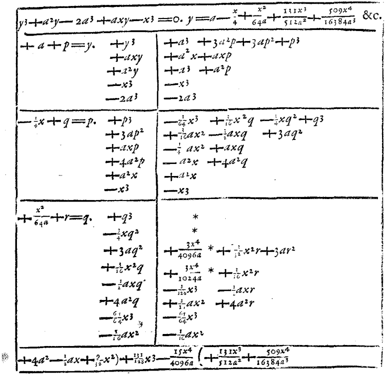
$$\left.\begin{aligned}\phi(y^{[n]}) &= \phi'(\zeta)(y^{[n]}-\zeta) + O((y^{[n]}-\zeta)^2)\\ \phi'(y^{[n]}) &= \phi'(\zeta) + O(y^{[n]}-\zeta)\end{aligned}\right\} \Rightarrow y^{[n]} - \zeta = \frac{\phi(y^{[n]})}{\phi'(y^{[n]})} + O((y^{[n]}-\zeta)^2),$$

$$\Rightarrow y^{[n+1]} - \zeta = O((y^{[n]}-\zeta)^2).$$

Symbolic Iteration

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

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



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

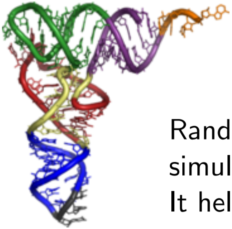
$$y^{[1]} = 1$$

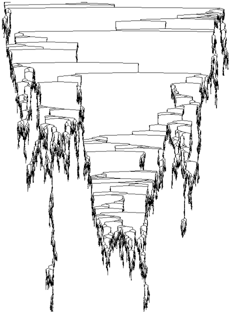
$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$

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

[Newton 1671]

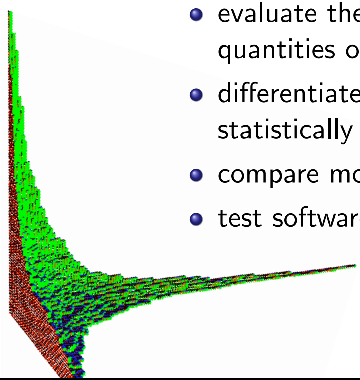
Random Generation in Combinatorics

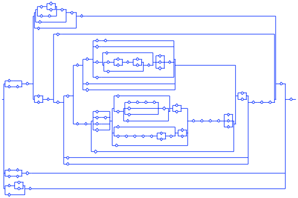




Random generation of large objects = simulation in the discrete world.
It helps

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

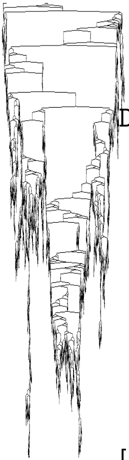




Recursive Method

Binary Trees: $\mathcal{B} = \mathcal{Z} \cup \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

b_k: nb. binary trees with k nodes (Catalan)



$\texttt{DrawBinTree}(n) = \{$

If $n = 1$ return $\mathcal{Z}$

Else $\{$

$U := \text{Uniform}([0,1]); k := 0; S := 0;$

while $(S < U)\{k := k+1; S := S + b_k b_{n-k-1}/b_n;\}$

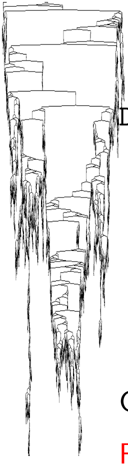
return $\mathcal{Z} \times \texttt{DrawBinTree}(k) \times \texttt{DrawBinTree}(n-k-1)\}\}$

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

Recursive Method

Binary Trees: $\mathcal{B} = \mathcal{Z} \cup \mathcal{Z} \times \mathcal{B} \times \mathcal{B}$

b_k: nb. binary trees with k nodes (Catalan)



$\texttt{DrawBinTree}(n) = \{$

If $n = 1$ return $\mathcal{Z}$

Else $\{$

$U := \text{Uniform}([0,1]); k := 0; S := 0;$

while $(S < U)\{k := k+1; S := S + b_k b_{n-k-1}/b_n;\}$

return $\mathcal{Z} \times \texttt{DrawBinTree}(k) \times \texttt{DrawBinTree}(n-k-1)\}\}$

Generalizes to many recursive structures.

Requires $b_0, \dots, b_n$.

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

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 (**oracle**).

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 (**oracle**).

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 (**oracle**).

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 (**oracle**).

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 $b \in \mathcal{B}$.

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 (**oracle**).

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.

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 $b \in \mathcal{B}$.

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**).

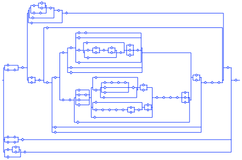
Framework: Constructible Species

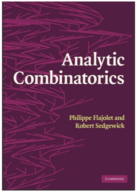
A small set of species

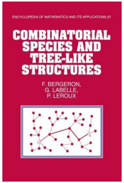
$1, \mathcal{Z}, \times, +, \textsc{Seq}, \textsc{Set}, \textsc{Cyc},$
cardinality constraints that are finite unions of intervals,
used recursively.

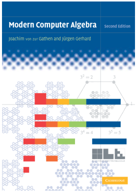
Examples:

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







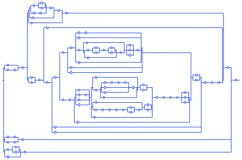


Framework: Constructible Species

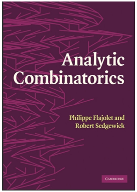
A small set of species

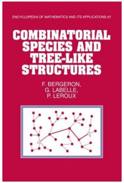
$1, \mathcal{Z}, \times, +, \textsc{Seq}, \textsc{Set}, \textsc{Cyc},$
cardinality constraints that are finite unions of intervals,
used recursively (when it makes sense).

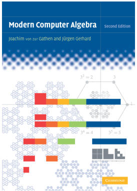
Examples:



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







Results (1/2): Fast Enumeration

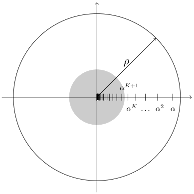
Theorem (Enumeration in Quasi-Optimal Complexity)

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. A numerical iteration converging to $\mathbf{Y}(\alpha)$ in the labelled case (inside the disk);
2. A numerical iteration converging to the sequence $\mathbf{Y}(\alpha), \mathbf{Y}(\alpha^2), \mathbf{Y}(\alpha^3), \ldots$ for $\| \cdot \|_\infty$ in the unlabelled case (inside the disk).



Examples (I): Polynomial Systems

Random generation following given XML grammars

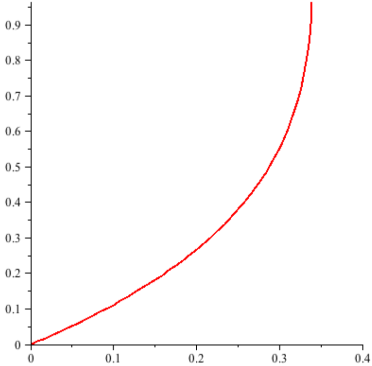
Grammar	nb eqs	max deg	nb sols	oracle (s.)	FGb (s.)
rss	10	5	2	0.02	0.03
PNML	22	4	4	0.05	0.1
xslt	40	3	10	0.4	1.5
relaxng	34	4	32	0.4	3.3
xhtml-basic	53	3	13	1.2	18
mathml2	182	2	18	3.7	882
xhtml	93	6	56	3.4	1124
xhtml-strict	80	6	32	3.0	1590
xmlschema	59	10	24	0.5	6592
SVG	117	10		5.8	$>$1.5Go
docbook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

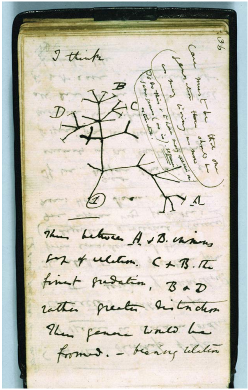
[Darrasse 2008]

Example (II): A Non-Polynomial “System”

Unlabelled rooted trees:

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



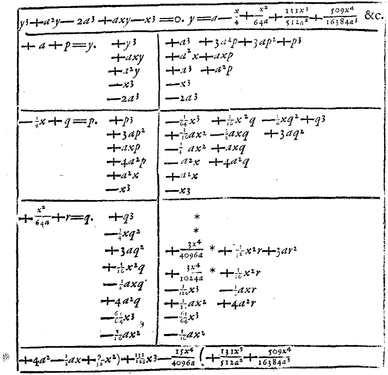


II Newton Iteration for Power Series

Symbolic Iteration

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

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



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

$+a+p=y.$	$+y^3$ $+axy$ $+a^2y$ $-x^3$ $-2a^3$	$+a^3 + 3a^2p + 3ap^2 + p^3$ $+a^2x + axp$ $+a^3 + a^2p$ $-x^3$ $-2a^3$
$-\frac{1}{4}x+q=p.$	$+p^3$ $+3ap^2$ $+axp$ $+4a^2p$ $+a^2x$ $-x^3$	$-\frac{1}{64}x^3 + \frac{3}{16}x^2q - \frac{3}{4}xq^2 + q^3$ $+\frac{3}{16}ax^2 - \frac{3}{2}axq + 3aq^2$ $-\frac{1}{4}ax^2 + axq$ $-a^2x + 4a^2q$ $+a^2x$ $-x^3$
$+\frac{x^2}{64a}+r=q.$	$+q^3$ $-\frac{3}{4}xq^2$ $+3aq^2$ $+\frac{3}{16}x^2q$ $-\frac{1}{2}axq$ $+4a^2q$ $-\frac{65}{64}x^3$ $-\frac{1}{16}ax^2$	* * $+\frac{3x^4}{4096a}$ * $+\frac{3}{32}x^2r + 3ar^2$ $+\frac{3x^4}{1024a}$ * $+\frac{3}{16}x^2r$ $-\frac{1}{128}x^3 \quad -\frac{1}{2}axr$ $+\frac{1}{32}ax^2 \quad +4a^2r$ $-\frac{65}{64}x^3$ $-\frac{1}{16}ax^2$

$+4a^2 - \frac{1}{2}ax + \frac{3}{32}x^2) + \frac{131}{128}x^3 - \frac{15x^4}{4096a} \left(+ \frac{131x^3}{512a^2} + \frac{509x^4}{16384a^3}\right.$

$$y^{[0]} = 0$$
$$y^{[1]} = 1$$
$$y^{[2]} = 1 + z + 2z^2 + 4z^3 + 8z^4 + 16z^5 + 32z^6 + 64z^7 + \dots$$
$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$

[Newton 1671]

Newton Iteration for Power Series has Good Complexity

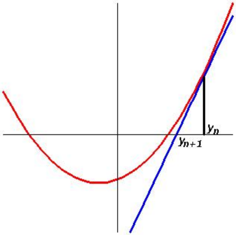
To solve $\phi(y) = 0$, iterate

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

Quadratic convergence

$\Updownarrow$

Divide-and-Conquer

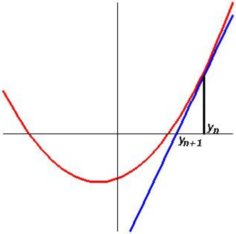


Newton Iteration for Power Series has Good Complexity

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
$\Updownarrow$
Divide-and-Conquer



To solve at precision N

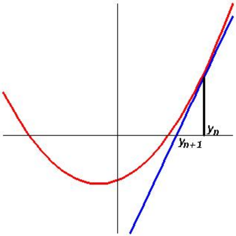
1. Solve at precision $N/2$;
2. Compute ϕ and ϕ' there;
3. Solve for $u^{[n+1]}$.

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

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
$\Updownarrow$
Divide-and-Conquer

To solve at precision N

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

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

Useful in conjunction with fast multiplication (e.g., FFT):

- 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: 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.

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

Inverses for Series-Parallel Graphs

$$(G,S,P)=\mathbf{H}(G,S,P).$$

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

Newton iteration:

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

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

Inverses for Series-Parallel Graphs

$$(G,S,P)=\mathbf{H}(G,S,P).$$

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

Newton iteration:

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

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

$\Rightarrow$ Wanted: efficient exp.

From the Inverse to the Exponential

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

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

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

[Brent 1975; Hanrot-Zimmermann 2002]

Application: Power Sums

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

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:

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

Application: Power Sums

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

Fast conversion using the generating series:

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

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

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

Fast conversion using the generating series:

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

Application: composed product and sums

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

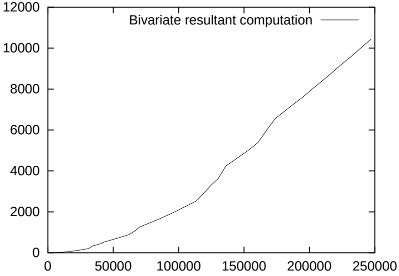
Easy in Newton representation: $\sum \alpha^s \sum \beta^s = \sum (\alpha\beta)^s$ and

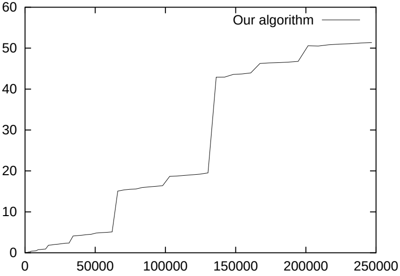
$$\sum \frac{\sum (\alpha + \beta)^s}{s!} t^s = \left(\sum \frac{\sum \alpha^s}{s!} t^s\right) \left(\sum \frac{\sum \beta^s}{s!} t^s\right).$$

[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

Conclusion for Series-Parallel Graphs

$$\mathcal{G} = \mathcal{S} + \mathcal{P}, \quad \mathcal{S} = \textsc{Seq}_{>0}(\mathcal{Z} + \mathcal{P}), \quad \mathcal{P} = \textsc{Set}_{>1}(\mathcal{Z} + \mathcal{S})$$

compiles into the Newton iteration:

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

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.

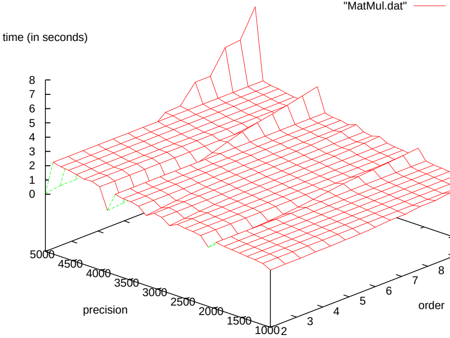
Algorithm

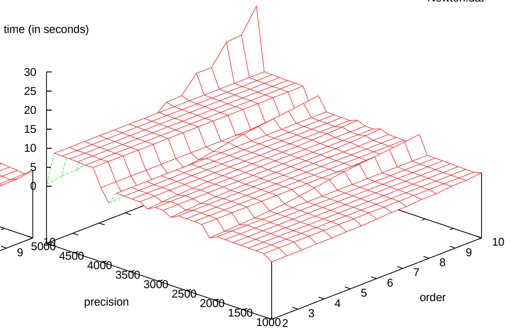
1. Convert into a system $\Phi : Y \mapsto Y' - A(t)Y$ $(D\Phi = \Phi)$;
2. $D\Phi|_Y(U) = \Phi(Y)$ rewrites $U' - AU = Y' - AY$;
3. Variation of constants: $U = Y\int Y^{-1}(Y' - AY)$;
4. Y^{-1} by Newton iteration too.

Special case: recover good exponential.

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

Timings



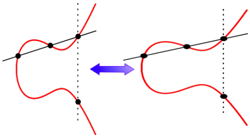


Polynomial matrix multiplication vs. solving $Y' = AY$.

Non-Linear Differential Equations

Example from cryptography:

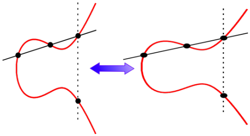
$$\phi : y \mapsto (x^3 + Ax + B)y'^2 - (y^3 + \tilde{A}y + \tilde{B}).$$



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]

III Combinatorics

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)

$\mathcal{A}\cup\mathcal{B}$	$A(z)+B(z)$
$\mathcal{A}\times\mathcal{B}$	$A(z)\times B(z)$
$\textsc{Seq}(\mathcal{C})$	$\frac{1}{1-C(z)}$
$\mathcal{A}'$	$A'(z)$
$\textsc{Cyc}(\mathcal{C})$	$\log\frac{1}{1-C(z)}$
$\textsc{Set}(\mathcal{C})$	$\exp(C(z))$

Consequences:

1. Newton 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 and Gen.	Fcns (labelled)	(unlabelled)
$\mathcal{A}\cup\mathcal{B}$	$A(z)+B(z)$	$A(z)+B(z)$
$\mathcal{A}\times\mathcal{B}$	$A(z)\times B(z)$	$A(z)\times B(z)$
$\textsc{Seq}(\mathcal{C})$	$\frac{1}{1-C(z)}$	$\frac{1}{1-C(z)}$
$\mathcal{A}'$	$A'(z)$	—
$\textsc{Cyc}(\mathcal{C})$	$\log\frac{1}{1-C(z)}$	$\sum_{k\geq1}\frac{\phi(k)}{k}\log\frac{1}{1-C(z^k)}$
$\textsc{Set}(\mathcal{C})$	$\exp(C(z))$	$\exp(\sum C(z^i)/i)$

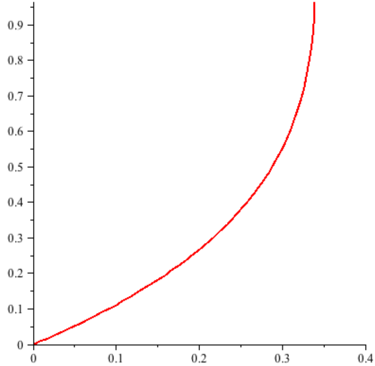
Consequences:

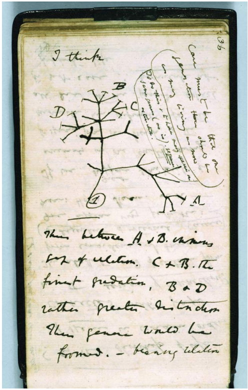
1. Newton for EGFs easy;
2. Pólya operators for ogfs;
3. Newton iteration more difficult for ogfs.

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

Unlabelled rooted trees:

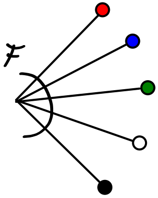
$$f(x) = x \exp\left(f(x) + \frac{1}{2} f(x^2) + \frac{1}{3} f(x^3) + \cdots\right)$$





Mini-Introduction to Species Theory

- Species $\mathcal{F}$:

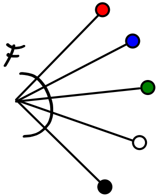


Examples:

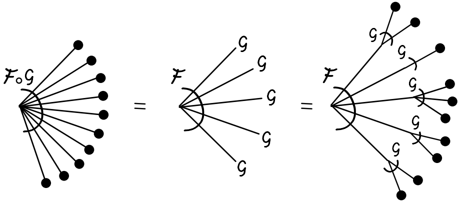
- 0, $\mathcal{Z}$, 1;
- SET;
- SEQ, CYC.

Mini-Introduction to Species Theory

- Species $\mathcal{F}$:



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

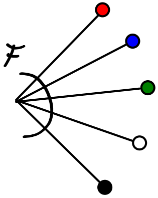


Examples:

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

Mini-Introduction to Species Theory

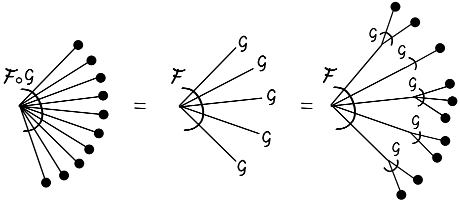
- Species $\mathcal{F}$:



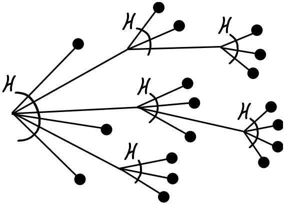
Examples:

- 0, $\mathcal{Z}$, 1;
- Set;
- Seq, Cyc.

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

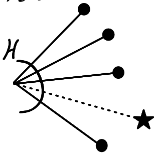


- $\mathcal{Y} = \mathcal{H}(\mathcal{Z}, \mathcal{Y})$



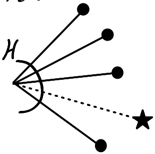
Derivative

$\mathcal{H}'$:

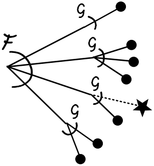


Derivative

$\mathcal{H}'$:

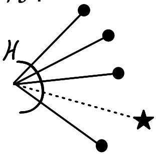


$(\mathcal{F} \circ \mathcal{G})'$:

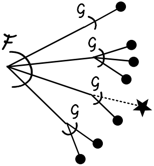


Derivative

$\mathcal{H}'$:



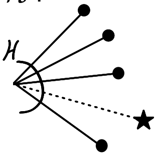
$(\mathcal{F} \circ \mathcal{G})'$:



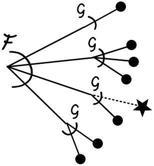
Huet's zipper

Derivative

$\mathcal{H}'$:

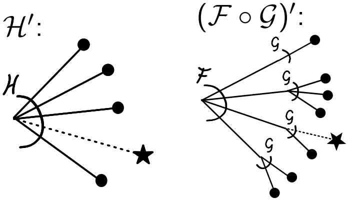


$(\mathcal{F} \circ \mathcal{G})'$:



species	derivative
$\mathcal{A}+\mathcal{B}$	$\mathcal{A}'+\mathcal{B}'$
$\mathcal{A}\cdot\mathcal{B}$	$\mathcal{A}'\cdot\mathcal{B}+\mathcal{A}\cdot\mathcal{B}'$
$\textsc{Seq}(\mathcal{B})$	$\textsc{Seq}(\mathcal{B})\cdot\mathcal{B}'\cdot\textsc{Seq}(\mathcal{B})$
$\textsc{Cyc}(\mathcal{B})$	$\textsc{Seq}(\mathcal{B})\cdot\mathcal{B}'$
$\textsc{Set}(\mathcal{B})$	$\textsc{Set}(\mathcal{B})\cdot\mathcal{B}'$

Derivative



species	derivative
$\mathcal{A}+\mathcal{B}$	$\mathcal{A}'+\mathcal{B}'$
$\mathcal{A}\cdot\mathcal{B}$	$\mathcal{A}'\cdot\mathcal{B}+\mathcal{A}\cdot\mathcal{B}'$
$\textsc{Seq}(\mathcal{B})$	$\textsc{Seq}(\mathcal{B})\cdot\mathcal{B}'\cdot\textsc{Seq}(\mathcal{B})$
$\textsc{Cyc}(\mathcal{B})$	$\textsc{Seq}(\mathcal{B})\cdot\mathcal{B}'$
$\textsc{Set}(\mathcal{B})$	$\textsc{Set}(\mathcal{B})\cdot\mathcal{B}'$

Example:

$$\boldsymbol{\mathcal{H}}(\mathcal{G},\mathcal{S},\mathcal{P}) := (\mathcal{S}+\mathcal{P}, \mathsf{Seq}_{>0}(\mathcal{Z}+\mathcal{P}), \mathsf{Set}_{>1}(\mathcal{Z}+\mathcal{S}))\,.$$

$$\frac{\partial\boldsymbol{\mathcal{H}}}{\partial\boldsymbol{\mathcal{Y}}} = \begin{pmatrix} \varnothing & 1 & 1 \\ \varnothing & \varnothing & \mathsf{Seq}(\mathcal{Z}+\mathcal{P})\cdot 1\cdot\mathsf{Seq}(\mathcal{Z}+\mathcal{P}) \\ \varnothing & \mathsf{Set}_{>0}(\mathcal{Z}+\mathcal{S})\cdot 1 & \varnothing \end{pmatrix}$$

Joyal's Implicit Species Theorem

Theorem

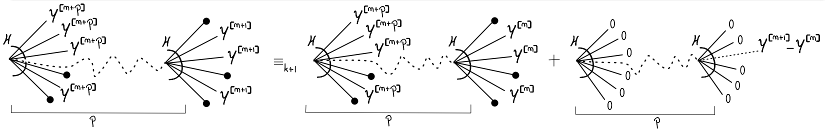
If $\mathcal{H}(0,0) = 0$ and $\partial\mathcal{H}/\partial\mathcal{Y}(0,0)$ is nilpotent, then $\mathcal{Y} = \mathcal{H}(\mathcal{Z},\mathcal{Y})$ has a unique solution, limit of

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

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



Joyal's Implicit Species Theorem

Theorem

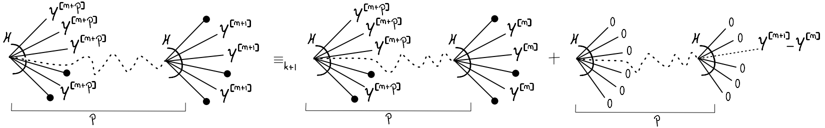
If $\mathcal{H}(0,0) = 0$ *and* $\partial\mathcal{H}/\partial\mathcal{Y}(0,0)$ *is nilpotent, then* $\mathcal{Y} = \mathcal{H}(\mathcal{Z},\mathcal{Y})$ *has a unique solution, limit of*

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

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



We prove an iff when no 0 coordinate.

Newton Iteration for Binary Trees

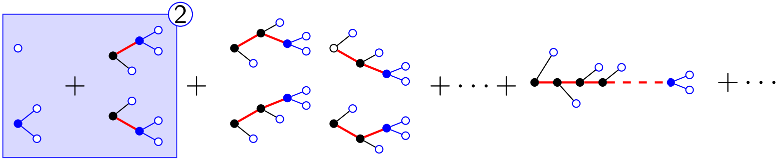
$$\mathcal{Y} = 1 \cup \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y}$$

Newton Iteration for Binary Trees

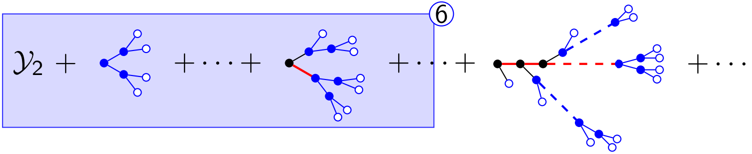
$$\mathcal{Y} = 1 \cup \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y}$$

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

$\mathcal{Y}_2 =$



$\mathcal{Y}_3 =$



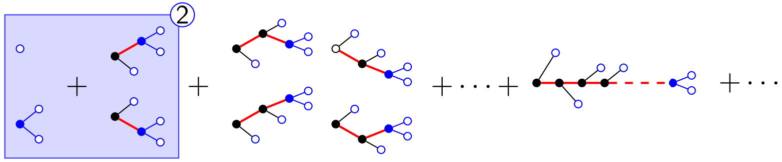
[Décoste, Labelle, Leroux 1982]

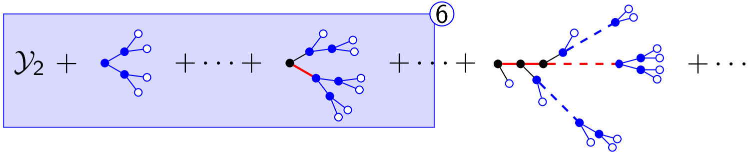
Newton Iteration for Binary Trees

$$\mathcal{Y} = 1 \cup \mathcal{Z} \times \mathcal{Y} \times \mathcal{Y}$$

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

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

$\mathcal{Y}_2 =$ 

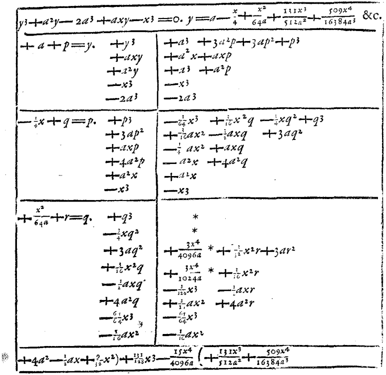
$\mathcal{Y}_3 =$ 

[Décoste, Labelle, Leroux 1982]

Symbolic Iteration

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

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



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

$+a+p=y.$	$+y^3$ $+axy$ $+a^2y$ $-x^3$ $-2a^3$	$+a^3 + 3a^2p + 3ap^2 + p^3$ $+a^2x + axp$ $+a^3 + a^2p$ $-x^3$ $-2a^3$
$-\frac14 x + q = p.$	$+p^3$ $+3ap^2$ $+axp$ $+4a^2p$ $+a^2x$ $-x^3$	$-\frac{1}{64}x^3 + \frac{3}{16}x^2q - \frac34 xq^2 + q^3$ $+\frac{3}{16}ax^2 - \frac32 axq + 3aq^2$ $-\frac14 ax^2 + axq$ $-a^2x + 4a^2q$ $+a^2x$ $-x^3$
$+\frac{x^2}{64a} + r = q.$	$+q^3$ $-\frac34 xq^2$ $+3aq^2$ $+\frac{3}{16}x^2q$ $-\frac12 axq$ $+4a^2q$ $-\frac{65}{64}x^3$ $-\frac{1}{16}ax^2$	* * $+\frac{3x^4}{4096a}$ * $-\frac{3}{32}x^2r + 3ar^2$ $+\frac{3x^4}{1024a}$ * $+\frac{3}{16}x^2r$ $-\frac{1}{128}x^3 \quad -\frac12 axr$ $+\frac{1}{16}ax^2 \quad +4a^2r$ $-\frac{65}{64}x^3$ $-\frac{1}{16}ax^2$

$+4a^2 - \frac12 ax + \frac{9}{32}x^2) + \frac{131}{128}x^3 - \frac{15x^4}{4096a}\ (+\frac{131x^3}{512a^2} + \frac{509x^4}{16384a^3}$

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

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

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

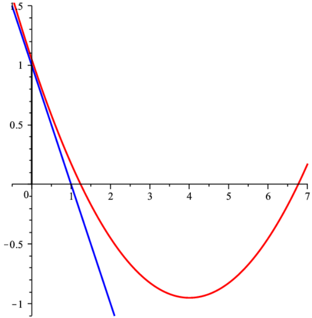
$$y^{[3]} = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + 428z^7 + \dots$$

[Newton 1671]

Numerical Iteration

$$\phi(y) = 1 + \frac{1}{8}y^2 - y$$

$$y^{[n+1]} = y^{[n]} - \frac{\phi(y^{[n]})}{\phi'(y^{[n]})} = y^{[n]} + \frac{1 + {y^{[n]}}^2/8 - y^{[n]}}{1 - y^{[n]}/4}$$



$$\begin{aligned}
y^{[0]} &= 0. \\
y^{[1]} &= 1.0000000000000000000000000 \\
y^{[2]} &= 1.1666666666666666666666666 \\
y^{[3]} &= 1.1715686274509803921568627 5 \\
y^{[4]} &= 1.1715728752506201787474088 4 \\
y^{[5]} &= 1.1715728752538099023966207 5
\end{aligned}$$

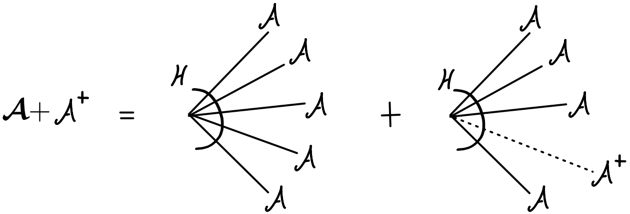
Combinatorial Newton Iteration

Theorem (essentially Labelle)

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

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

has contact $2k+1$ with it.



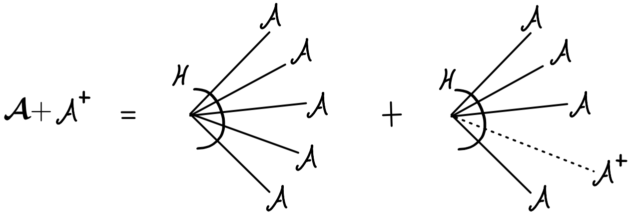
Combinatorial Newton Iteration

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

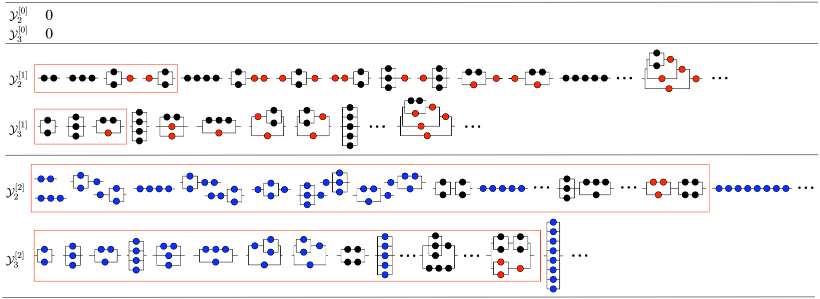
$$\mathcal{A} + \sum_{i \geq 0} \left(\frac{\partial \mathcal{H}}{\partial \mathcal{Y}}(\mathcal{Z}, \mathcal{A}) \right)^i \cdot (\mathcal{H}(\mathcal{Z}, \mathcal{A}) - \mathcal{A})$$

has contact $2k+1$ with it.



Generation by increasing Strahler numbers.

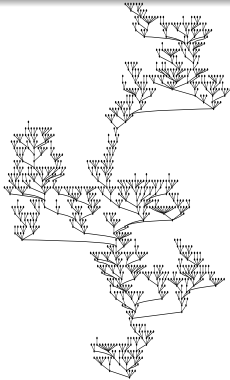
Newton Iteration for Series-Parallel Graphs



$$\begin{pmatrix}\mathcal{S}^{[n+1]}\\ \mathcal{P}^{[n+1]}\end{pmatrix} = \begin{pmatrix}\mathcal{S}^{[n]}\\ \mathcal{P}^{[n]}\end{pmatrix} + \left(\sum_{k\geq 0}\begin{pmatrix}0 & \textsc{Seq}^2(\mathcal{Z}+\mathcal{P}^{[n]})-1\\ \textsc{Set}_{>0}(\mathcal{Z}+\mathcal{S}^{[n]}) & 0\end{pmatrix}^k\right)\begin{pmatrix}\textsc{Seq}_{>1}(\mathcal{Z}+\mathcal{P}^{[n]})-\mathcal{S}^{[n]}\\ \textsc{Set}_{>0}(\mathcal{Z}+\mathcal{S}^{[n]})-\mathcal{P}^{[n]}\end{pmatrix}.$$

Example: Unlabelled Rooted Trees

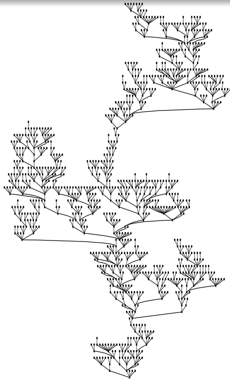
1. Combinatorial equation: $\mathcal{Y} = \mathcal{Z} \cdot \mathrm{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$;



Example: Unlabelled Rooted Trees

1. Combinatorial equation: $\mathcal{Y} = \mathcal{Z} \cdot \mathrm{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$;
2. Combinatorial Newton iteration:

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



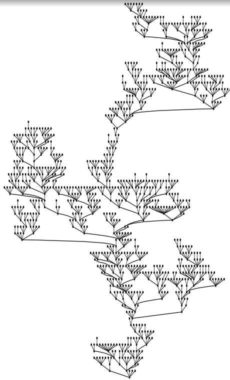
Example: Unlabelled Rooted Trees

1. Combinatorial equation: $\mathcal{Y} = \mathcal{Z} \cdot \mathrm{SET}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$;
2. Combinatorial Newton iteration:

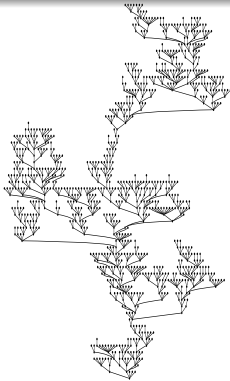
$$\mathcal{Y}^{[n+1]} = \mathcal{Y}^{[n]} + \mathrm{SEQ}(\mathcal{H}(\mathcal{Y}^{[n]})) \cdot (\mathcal{H}(\mathcal{Y}^{[n]}) \setminus \mathcal{Y}^{[n]})$$

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

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



Example: Unlabelled Rooted Trees



1. Combinatorial equation: $\mathcal{Y} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$;
2. Combinatorial Newton iteration:

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

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

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

4. Newton for OGF:

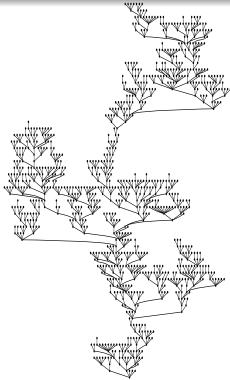
$$\tilde{Y}^{[n+1]} = \tilde{Y}^{[n]} + \frac{H(z, \tilde{Y}^{[n]}) - \tilde{Y}^{[n]}}{1 - H(z, \tilde{Y}^{[n]})}$$

0,

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

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

Example: Unlabelled Rooted Trees



1. Combinatorial equation: $\mathcal{Y} = \mathcal{Z} \cdot \textsc{Set}(\mathcal{Y}) =: \mathcal{H}(\mathcal{Z}, \mathcal{Y})$;
2. Combinatorial Newton iteration:
3. OGF equation: $\tilde{Y}(z) = H(z, \tilde{Y}(z))$

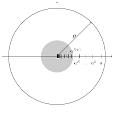
$$\tilde{Y}(z) = z \exp(\tilde{Y}(z) + \frac{1}{2}\tilde{Y}(z^2) + \frac{1}{3}\tilde{Y}(z^3) + \cdots)$$

4. Newton for OGF:

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

5. Numerical iteration:

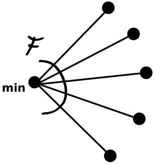
n	$\tilde{Y}^{[n]}(0.3)$	$\tilde{Y}^{[n]}(0.3^2)$	$\tilde{Y}^{[n]}(0.3^3)$
0	0	0	0
1	.43021322639	0.99370806338e-1	0.27759817516e-1
2	.54875612912	0.99887132154e-1	0.27770629187e-1
3	.55709557053	0.99887147197e-1	0.27770629189e-1
4	.55713907945	0.99887147198e-1	0.27770629189e-1
5	.55713908064	0.99887147198e-1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



Linear Species and Ordered Structures

The underlying sets are ordered

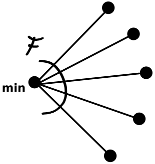
$\int \mathcal{F}$:



Linear Species and Ordered Structures

The underlying sets are ordered

$\int \mathcal{F}$:



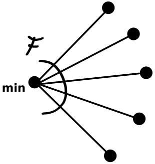
Examples

- increasing trees: $\mathcal{Y} = \mathcal{Z} + \int \mathcal{F}(\mathcal{Y})$;

Linear Species and Ordered Structures

The underlying sets are ordered

$\int \mathcal{F}$:



Examples

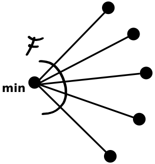
- increasing trees: $\mathcal{Y} = \mathcal{Z} + \int \mathcal{F}(\mathcal{Y})$;
- alternating permutations (odd/even):

$$\mathcal{A}_e = \int \mathcal{A}_e \mathcal{A}_o, \quad \mathcal{A}_o = \mathcal{Z} + \int \mathcal{A}_o^2;$$

Linear Species and Ordered Structures

The underlying sets are ordered

$\int \mathcal{F}$:



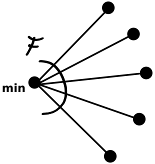
Examples

- increasing trees: $\mathcal{Y} = \mathcal{Z} + \int \mathcal{F}(\mathcal{Y})$;
- alternating permutations (odd/even):
$$\mathcal{A}_e = \int \mathcal{A}_e \mathcal{A}_o, \quad \mathcal{A}_o = \mathcal{Z} + \int \mathcal{A}_o^2;$$
- cycles: $\textsc{Cyc}(\mathcal{A}) = \int \textsc{Seq}(\mathcal{A})\mathcal{A}'$;

Linear Species and Ordered Structures

The underlying sets are ordered

$\int \mathcal{F}$:

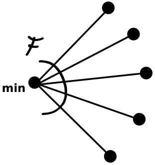


Examples

- increasing trees: $\mathcal{Y} = \mathcal{Z} + \int \mathcal{F}(\mathcal{Y})$;
- alternating permutations (odd/even):
$$\mathcal{A}_e = \int \mathcal{A}_e \mathcal{A}_o, \quad \mathcal{A}_o = \mathcal{Z} + \int \mathcal{A}_o^2;$$
- cycles: $\mathrm{C{\small YC}}(\mathcal{A}) = \int \mathrm{S{\small EQ}}(\mathcal{A})\mathcal{A}'$;
- sets: $\mathrm{S{\small ET}}(\mathcal{A}) = 1 + \int \mathrm{S{\small ET}}(\mathcal{A})\mathcal{A}'$.

Linear Species and Ordered Structures

$\int \mathcal{F}$:



Examples

- increasing trees: $\mathcal{Y} = \mathcal{Z} + \int \mathcal{F}(\mathcal{Y})$;
- alternating permutations (odd/even):
$$\mathcal{A}_e = \int \mathcal{A}_e \mathcal{A}_o, \quad \mathcal{A}_o = \mathcal{Z} + \int \mathcal{A}_o^2;$$
- cycles: $\mathrm{CYC}(\mathcal{A}) = \int \mathrm{SEQ}(\mathcal{A})\mathcal{A}'$;
- sets: $\mathrm{SET}(\mathcal{A}) = 1 + \int \mathrm{SET}(\mathcal{A})\mathcal{A}'$.

Theorem (Enumeration in Quasi-Optimal Complexity)

First N coefficients of the solution of

$$\boldsymbol{\mathcal{Y}}(\mathcal{Z}) = \boldsymbol{\mathcal{H}}(\mathcal{Z}, \boldsymbol{\mathcal{Y}}(\mathcal{Z})) + \int_0^{\mathcal{Z}} \boldsymbol{\mathcal{G}}(\mathcal{T}, \boldsymbol{\mathcal{Y}}(\mathcal{T}))\, d\mathcal{T}$$

with $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{G}}$ constructible, in $O(N \log N)$ operations.

IV 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:
  - Well-defined systems (with 1);
  - Majorant species;
  - 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:
  - Well-defined systems (with 1);
  - Majorant species;
  - PowerSet (it is not a species).

THE END