Analytic Random Generation of Combinatorial Objects

POTLATCH 2013
Nov. 23rd 2013
University of Victoria

SFU
Jérémie Lumbroso

I. Introduction

About combinatorial classes, specification generating function, and basic random generation

decomposable combinatorial classes

a class $\mathcal{A}$ is a decomposable combinatorial class if:

- described by symbolic rules (= grammar)

$\mathcal{Z}$, ε	$+$, $\times$, Seq, Set, Cyc, ...
building blocks	ways to combine them

- possible recursive (defined using itself)
- the number a_n of objects of size n is finite

example: binary trees counted by external nodes

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

a binary tree   leaf   or   two subtrees
(each defined recursively in same way)

all binary trees with 4 leaves $(b_4 = 5)$

random generation of combinatorial structures

let $\mathcal{A}$ be a class, with a_n objects of size n, this means drawing an object of size n is uniform:

$$\mathbb{P}_n[\alpha \in \mathcal{A}_n] = \frac{1}{a_n}$$

some methods:

- ad-hoc methods to deal with specific classes: Remy's algorithm (binary trees), Hook formula (Young tableaux), or more generally bijection/rejection methods (random walks, etc.)
- automatic methods to deal with all decomposable classes: **recursive method** [Nijenhuis and Wilf, Flajolet *et al.*], but requires precomputing all enumeration coefficients up until n: a_0, a_1, ..., a_n

some applications:

- **analysis:** study specific properties/statistics of huge objects through simulation
  - generate many random objects, and empirically study properties
  - compare real data with (randomly generated) uniform data: in genetics, in **poetry** [Gasparov 1987]
- **testing:** generate input for algorithm/server to test robustness and ability to withstand heavy loads [Mougenot *et al.* 2009]

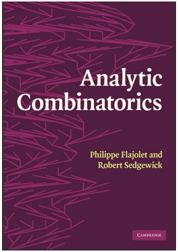
symbolic method [Flajolet & Sedgewick 09]

the **generating function** $A(z)$ of class $\mathcal{A}$ encodes, within a function, the complete enumeration (the number of objects for each size) of the class:

$$A(z) = \sum_{n=0}^{\infty} a_n z^n$$

- in the general case, this generating function (GF) is a formal object; however the GF of decomposable classes is often convergent
- dictionary: correspondence which exactly relates specific. and GF

construction	specification	GF
neutral element	ε	1
atome	$\mathcal{Z}$	z
union	$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$
Cartesian product	$\mathcal{A} \times \mathcal{B}$	$A(z) \cdot B(z)$
sequence	$\mathrm{Seq}(\mathcal{A})$	$\frac{1}{1-A(z)}$



example: class $\mathcal{B}$ of binary trees

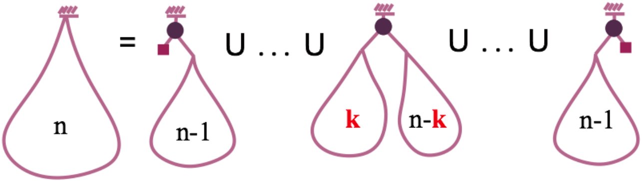
$$\mathcal{B} = \mathcal{Z} + \mathcal{B} \times \mathcal{B} \quad \Rightarrow \quad B(z) = z + B(z) \cdot B(z) = \frac{1 - \sqrt{1-4z}}{2}$$

the "recursive" method

[Nijenhuis & Wilf 1978; Flajolet, Zimmerman & Van Cutsem 1994]

divide and conquer: sample objects of size *n*

by sampling objects of size *1*, ..., *n-1*



uses the recurrences of GF to determine algorithms

efficient and generic for all specifiable classes, exact generation in $O(n \log n)$

Drawbacks

- requires preprocessing b_i for $1 \leqslant i \leqslant n$ and storing coefficients, space $O(n^2)$
- drawing prob. law for k is costly, in $O(k)$

(can be improved with "boustrophedonic" trick)

GenBinTree(n) :=

draw k following law $\mathbb{P}[K = k] = \dfrac{b_k \cdot b_{n-k}}{b_n}$

return <GenBinTree(k), GenBinTree(n-k)>

b_n := binary trees of size n

II. Analytic samplers

("Boltzmann" samplers)

$$\mathbb{P}_z[N = n] = \frac{f_n z^n}{F(z)}$$

randomly generating objects

by evaluating their GF

analytic random samplers

[Duchon, Flajolet, Louchard & Schaeffer 2002]

approximate-size sampling allows for new approach

let $\mathcal{C}$ be a class, we draw an object $\gamma \in \mathcal{C}$ with probability

$$\mathbb{P}_z[\gamma] = \frac{z^{|\gamma|}}{C(z)}$$

size of object

generat. funct. of comb. class

- uniformity at given size (two obj. same size = same prob. being drawn)

$$\mathbb{P}_z[\gamma \in \mathcal{C} \mid |\gamma| = n] = \frac{1}{c_n}$$

- **idea:** by evaluating GF, get a biased average of coefficients
- the probability distribution ("Power Series Distribution") has all the same good algebraic properties as GF
- later we can see how to control the size

the result is an elegant and simple translation to algorithms
(here for the labeled case)

construction	**algorithm**
$\mathcal{A} = \varepsilon$ or $\mathcal{Z}$	$\Gamma\mathcal{A}(z) :=$ **return** $\square$ or $\blacksquare$
$\mathcal{A} = \mathcal{B} + \mathcal{C}$	$\Gamma\mathcal{A}(z) :=$ **if** $\mathrm{Ber}(B(z)/(B(z)+C(z))) = 1$ **then return** $\Gamma\mathcal{B}(z)$ **else return** $\Gamma\mathcal{C}(z)$
$\mathcal{A} = \mathcal{B} \times \mathcal{C}$	$\Gamma\mathcal{A}(z) :=$ **return** $< \Gamma\mathcal{B}(z); \Gamma\mathcal{C}(z) >$
$\mathcal{A} = \mathrm{Seq}\,(\mathcal{B})$	$\Gamma\mathcal{A}(z) := k \leftarrow \mathrm{Geo}(A(z))$; **return** k indep. $\Gamma\mathcal{B}(z)$
$\mathcal{A} = \mathrm{Set}\,(\mathcal{B})$	$\Gamma\mathcal{A}(z) := k \leftarrow \mathrm{Poi}(A(z))$; **return** k indep. $\Gamma\mathcal{B}(z)$
$\mathcal{A} = \mathrm{Cyc}\,(\mathcal{B})$	$\Gamma\mathcal{A}(z) := k \leftarrow \mathrm{Loga}(A(z))$; **return** k indep. $\Gamma\mathcal{B}(z)$

Definition

An analytic sampler for class $\mathcal{C}$, with generating function $C(z) = \sum c_n z^n$, is an algorithm ΓC which returns any object $\gamma \in \mathcal{C}$ with probability:

$$\mathbb{P}_z[\gamma] = \frac{c_n z^n}{C(z)}.$$

- when $\mathcal{A} = \{\varepsilon\}$ or $\mathcal{A} = \{\mathcal{Z}\}$, $\Gamma\mathcal{A}(z) :=$ **return** $\square$ or $\blacksquare$
  Proof (that it's an analytic sampler): it always returns an element from a singleton (containing a neutral element and atom resp.):

$$\mathbb{P}_z[\square] = \frac{z^0}{z^0} = 1 \qquad \mathbb{P}_z[\blacksquare] = \frac{z^1}{z^1} = 1$$

- when $\mathcal{A} = \mathcal{B} + \mathcal{C}$, $\Gamma\mathcal{A}(z) :=$ **if** $\mathrm{Ber}(B(z)/(B(z) + C(z))) = 1$ **then return** $\Gamma\mathcal{B}(z)$ **else return** $\Gamma\mathcal{C}(z)$
  Proof: prob. of drawing $\beta \in \mathcal{A}$ (when $\beta \in \mathcal{B}$)

$$\mathbb{P}_z[\beta \in \mathcal{A}] = \frac{B(z)}{B(z) + C(z)} \cdot \mathbb{P}_z[\beta \in \mathcal{B}] = \frac{B(z)}{B(z) + C(z)} \cdot \frac{z^{|\beta|}}{B(z)} = \frac{z^{|\alpha|}}{A(z)}$$

- when $\mathcal{A} = \mathcal{B} \times \mathcal{C}$, $\Gamma\mathcal{A}(z) :=$ **return** $< \Gamma\mathcal{B}(z); \Gamma\mathcal{C}(z) >$
  Proof: let $\alpha \in \mathcal{A}$, $\alpha = (\beta, \gamma)$,

$$\mathbb{P}_z[\alpha] = \mathbb{P}_z[\beta] \cdot \mathbb{P}_z[\gamma] = \frac{z^{|\beta|}}{B(z)} \cdot \frac{z^{|\gamma|}}{C(z)} = \frac{z^{|\beta|+|\gamma|}}{B(z) \cdot C(z)} = \frac{z^{|\alpha|}}{A(z)}$$

first example: binary trees

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

first example: binary trees

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

$$B(z) = z + B(z)^2 = \frac{1 - \sqrt{1 - 4z}}{2}$$

first example: binary trees

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

$$B(z) = z + B(z)^2 = \frac{1 - \sqrt{1 - 4z}}{2}$$

$$\begin{aligned}\Gamma\mathcal{B}(z) := &\ \textbf{if } \mathrm{Ber}(z/B(z)) = 1 \textbf{ then return } \blacksquare \\ &\ \textbf{else return } < \Gamma\mathcal{B}(z); \Gamma\mathcal{B}(z) >\end{aligned}$$

Untitled-1

```
(* Specification:   B = Z + B^2   *)

In[4]:= Solve[B == z + B^2, B]
```

Out[4]= $\left\{\left\{B \to \frac{1}{2}\left(1 - \sqrt{1 - 4 z}\right)\right\}, \left\{B \to \frac{1}{2}\left(1 + \sqrt{1 - 4 z}\right)\right\}\right\}$

In[5]:= B[z_] := $\frac{1}{2}\left(1 - \sqrt{1 - 4 z}\right)$

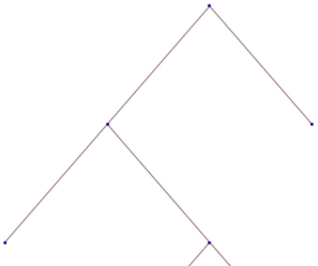
```
In[6]:= AnaBinTree[z_] :=
         If[RandomVariate[BernoulliDistribution[z / B[z]]] == 1,
          (*if*) Return[{}],
          (*else*) Return[{AnaBinTree[z], AnaBinTree[z]}]]

In[7]:= AnaBinTree[0.25]

Out[7]= {{{}, {{}, {}}}, {}}

In[36]:= DrawTree[Out[7]]

Out[36]//TreeForm=
```



RECURSIVE version [Flajolet *et al.* 1994]

RecBT(n) := {
  if $n = 1$ **then** return Leaf
  else
    k from distr. $\mathbb{P}[K = k] = (b_k \cdot b_{n-k})/b_n$
    return Node(RecBT(k), RecBT($n - k$))
}

"BOLTZMANN" vers. [Duchon *et al.* 02]

AnaBT(z) := {
  if $\mathrm{Ber}(z/B(z)) = 1$ **then** return Leaf
  else
    return Node(AnaBT(z), AnaBT(z))
}

Noteworthy, in "Boltzmann"/analytic random sampling, the randomization is **global**: the same law is calculated in all recursive calls.

second example: general trees (any number of children)

$$\mathcal{G} = \mathcal{Z} \times \mathsf{Seq}\,(\mathcal{G})$$

second example: general trees (any number of children)

$$\mathcal{G} = \mathcal{Z} \times \mathsf{Seq}\,(\mathcal{G})$$

$$G(z) = \frac{z}{1 - G(z)} = \frac{1 - \sqrt{1 - 4z}}{2}$$

second example: general trees (any number of children)

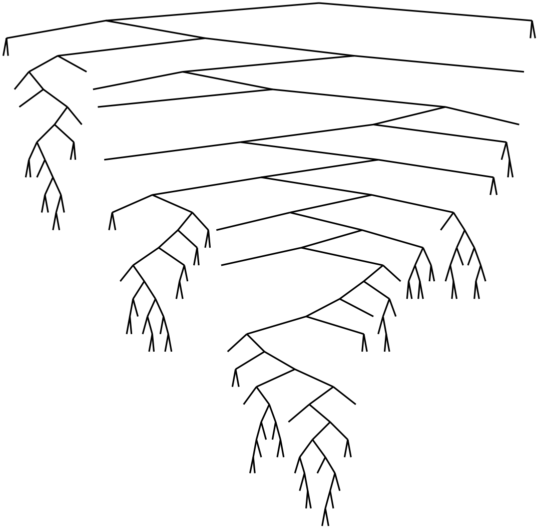
$$\mathcal{G} = \mathcal{Z} \times \mathsf{Seq}\,(\mathcal{G})$$

$$G(z) = \frac{z}{1 - G(z)} = \frac{1 - \sqrt{1 - 4z}}{2}$$

$$\begin{aligned} \Gamma\mathcal{G}(z) &:= \text{let } k = \mathrm{Geo}(G(z)) \\ &\quad \textbf{return} \;\; < \blacksquare; \Gamma\mathcal{G}(z); \ldots; \Gamma\mathcal{G}(z) > \qquad k \text{ times} \end{aligned}$$

Otter tree

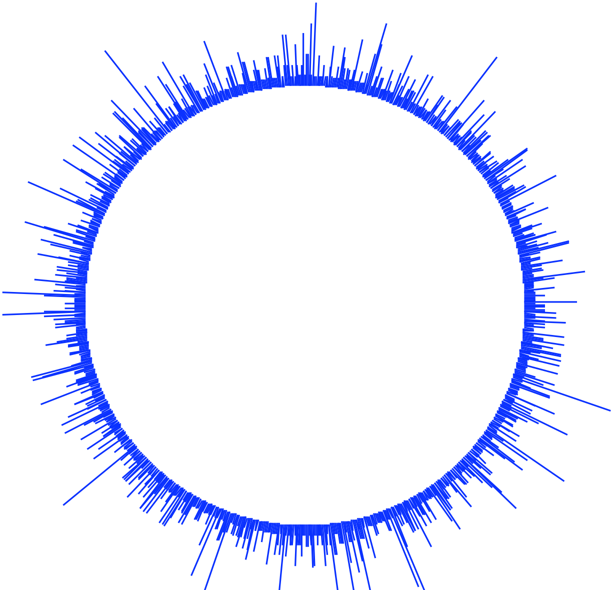
$$\mathcal{O} = Z + \mathrm{MSet}_2(\mathcal{O})$$



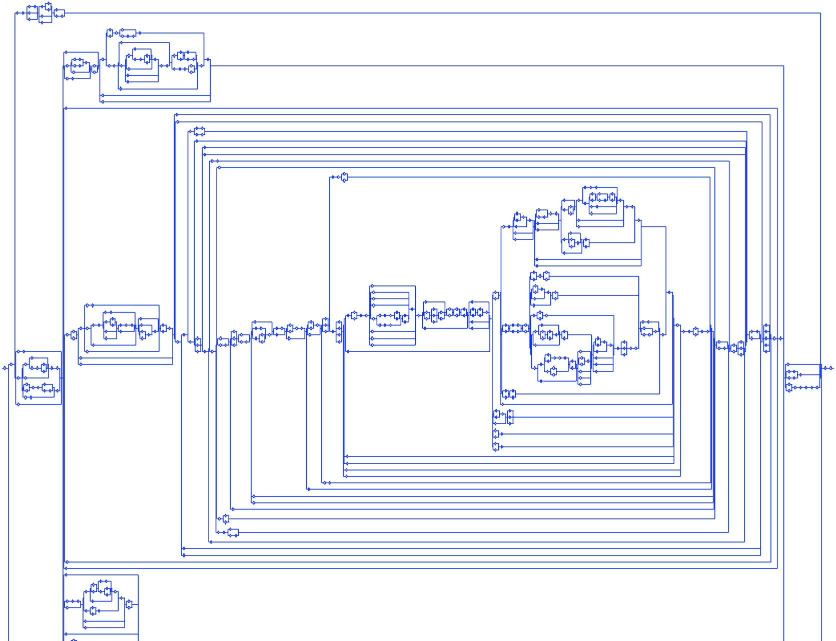
picture by Carine Pivoteau

Circular composition (size about 2000)

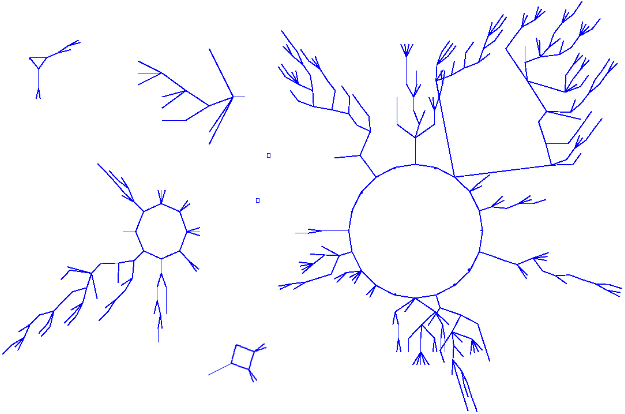
$$\mathcal{C} = \mathrm{Cyc}\left(\mathrm{Seq}\left(\mathcal{Z}\right)\right)$$



Series-parallel graphs (size about 500)



Functional graph



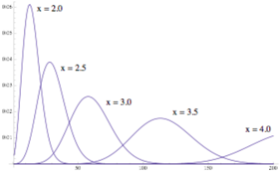
III. Size matters

Analytic samplers efficiently draw objects
but following some "arbitrary" distribution

How to make this useful?

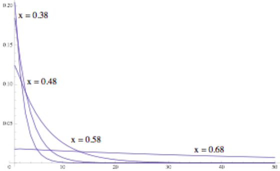
size control: rejection & its cost (for typical classes)

"bumpy"



$$P(z) = e^{e^z - 1}$$

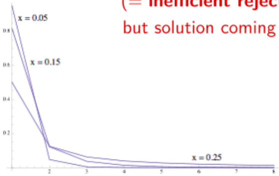
"flat"



$$S(z) = \frac{1}{1 - (e^z - 2)}$$

"peaked"

(= **inefficient rejection**
but solution coming up)



$$B(z) = \frac{1 - \sqrt{1 - 4z}}{2}$$

- solve/approximate expected value to find z targeting size

$$\mathbb{P}_z[N = n] = \frac{f_n z^n}{F(z)} \quad \Rightarrow \quad \mathbb{E}_z[N] = z\frac{F'(z)}{F(z)} \quad \Rightarrow \quad z_n = \ldots$$

- size distribution of samplers depends on *type of singularity* of generating function

$$f(z) \underset{z \to \rho}{\sim} P(z) + c_0(1 - z/\rho)^{-\alpha} + o((1 - z/\rho)^{-\alpha}), \quad \alpha \in \mathbb{R} \setminus \{0, -1, -2, \ldots\}$$

ρ : radius of convergence $\qquad \alpha$: singular exponent

- ⚠ size distribution affects *rejection complexity*; for "bumpy" and "flat" approx. in **O(1) loops** and exact in **O(n) loops**

approximate-size rejection

loop
obj $:= \Gamma B(z)$
while $|\text{size}(\text{obj}) - n| > \epsilon$

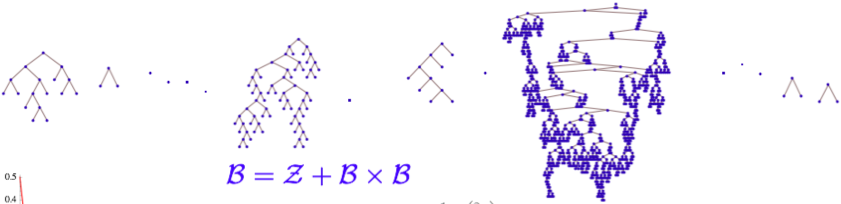
exact-size rejection

loop
obj $:= \Gamma B(z)$
while $\text{size}(\text{obj}) \neq n$

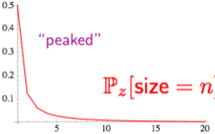
for target size n and tolerance $\epsilon = \ldots\%$

ex.: size distribution of sampler for binary trees

9, 2, 1, 1, 1, 1, 36, 1, 6, 1, 449, 1, 1, 1, 2, 2,



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



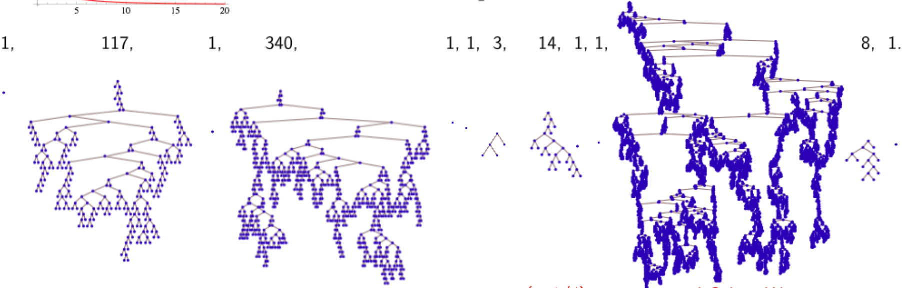
$$\mathbb{P}_z[\text{size} = n] = \frac{b_n z^n}{B(z)}$$

$$b_{n+1} := \frac{1}{n+1}\binom{2n}{n}$$

$$B(z) := \frac{1 - \sqrt{1-4z}}{2}$$

3107,

1, 117, 1, 340, 1, 1, 3, 14, 1, 1, 8, 1.



$z = \rho \ (= 1/4) \quad \Rightarrow \quad$ critical Galton Watson process

efficient rejection for "peaked" classes

Pointing: if $\mathcal{A}$ is a class, then $\mathcal{C} = \mathcal{A}^\bullet$ is the class obtained from all possible ways to *distinguish one atom* of objects of $\mathcal{A}$.

$$c_n = n \cdot a_n \qquad C(z) = z\frac{\mathrm{d}}{\mathrm{d}z}A(z)$$

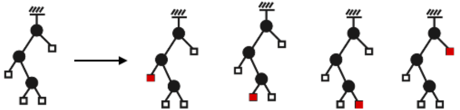
improve the prevalence of larger objects in size distr.

- reshapes size distribution while **preserving uniformity at given size**
- changes profile from "peaked" (inefficient) to "flat" (efficient)

$$\mathcal{B} = \mathcal{Z} + \mathcal{B} \times \mathcal{B} \quad \Longrightarrow \quad \begin{cases} \mathcal{B} & = \quad \mathcal{Z} + \mathcal{B} \times \mathcal{B} \\ \mathcal{B}^\bullet & = \quad \mathcal{Z} + \mathcal{B}^\bullet \times \mathcal{B} + \mathcal{B} \times \mathcal{B}^\bullet. \end{cases}$$

1, 1, 3, 9, 1, 1, 1, 5, 12, 1, 1, 1108, 1, 1, 1, 4, 1, 42, 5, 4, 16, 3, 1, 1, 2, 1, 1, 2, 3, 1, 1, 341, 1, 2, 18, 8, 1, 14, 30, 1, 2, 1, 114, 1, 1, 4, 3, 2, 2, 1, 2, 4, 1, 1, 1, 360, 1, 1, 3, 1, 2, 3, 1, 1, 1, 417, 1, 3, 3, 429, 1, 16, 1, 1, 1, 1, 1, 1, 1, 55, 38, 1, 1, 1, 1, 1, 2, 1, 1, 4, 1, 1, 14, 3, 2, 1, 3, 3, 1, 1

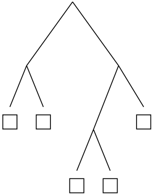
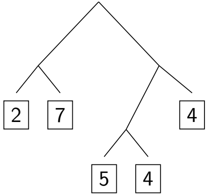
2995, 4, 18, 575, 191, 6, 2697, 2656, 665, 503, 1, 488, 433, 250, 7458, 165, 32, 368, 1384, 1487, 756, 636, 50, 1520, 4974, 866, 1346, 14, 6289, 9, 3775, 85, 687, 79, 6228, 947, 1325, 8, 1, 65, 1, 375, 307, 31, 12, 32, 184, 1094, 2824, 3282, 383, 188, 1435, 277, 1340, 52, 4659, 2089, 3423, 244, 17, 396, 23, 5, 2129, 1330, 9760, 2403, 520, 197, 1816, 9, 249, 867, 799, 59, 62, 1758, 19, 4393, 1783, 1, 373, 146, 363, 5154, 2494, 114, 1137, 1, 1887, 136, 43, 87, 79, 67, 21, 867, 72, 2



a tree of size 4
is "copied" 4 times

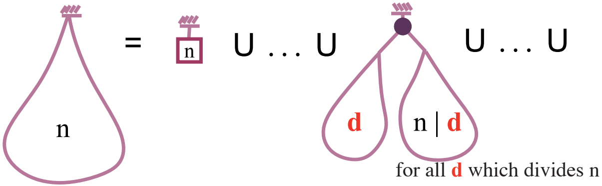
IV. Advanced example: Dirichlet sampling

Multiplicative object that cannot be generated any other way

ADDITIVE (traditional objects)	MULTIPLICATIVE[1]
$\alpha \in \mathcal{A},\ \beta \in \mathcal{B}\ \ \lvert(\alpha,\beta)\rvert = \lvert\alpha\rvert + \lvert\beta\rvert$	$\alpha \in \mathcal{A},\ \beta \in \mathcal{B}\ \ \lvert(\alpha,\beta)\rvert = \lvert\alpha\rvert \cdot \lvert\beta\rvert$
unique atom $\mathcal{Z}$ of unit size 1	**infinity** of atoms, $\mathcal{Z}_m$ $(m \in \mathbb{Z}_{>0})$
$\mathcal{A} = \mathcal{Z} + \mathcal{A} \times \mathcal{A}$	$\mathcal{M} = \mathcal{I} \setminus \mathcal{Z}_1 + \mathcal{M} \times \mathcal{M}$
	
$1+1+1+1+1=5$	$2 \times 7 \times 5 \times 4 \times 4 = 1120$
Ordinary GF or Exponential GF	Dirichlet GF
$\sum_{k=0}^{\infty} a_k z^k \qquad \sum_{k=0}^{\infty} \frac{a_k}{k!} z^k$	$\sum_{k=1}^{\infty} a_k \frac{1}{k^s}$

[1] First considered from a symbolic/combinatoric perspective by Hwang (1994).

recursive method: not efficient for multiplicative objects



$$b_n = 1 + \sum_{\substack{d|n \\ 1<d<n}} b_d \cdot b_{n|d}$$

PROBLEMS of efficiency

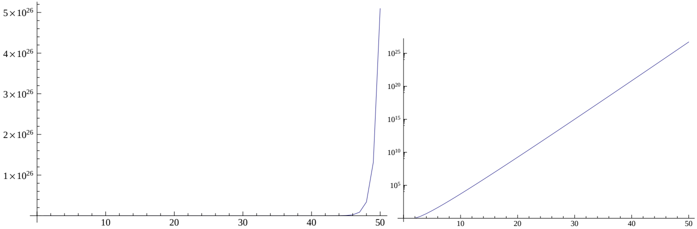
- sizes (wrt. number of "nodes") exponentially larger than for additive objects
- requires **factor decomposition** which is (too) costly

PROBLEMS of quality

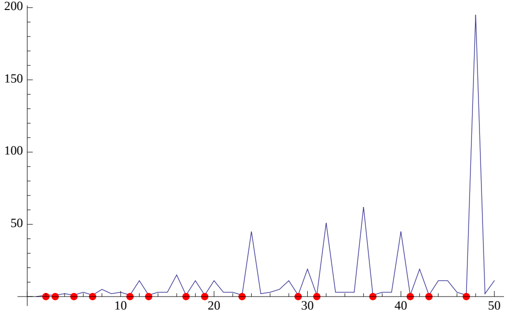
- size distribution is highly irregular

size distributions (# obj. of given size)

additive binary trees (plot then logplot)



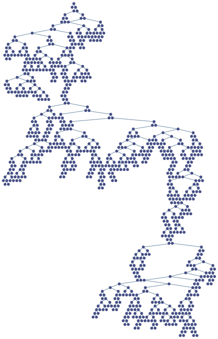
multiplicative (binary) branching factorizations



extending the idea to multiplicative objects

Theorem [Bodini & L. 2012]. Let $\mathcal{C}$ be a multiplicative combinatorial class described with: disjoint union, cartesian product, sequence, well-founded recursion, etc.

Under some hypotheses on the generating function, a Dirichlet sampler for $\mathcal{C}$ can generate an object of size n, with some error $\varepsilon \in (0,1)$, in $O(\log(n)^2)$ worst-case time complexity.

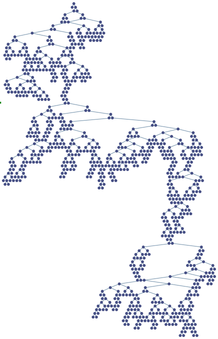


extending the idea to multiplicative objects

Theorem [Bodini & L. 2012]. Let $\mathcal{C}$ be a multiplicative combinatorial class described with: disjoint union, cartesian product, sequence, well-founded recursion, etc.

Under some hypotheses on the generating function, a Dirichlet sampler for $\mathcal{C}$ can generate an object of size n, with some error $\varepsilon \in (0,1)$, in $O(\log(n)^2)$ worst-case time complexity.

- Zeta-distributed atoms sampled in $O(1)$ [Devroye 1986]
- resorts to analytic number theory: specifically Delange's Tauberian theorem, as equivalent of Flajolet-Odlyzko transfer theorem in additive combinatorics
- tuning of control parameter completely different: in additive analytic ("Boltzmann") sampling, direct inversion of expected value; here expected value is infinite and requires *ad-hoc* tuning informed from theorem



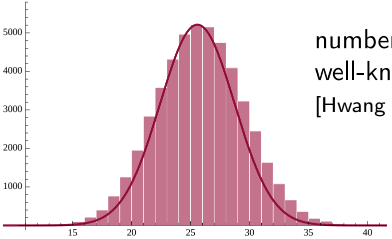
ordered factorizations, $\mathcal{F} := \mathsf{Seq}\left(\mathcal{I} \setminus \mathcal{Z}_1\right)$

$$
\begin{array}{l}
\Gamma\mathrm{D}_s\left[\mathcal{F}\right] := \{ \\
\qquad \lambda \leftarrow \zeta(s) - 1; \\
\qquad K \in \mathrm{Geo}(\lambda); \\
\qquad \text{return } \big(\underbrace{\Gamma\mathrm{D}_s\left[\mathcal{I} \setminus \mathcal{Z}_1\right], \ldots, \Gamma\mathrm{D}_s\left[\mathcal{I} \setminus \mathcal{Z}_1\right]}_{K \text{ times}}\big) \\
\}
\end{array}
$$

```
OrderedFactorization[10^200, 0.5] // AbsoluteTiming

{17.002826, {598,
  94 315 438 343 755 964 449 064 464 145 270 360 907 587 302 431 535 020 906 407 \
   589 438 865 191 662 481 620 456 946 846 202 450 914 444 733 710 252 639 029 \
   394 242 922 918 929 394 271 546 094 283 086 276 198 942 107 362 365 753 807 \
   339 520 000 000 000 000 000 000 000,
  {4, 3, 5, 131, 2, 9, 5, 3, 4, 3, 4, 3, 51, 5, 2, 7, 3, 3, 3, 2, 2, 2, 4,
   2, 3, 5, 3, 3, 23, 3, 3, 6, 5, 10, 2, 6, 6, 2, 22, 2, 2, 3, 18, 242,
   3, 7, 3, 4, 2, 379, 4, 2, 7, 2, 9, 3, 12, 2, 46, 7, 2, 4, 9, 2, 3, 7,
   2, 11, 2, 3, 2, 3, 5, 6, 2, 2, 9, 9, 5, 20, 24, 35, 4, 2, 4, 2, 4, 2,
   2, 2, 5, 2, 2, 3, 6, 3, 2, 5, 22, 3, 13, 16, 2, 3, 2, 3, 4, 2, 21, 4,
   2, 2, 6, 3, 4, 4, 6, 70, 13, 3, 10, 3, 2, 3, 894, 4, 14, 2, 2, 22, 6,
   4, 2, 3, 13, 3, 11, 2, 3, 7, 53, 4, 2, 3, 47, 3, 77, 2, 2, 2, 4, 6, 6,
   6, 3, 2, 7, 4, 2, 8, 2, 3, 2, 53, 3, 4, 33, 2, 2, 6, 4, 3, 7, 15, 3, 7,
   222, 9, 7, 3, 3, 18, 2, 12, 2, 2, 2, 2, 2, 29, 5, 9, 2, 305, 904, 2,
   2, 12, 7, 2, 2, 4, 2, 3, 2, 54, 2, 27, 9, 18, 2, 3, 41, 8, 2, 44, 2,
   3, 2, 4, 2, 3, 2, 3, 2, 4, 17, 4, 5, 2, 5, 2, 53, 8, 2, 40, 2, 2, 4,
   2, 3, 3, 4, 6, 3, 2, 2, 2, 15, 13, 14, 7, 14, 3, 2, 3, 7, 3, 8, 2, 2,
   33, 3, 4, 3, 7, 523, 3, 10, 3, 3, 2, 12, 3, 86, 67, 4, 2, 2, 2, 2}}}
```

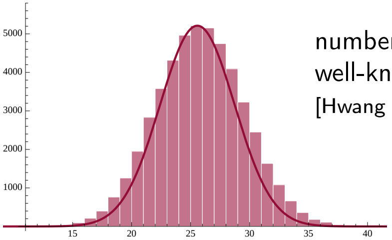
ordered factorizations, $\mathcal{F} := \mathrm{Seq}(\mathcal{I} \setminus \mathcal{Z}_1)$



number of factors in random ordered factorizations well-known to be **normally distributed**

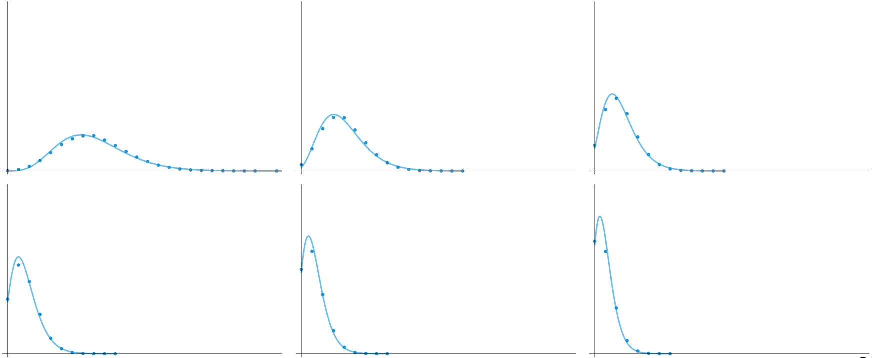
[Hwang 1999] [Hwang and Janson 2009]

ordered factorizations, $\mathcal{F} := \mathrm{Seq}\,(\mathcal{I} \setminus \mathcal{Z}_1)$



number of factors in random ordered factorizations
well-known to be **normally distributed**
[Hwang 1999] [Hwang and Janson 2009]

\# of factors equal to $m = 2, 3, \ldots$ is **gamma distributed** (conjectured then proven)



other developments

other developments

Expressivity

- colored objects [Bodini, Jacquot 2006]
- multi-dimensional generation [Bodini, Ponty 2010; Bodini, L., Ponty 2014]
- holonomic specification [Bacher, Bodini, Jacquot 2013]
- planar graphs [Fusy et al. 2008]

Implementation

- oracle evaluation [Pivoteau, Salvy, Soria, 2008]
- bit complexity [Flajolet, Pelletier, Soria, 2011]
- approximate-evaluation-rejection [Bodini, Lumbroso 2014]

Other

- use the samplers as a proof model [Steger, Panagiotou] conclusion