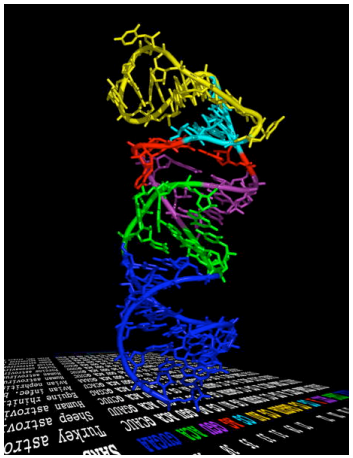
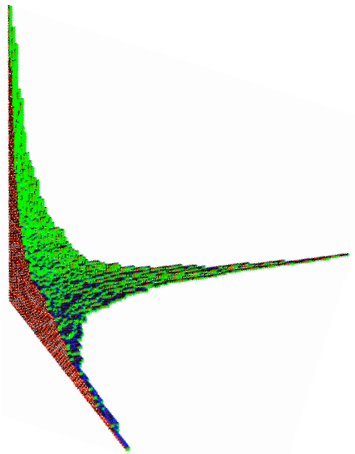
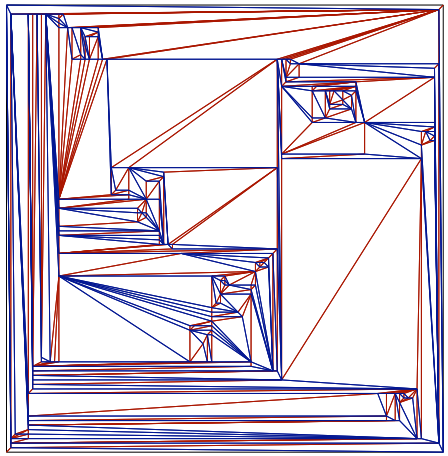
# Effective Boltzmann sampling

Carine Pivoteau

based on work by P. Duchon, P. Flajolet, E. Fusy,
G. Louchard, C. Pivoteau and G. Schaeffer







# Plan of the talk

# Context

Random sampling under Boltzmann model

- approximate size sampling,
- size distribution spread over the whole combinatorial class, but uniform for a sub-class of objects of the same size,
- control parameter,
- automatized sampling: the sampler is compiled from specification automatically,
- very large objects can be sampled.

1. *Boltzmann samplers for the random generation of combinatorial structures.* P. Duchon, P. Flajolet, G. Louchard, G. Schaeffer. Combinatorics, Probability and Computing, 13(4-5):577-625, 2004. Special issue on Analysis of Algorithms.
2. *Boltzmann sampling of unlabelled structures.* Ph. Flajolet, E. Fusy, C. Pivoteau. Proceedings of ANALCO07, january 2007.

# Boltzmann model

# Framework: constructible classes

| | specification | ordinary g.f. (unlabelled) | exponential g.f. (labelled) |
|---|---|---|---|
| $\varepsilon$ / atom | $1 \quad / \quad \mathcal{Z}$ | $1 \quad / \quad x$ | $1 \quad / \quad x$ |
| Union | $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ | $C(x) = A(x) + B(x)$ | $\hat{C}(x) = A(x) + B(x)$ |
| Product | $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ | $C(x) = A(x) \times B(x)$ | $\hat{C}(x) = A(x) \times B(x)$ |
| Sequence | $\mathcal{C} = \textsc{Seq}(\mathcal{A})$ | $C(x) = \frac{1}{1-A(x)}$ | $\hat{C}(x) = \frac{1}{1-A(x)}$ |
| Set | $\mathcal{C} = \textsc{Set}(\mathcal{A})$ | $\exp\left(\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} A(z^k)\right)$ | $\hat{C}(x) = \exp(A(x))$ |
| Multiset | $\mathcal{C} = \textsc{MSet}(\mathcal{A})$ | $\exp\left(\sum_{k=1}^{\infty} \frac{1}{k} A(z^k)\right)$ | – |
| Cycle | $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$ | $\sum_{k=1}^{\infty} \frac{1}{k} \log \frac{1}{1 - A(z^k)}$ | $\hat{C}(x) = \log \frac{1}{1-A(z^k)}$ |

# Model definition

## Definition

In the unlabelled case, Boltzmann model assigns to any object $c \in \mathcal{C}$ the following probability:

$$\mathbb{P}_x(c) = \frac{x^{|c|}}{C(x)}$$

In the labelled case, this probability becomes:

$$\mathbb{P}_x(c) = \frac{1}{\hat{C}(x)} \frac{x^{|c|}}{|c|!}$$

A Boltzmann sampler $\Gamma C(x)$ for the class $\mathcal{C}$ is a process that produces objects from $\mathcal{C}$ according to this model.

$\rightarrow$ 2 object of the same size will be drawn with the same probability.

# Size distribution

The probability of drawing an object of size $N$ is then:

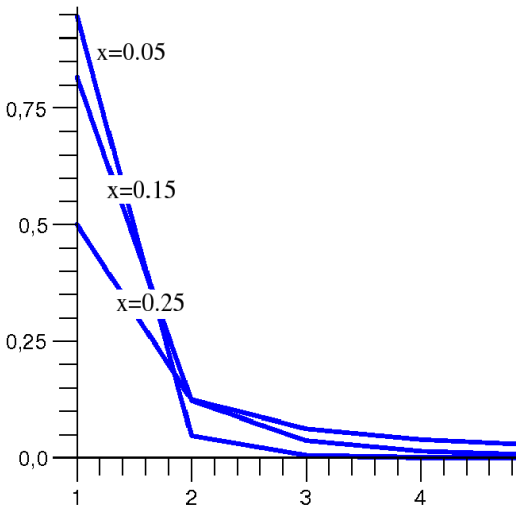
$$\mathbb{P}_x(N = n) = \sum_{|c|=n} \mathbb{P}_x(c) = \frac{C_n x^n}{C(x)} \quad \text{or} \quad \frac{C_n x^n}{n!\hat{C}(x)}$$

Then, the expected size of an object drawn by a generator with parameter $x$ is:

$$\mathbb{E}_x(N) = x\frac{C'(x)}{C(x)} \quad \text{or} \quad x\frac{\hat{C}'(x)}{\hat{C}(x)}$$

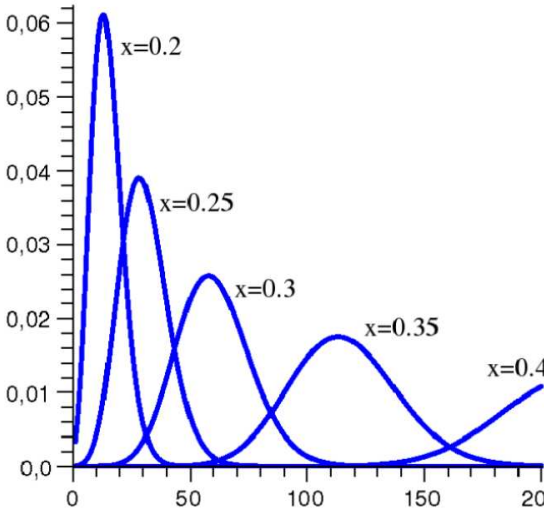
# Size distribution – 2

General plane trees



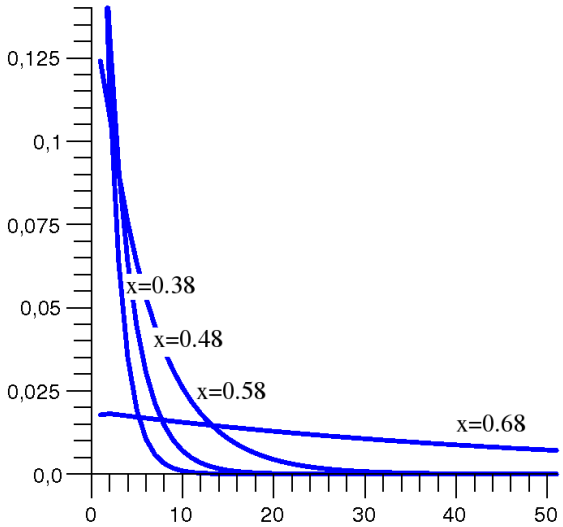

**Peaked**

Set partitions




**Bumpy**

Surjections




**Flat**

# Algorithms

1 Boltzmann model

2 Algorithms
- Basic unlabelled constructions
- Labelled classes
- Back to unlabelled

3 Technical matters

# Unions, products

## Disjoint unions

Boltzmann sampler $\Gamma C$ for $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$:

With probability $\frac{A(x)}{C(x)}$ do $\Gamma A(x)$ else do $\Gamma B(x)$ $\quad \rightarrow$ Bernoulli.

$$\text{if } \gamma \in \mathcal{A} \text{ then } \mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{A(x)} \cdot \frac{A(x)}{C(x)} = \frac{x^{|\gamma|}}{C(x)}.$$

## Products

Boltzmann sampler $\Gamma C$ for $\mathcal{C} = \mathcal{A} \times \mathcal{B}$:

Generate a pair $\langle\ \Gamma A(x)\ ,\ \Gamma B(x)\ \rangle$ $\quad \rightarrow$ independent calls.

$$\text{if } \gamma = \alpha \times \beta \in \mathcal{C} \text{ then } \mathbb{P}_x(\gamma) = \frac{x^{|\alpha|}}{A(x)} \cdot \frac{x^{|\beta|}}{B(x)} = \frac{x^{|\gamma|}}{C(x)}.$$

Remark: $A(x)$, $B(x)$ and $C(x)$ are given by an ***oracle***.

# Sequences

## Sequences

Boltzmann sampler $\Gamma C$ for $\mathcal{C} = \textsc{Seq}(\mathcal{A})$:
Generate $k$ according to a geometric law of parameter $A(x)$
Generate a $k$-tuple $\langle\ \Gamma A(x)\ , \ldots ,\ \Gamma A(x)\ \rangle \rightarrow$ independent calls.

*Proof.*
Recursive equation: $\mathcal{C} = \mathbf{1} + \mathcal{A}\mathcal{C}$ with $+, \times$ constructions.

With proba. $A(x)$, call $\Gamma A(x)$ and call recursively $\Gamma C(x)$; else stop.

Number of successful trials of $\text{Bern}(A(x))$ is $\text{Geom}(A(X))$.

Remark: $A(x)$ is given by an ***oracle***.

# Examples of specifications with $\{\cup, \times, \mathbf{Seq}\}$

**Regular** specifications (non recursive).

- integer compositions,
- polyominos that have rational g.f.: column-convex,

- regular languages,
- binary words without long runs:

$$\mathcal{L} = \textsc{Seq}_m(b) \times \textsc{Seq}(a \times \textsc{Seq}_m(a) \times b \times \textsc{Seq}_m(b)) \times \textsc{Seq}_m(a)$$

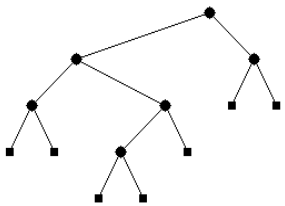
where $\textsc{Seq}_m(a) = a + aa + \cdots + a^m$.

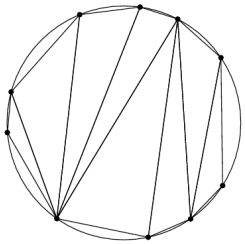
- ...

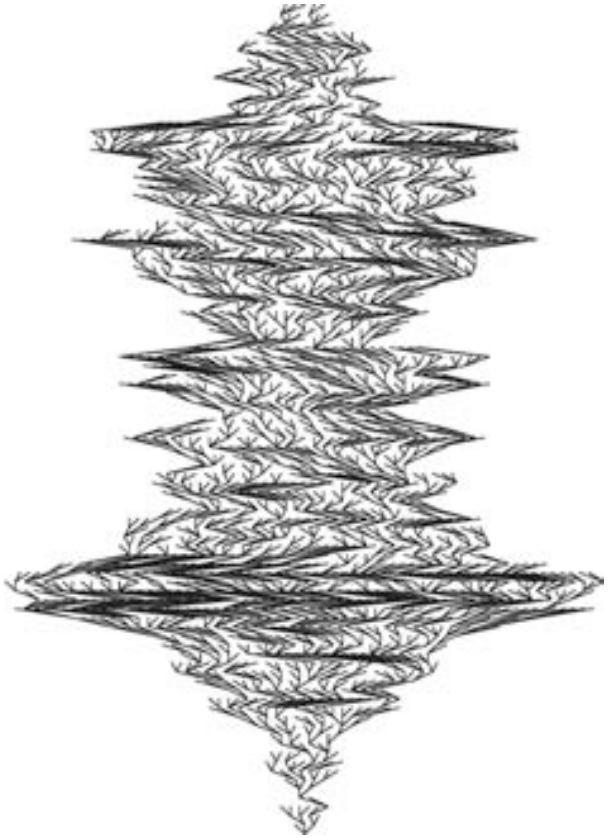
# Specifications with $\{\cup, \times, \mathbf{Seq}\}$ and recursion

**Context-free** specifications.

- binary, $k$-ary, 2–3–4 trees, ...,
- triangulations,
- general planar rooted trees,
- noncrossing graphs,
- ...



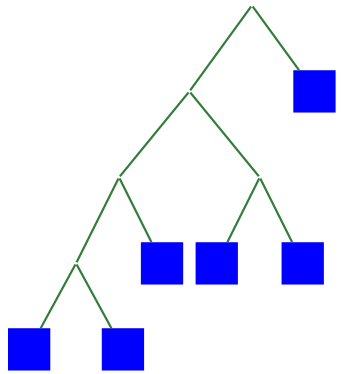




# Binary trees

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

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



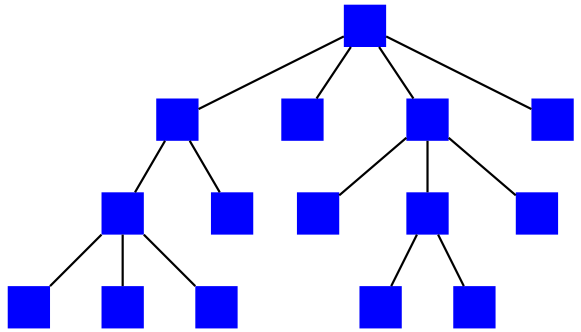
## Algorithm: $\Gamma B(x)$

$b \leftarrow \text{Bern}(x/B(x))$;
**if** $b = 1$ **then**
  Return ■
**else**
  Return $\langle\ \Gamma B(x)\ ,\ \Gamma B(x)\ \rangle$ ;
**end if**

# General planar rooted trees

$$\mathcal{T} = \mathcal{Z} \times \textsc{Seq}(\mathcal{T})$$

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



**Algorithm: $\Gamma B(x)$**

$k \leftarrow \text{Geom}(T(x))$;
*children* $\leftarrow$ the $k$-tuple $\langle\ \Gamma T(x), \ldots, \Gamma T(x)\ \rangle$;
Return ■ $\times$ *children* ;

# Labelled classes

Same algorithms, with exponential g.f.

| construction | sampler |
|---|---|
| $\mathcal{C} = \emptyset$ or $\mathcal{Z}$ | $\Gamma C(x) := \varepsilon$ or atom |
| $\mathcal{C} = \mathcal{A} + \mathcal{B}$ | $\Gamma C(x) := \text{Bern}\, \frac{\hat{A}(x)}{\hat{C}(x)} \longrightarrow \Gamma A(x) \mid \Gamma B(x)$ |
| $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ | $\Gamma C(x) := \langle\, \Gamma A(x)\ ;\ \Gamma B(x)\, \rangle$ |
| $\mathcal{C} = \textsc{Seq}(\mathcal{A})$ | $\Gamma C(x) :=\ \text{Geom}\, \hat{A}(x) \Longrightarrow \Gamma A(x)$ |

**Put the labels at the end !**

# Labelled sets and cycles

## Sets

Boltzmann sampler $\Gamma C$ for $\mathcal{C} = \textsc{Set}(\mathcal{A})$:
Generate $k$ according to a Poisson law of parameter $A(x)$
Generate a $k$-tuple $\langle\, \Gamma A(x)\, , \ldots ,\, \Gamma A(x)\, \rangle$

Poisson law: $\mathbb{P}(X = k) = e^{-\lambda} \dfrac{\lambda^k}{k}!$

## Cycles

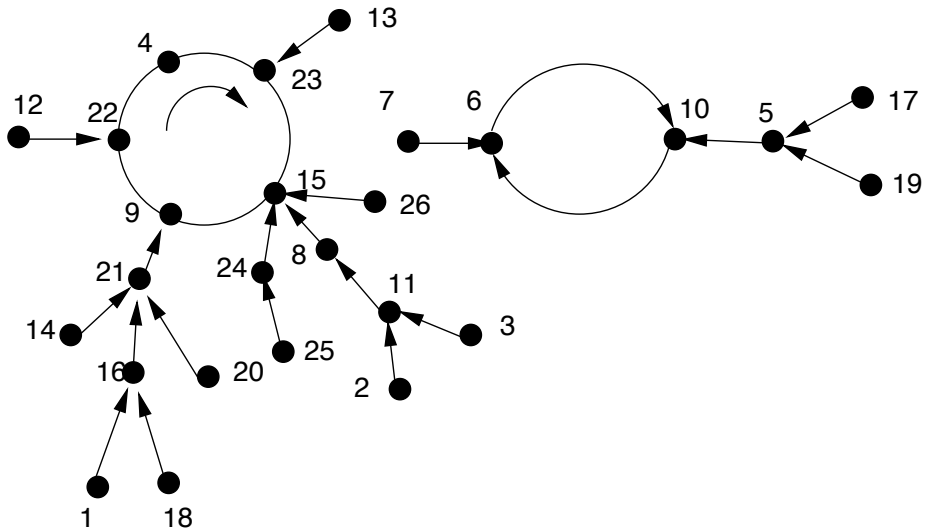
Boltzmann sampler $\Gamma C$ for $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$:
Generate $k$ according to a logarithmic law of parameter $A(x)$
Generate a $k$-tuple $\langle\, \Gamma A(x)\, , \ldots ,\, \Gamma A(x)\, \rangle$

Logarithmic law: $\mathbb{P}(X = k) = \dfrac{1}{\log(1-\lambda)^{-1}} \dfrac{\lambda^k}{k}$

Remark: the laws are given by simple sequential algorithms

# Examples of possible labelled classes

- permutations, derangements, involutions,
- surjections,
- set partitions,
- necklaces,
- labelled (planar) trees,
- functional graphs,
- ...



**Theorem (Labelled free Boltzmann samplers)**

*For a labelled class $\mathcal{C}$ specified (poss. recursively) using the following constructions:*

$$\varepsilon,\ \mathcal{Z},\ +,\ \times,\ \textsc{Seq},\ \textsc{Set},\ \textsc{Cyc}$$

*The free Boltzmann sampler $\Gamma C(x)$ operates in linear time in the size of the object produced.*

- oracle complexity is not involved,
- size is not controlled (yet).

# To begin: $\mathrm{MSet}_2$

$\mathrm{MSET}_2(\mathcal{A}) \cong$ unordered set of two objects of $\mathcal{A}$

$$\begin{aligned} \mathcal{C} &= \mathrm{MSET}_2(\mathcal{A}) \\ C(z) &= \frac{1}{2}A^2(z) + \frac{1}{2}A(z^2) \qquad [\ 2\mathrm{MSET}_2(\mathcal{A}) = \mathcal{A}^2 + \Delta\mathcal{A}^2\ ] \end{aligned}$$

## Algorithm: $\Gamma C(x)$

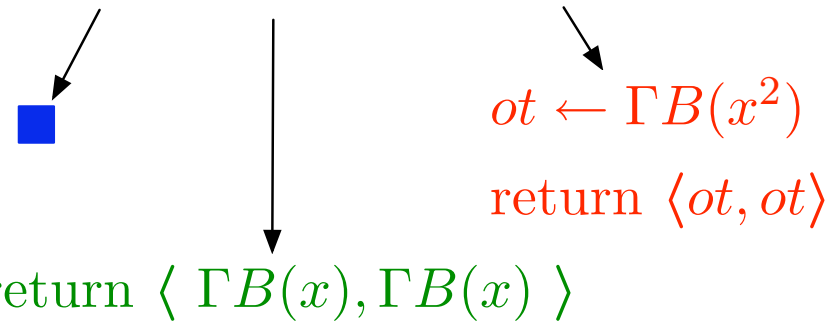
**if** $Bern\left(\frac{1}{2}\frac{A^2(x)}{C(x)}\right) = 1$ **then**
  Return $\langle\ \Gamma A(x), \Gamma A(x)\ \rangle$
**else**
  $a \leftarrow \Gamma A(x^2)$;
  Return $\langle a, a\rangle$;
**end if**

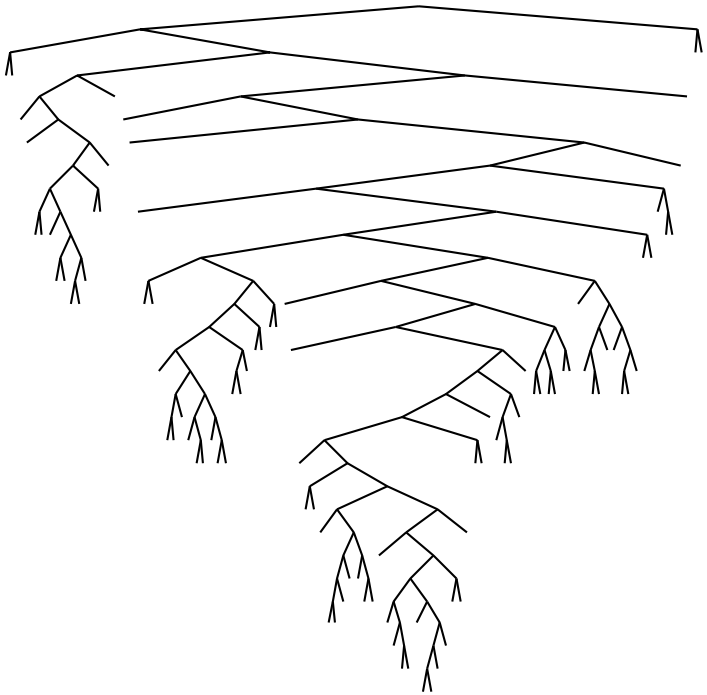
# A simple example

Unlabelled binary trees
(Otter trees)

$$\mathcal{B} = \mathcal{Z} + \mathrm{MSET}_2(\mathcal{B})$$

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



# MSet: the general case

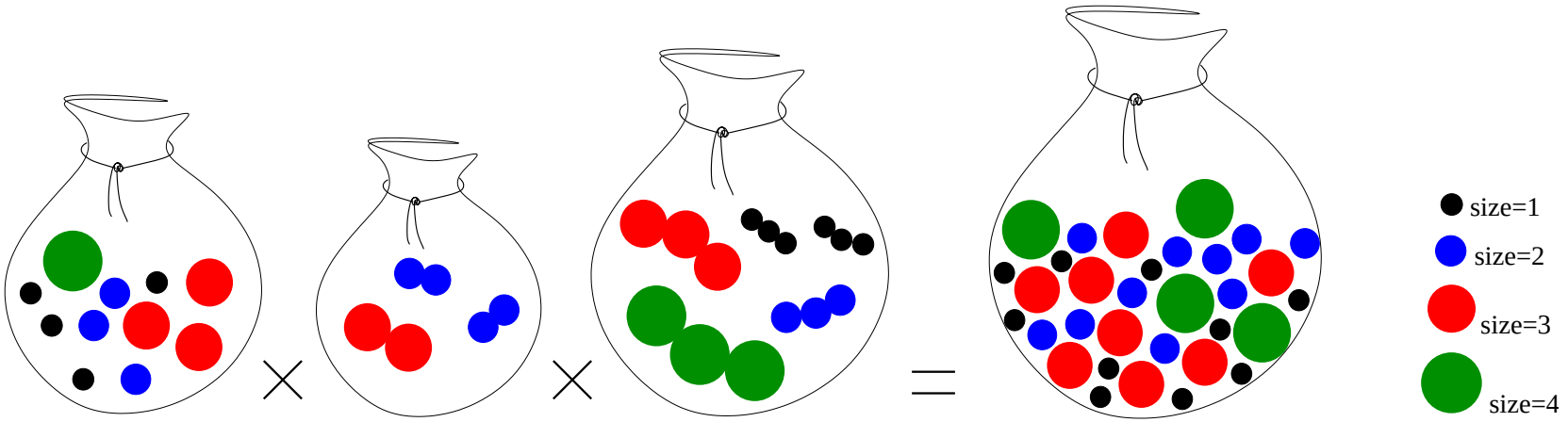
$$\mathcal{M} = \textsc{MSet}(\mathcal{A}) \cong \prod_{\gamma \in \mathcal{A}} \textsc{Seq}(\gamma) \quad \Rightarrow \quad M(z) = \prod_{\gamma \in \mathcal{A}} (1 - z^{|\gamma|})^{-1}$$

$$M(z) = \exp\left( \sum_{k=1}^{\infty} \frac{1}{k} A(z^k) \right) = \prod_{k=1}^{\infty} \exp\left( \frac{1}{k} A(z^k) \right)$$

# MSet: the general case

$$\mathcal{M} = \textsc{MSet}(\mathcal{A}) \cong \prod_{\gamma \in \mathcal{A}} \textsc{Seq}(\gamma) \quad \Rightarrow \quad M(z) = \prod_{\gamma \in \mathcal{A}} (1 - z^{|\gamma|})^{-1}$$

$$M(z) = \exp\left( \sum_{k=1}^{\infty} \frac{1}{k} A(z^k) \right) = \prod_{k=1}^{\infty} \exp\left( \frac{1}{k} A(z^k) \right)$$



# MSet

## Algorithm $\Gamma MSet[\mathcal{A}](x)$

- Draw $k$, the max. index of a subset, depending on $x$;
- For each index $i$ of a subset until $k-1$
  - Draw the number $p$ of elements to sample, according to a Poisson law of parameter $\frac{1}{i}A(x^i)$.
  - Call $\Gamma A(x^i)$ $p$ times, and each time, add $i$ copies of the result to the multiset.
- for index $k$, draw the number $p$ of elements to generate, according to a non zero Poisson law.

Proof: begin with the product sampler and use $\text{Geom}(\lambda) \equiv \sum_k \text{Pois}(\frac{\lambda^k}{k})$.

# How to compute the max. index for $\mathcal{C} = \mathrm{MSet}(\mathcal{A})$

Draw $U$, uniformly at random in $[0,1]$ and find $k$ such that:

$$\frac{\exp\left(\sum_{i=1}^{k-1} \frac{1}{i} A(x^i)\right)}{C(x)} < U \leq \frac{\exp\left(\sum_{i=1}^{k} \frac{1}{i} A(x^i)\right)}{C(x)}$$

with i.e.:

$$\sum_{i=k+1}^{\infty} \frac{1}{i} A(x^i) < \log \frac{1}{U} \leq \sum_{i=k}^{\infty} \frac{1}{i} A(x^i)$$

## Algorithm

$U \leftarrow \mathrm{rand}(0,1)$; $V \leftarrow \log \frac{1}{U}$;
$p \leftarrow \log C(x)$; $k \leftarrow 0$;
**while** $V < p$ **do**
$\quad k \leftarrow k+1$; $p \leftarrow p - \frac{A(x^k)}{k}$;
Return $k$;

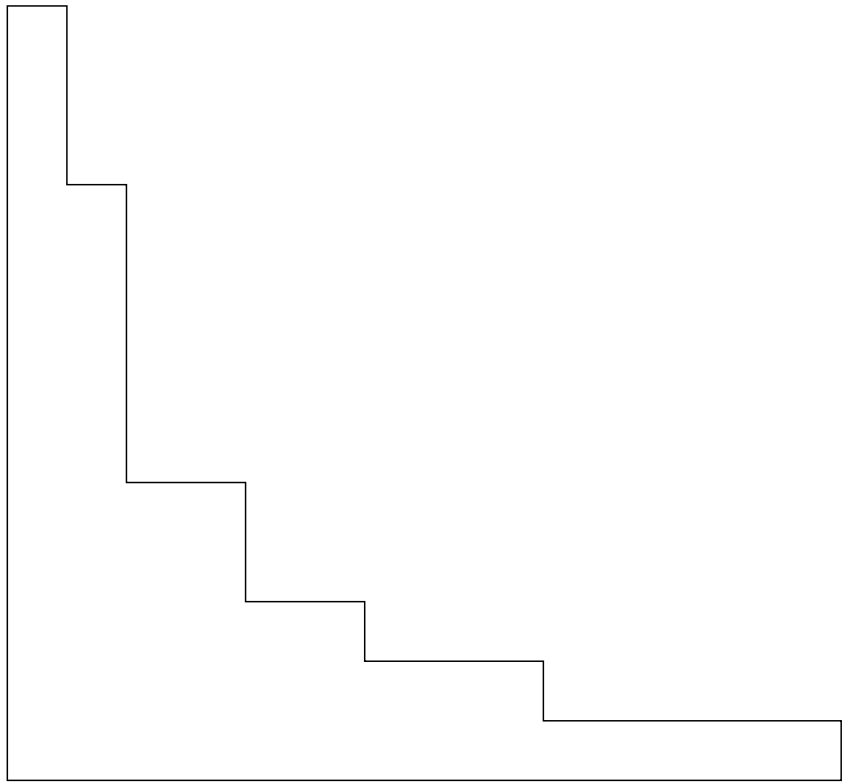
# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

$\boldsymbol{k = 3};$

$p_1 = 5;$

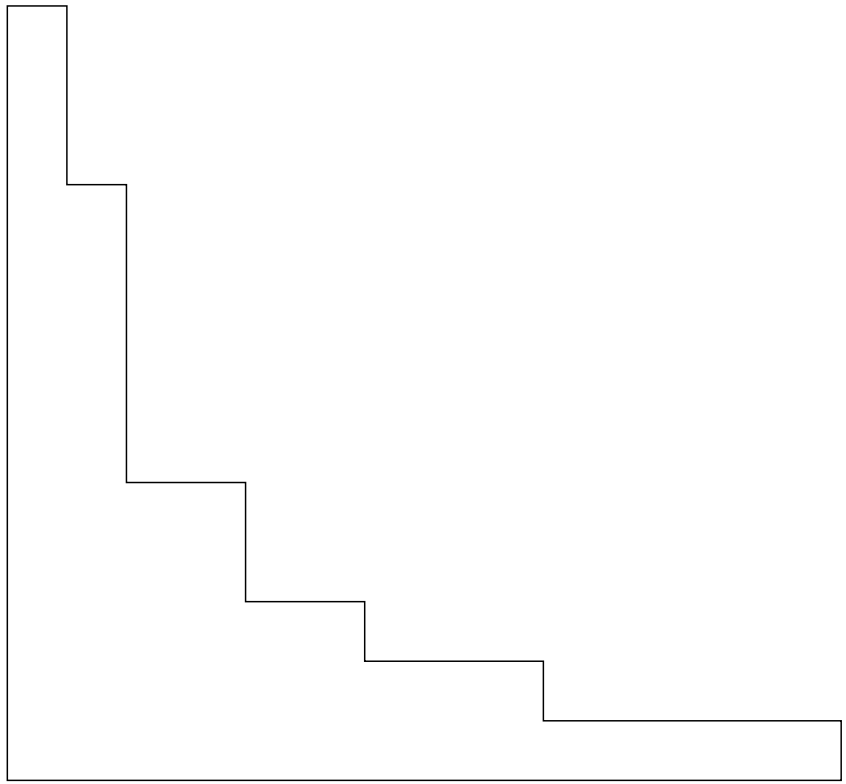
$p_2 = 3;$

$p_3 = 1;$



# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

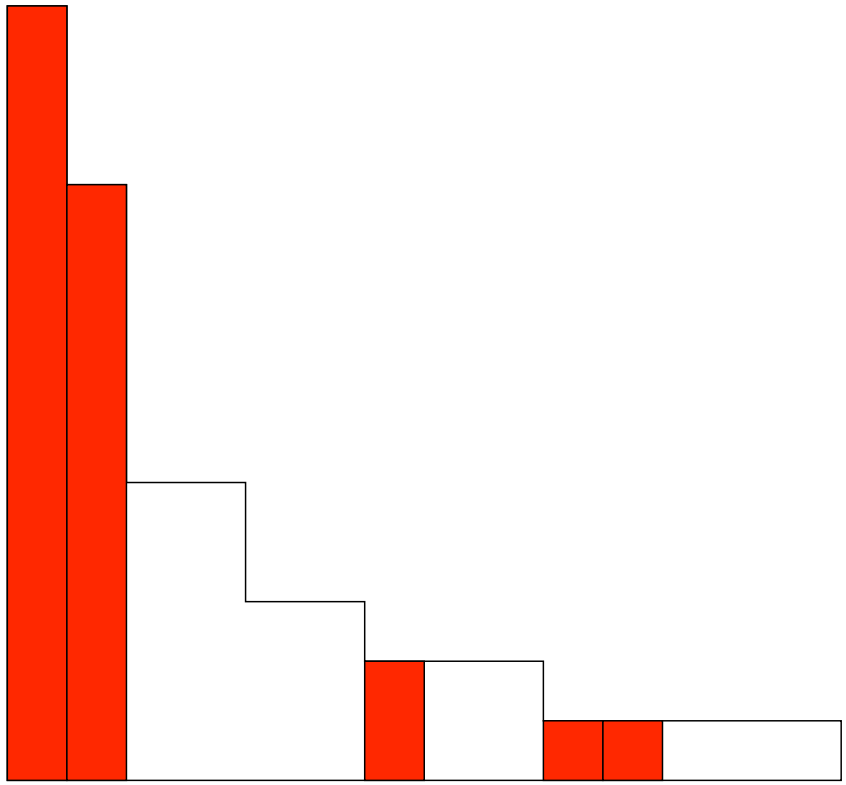
$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$



# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$

$\rightarrow 13,\ 10,\ 2,\ 1,\ 1$



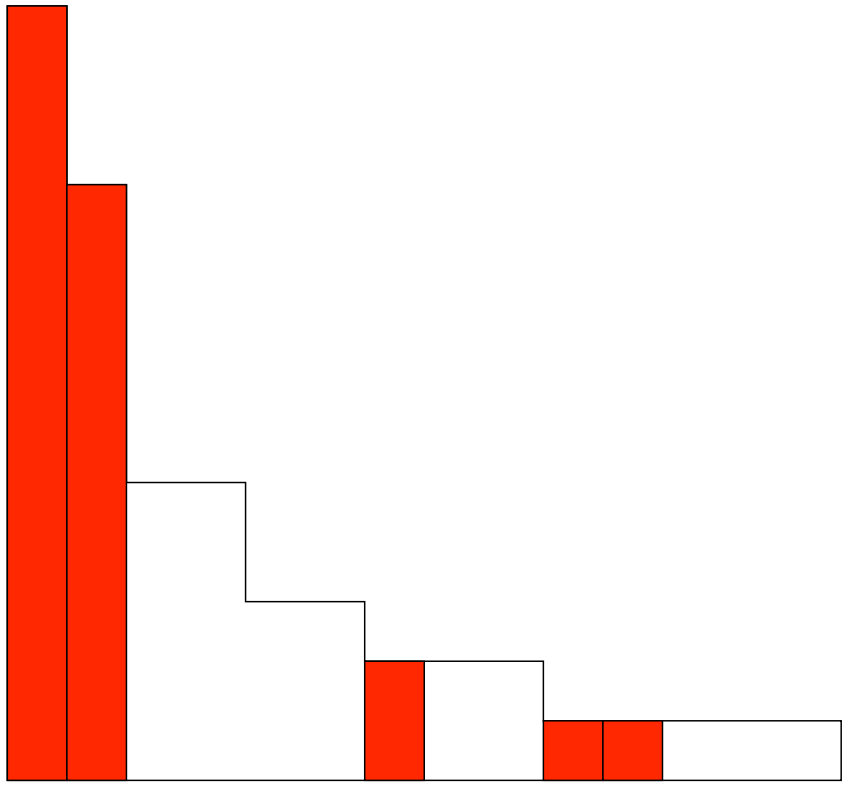
# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

$\boldsymbol{k} = 3;$

$p_1 = 5;$

$p_2 = 3;$

$p_3 = 1;$



# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

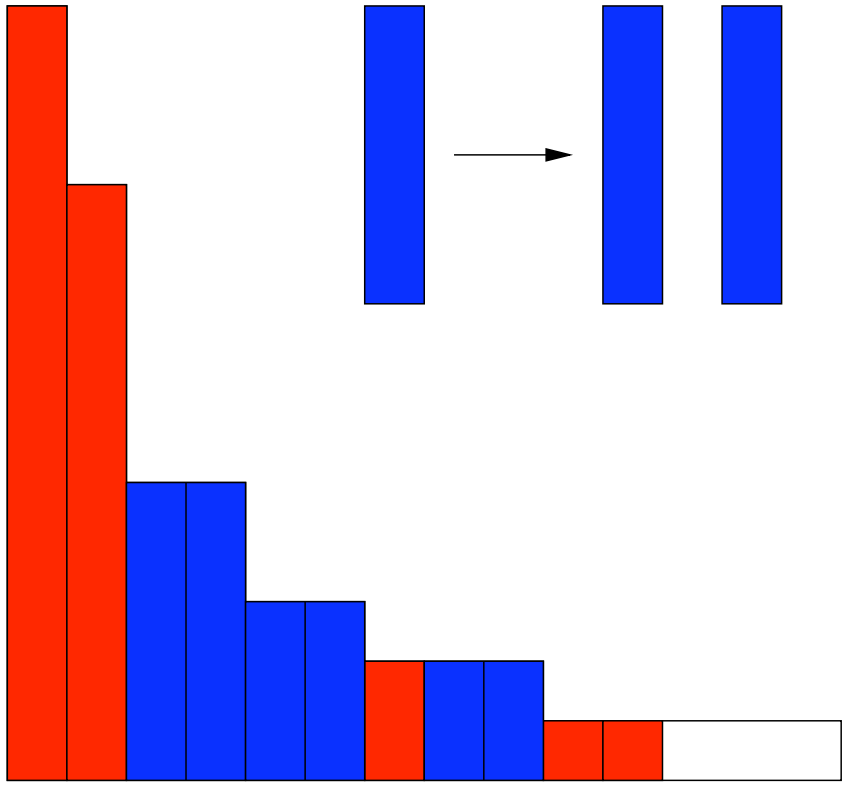
$\boldsymbol{k} = 3;$

$p_1 = 5;$

$p_2 = 3;$

$p_3 = 1;$

$\rightarrow 5,\ 3,\ 2$



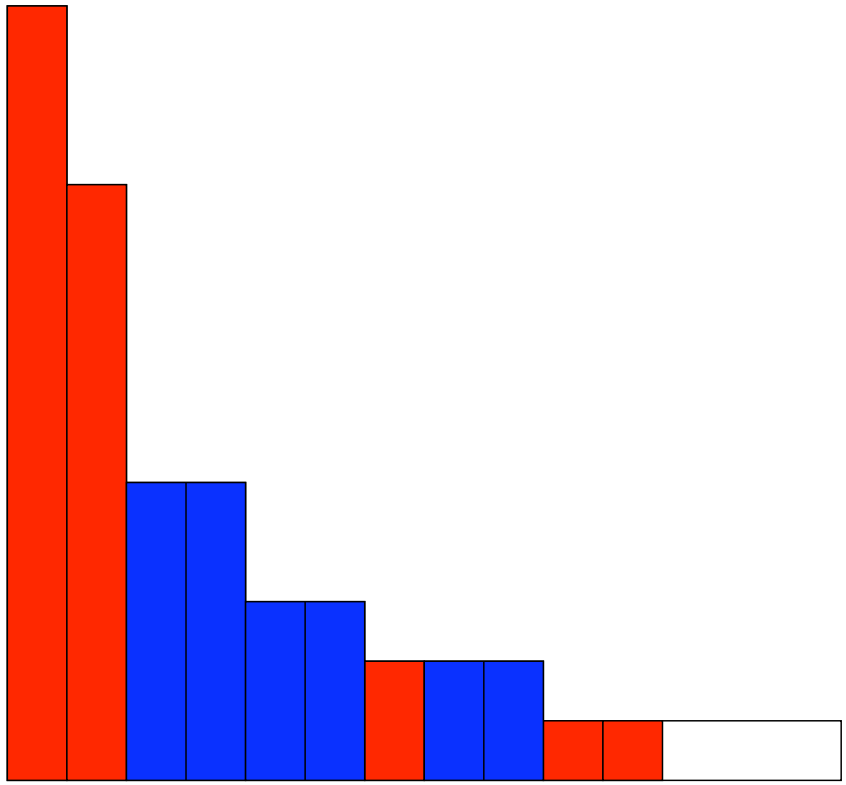
# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

$\boldsymbol{k} = 3;$

$p_1 = 5;$

$p_2 = 3;$

$p_3 = 1;$



# Sampling of a partition $\mathrm{MSet}(\mathcal{Z} \times \mathrm{Seq}(\mathcal{Z}))$

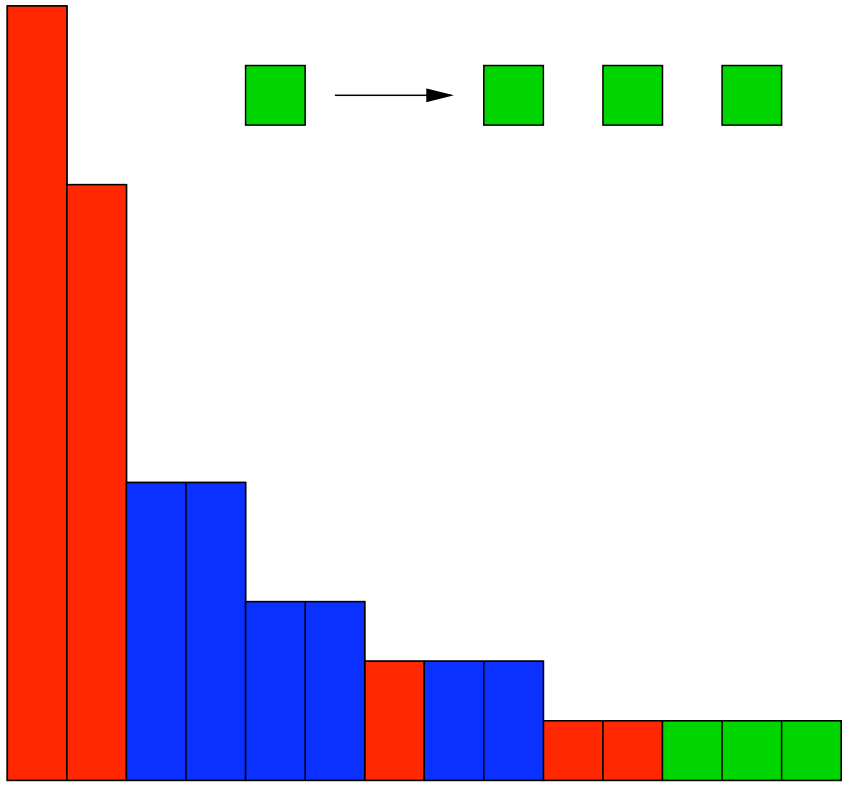
$\boldsymbol{k} = 3;$

$p_1 = 5;$

$p_2 = 3;$

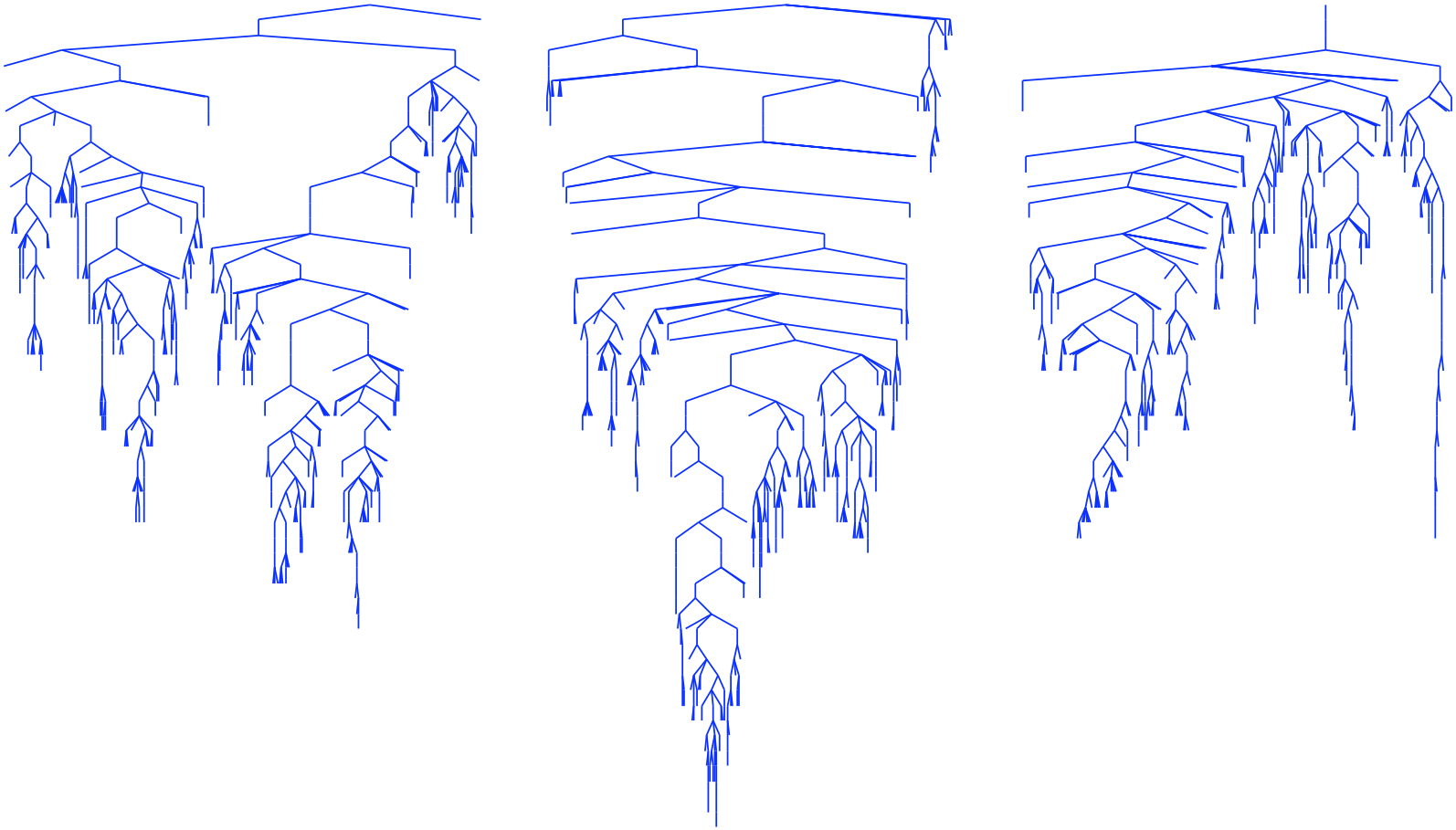
$p_3 = 1;$

$\rightarrow 1$



# Cayley trees

$$\mathcal{T} = \mathcal{Z} \times \textsc{MSet}(\mathcal{T})$$



# From MSet to PSet

**Principle** : Use the following non ambiguous decomposition:

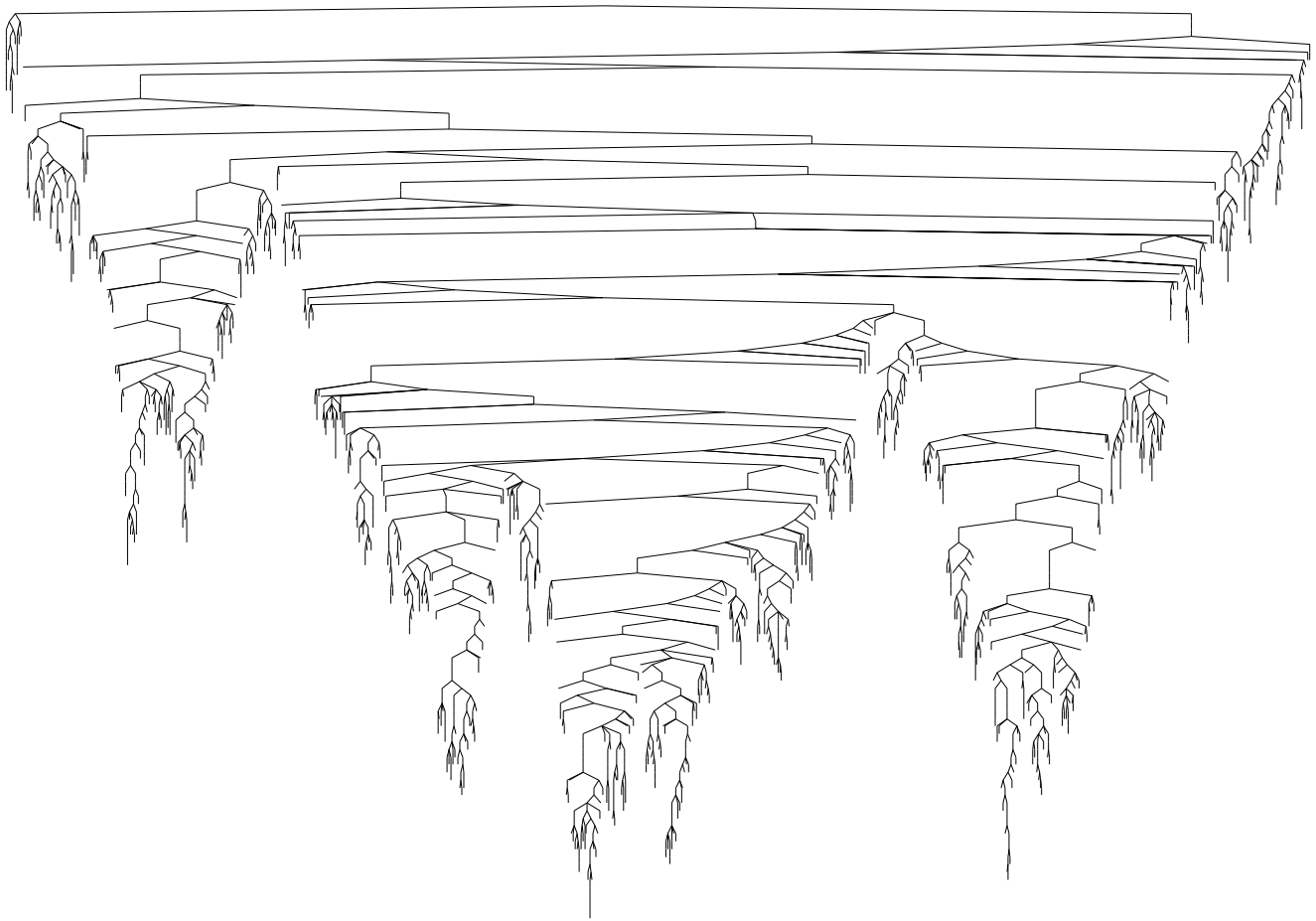
$$\begin{aligned} \textsc{MSet}(\mathcal{A}) &= \textsc{PSet}(\mathcal{A}) \times \textsc{MSet}(\mathcal{A}^{(2)}) \\ \prod_\gamma \frac{1}{1-z^{|\gamma|}} = \prod_\gamma \frac{1+z^{|\gamma|}}{1-z^{2|\gamma|}} &= \prod_\gamma (1+z^{|\gamma|}) \prod_\gamma \frac{1}{1-z^{2|\gamma|}} \end{aligned}$$

The algo. $\Gamma PSet[\mathcal{A}](x)$ to sample a powerset of objects of $\mathcal{A}$ is:

- Sample a multiset with $\Gamma MSet[\mathcal{A}](x)$,
- Extract the corresponding powerset :
  - by removing objects with even multiplicity,
  - and keeping only one occurrence of objects with **odd multiplicity**.

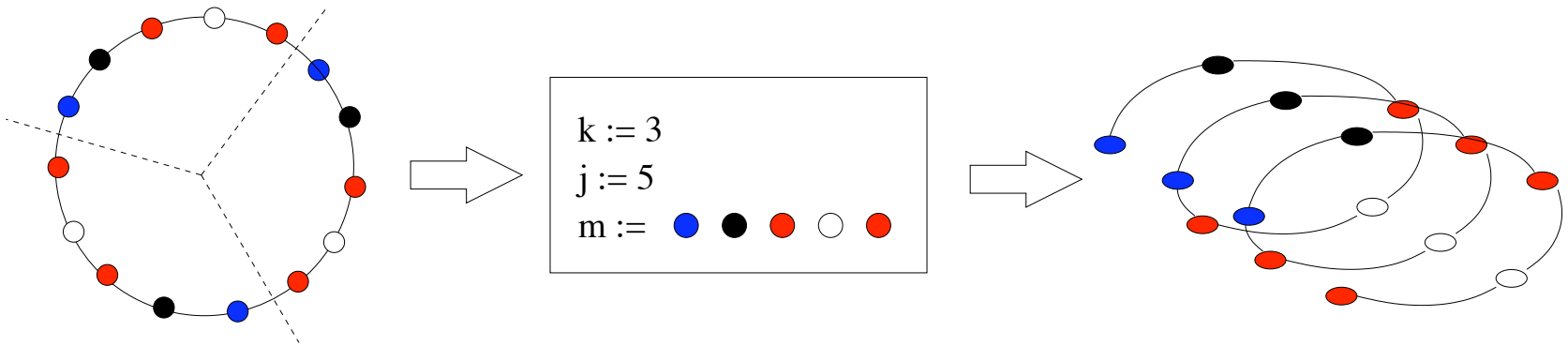
# Trees without twins

$$\mathcal{T} = \mathcal{Z} \times \textsc{PSet}(\mathcal{T})$$



# Cycles

$$\mathcal{C} = \textsc{Cyc}(\mathcal{A}) \quad \Rightarrow \quad C(z) = \sum_{k \geq 1} \frac{\varphi(k)}{k} \log \frac{1}{1 - A(z^k)}$$



# Algorithm

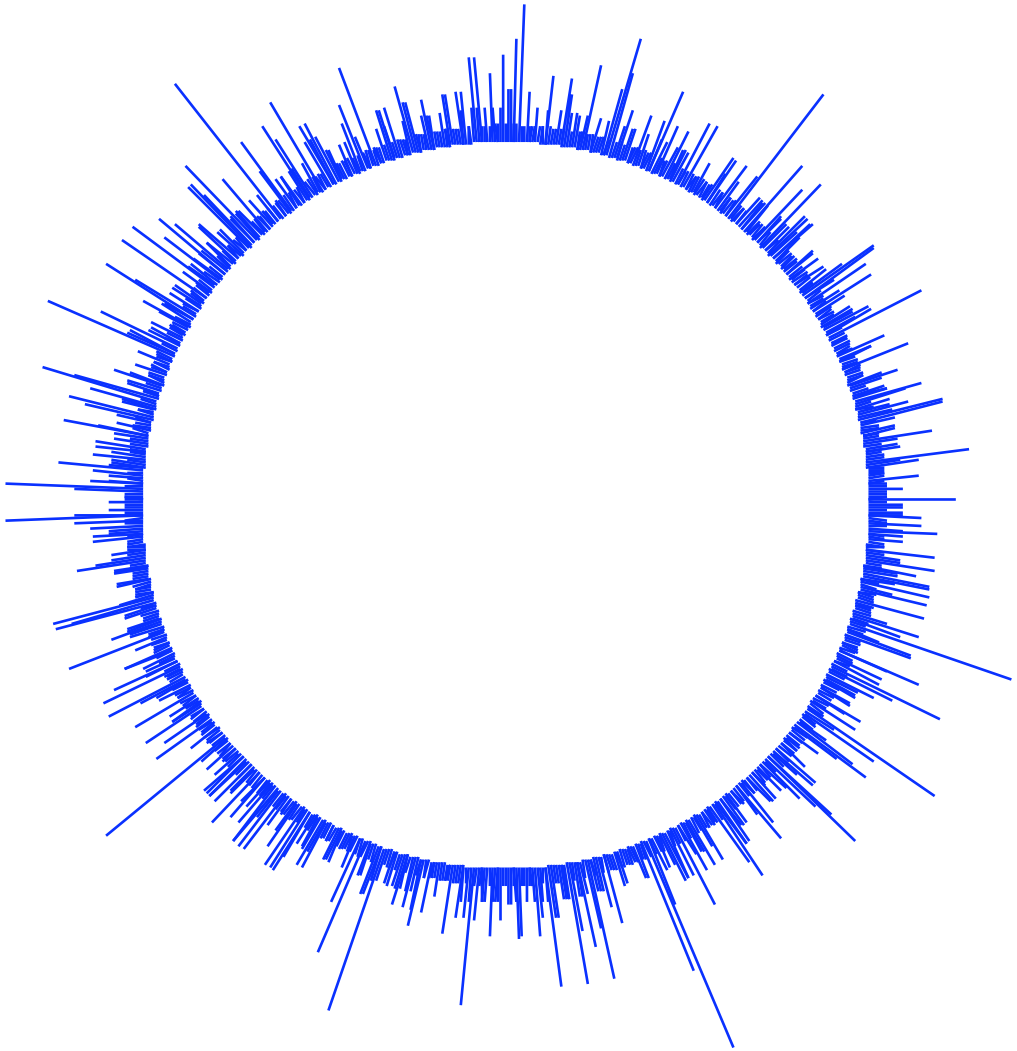
**$\Gamma Cyc[\mathcal{A}](x)$**

- Draw the replication order $k$ of the cycle.
- Draw the length $j$ of the pattern according to a logarithmic law of parameter $A(x^k)$.
- Draw the pattern $m$, calling $\Gamma A(x^k)$ $j$ times.
- Return a cycle composed of $k$ copies of $m$.

Proof: find $\mathbb{P}_x(\gamma)$ for any $\gamma \in \textsc{Cyc}(\mathcal{A})$, using $\mathbb{P}(k,j) = \frac{\varphi(k)}{kj} \frac{A(x^k)^j}{C(x)}$ and $\sum_{k|m} \varphi(k) = m$.

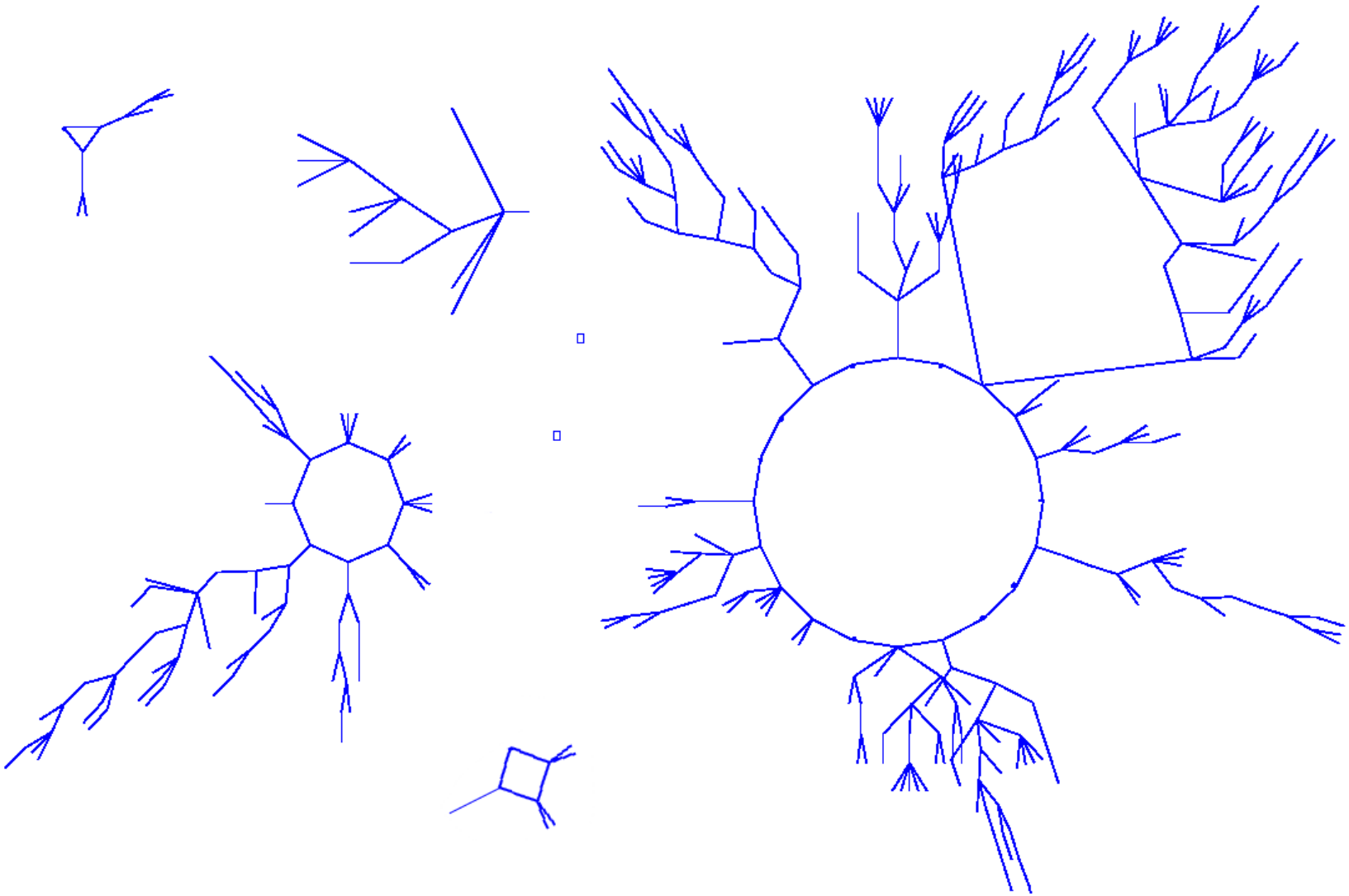
# Cyclic compositions

$$\begin{aligned} \mathcal{C} &= \textsc{Cyc}(\mathcal{Z} \times \textsc{Seq}(\mathcal{Z})) \\ C(z) &= \sum_{k=1}^{\infty} \frac{\varphi(k)}{k} \log \frac{1}{1 - \frac{z^k}{1-z^k}} \end{aligned}$$



# Functional graphs

$$\mathcal{G} = \textsc{Set}(\mathcal{C}),\ \mathcal{C} = \textsc{Cyc}(\mathcal{T}),\ \mathcal{T} = \mathcal{Z} \times \textsc{MSet}(\mathcal{T})$$



# $\mathbf{MSet}_k(\mathcal{A})$ et $\mathbf{Cyc}_k(\mathcal{A})$

Samplers are deduced from the generating functions:

- $\textsc{MSet}_k(\mathcal{A})$

$$M_k(z) = [u^k] \exp\left( uA(z) + \frac{u^2}{2}A(z^2) + \frac{u^3}{3}A(z^3) + \dots \right)$$

*Example:* $M_3(z) = \frac{1}{6}A(z)^3 + \frac{1}{2}A(z)A(z^2) + \frac{1}{3}A(z^3)$

- $\textsc{Cyc}_k(\mathcal{A})$

$$C_k(z) = [u^k] \sum_{l=1}^{\infty} \frac{\varphi(l)}{l} \log \frac{1}{1 - u^l A(z^l)}$$

*Example:* $C_4(z) = \frac{1}{4}A(z)^4 + \frac{1}{4}A(z^2)^2 + \frac{1}{2}A(z^4)$

# Other constraints

- $\textsc{MSet}_{<k}$, $\textsc{PSet}_{<k}$, $\textsc{Cyc}_{<k}$

  Union over $i < k$ of sets or cycles with $i$ elements.

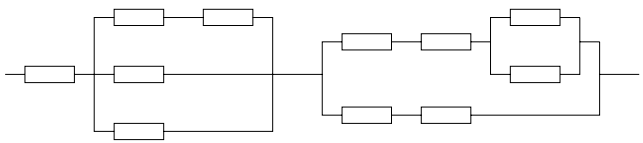
  *Example:* $\textsc{Cyc}_{<k}(\mathcal{A}) = \bigcup_{i=1}^{k-1} \textsc{Cyc}_i(\mathcal{A})$

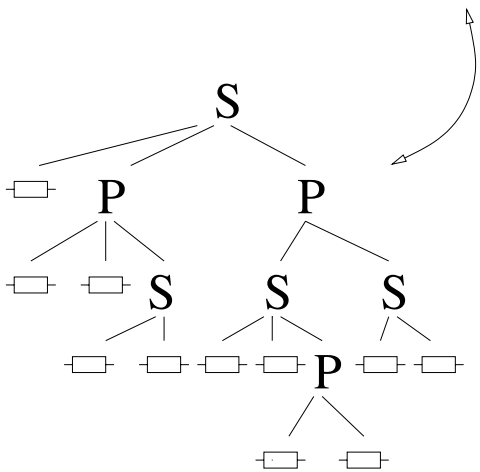
- $\textsc{PSet}_k$, $\textsc{MSet}_{>k}$, $\textsc{PSet}_{>k}$, $\textsc{Cyc}_{>k}$

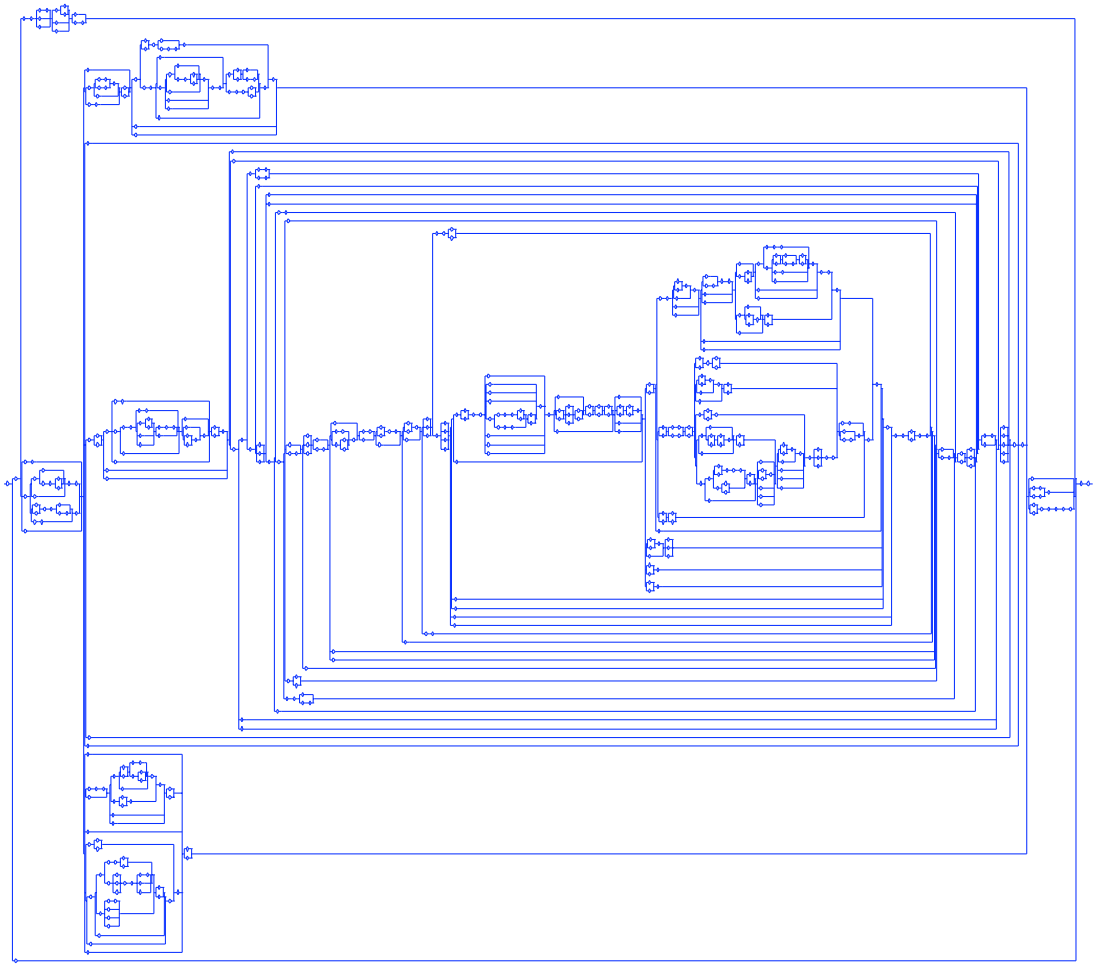
  Rejection.

# Series-parallel circuits

$$\begin{aligned}\mathcal{C} &= \mathcal{P} + \mathcal{S} + \mathcal{Z}\\ \mathcal{S} &= \textsc{Seq}_{\geq 2}(\mathcal{P} + \mathcal{Z})\\ \mathcal{P} &= \textsc{MSet}_{\geq 2}(\mathcal{S} + \mathcal{Z})\end{aligned}$$





**Theorem (Unlabelled free Boltzmann samplers)**

*For an unlabelled class $\mathcal{C}$ specified (poss. recursively) using the following constructions:*

$$\varepsilon,\ \mathcal{Z},\ +,\ \times,\ \textsc{Seq},\ \textsc{Seq}_k,\ \textsc{MSet},\ \textsc{MSet}_k,\ \textsc{Cyc},\ \textsc{Cyc}_k$$

*The free Boltzmann sampler $\Gamma C(x)$ operates in linear time in the size of the object produced.*

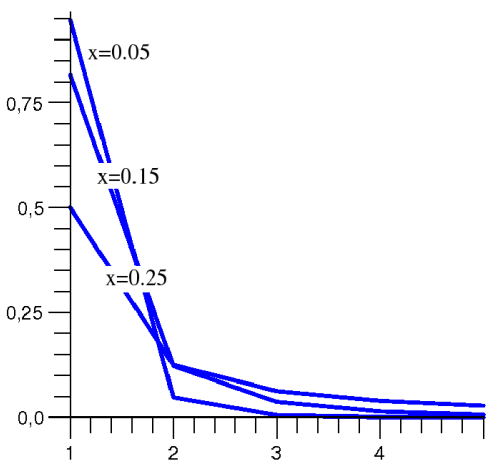
SET: not so bad!

if $\rho < 1$ then the *overhead* (total size of the discarded elements) is bounded by a constant.

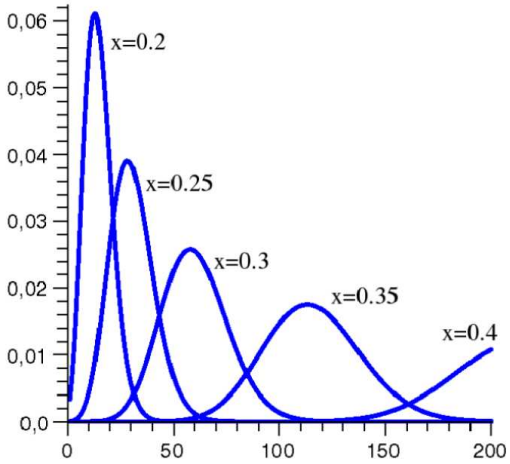
# Technical matters

# Size control – parameter tuning

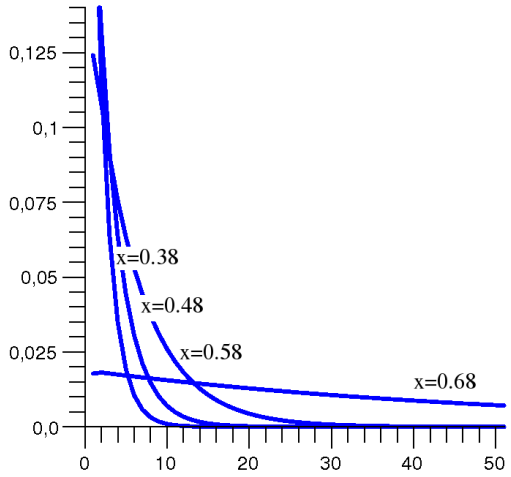
- Free samplers: produce objects with randomly varying sizes!
- Tuned samplers: choose $x$ so that expected size is $n$.
- Size distribution of free sampler determines complexity.




Peaked




Bumpy




Flat

# Size control – 2

Cost of tuned Boltzmann samplers for a labelled or unlabelled class $\mathcal{C}$ specified (poss. recursively) using the following constructions:

$\varepsilon$, $\mathcal{Z}$, $+$, $\times$, $\textsc{Seq}$, $\textsc{Seq}_k$, $\textsc{MSet}$, $\textsc{MSet}_k$, $\textsc{Cyc}$, $\textsc{Cyc}_k$

**Theorem (Bumpy type)**

*Approximate-size complexity* $= \mathcal{O}(n)$. *Exact size* $= o(n^2)$.

**Theorem (Flat type)**

*Approximate-size complexity* $= \mathcal{O}(n)$. *Exact-size* $= o(n^2)$.

**Theorem (Peaked type)**

*Approximate-size complexity* $= \mathcal{O}(n^2)$ $\rightarrow$ *pointing*.

# Pointing

If $\mathcal{A}$ is a class, then $\mathcal{C} = \mathcal{A}^\bullet$ is the set of objects with one atom pointed, and

$$C_n = nA_n, \qquad C(z) = z\frac{d}{dz}A(z).$$

*Uniformity at given size is preserved, but size profile is altered.*
Transforms peaked (inefficient) to flat (efficient).
For example, binary trees $\mathcal{B}$:

$$\mathcal{B} = \mathcal{Z} + \mathcal{B} \times \mathcal{B} \quad \Longrightarrow \quad \left\{ \begin{array}{lcl} \mathcal{B} & = & \mathcal{Z} + \mathcal{B} \times \mathcal{B} \\ \mathcal{B}^\bullet & = & \mathcal{Z}^\bullet + \mathcal{B}^\bullet \times \mathcal{B} + \mathcal{B} \times \mathcal{B}^\bullet. \end{array} \right.$$

It works for all simple families of trees.

# Singular samplers

- critical sequences


  exact-size: $\mathcal{O}(n)$.

- regular languages
  if the automaton recognizing $\mathcal{L}$ is strongly connected:
  exact-size: $\mathcal{O}(n)$.

- trees
  with early interrupted execution:
  approx-size: $\mathcal{O}(n)$, exact-size: $\mathcal{O}(n^2)$.

# Concluding remarks

# Already done

**BaNi06** *Accessible and deterministic automata: enumeration and Boltzmann samplers*, by F. Bassino C. Nicaud. In *Fourth Colloquium on Mathematics and Computer Science.*

**BoFuPi06** *Random sampling of plane partitions*, by O. Bodini, E. Fusy, and C. Pivoteau. In *GASCOM*-2006.

**DaSo07** *Degree distribution of random Apollonian network structures and Boltzmann sampling*, by A. Darrasse and M. Soria. In *International Conference on Analysis of Algorithms*, 2007, *DIMACS.*

**Fusy05** *Quadratic exact-size and linear approximate-size random sampling of planar graphs*, by E. Fusy. In *International Conference on Analysis of Algorithms*, 2005, *DMTCS* Conference Volume AD (2005), pp. 125-138.

**PaWe07** *Properties of Random Graphs via Boltzmann Samplers*, by K. Panagiotou and A. Weißl. In *International Conference on Analysis of Algorithms*, 2007, *DIMACS.*

**Ponty06** *Modélisation de séquences génomiques structurées, génération aléatoire et application*, by Yann Ponty, PhD Thesis, Université Paris-Sud, 2006.

...

# Coming soon...?

- other constructions: box operator, shuffle, ...
- multivariate Boltzmann samplers,
- automatic oracle, singularities,
- discrete samplers,
- specialized samplers,
- new applications,
- ...