Random generation of combinatorial structures using Boltzmann samplers

Éric Fusy

Algorithms Project INRIA Rocquencourt

Plan

Plan

1. Recursive counting and recursive sampling
2. Generating functions and Boltzmann samplers
3. New constructions for Boltzmann samplers
4. Random sampling of plane partitions

Plan

1. Recursive counting and recursive sampling
2. Generating functions and Boltzmann samplers
3. New constructions for Boltzmann samplers
4. Random sampling of plane partitions

Plan

1. **Recursive counting and recursive sampling**
2. Generating functions and Boltzmann samplers
3. New constructions for Boltzmann samplers
   - Multisets
   - Cycles
4. Random sampling of plane partitions

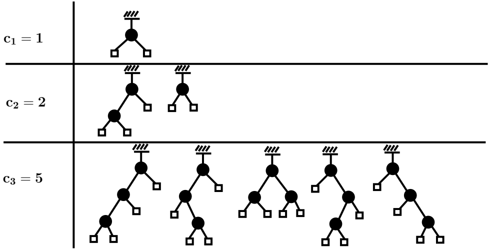
Combinatorial classes

A combinatorial class $\mathcal{C}$ is a set of objects such that

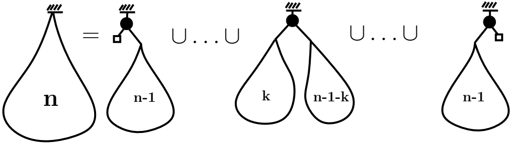
- Each object $\gamma \in \mathcal{C}$ has a size $|\gamma| \in \mathbb{N}$ (e.g. number of vertices in a graph)
- For $n \geq 0$, the number of objects of size n in $\mathcal{C}$ is finite. This number c_n is called the nth coefficient of $\mathcal{C}$

Example: binary trees

- Each inner node has a left son and a right son
- The size is the number of inner nodes



$$c_n = c_0 c_{n-1} + \ldots + c_k c_{n-1-k} + \ldots + c_{n-1} c_0$$

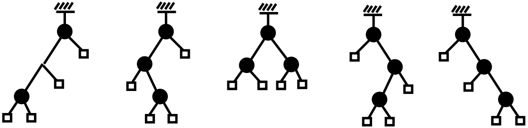


Random generation at a fixed size n

For a given size $n \geq 0$, we want a procedure that picks up an object of size n under the uniform distribution:

$$\Pr(\gamma) = \frac{1}{c_n} \text{ for each } \gamma \in \mathcal{C} \text{ of size } n$$

Binary trees: uniform distribution for $n = 3$

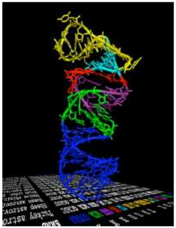


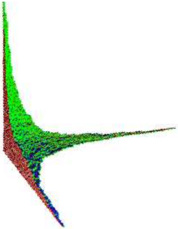
$\Pr = \frac{1}{5}$ $\Pr = \frac{1}{5}$ $\Pr = \frac{1}{5}$ $\Pr = \frac{1}{5}$ $\Pr = \frac{1}{5}$

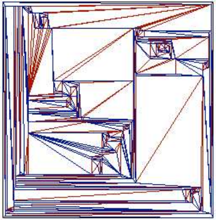
Motivations for random generation

- Observation of asymptotic phenonema

- Generation of structures for experimentation :
  - bio-informatics
  - software testing
  - ...





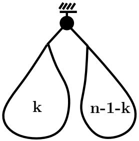


A general procedure: the recursive method

Idea: (Nijenhuis and Wilf'78, Flajolet et al'94) compute the correct probability at the root of the decomposition from the coefficients to have the uniform distribution.

$$\mathbf{c_n = c_0 c_{n-1} + \ldots + c_k c_{n-1-k} + \ldots + c_{n-1} c_0}$$

Proportion of

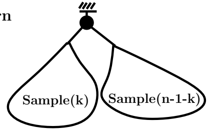


$$= \frac{\mathbf{c_k c_{n-1-k}}}{\mathbf{c_n}}$$

⇒ Recursive algorithm:

Sample(n): 1) choose $k \in \{0, \ldots, n-1\}$ **under the distribution** $\boxed{\Pr(\mathbf{k}) = \frac{\mathbf{c_k c_{n-1-k}}}{\mathbf{c_n}}}$

2) return



Drawback: large auxiliary memory

To sample at size n, we need the coefficients $c_1, \ldots, c_n$
$\Rightarrow$ this requires a quadratic number of bits

n	c_n	n	c_n	n	c_n
1	1	11	58786	21	24466267020
2	2	12	208012	22	91482563640
3	5	13	742900	23	343059613650
4	14	14	2674440	24	1289904147324
5	42	15	9694845	25	4861946401452
6	132	16	35357670	26	18367353072152
7	429	17	129644790	27	69533550916004
8	1430	18	477638700	28	263747951750360
9	4862	19	1767263190	29	1002242216651368
10	16796	20	6564120420	30	3814986502092304

1. Recursive counting and recursive sampling

2. **Generating functions and Boltzmann samplers**

3. New constructions for Boltzmann samplers
   - Multisets
   - Cycles

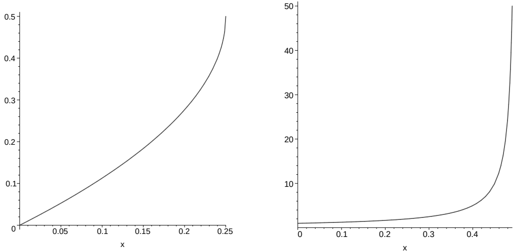
4. Random sampling of plane partitions

Generating functions

The generating function of a class $\mathcal{C}$ is defined as:

$$C(x) := \sum_{\gamma \in \mathcal{C}} x^{|\gamma|} = \sum_{n \geq 0} c_n x^n$$

There is a critical value $\rho > 0$ such that the sum defining the generating function converges for $x < \rho$ and not for $x > \rho$



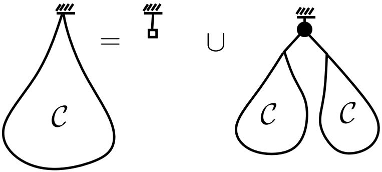
Generating functions: rules of calculations

- Disjoint union: $\boxed{\mathcal{C} = \mathcal{A} \cup \mathcal{B} \Rightarrow C(x) = A(x) + B(x)}$

  Proof: $\sum_{\gamma \in \mathcal{A} \cup \mathcal{B}} x^{|\gamma|} = \sum_{\gamma \in \mathcal{A}} x^{|\gamma|} + \sum_{\gamma \in \mathcal{B}} x^{|\gamma|}$

- Cartesian prod: $\boxed{\mathcal{C} = \mathcal{A} \times \mathcal{B} \Rightarrow C(x) = A(x) \cdot B(x)}$

  Proof: $\sum_{(\gamma_1, \gamma_2) \in \mathcal{A} \times \mathcal{B}} x^{|(\gamma_1, \gamma_2)|} = \sum_{\gamma_1, \gamma_2} x^{|\gamma_1| + |\gamma_2|} = \sum_{\gamma_1 \in \mathcal{A}} x^{|\gamma_1|} \sum_{\gamma_2 \in \mathcal{B}} x^{|\gamma_2|}$



$$\boxed{C(x) = 1 \quad + \quad C(x) x C(x)}$$

simpler than $c_n = c_0 c_{n-1} + \ldots + c_k c_{n-1-k} + \ldots + c_{n-1} c_0.$

Boltzmann samplers

- Introduced by Duchon, Flajolet, Louchard and Schaeffer (2002)
- Relax the constraint of fixed size (cf recursive method) for random generation.
- The distribution is spread over all objects of the class.
- An object is drawn with probability proportional to the exponential of its size (cf statistical physics)

Boltzmann samplers: definition

- Let $\mathcal{C}$ be an unlabelled combinatorial class
  (e.g. binary trees)
  Ordinary generating function:

$$C(x) = \sum_{\gamma \in \mathcal{C}} x^{|\gamma|} = \sum_{n \geq 0} c_n x^n,$$

  where $|\gamma|$ is the size of γ.

- Given $x > 0$ ($x \leq \rho_C$) a fixed real value,
  a Boltzmann sampler $\Gamma C(x)$ is a procedure that draws each object γ of $\mathcal{C}$ with probability:

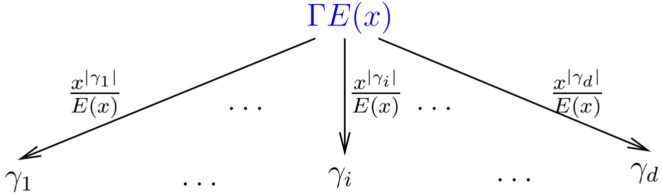
$$\boxed{\Pr(\gamma) = \frac{x^{|\gamma|}}{C(x)}}$$

Analogy with statistical physics

Combinatorics ⟷	**Stat. Phys.**
Structure γ	State s
size n	Energy E
Generating function $C(x) = \sum_\gamma x^{\lvert\gamma\rvert}$	Partition function $Z = \sum_{\mathrm{s}} e^{-\beta E}$
Boltz: $\mathbb{P}(\gamma) = \frac{x^{\lvert\gamma\rvert}}{C(x)}$	Boltz: $\mathbb{P}(s) = \frac{e^{-\beta E}}{Z}$

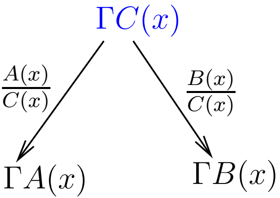
Finite sets

Let $\mathcal{E} = (\gamma_1, \ldots, \gamma_d)$ $\qquad E(x) = \sum_{i=1}^{d} x^{|\gamma_i|}$



The basic construction rules: union

Union: Let $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$. Assume we have Boltzmann samplers $\Gamma A(x)$ for $\mathcal{A}$ and $\Gamma B(x)$ for $\mathcal{B}$. Define $\Gamma C(x)$ as:



Then $\Gamma C(x)$ is a Boltzmann sampler for $\mathcal{A} \cup \mathcal{B}$.

Proof: Let $\gamma \in \mathcal{A} \cup \mathcal{B}$

- If $\gamma \in \mathcal{A}$, then $\Pr(\gamma) = \frac{A(x)}{C(x)} \cdot \frac{x^{|\gamma|}}{A(x)} = \frac{x^{|\gamma|}}{C(x)}$.
- If $\gamma \in \mathcal{B}$, then $\Pr(\gamma) = \frac{B(x)}{C(x)} \cdot \frac{x^{|\gamma|}}{B(x)} = \frac{x^{|\gamma|}}{C(x)}$.

The basic construction rules: product

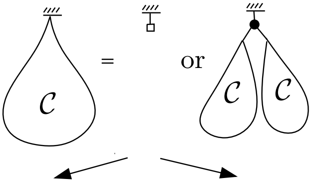
Product: Let $\mathcal{C} = \mathcal{A} \times \mathcal{B}$. Assume we have Boltzmann samplers $\Gamma A(x)$ for $\mathcal{A}$ and $\Gamma B(x)$ for $\mathcal{B}$. Define $\Gamma C(x)$ as:

$$\begin{array}{lll} \Gamma C(x) & : & \gamma_1 \leftarrow \Gamma A(x) \\ & & \gamma_2 \leftarrow \Gamma B(x) \\ & & \text{return } (\gamma_1, \gamma_2) \end{array}$$

Then $\Gamma C(x)$ is a Boltzmann sampler for $\mathcal{A} \cup \mathcal{B}$:
Proof: an object $\gamma = (\gamma_1, \gamma_2)$ has probability:

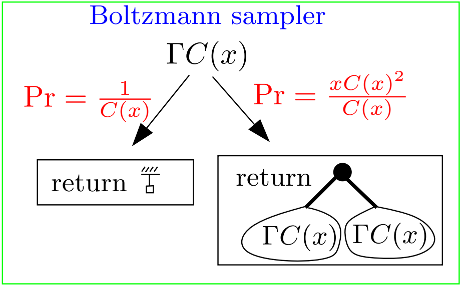
$$\frac{x^{|\gamma_1|}}{A(x)} \frac{x^{|\gamma_2|}}{B(x)} = \frac{x^{|\gamma_1|+|\gamma_2|}}{A(x) \cdot B(x)} = \frac{x^{|\gamma|}}{C(x)}$$

Example: binary trees



Generating function

$$C(x) = 1 + xC(x)^2$$



A first Sampling dictionary

Theorem

A Boltzmann sampler can be designed for any class having a recursive specification with the constructions $\cup$ and $\times$. The complexity is linear in the size of the output object.

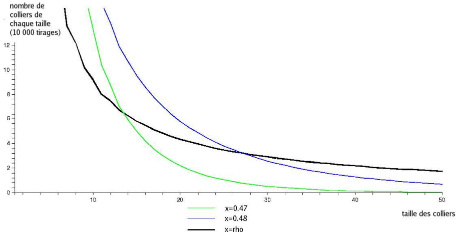
construction	generator
$\mathcal{C} = \varnothing$	$\Gamma C(x) :=$ return $\varnothing$
$\mathcal{C} = \{\bullet\}$	$\Gamma C(x) :=$ return $\{\bullet\}$
$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$\Gamma C(x) := \left(\text{Bern}\ \frac{A(x)}{C(x)} \longrightarrow \Gamma A(x) \mid \Gamma B(x) \right)$
$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$\Gamma C(x) := (\ \Gamma A(x); \Gamma B(x)\ )$

Choosing x to draw objects around a target-size

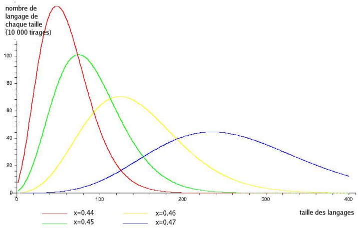
The probability of drawing an object of size n is:

$$\mathbb{P}_x(\text{Size} = n) = \sum_{|\gamma|=n} \frac{x^{|\gamma|}}{C(x)} = \frac{c_n x^n}{C(x)}$$

$\rightarrow$ size distributions for different values of x.



Bicolored necklaces



Finite languages

Boltzmann samplers *vs* the recursive method

	Boltzmann	recursive method
size distribution	$Pr(size = n) = \frac{c_n x^n}{C(x)}$	fixed size n
auxiliary memory	$\mathcal{O}(\log(n))$	$\mathcal{O}(n^2)$
time per generation	$\mathcal{O}(n^2)$ Exact $\mathcal{O}(n)$ Approx	$\mathcal{O}(n \log(n))$ Exact

New constructions for Boltzmann samplers

(with Philippe Flajolet and Carine Pivoteau)

1. Recursive counting and recursive sampling
2. Generating functions and Boltzmann samplers
3. **New constructions for Boltzmann samplers**
   - Multisets
   - Cycles
4. Random sampling of plane partitions

A first construction: MSet_2

$\textsc{MSet}_2(\mathcal{A}) \cong$ unorderer pairs of objects de $\mathcal{A}$

$$\mathcal{C} = \textsc{MSet}_2(\mathcal{A})$$

$$C(z) = \frac{1}{2}A^2(z) + \frac{1}{2}A(z^2) \qquad \left[\ 2\textsc{MSet}_2(\mathcal{A}) = \mathcal{A}^2 + \Delta\mathcal{A}^2\ \right]$$

Algorithm: $\Gamma C(x)$

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

Proof that the algorithm is correct

Proof.

Let $\gamma = \langle \gamma_1, \gamma_2 \rangle$ be an unordered pair of objects of $\mathcal{A}$. The probability that γ is drawn by $\Gamma C(x)$ is:

1. $\gamma_1 \neq \gamma_2$

$$\mathbb{P}_x(\gamma) = \frac{1}{2} \frac{A(x)^2}{C(x)} \, \frac{x^{|\gamma_1|}}{A(x)} \, \frac{x^{|\gamma_2|}}{A(x)} \times 2 = \frac{x^{|\gamma|}}{C(x)}$$

2. $\gamma_1 = \gamma_2$

$$\mathbb{P}_x(\gamma) = \frac{1}{2} \frac{A(x)^2}{C(x)} \, \frac{x^{|\gamma_1|}}{A(x)} \, \frac{x^{|\gamma_1|}}{A(x)} + \frac{1}{2} \frac{A(x^2)}{C(x)} \, \frac{x^{2|\gamma_1|}}{A(x^2)} = \frac{x^{|\gamma|}}{C(x)}$$

□

Application

Unembedded binary trees
(Otter trees)

$$\mathcal{B} = \mathcal{Z} + \mathrm{MS{\scriptsize ET}}_2(\mathcal{B})$$

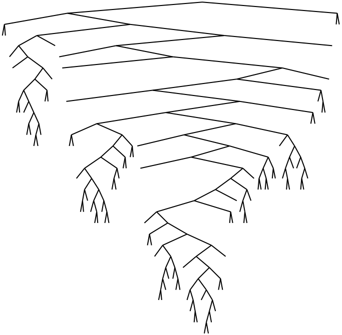
$$B(z) = z + \frac{1}{2}B^2(z) + \frac{1}{2}B(z^2)$$

$\rightarrow$ □

$\rightarrow$ ■$\langle\ \Gamma B(x), \Gamma B(x)\ \rangle$

$\rightarrow$ $ot \leftarrow \Gamma B(x^2)$;

return ■$\langle\ ot, ot\ \rangle$;



Application

Unembedded binary trees
(Otter trees)

$$\mathcal{B} = \mathcal{Z} + \text{MSet}_2(\mathcal{B})$$

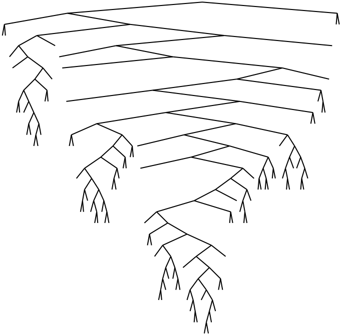
$$B(z) = z + \frac{1}{2}B^2(z) + \frac{1}{2}B(z^2)$$

$\rightarrow$ $\square$

$\rightarrow$ $\blacksquare\langle\, \Gamma B(x), \Gamma B(x)\,\rangle$

$\rightarrow$ $ot \leftarrow \Gamma B(x^2);$

return $\blacksquare\langle\, ot, ot\,\rangle;$



Application

Unembedded binary trees
(Otter trees)

$$\mathcal{B} = \mathcal{Z} + \textsc{MSet}_2(\mathcal{B})$$

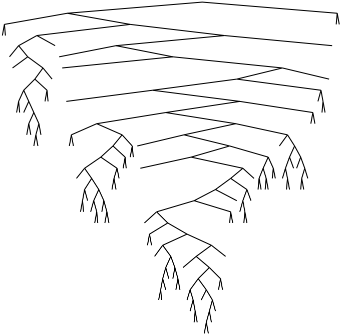
$$B(z) = z + \frac{1}{2}B^2(z) + \frac{1}{2}B(z^2)$$

$\rightarrow$ $\square$

$\rightarrow$ $\blacksquare\langle\ \Gamma B(x), \Gamma B(x)\ \rangle$

$\rightarrow$ $ot \leftarrow \Gamma B(x^2);$

return $\blacksquare\langle\ ot, ot\ \rangle;$

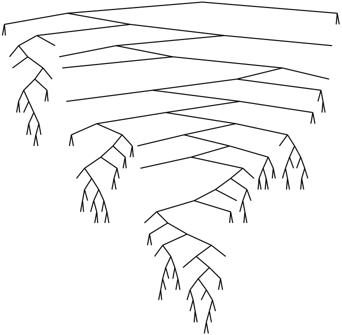


Application

Unembedded binary trees
(Otter trees)

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

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



$\rightarrow$ □

$\rightarrow$ ■$\langle\ \Gamma B(x), \Gamma B(x)\ \rangle$

$\rightarrow$ $ot \leftarrow \Gamma B(x^2)$;

return ■$\langle\ ot, ot\ \rangle$;

General MSet

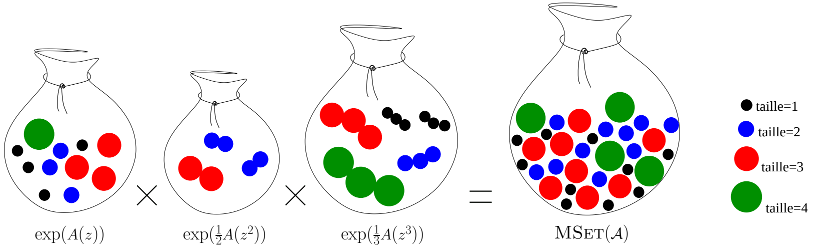
Let $\mathcal{M} := \textsc{MSet}(\mathcal{A})$ be the class of all multisets of objects of $\mathcal{A}$.

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

General MSet

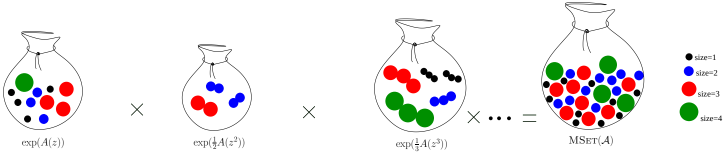
Let $\mathcal{M} := \text{MSet}(\mathcal{A})$ be the class of all multisets of objects of $\mathcal{A}$.

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



Algorithm for general MSet

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



repeat Poiss$(A(x))$ times $\gamma \leftarrow \Gamma A(x)$ add γ	repeat Poiss$(\frac{A(x^2)}{2})$ times $\gamma \leftarrow \Gamma A(x^2)$ add 2 copies of γ	repeat Poiss$(\frac{A(x^3)}{3})$ times $\gamma \leftarrow \Gamma A(x^3)$ add 3 copies of γ

Algorithm for general MSet (2)

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

$k \longleftarrow \text{MaxIndex}(x)$;
for i **from** 1 **to** $k-1$ **do** {
  $p \longleftarrow \text{Pois}\left(\frac{1}{i}A(x^i)\right)$;
  repeat p {
    $\gamma \leftarrow \Gamma A(x^i)$;
    Add $\boldsymbol{i}$ copies of γ to the MSet
  }
}
$p \longleftarrow \text{Pois}_{\geq 1}\left(\frac{1}{k}A(x^k)\right)$;
repeat p {
  $\gamma \leftarrow \Gamma A(x^k)$;
  Add $\boldsymbol{k}$ copies of γ to the MSet
}

Algorithm for general MSet (2)

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

$k \longleftarrow \text{MaxIndex}(x)$;
for i **from** 1 **to** $k-1$ **do** {
  $p \longleftarrow \text{Pois}\left(\frac{1}{i}A(x^i)\right)$;
  repeat p {
    $\gamma \leftarrow \Gamma A(x^i)$;
    Add $\boldsymbol{i}$ copies of γ to the MSet
  }
}
$p \longleftarrow \text{Pois}_{\geq 1}\left(\frac{1}{k}A(x^k)\right)$;
repeat p {
  $\gamma \leftarrow \Gamma A(x^k)$;
  Add $\boldsymbol{k}$ copies of γ to the MSet
}

Algorithm for general MSet (2)

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

$k \longleftarrow \text{MaxIndex}(x)$;
for i **from** 1 **to** $k-1$ **do** {
  $p \longleftarrow \text{Pois}\left(\frac{1}{i}A(x^i)\right)$;
  repeat p {
    $\gamma \leftarrow \Gamma A(x^i)$;
    Add $\boldsymbol{i}$ copies of γ to the MSet
  }
}
$p \longleftarrow \text{Pois}_{\geq 1}\left(\frac{1}{k}A(x^k)\right)$;
repeat p {
  $\gamma \leftarrow \Gamma A(x^k)$;
  Add $\boldsymbol{k}$ copies of γ to the MSet
}

Algorithm for general MSet (2)

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

$k \longleftarrow \text{MaxIndex}(x)$;
for i **from** 1 **to** $k-1$ **do** {
  $p \longleftarrow \text{Pois}\left(\frac{1}{i}A(x^i)\right)$;
  repeat p {
    $\gamma \leftarrow \Gamma A(x^i)$;
    Add ***i*** copies of γ to the MSet
  }
}
$p \longleftarrow \text{Pois}_{\geq 1}\left(\frac{1}{k}A(x^k)\right)$;
repeat p {
  $\gamma \leftarrow \Gamma A(x^k)$;
  Add ***k*** copies of γ to the MSet
}

Algorithm for general MSet (2)

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

$k \longleftarrow \mathrm{MaxIndex}(x)$;
for i **from** 1 **to** $k-1$ **do** {
    $p \longleftarrow \mathrm{Pois}\left(\frac{1}{i}A(x^i)\right)$;
    repeat p {
        $\gamma \leftarrow \Gamma A(x^i)$;
        Add $\boldsymbol{i}$ copies of γ to the MSet
    }
}
$p \longleftarrow \mathrm{Pois}_{\geq 1}\left(\frac{1}{k}A(x^k)\right)$;
repeat p {
    $\gamma \leftarrow \Gamma A(x^k)$;
    Add $\boldsymbol{k}$ copies of γ to the MSet
}

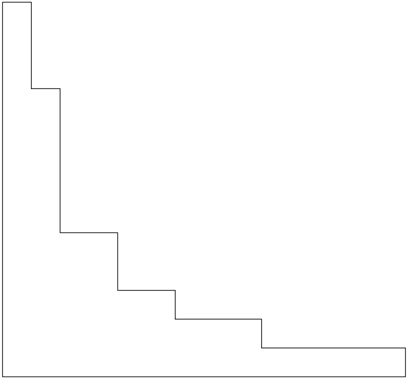
Result

Theorem

If we have a Boltzmann sampler $\Gamma A(x)$ *for* $\mathcal{A}$, *then* $\Gamma MSet[\mathcal{A}](x)$ *is a Boltzmann sampler for* $\textsc{MSet}(\mathcal{A})$.

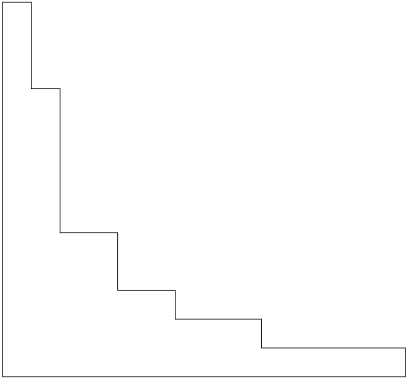
Generation of a partition (multiset of integers)

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



Generation of a partition (multiset of integers)

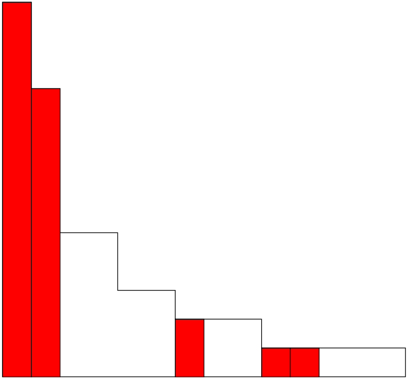
$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$



Generation of a partition (multiset of integers)

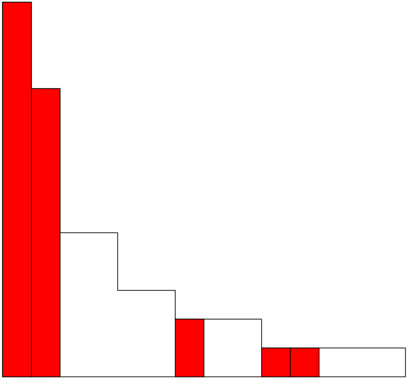
$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$

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



Generation of a partition (multiset of integers)

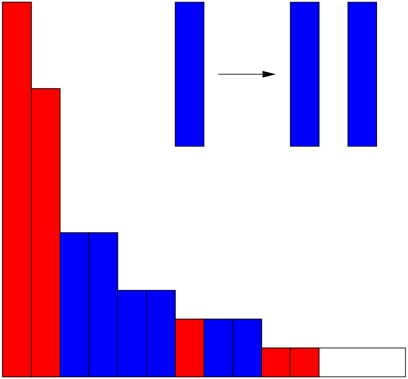
$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$



Generation of a partition (multiset of integers)

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

$\rightarrow 5,\ 3,\ 2$



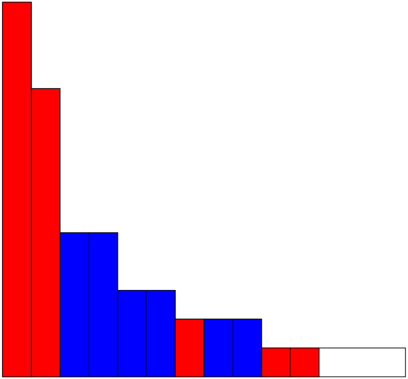
Generation of a partition (multiset of integers)

$\boldsymbol{k} = 3;$

$p_1 = 5;$

$p_2 = 3;$

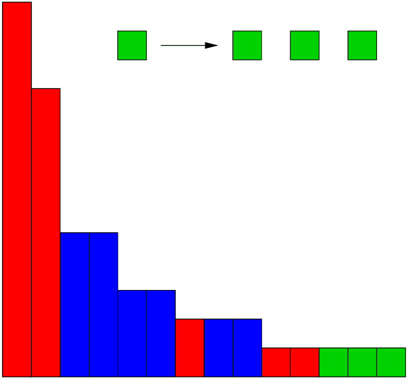
$p_3 = 1;$



Generation of a partition (multiset of integers)

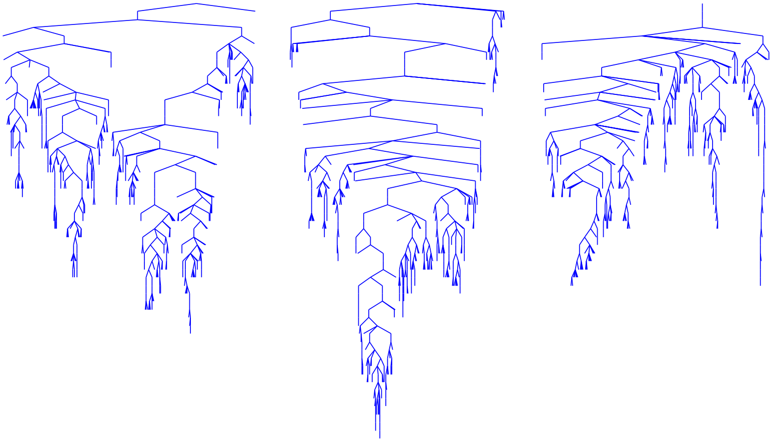
$\boldsymbol{k} = 3;$
$p_1 = 5;$
$p_2 = 3;$
$p_3 = 1;$

$\rightarrow 1$



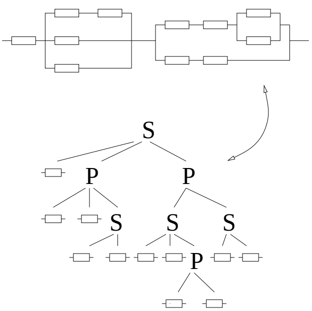
General unembedded trees

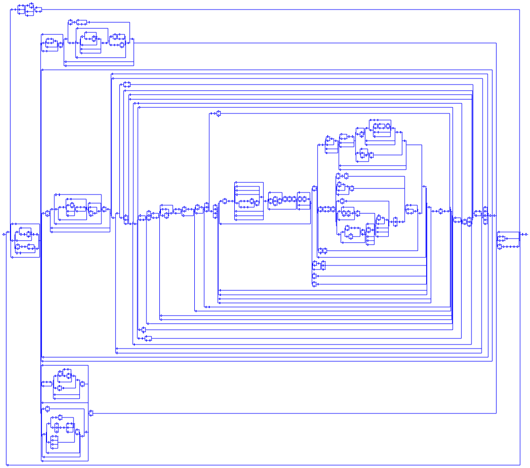
$$\mathcal{T} = \mathcal{Z} \times \mathrm{MS{\small ET}}(\mathcal{T})$$



Series-parallel circuits

$$\mathcal{C} = \mathcal{P} + \mathcal{S} + \mathcal{Z}$$
$$\mathcal{S} = \mathrm{SEQ}_{\geq 2}(\mathcal{P} + \mathcal{Z})$$
$$\mathcal{P} = \mathrm{MSET}_{\geq 2}(\mathcal{S} + \mathcal{Z})$$

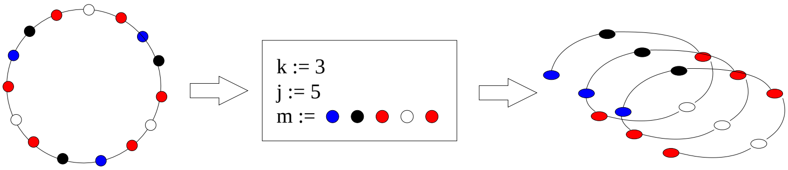




1. Recursive counting and recursive sampling
2. Generating functions and Boltzmann samplers
3. New constructions for Boltzmann samplers
   - Multisets
   - Cycles
4. Random sampling of plane partitions

Generating function for 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

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

$k \longleftarrow \text{ReplicationOrder}(x);$ # how many copies of s

$j \longleftarrow \text{Loga}\left(A(x^k)\right);$ length of s

$s \longleftarrow \underbrace{\Gamma A(x^k), \ldots, \Gamma A(x^k)}_{j\ times};$ sequence

Return $(\underbrace{m, \ldots, m}_{k\ times});$

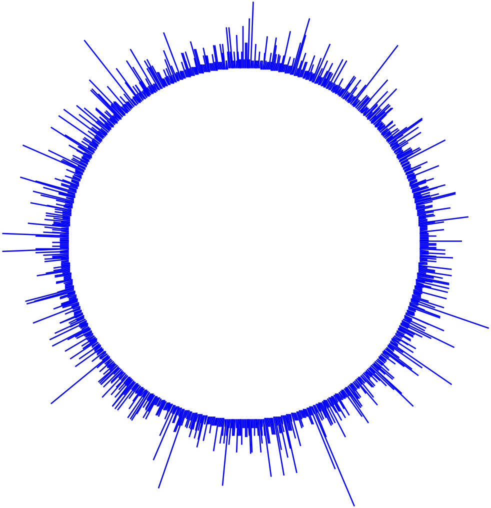
Definition: X follows a Loga law of parameter λ if:

$$\mathbb{P}(X = j) = \frac{1}{\log(1-\lambda)^{-1}} \frac{\lambda^j}{j}$$

Cycles of integers

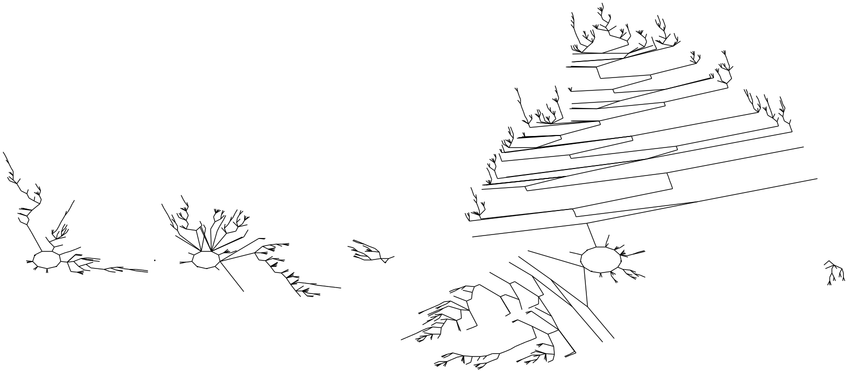
$$\mathcal{C} = \textsc{Cyc}(\mathcal{Z} \times \textsc{Seq}(\mathcal{Z}))$$

$$C(z) = \sum_{k=1}^{\infty} \frac{\varphi(k)}{k} \log \frac{1}{1 - \frac{z^k}{1-z^k}}$$



Functional graphs

$$\begin{array}{rcl} \mathcal{M} & = & \text{MSet}(\mathcal{C}) \\ \mathcal{C} & = & \text{Cyc}(\mathcal{G}) \\ \mathcal{G} & = & \mathcal{Z} \times \text{MSet}(\mathcal{G}) \end{array}$$



Result

Theorem

- *If a combinatorial class $\mathcal{C}$ has a recursive specification with the constructions*

$$\{\cup, \times, \text{Seq}, \text{MSet}, \text{Cyc}\}$$

then a **Boltzmann sampler** *can be designed for $\mathcal{C}$.*

- *This list of constructions can be completed by constructions with cardinality restrictions (e.g.* MSet_2*)*

This applies for several classes:

- partitions, compositions, necklaces,...
- trees,
- regular languages,
- functional graphs,...

Result

Theorem

- *If a combinatorial class $\mathcal{C}$ has a recursive specification with the constructions*

$$\{\cup, \times, \textsc{Seq}, \textsc{MSet}, \textsc{Cyc}\}$$

then a **Boltzmann sampler** *can be designed for $\mathcal{C}$.*

- *This list of constructions can be completed by constructions with cardinality restrictions (e.g.* $\textsc{MSet}_2$*)*

This applies for several classes:

- partitions, compositions, necklaces,...
- trees,
- regular languages,
- functional graphs,...

Complexity

Theorem

By choosing a suitable value for the parameter x, one has the following complexities for approximate size and fixed size:

Integer partitions of size n	$O(\sqrt{n}\log n)$	$O(n)$
Unembedded trees of size n	$O(n)$	$O(n^2)$
Necklaces, circular compositions of size n	$O(n)$	$O(n)$
Mobiles of size n	$O(n)$	$O(n^2)$
Functional graphs of size n	$O(n)$	$O(n\sqrt{n})$

Random sampling of plane partitions

(with Olivier Bodini and Carine Pivoteau)