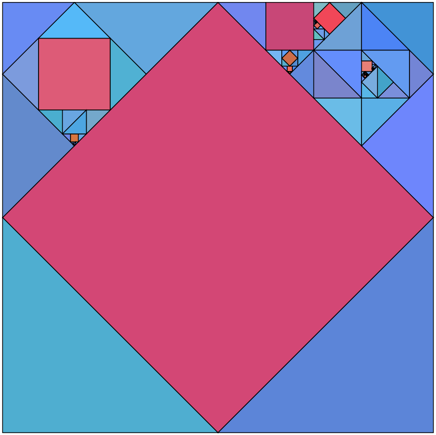
Boltzmann sampling in linear time: a promise from two years ago

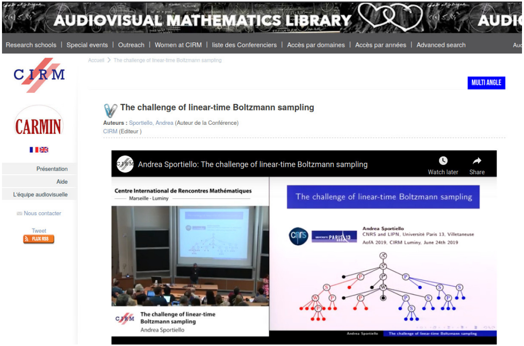
Andrea Sportiello
CNRS and LIPN
Université Sorbonne Paris Nord
Villetaneuse, France

AofA 2021
Alpen-Adria-Universität Klagenfurt
14-17 June 2021

based on
https://arxiv.org/abs/2105.12881



Two years have past; two summers, with the length / Of two long winters!



me at AofA 2019!

I was giving a talk with title **The challenge of linear-time Boltzmann sampling**

Some results, some hopes. . .

Here, I'll tell you about one hope that has been fulfilled: **Boltzmann sampling of irreducible context-free structures in linear time**

video: `https://library.cirm-math.fr/Record.htm?idlist=2&record=19286312124910045949`

slides: `https://www.cirm-math.fr/RepOrga/1940/Slides/Sportiello.pdf`

Irreducible context-free structures

A context-free structure is a class $\mathcal{Y} = \bigcup_n \mathcal{Y}_n$ of configurations whose combinatorial specification leads to a system of m equations the gen. function $Y^{(1)}(z) = \sum_n z^n |\mathcal{Y}_n|$ being the first component

$$\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z)) \,.$$

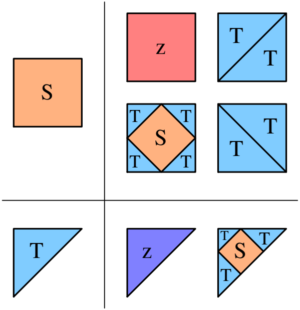
If the system is irreducible in a certain sense, the Drmota–Lalley–Woods Theorem applies, and $|\mathcal{Y}_n| \sim K \rho^{-n} n^{-\frac{3}{2}}$.
Also, Perron–Frobenius Theory applies to the matrix

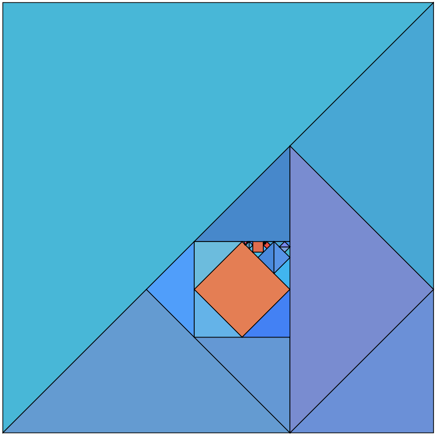
$$K = \{K_{\alpha\beta}\}_{1 \leq \alpha, \beta \leq m} = \frac{\partial}{\partial Y^{(\beta)}} \Phi^{(\alpha)}(z, \vec{Y})$$

The simplest situation is $m = 1$ and $\Phi(z, y) = z\,\phi(y)$.
This corresponds to simply-generated (rooted planar) trees, where each node counts as an unit, and of Łukasiewicz excursions, that is lattice paths in the upper-half plane with steps of the form $(+1, h)$ for $h \geq -1$ (the famous bijection is based on the depth-first search countour of the tree)

An example: subdiving a square into squares and triangles

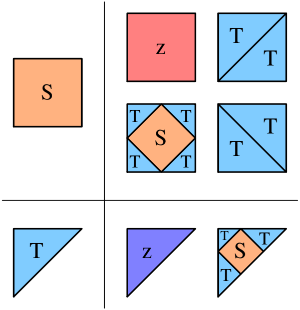
$$\begin{cases} S &= z + 2T^2 + ST^4 \\ T &= z + ST^3 \end{cases}$$

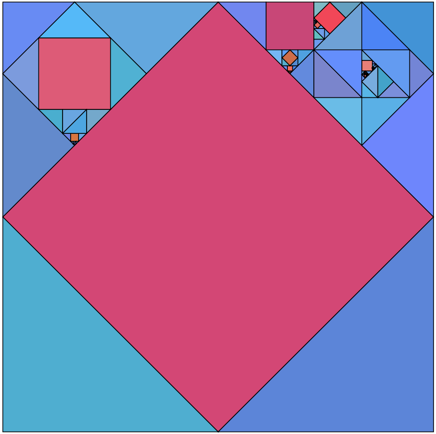




An example: subdiving a square into squares and triangles

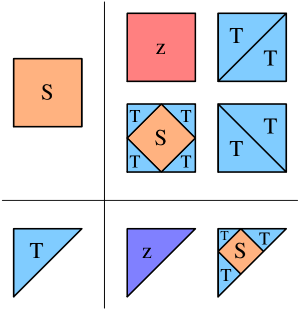
$$\begin{cases} S &= z + 2T^2 + ST^4 \\ T &= z + ST^3 \end{cases}$$

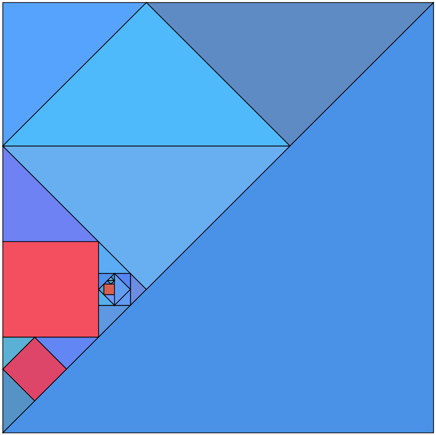




An example: subdiving a square into squares and triangles

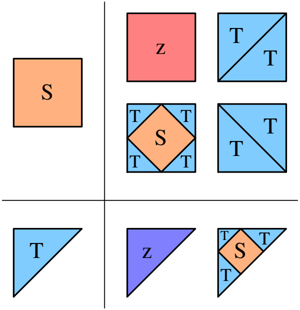
$$\begin{cases} S = z + 2T^2 + ST^4 \\ T = z + ST^3 \end{cases}$$

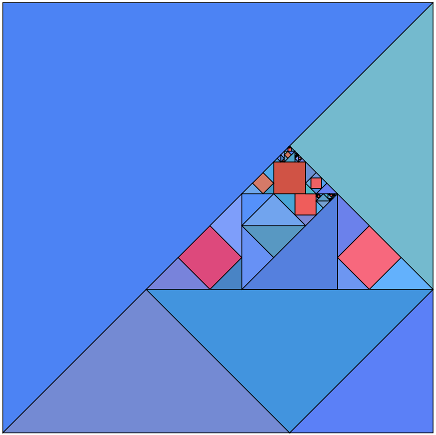




An example: subdiving a square into squares and triangles

$$\begin{cases} S = z + 2T^2 + ST^4 \\ T = z + ST^3 \end{cases}$$





Boltzmann sampling for combinatorial structures

Given a family of measures $\mu_n(X)$ on $\mathcal{Y}_n$, exact sampling is the problem of devising an efficient algorithm for sampling configurations $X \in Y_n$, with measure μ_n.

In the Boltzmann sampling paradigm, the combinatorial specification is turned into an algorithm for sampling from the 'Boltzmann' measure $\mu_{[z]}(X) = z^{|X|}\mu_n(X)/Y(z)$ and you are temped to use the obvious algorithm

repeat
  $X \leftarrow \mu_{[z]}$
until $|X| = n$;
return X

Boltzmann method

Duchon, Flajolet, Louchard and Schaeffer,
Boltzmann Samplers for the Random Generation of Combinatorial Structures

Boltzmann sampling for bridges

Some structures can be put in bijection with lattice bridges,
that is directed walks in $\mathbb{Z}^2$, from $(0,0)$ to $P_n = (n,0)$ (or to $(n,-1)$)
Now $X = (x_1, \dots, x_k)$, and $\mu_n(X) = \left(\prod_j p(x_j)\right) \mathbb{1}(\sum_j x_j = P_n)$
In this case, the Boltzmann idea is to change $p(x)$ into
$p_{[z]}(x) \propto z^{x_2} p(x)$, with z tuned as to have average zero drift
and you are temped to use the obvious algorithm

---

repeat
  $p = (0,0)$;
  repeat
    $x_j \leftarrow p_{[z]}$;
    $p = p + x_j$;
  until $p_1 \geq n$;
until $p = P_n$;
return $(x_1, x_2, \dots)$

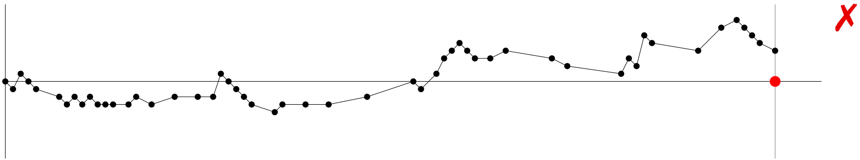
Boltzmann method for bridges

Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1 - x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

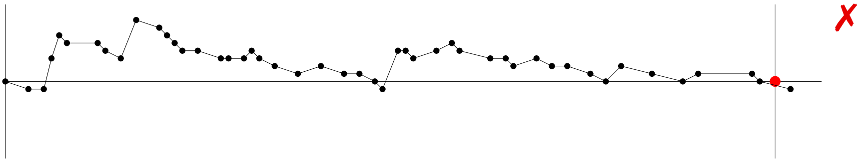


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1-x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

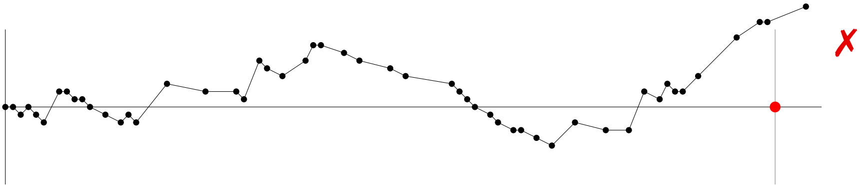


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1-x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

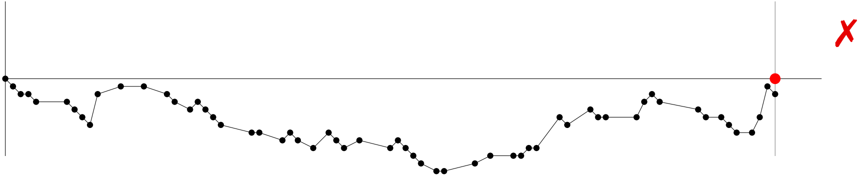


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1 - x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

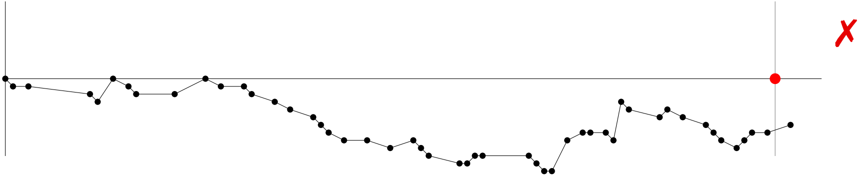


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1 - x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

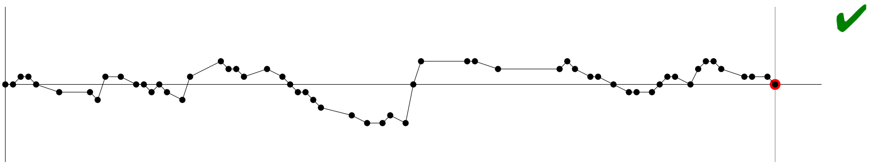


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1 - x_2} \mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)

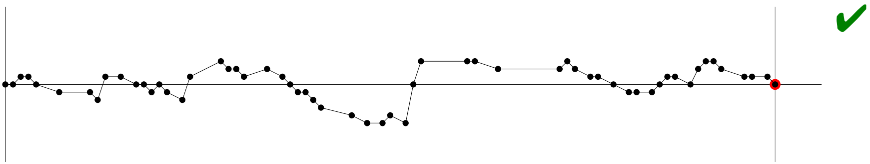


Complexity of the Boltzmann Method

The typical complexity of the Boltzmann Method,
for structures in the smooth inverse-function schema, is $T(n) \sim n^2$

If we are in the Bridge case, the analysis is simpler
and the complexity is smaller, $T(n) \sim n^{\frac{3}{2}}$
Indeed, a single run takes time $\sim n$,
but the probability of reaching P_n is only $\sim 1/\sqrt{n}$.

(example with $p_{[z]}(x_1, x_2) = 2^{-x_1-x_2}\mathbb{1}(x_1 \geq 1, x_2 \geq -1)$)



We want a new idea for ‘accelerating’ the Boltzmann Method, and reach linear complexity

BBHL algorithm: ‘the mother of all linear algorithms’

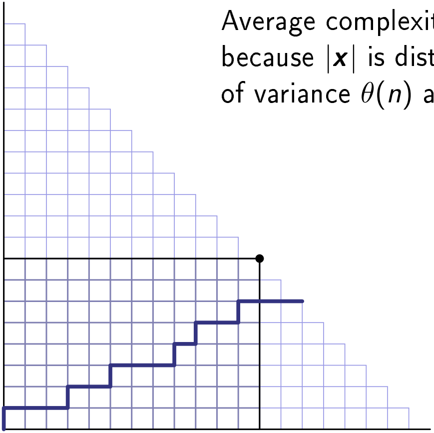
Can we really reach linearity in sampling bridges?
Yes! The BBHL's BalancedShuffle does it in a simple case

Bacher, Bodini, Hollender and Lumbroso,
MergeShuffle: A Very Fast, Parallel Random Permutation Algorithm

The problem: exact sampling of strings in $\{\bullet, \circ\}^n$ with $\#\{\bullet\} = k$
BBHL solves it in linear time and optimal random-bit complexity

BBHL algorithm: 'the mother of all linear algorithms'

First naïve idea: the Boltzmann Method in the bridge case.
Sample n variables $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, i.i.d. with Bern_p, (with $p = k/n$). Restart if $|\boldsymbol{x}| \neq k$.



Average complexity: $\sim n^{\frac{3}{2}}$,
because $|\boldsymbol{x}|$ is distributed roughly as a Gaussian of variance $\theta(n)$ and mean k.

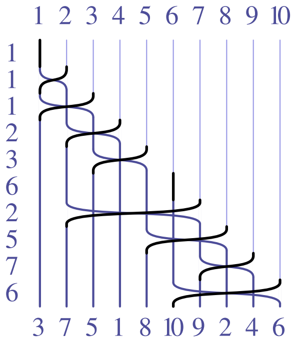
Second naïve idea: project down from Fisher–Yates

The Fisher–Yates algorithm samples a random permutation $\sigma \in \mathfrak{S}_n$ with optimal random-bit complexity: $T_{\mathrm{rand}}(n) \simeq \ln n! \simeq n(\ln n - 1)$

It works by sampling $\boldsymbol{y} \in \{1\} \times \{1,2\} \times \{1,2,3\} \times \cdots \times \{1,\ldots,n\}$, and doing as follows:

Then, 'projecting down' means $x_i = 1$ iff $\sigma^{-1}(i) \leq k$

Average complexity: $\sim n \ln n$, because, even if Fisher–Yates is optimal, the projection throws away most of the information

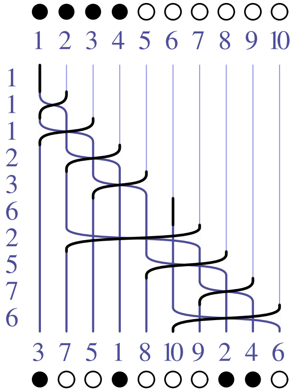


BBHL algorithm: 'the mother of all linear algorithms'

Second naïve idea: project down from Fisher–Yates

The Fisher–Yates algorithm samples a random permutation $\sigma \in \mathfrak{S}_n$ with optimal random-bit complexity: $T_{\mathrm{rand}}(n) \simeq \ln n! \simeq n(\ln n - 1)$

It works by sampling $\boldsymbol{y} \in \{1\} \times \{1,2\} \times \{1,2,3\} \times \cdots \times \{1,\ldots,n\}$, and doing as follows:

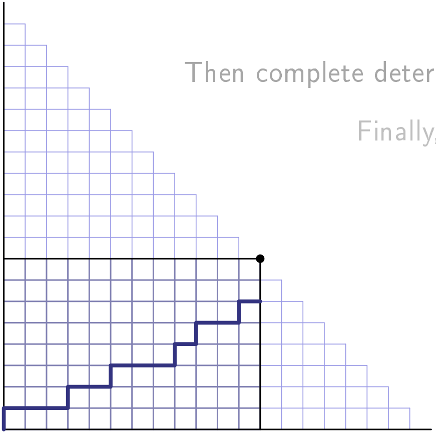


Then, 'projecting down' means $x_i = 1$ iff $\sigma^{-1}(i) \leq k$

Average complexity: $\sim n \ln n$, because, even if Fisher–Yates is optimal, the projection throws away most of the information

BBHL algorithm: 'the mother of all linear algorithms'

The good idea: Sample the n variables $\boldsymbol{x} = (x_1, \dots, x_n) \in \{0,1\}^n$, i.i.d. with Bern_p, one by one up to when you have k entries $x_i = 1$, or $n - k$ entries $x_i = 0$.



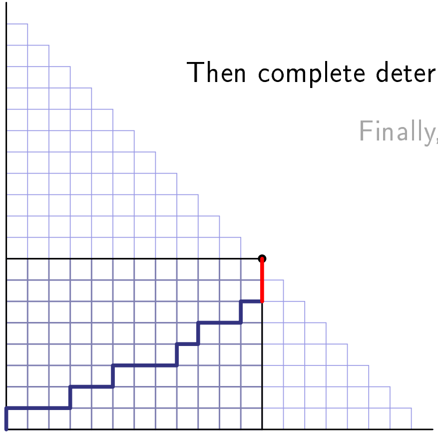
Then complete deterministically with what is needed,

Finally, perform Fisher–Yates shufflings on these last added steps.

Average complexity:
$T_{\mathrm{rand}}(n) = S[\mu] + \mathcal{O}(\sqrt{n} \ln n)$
because the final shuffles are a.s. just $\Theta(\sqrt{n})$

BBHL algorithm: 'the mother of all linear algorithms'

The good idea: Sample the n variables $\boldsymbol{x} = (x_1, \dots, x_n) \in \{0,1\}^n$, i.i.d. with Bern_p, one by one up to when you have k entries $x_i = 1$, or $n-k$ entries $x_i = 0$.



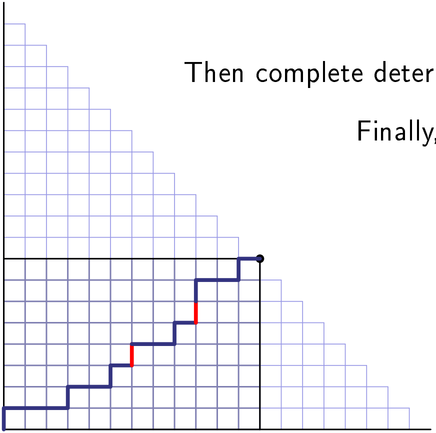
Then complete deterministically with what is needed,

Finally, perform Fisher–Yates shufflings on these last added steps.

Average complexity:
$T_{\mathrm{rand}}(n) = S[\mu] + \mathcal{O}(\sqrt{n} \ln n)$
because the final shuffles are a.s. just $\Theta(\sqrt{n})$

BBHL algorithm: 'the mother of all linear algorithms'

The good idea: Sample the n variables $\boldsymbol{x} = (x_1, \dots, x_n) \in \{0, 1\}^n$, i.i.d. with Bern_p, one by one up to when you have k entries $x_i = 1$, or $n - k$ entries $x_i = 0$.



Then complete deterministically with what is needed,

Finally, perform Fisher–Yates shufflings on these last added steps.

Average complexity:
$T_{\mathrm{rand}}(n) = S[\mu] + \mathcal{O}(\sqrt{n} \ln n)$
because the final shuffles are a.s. just $\Theta(\sqrt{n})$

Devroye Algorithm for simply-generated trees

In the case of simply-generated trees
(that can be related to bridges to $P_n = (n, -1)$, with steps $(+1, h)$)
an algorithm of Devroye, once complemented by BBHL, is optimal

Devroye, *Simulating Size-constrained Galton-Watson Trees*

idea: First sample how many steps of each type you have in total, according to a multinomial distribution, then put them in some canonical order, finally perform iteratively random BBHL shuffles

Pros:

✔ optimal random-bit complexity

Cons:

✗ use of float approximations for multinomial coefficients,

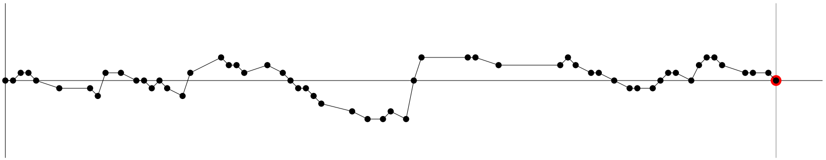
✗ need for extra tricks if the steps do not have finite support

✗ **cannot be used for higher-dimensional systems**, as the steps are not exchangeable random variables

The idea of our algorithm: a bridge example

How our algorithm works, in the example of bridge before:

- sample steps in $p_{\neq}(x) \propto p_{[z]}(x)\mathbb{1}(x \neq (1,-1))$,[†] up to reach the 'landing diagonal' D_n, at position $(n-m, m)$ (if you jump over, restart);
- introduce the acceptance rate $r_n(m)$ (if failed, restart);
- complete the path to $(n-1)$ with m steps $(1,-1)$;
- perform a BBHL shuffle of the steps, with parameters (n, m).

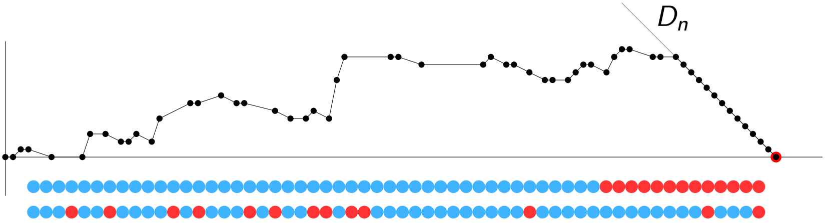


[†] In fact, a small deformation of it, namely a combination of a slightly sub-critical and super-critical measures.

The idea of our algorithm: a bridge example

How our algorithm works, in the example of bridge before:

- sample steps in $p_{\neq}(x) \propto p_{[z]}(x)\mathbb{1}(x \neq (1,-1))$,[†] up to reach the 'landing diagonal' D_n, at position $(n-m, m)$ (if you jump over, restart);
- introduce the acceptance rate $r_n(m)$ (if failed, restart);
- complete the path to $(n-1)$ with m steps $(1,-1)$;
- perform a BBHL shuffle of the steps, with parameters (n, m).



[†] In fact, a small deformation of it, namely a combination of a slightly sub-critical and super-critical measures.

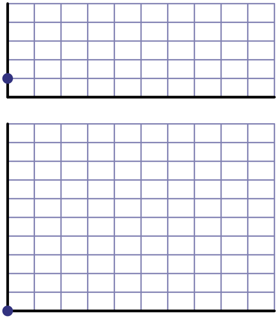
Context-free structures are coloured random trees

The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

A

stack size: 1 obj. size: 0

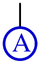


Context-free structures are coloured random trees

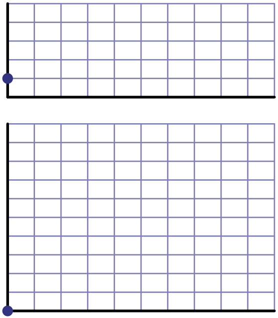
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

A



stack size: 1 obj. size: 0

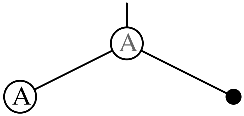


Context-free structures are coloured random trees

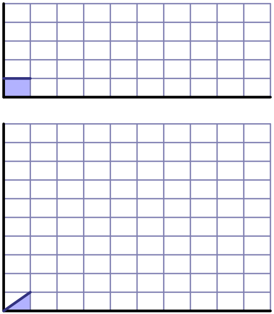
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

Az



stack size: 1 obj. size: 1

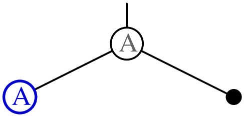


Context-free structures are coloured random trees

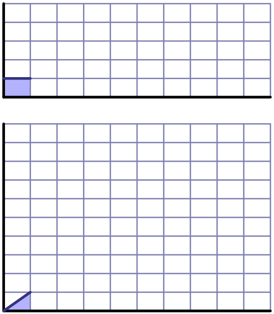
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

Az



stack size: 1 obj. size: 1



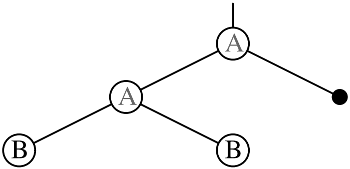
Context-free structures are coloured random trees

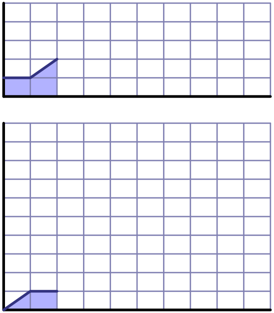
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

BBz

stack size: 2 obj. size: 1





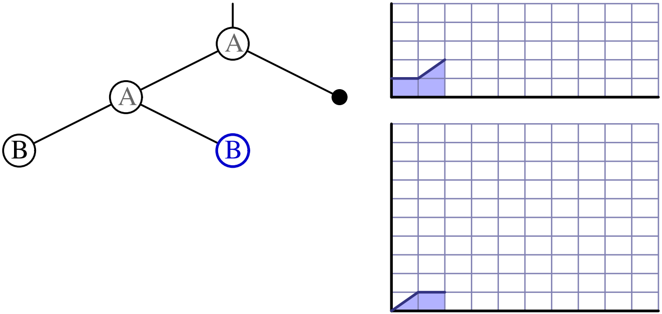
Context-free structures are coloured random trees

The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

BBz

stack size: 2 obj. size: 1

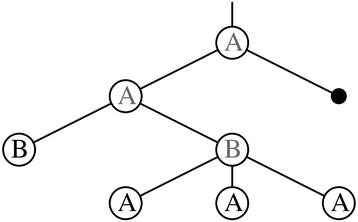


Context-free structures are coloured random trees

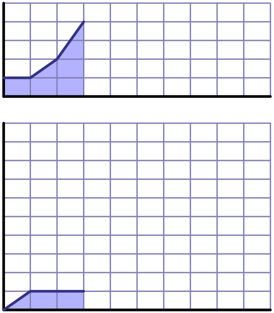
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = A\,z + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BAAAz$



stack size: 4 obj. size: 1



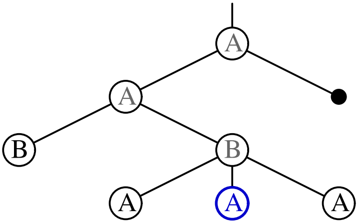
Context-free structures are coloured random trees

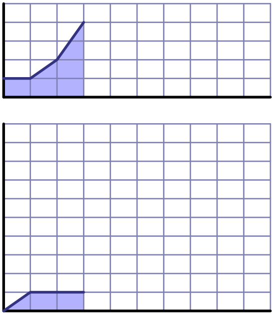
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BAAAz$

stack size: 4 obj. size: 1





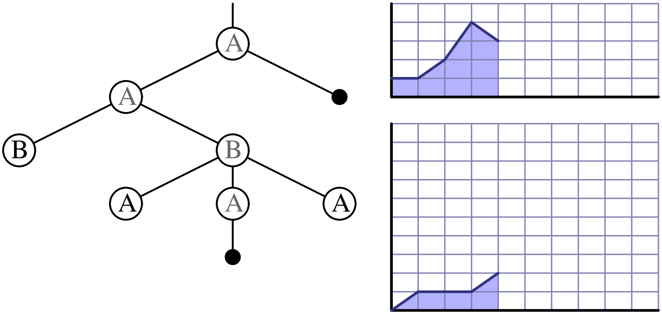
Context-free structures are coloured random trees

The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BAzAz$

stack size: 3 obj. size: 2



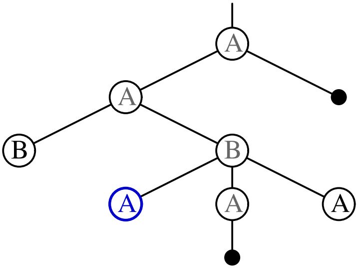
Context-free structures are coloured random trees

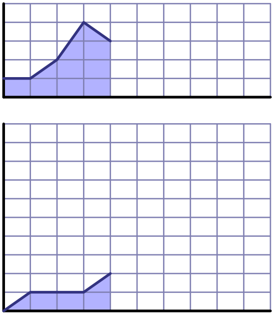
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BAzAz$

stack size: 3 obj. size: 2





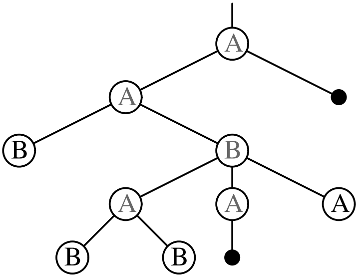
Context-free structures are coloured random trees

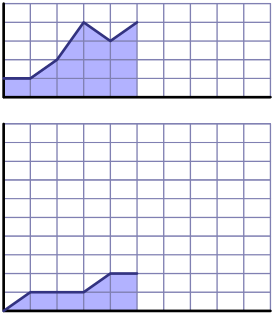
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = A\,z + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

BBBzAz

stack size: 4 obj. size: 2



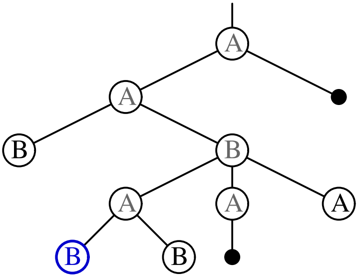


Context-free structures are coloured random trees

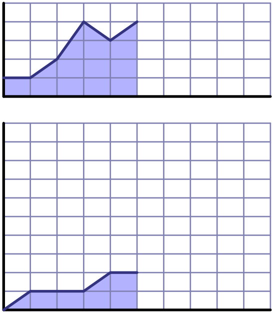
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BBBzAz$



stack size: 4 obj. size: 2

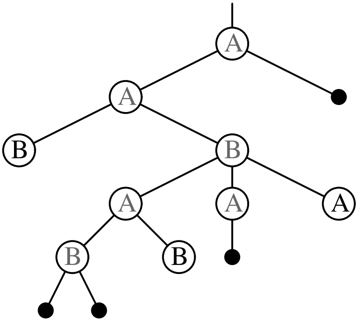


Context-free structures are coloured random trees

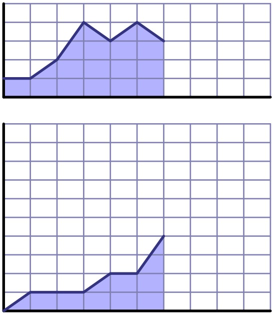
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

BzzBzAz



stack size: 3 obj. size: 4

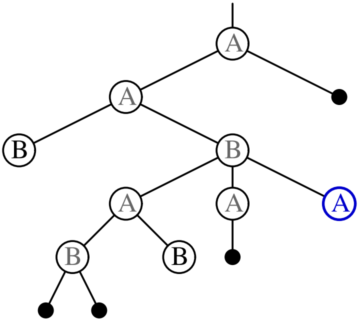


Context-free structures are coloured random trees

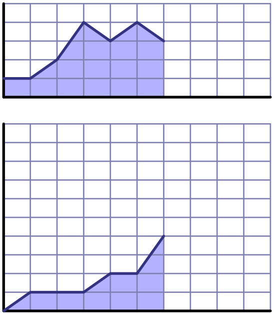
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = A\,z + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BzzBzAz$



stack size: 3 obj. size: 4

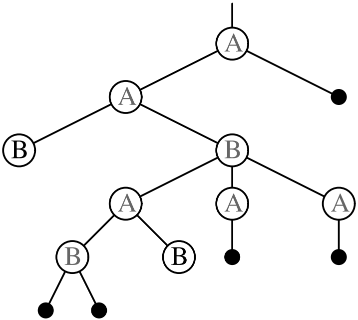


Context-free structures are coloured random trees

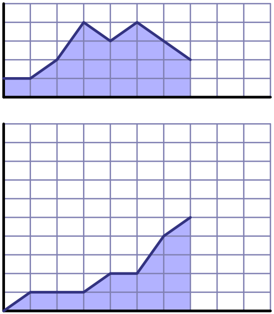
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

BzzBzzz



stack size: 2 obj. size: 5



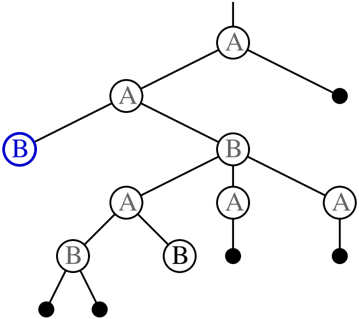
Context-free structures are coloured random trees

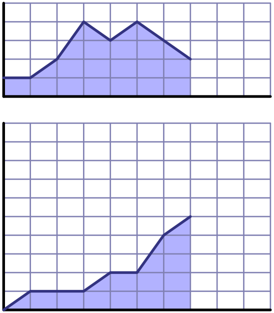
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = A\,z + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$BzzBzzz$

stack size: 2 obj. size: 5





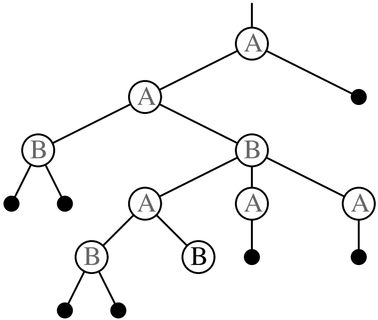
Context-free structures are coloured random trees

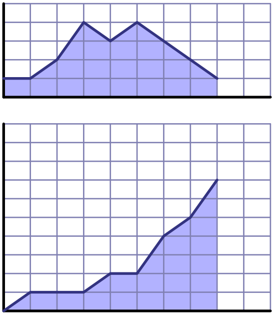
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

zzzzBzzz

stack size: 1 obj. size: 7





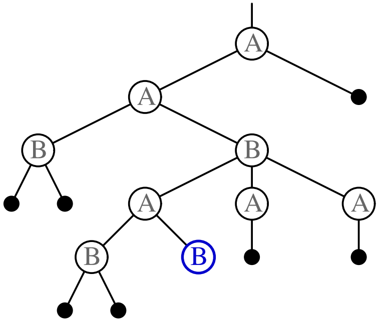
Context-free structures are coloured random trees

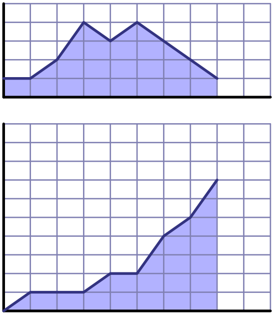
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = A\,z + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

$zzzz\,B\,zzz$

stack size: 1 obj. size: 7



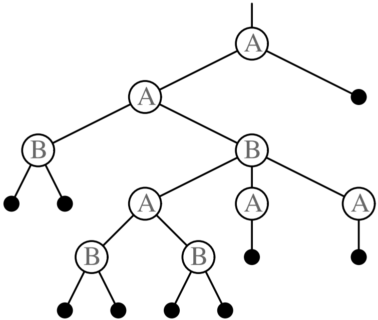


Context-free structures are coloured random trees

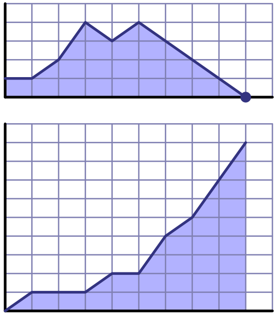
The combinatorial specification associated to a system $\vec{Y}(z) = \vec{\Phi}(z, \vec{Y}(z))$ translates into a Galton–Watson process, which, in turns, can be seen as a random rewriting system

Example: for $\begin{cases} A = Az + B^2 + z \\ B = A^3 + z^2 \end{cases}$ we could get

zzzzzzzzz



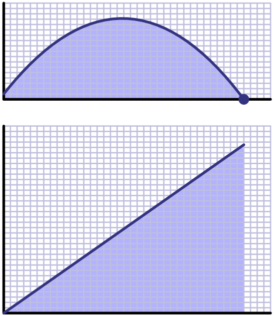
stack size: 0 obj. size: 9



From trees to bridges

In the limit, the stack size profile is an excursion
while the object size profile is a straight line

The Cyclic Lemma allows to relate the exact sampling
of excursions and of bridges

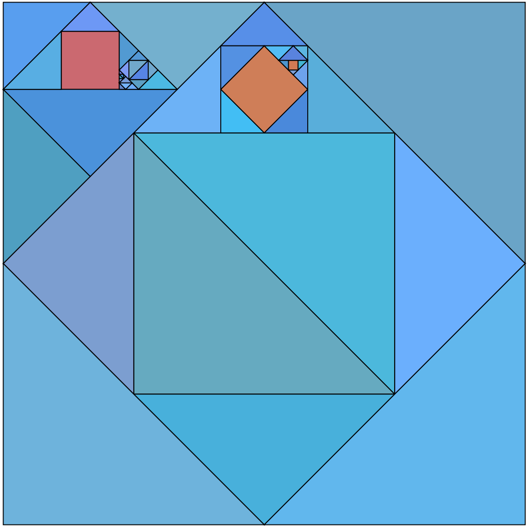


However, for a generic specification
we have coloured nodes, and the size
is the number of leaves, not of nodes.

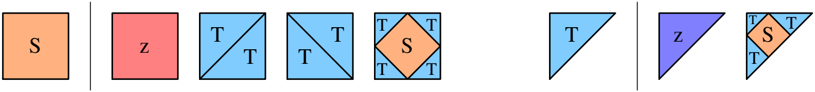
As a result, the bridges have a variable
number of steps, and non-local correlations

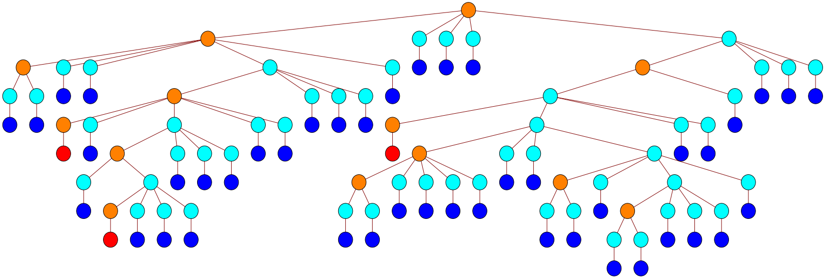
Neither Devroye nor BBHL (nor anything else)
apply as is, and we need some new idea. . .

The trees in our example



The trees in our example

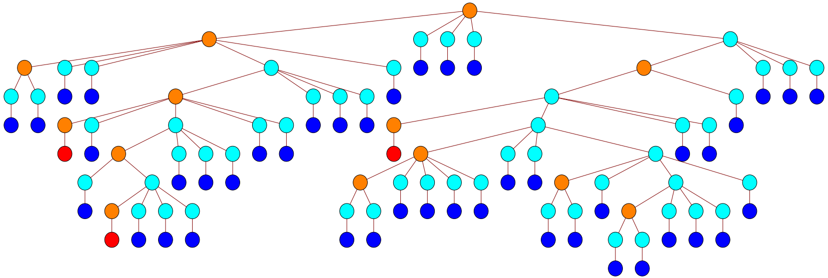




The size is the number of leaves: 3 (squares) + 44 (triangles)

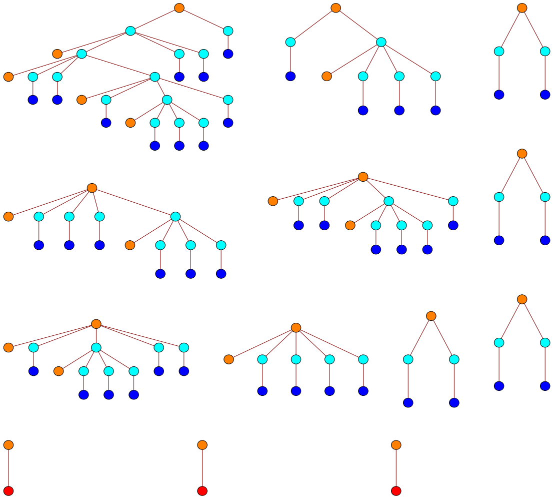
The bridges in our example

Break the tree into subtrees at all $Y^{(1)}$-nodes



The bridges in our example

Break the tree into subtrees at all $Y^{(1)}$-nodes



Our bridges in general

Breaking the bridges in this way leads to exchangeable steps x,
where x_1 is the number of z-leaves in the subtree,
and $x_2 + 1$ is the number of $Y^{(1)}$-leaves.
So, we just have to run our algorithm for bridges