# Uniform generation of infinite concurrent runs

## The case of trace monoids


Samy Abbes[1] & Vincent Jugé[2]

1: Université Paris Diderot (IRIF) — 2: Université Paris-Est Marne-la-Vallée (LIGM)


13/03/2018

# Contents

# Heaps of pieces and trace monoids

## Heap of pieces

- Pieces:


## Trace monoid

- Alphabet:

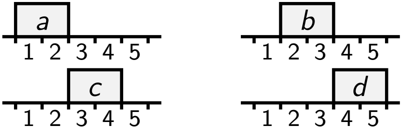
$$\Sigma = \{a, b, c, d\}$$

# Heaps of pieces and trace monoids

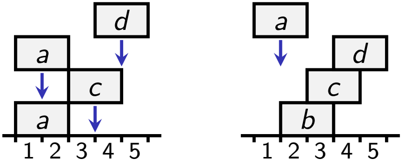
## Heap of pieces

- Pieces:


- Horizontal layout:




- Vertical heaps:




## Trace monoid

- Alphabet:

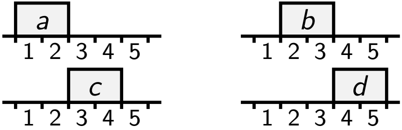
$$\Sigma = \{a, b, c, d\}$$

# Heaps of pieces and trace monoids

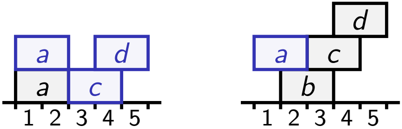
## Heap of pieces

- Pieces:


- Horizontal layout:




- Vertical heaps:




## Trace monoid

- Alphabet:

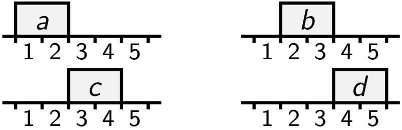
$$\Sigma = \{a, b, c, d\}$$

# Heaps of pieces and trace monoids

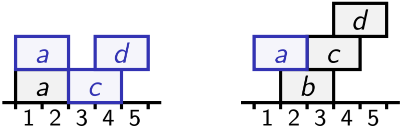
## Heap of pieces

- Pieces:


- Horizontal layout:




- Vertical heaps:




## Trace monoid

- Alphabet:

$$\Sigma = \{a, b, c, d\}$$

- Dependence relation:

$$\mathbf{D}=\{\{a,b\},\{b,c\},\{c,d\}\}$$

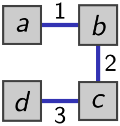
- Trace monoid:

$$\mathcal{M}=\langle a,b,c,d \,|\, ac=ca, ad=da, bd=db\rangle^+$$

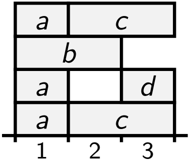

Dependency graph

# Heaps of pieces and dependency graph

## Dependency graph

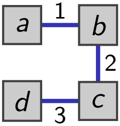


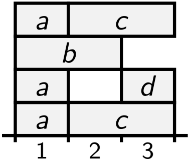

## Heap of pieces

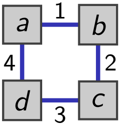


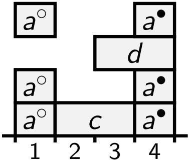
# Heaps of pieces and dependency graph

**Dependency graph** | **Heap of** (disconnected) **pieces**



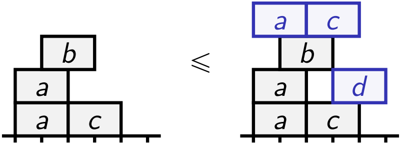



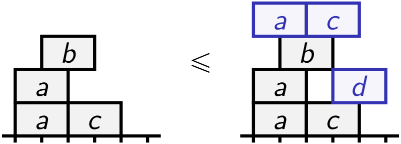
# Heaps of pieces and left divisibility

## Heap of pieces



# Heaps of pieces and left divisibility

## Heap of pieces

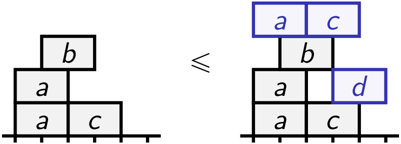



Can you pick one **infinite** heap **uniformly at random**?

1. Define your preferred notion of heap length
2. Study uniform distributions on traces of length $k$: what if $k \to \infty$?

# Heaps of pieces and left divisibility

**Heap of pieces**



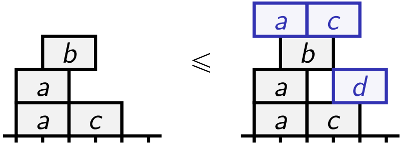

Can you pick one **infinite** heap **uniformly at random**?

1. Heap length = #pieces in the heap
2. **Weak convergence** of distributions:

$$\mu_k \to \nu \Leftrightarrow (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leqslant \xi] \to \mathbb{P}_\nu[x \leqslant \xi])$$

# Heaps of pieces and left divisibility

## Heap of pieces




Can you pick one **infinite** heap **uniformly at random**?

1. Heap length $=$ #pieces in the heap
2. **Weak convergence** of distributions:

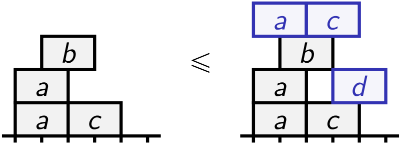
$$\mu_k \to \nu \Leftrightarrow (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leqslant \xi] \to \mathbb{P}_\nu[x \leqslant \xi])$$

**Theorem (Abbes & Mairesse 2015)**

If $\mu_k$ is the unifom measure on $\mathcal{M}_k = \{\zeta \in \mathcal{M} : |\zeta| = k\}$, $\nu$ **exists** and is the **critical Bernoulli** distribution of $\mathcal{M}$.

# Heaps of pieces and left divisibility

## Heap of pieces




Can you pick one **infinite** heap **uniformly at random**?

1. Heap length = #pieces in the heap
2. **Weak convergence** of distributions:

$$\mu_k \to \nu \Leftrightarrow (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leqslant \xi] \to \mathbb{P}_\nu[x \leqslant \xi])$$

**Theorem (Abbes & Mairesse 2015) — not constructive!**

If $\mu_k$ is the unifom measure on $\mathcal{M}_k = \{\zeta \in \mathcal{M} : |\zeta| = k\}$, $\nu$ **exists** and is the **critical Bernoulli** distribution of $\mathcal{M}$.

# Bernoulli distributions

## Definition

A probability measure $\mu$ on $\mathcal{M}$ is:

- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$

# Bernoulli distributions

## Definition

A probability measure $\mu$ on $\mathcal{M}$ is:

- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$

Which are the **possible values** of the parameter $p$?

# Bernoulli distributions

## Definition

A probability measure $\mu$ on $\mathcal{M}$ is:

- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$

Which are the **possible values** of the parameter $p$?

Uniform Bernoulli $\Rightarrow \mathbb{P}_\mu[x \leqslant \zeta] = p^{|x|} \Rightarrow \mathbb{P}_\mu[x = \zeta] \propto p^{|x|}$.

# Bernoulli distributions

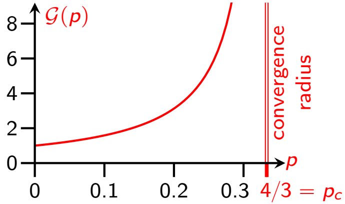
## Definition

A probability measure $\mu$ on $\mathcal{M}$ is:

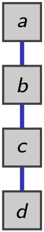
- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$

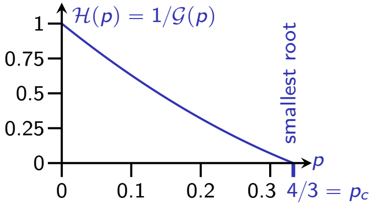
Which are the **possible values** of the parameter $p$?

Uniform Bernoulli $\Rightarrow \mathbb{P}_\mu[x \leqslant \zeta] = p^{|x|} \Rightarrow \mathbb{P}_\mu[x = \zeta] \propto p^{|x|}$.



Generating function $\mathcal{G}(p)$



Möbius polynomial $\mathcal{H}(p)$

# Bernoulli distributions

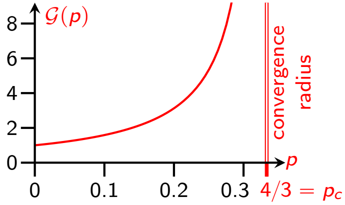
## Definition

A probability measure $\mu$ on $\mathcal{M}$ is:

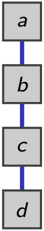
- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$
- **critical Bernoulli** if $p = p_c$ (requires **infinite** heaps)

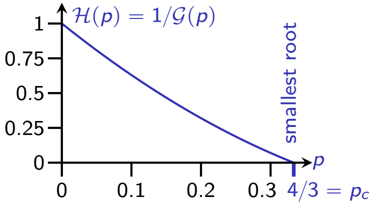
Which are the **possible values** of the parameter $p$? $0 \leqslant p \leqslant p_c$

$$\text{Uniform Bernoulli} \Rightarrow \mathbb{P}_\mu[x \leqslant \zeta] = p^{|x|} \Rightarrow \mathbb{P}_\mu[x = \zeta] \propto p^{|x|} \stackrel{?}{=} p^{|x|}\mathcal{H}(p).$$



Generating function $\mathcal{G}(p)$



Möbius polynomial $\mathcal{H}(p)$

# Bernoulli distributions

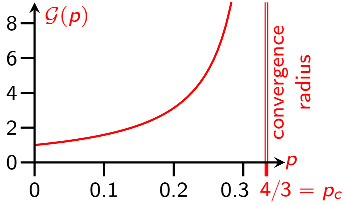
## Definition

A probability measure $\mu$ on $\overline{\mathcal{M}}$ is:

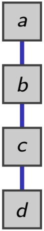
- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x\,\sigma \leqslant \zeta \mid x \leqslant \zeta] = \mathbb{P}_\mu[\sigma \leqslant \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leqslant \zeta] = p$
- **critical Bernoulli** if $p = p_c$ (requires **infinite** heaps)

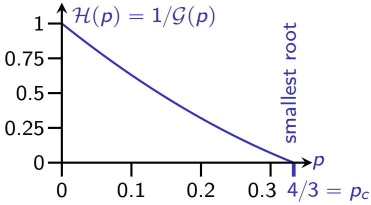
Which are the **possible values** of the parameter $p$? $0 \leqslant p \leqslant p_c$

$$\text{Uniform Bernoulli} \Rightarrow \mathbb{P}_\mu[x \leqslant \zeta] = p^{|x|} \Rightarrow \mathbb{P}_\mu[x = \zeta] \propto p^{|x|} = p^{|x|}\mathcal{H}(p).$$




Generating function $\mathcal{G}(p)$

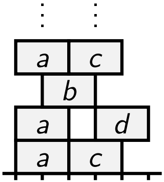



Möbius polynomial $\mathcal{H}(p)$

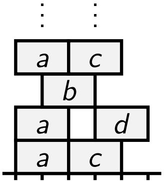
# Infinite heaps

## Heap of pieces

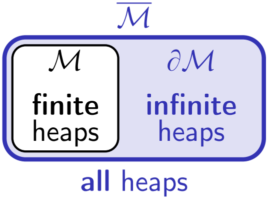


# Infinite heaps

**Heap of pieces**




**Sets of interest**

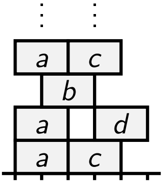



$\overline{\mathcal{M}}$ is the **compactification** of $\mathcal{M}$ for the topology induced by $\Uparrow \mathbf{x} = \{\zeta : \mathbf{x} \leqslant \zeta\}$

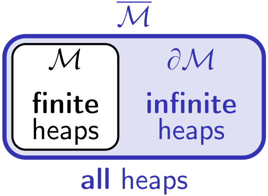
**Reminder**: $\mu_k \to \nu \Leftrightarrow (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leqslant \xi] \to \mathbb{P}_\nu[x \leqslant \xi])$

# Infinite heaps

**Heap of pieces**




**Sets of interest**




$\overline{\mathcal{M}}$ is the **compactification** of $\mathcal{M}$ for the topology induced by $\Uparrow \mathbf{x} = \{\zeta : \mathbf{x} \leqslant \zeta\}$

**Reminder**: $\mu_k \to \nu \Leftrightarrow (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leqslant \xi] \to \mathbb{P}_\nu[x \leqslant \xi])$

**Fact #1 (Abbes & Mairesse 2015)**

The limit $\nu$ is a distribution **on the set** $\overline{\mathcal{M}}$ with **support** $\partial\mathcal{M}$.⚠

# Contents

# Simulating the limit $\nu$

**Idea #1:** Pick $\zeta_k \sim \mu_k$, pick a piece **x** **wisely** and set $\zeta_{k+1} = \zeta_k \cdot x$

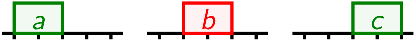
# Simulating the limit $\nu$

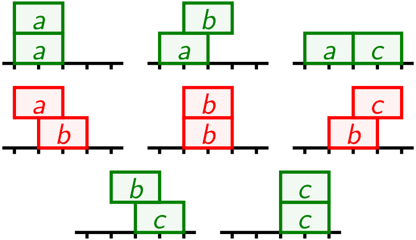
**Idea #1:** Pick $\zeta_k \sim \mu_k$, pick a piece **x** **wisely** and set $\zeta_{k+1} = \zeta_k \cdot x$
**Problem:** In general, $\zeta_{k+1}$ **cannot** be distributed according to $\mu_{k+1}$

**Example** in the monoid $\langle a, b, c \mid ac = ca \rangle^+$


Dependency graph

$\mu_1(\Uparrow a \cup \Uparrow c) = 2/3 \quad > \quad \mu_2(\Uparrow a \cup \Uparrow c) = 5/8$



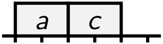



# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**

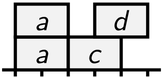
# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**



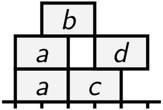
# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**



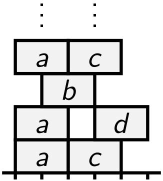
# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**



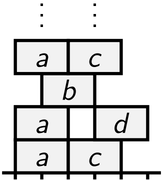
# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**



# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**

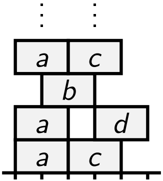



**Fact #2 (Abbes & Mairesse 2015)**

This approach works because $\nu$ is **Bernoulli**!

# Simulating the limit $\nu$

**Idea #2:** Simulate $\zeta \sim \nu$, **floor by floor**




**Fact #2 (Abbes & Mairesse 2015)**

This approach works because $\nu$ is **Bernoulli**!

**Problems:** Huge state space (exponential number of possible floors)
Adding one floor implies synchronising many pieces

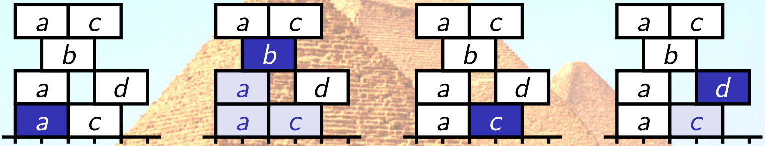
# Contents

# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **pyramids**

# Simulating the limit $\nu$ piece by piece⚠

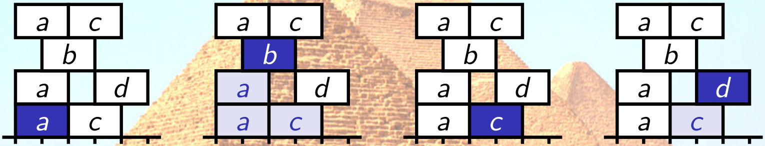
**Idea #3:** Decompose heaps recursively by using **pyramids**




$a$-pyramid $b$-pyramid $c$-pyramid $d$-pyramid

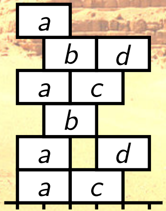
# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **pyramids**



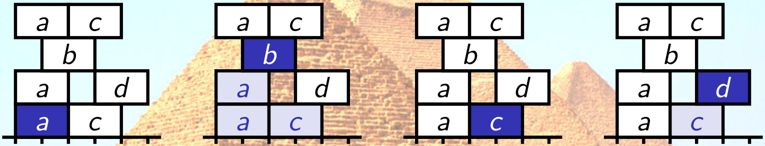

*a*-pyramid *b*-pyramid *c*-pyramid *d*-pyramid

**Example:** Recursive decomposition using *b*-pyramids



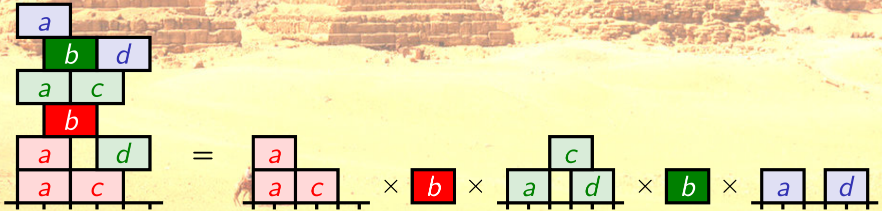
# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **pyramids**



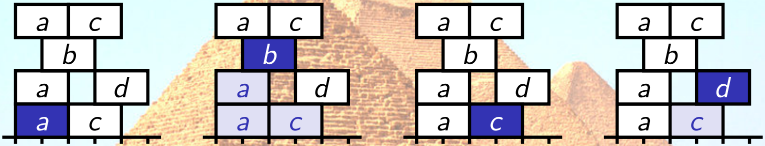

*a*-pyramid *b*-pyramid *c*-pyramid *d*-pyramid

**Example:** Recursive decomposition using *b*-pyramids



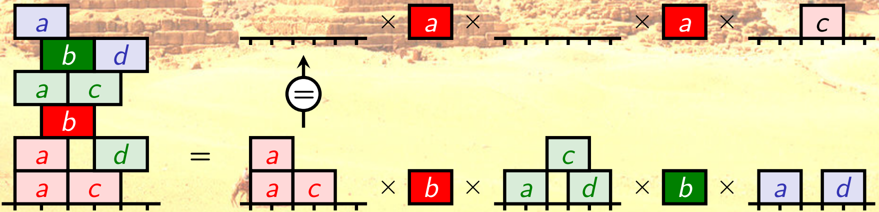
# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **pyramids**



$a$-pyramid $b$-pyramid $c$-pyramid $d$-pyramid

**Example:** Recursive decomposition using $b$-pyramids, then $a$-pyramids...



# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **independent pyramids**

**Theorem – continued (Abbes & J. 2017)**

If $\mathbf{a} \in \Sigma$ and $\nu$ is **Bernoulli** on $\overline{\mathcal{M}}(\Sigma)$, then

Heap

$=$

**a**-pyramid

$\times$

Heap

$+$

**a**-free heap

where right-hand side random variables are **independent**.

# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **independent pyramids**

**Theorem – continued (Abbes & J. 2017)**

If $\mathbf{a} \in \Sigma$ and $\nu$ is **Bernoulli** on $\overline{\mathcal{M}}(\Sigma)$, then


=

×

+


Heap = **a**-pyramid × Heap + **a**-free heap

where right-hand side random variables are **independent**.


=

× **A**

**a**-pyramid = **a**-free heap $\xi$ s.t. $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ × **A**

- $\mathbf{S}(\xi) = \{\sigma \in \Sigma : \xi \in \mathcal{M}^+ \cdot \sigma\}$
- $\mathbf{D}(\mathbf{a}) = \{\sigma \in \Sigma : \sigma \cdot \mathbf{a} \neq \mathbf{a} \cdot \sigma\}$

# Simulating the limit $\nu$ piece by piece⚠

**Idea #3:** Decompose heaps recursively by using **independent pyramids**

## Theorem – continued (Abbes & J. 2017)

If $X \subseteq \Sigma$, $\mathbf{a} \in \Sigma$ and $\nu$ is **Bernoulli** on $\overline{\mathcal{M}}(\Sigma)$, then

 $= \sum_{k=1}^{\infty}$  $^{k}$ $\times$  $+$ 

Heap $\xi$ s.t. $\mathbf{S}(\xi) \subseteq X$ — $\mathbf{a}$-pyramid — $\mathbf{a}$-free heap $\xi$ s.t. $\mathbf{S}(\mathbf{a} \cdot \xi) \subseteq X$ — $\mathbf{a}$-free heap $\xi$ s.t. $\mathbf{S}(\xi) \subseteq X$

where right-hand side random variables are **independent**.

 $=$  $\times$ A

$\mathbf{a}$-pyramid — $\mathbf{a}$-free heap $\xi$ s.t. $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$

- $\mathbf{S}(\xi) = \{\sigma \in \Sigma : \xi \in \mathcal{M}^+ \cdot \sigma\}$
- $\mathbf{D}(\mathbf{a}) = \{\sigma \in \Sigma : \sigma \cdot \mathbf{a} \neq \mathbf{a} \cdot \sigma\}$

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of $\mathbf{a}$-pyramids $\qquad (k \leftarrow \infty)$
- $\mathbf{a}$-pyramid = ( $\mathbf{a}$-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ $\mathbf{a}$
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $\quad (k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? $\quad (k \leftarrow$ **Geometric law**$)$

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $(k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? ($k \leftarrow$ **Geometric law**)
   - if $k = 0$: output $\xi = \zeta$

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $(k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? ($k \leftarrow$ **Geometric law**)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \nsubseteq X$: go back to step #1 (anticipated rejection)

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $(k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? ($k \leftarrow$ **Geometric law**)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \nsubseteq X$: go back to step #1 (anticipated rejection)
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X$: generate **a**-pyramids $\zeta_1, \ldots, \zeta_k$ and output $\xi = \zeta_1\cdot\zeta_2\cdots\zeta_k\cdot\zeta$

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $\quad (k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$ — if $\mathbf{a} \in X$, $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Leftrightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? $\quad$ ($k \leftarrow$ **Geometric law**)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \nsubseteq X$: go back to step #1 $\quad$ (anticipated rejection)
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X$: generate **a**-pyramids $\zeta_1, \ldots, \zeta_k$ and output $\xi = \zeta_1\cdot\zeta_2\cdots\zeta_k\cdot\zeta$

**Variant:** Choose $\mathbf{a} \in X$ and avoid rejection

# Our algorithm: Generating heaps distributed according to $\nu$

**Remarks:**

- if $\nu$ has support $\partial\mathcal{M}$, $\xi$ = infinite sequence of **a**-pyramids $\quad (k \leftarrow \infty)$
- **a**-pyramid = ( **a**-free sequence $\xi$ with $\mathbf{S}(\xi) \subseteq \mathbf{D}(\mathbf{a})$ ) $\cdot$ **a**
- $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Rightarrow \mathbf{S}(\zeta) \subseteq X$ — if $\mathbf{a} \in X$, $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X \Leftrightarrow \mathbf{S}(\zeta) \subseteq X$

**Auxiliary goal:** Generate a heap $\xi \in \mathcal{M}$ with $\mathbf{S}(\xi) \subseteq X$:

1. Generate an **a**-free heap $\zeta \in \mathcal{M}$ with $\mathbf{S}(\zeta) \subseteq X$
2. How many **a** should $\xi$ contain? $\quad$ ($k \leftarrow$ **Geometric law**)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \nsubseteq X$: go back to step #1 $\quad$ (anticipated rejection)
   - if $k \geqslant 1$ and $\mathbf{S}(\mathbf{a}\cdot\zeta) \subseteq X$: generate **a**-pyramids $\zeta_1, \ldots, \zeta_k$ and output $\xi = \zeta_1\cdot\zeta_2\cdots\zeta_k\cdot\zeta$

**Variant:** Choose $\mathbf{a} \in X$ and avoid rejection

**Running time/piece:** $n\,q$ or $n$ $\qquad$ **Read-only memory usage:** $n$ or $2^n$

where $\mathcal{M}' = \mathcal{M}(\Sigma\backslash\{a\})$ and $q = \mathcal{G}_{\mathcal{M}'}(p_c) \leqslant 1/p_c^n$ $\qquad$ ($q = n^{\Theta(n)}$ is possible)

# Contents

# A distributed simulation algorithm

Algorithm based on:

- precomputing and storing **Möbius polynomials** of sub-monoids
- decomposing heaps into **independent** pyramids/heaps in sub-monoids
- outputting pieces **one by one** with little synchronisation

# A distributed simulation algorithm

Algorithm based on:

- precomputing and storing **Möbius polynomials** of sub-monoids
- decomposing heaps into **independent** pyramids/heaps in sub-monoids
- outputting pieces **one by one** with little synchronisation

Two variants (mixtures are possible):

- small storage – anticipated rejection – rather low efficiency
- huge storage – no rejection – high, guaranteed efficiency

# A distributed simulation algorithm

Algorithm based on:

- precomputing and storing **Möbius polynomials** of sub-monoids
- decomposing heaps into **independent** pyramids/heaps in sub-monoids
- outputting pieces **one by one** with little synchronisation

Two variants (mixtures are possible):

- small storage – anticipated rejection – rather low efficiency
- huge storage – no rejection – high, guaranteed efficiency

Very efficient on graphs with:

- few cycles (small storage/high efficiency for a mix between variants)
- small tree-width (no preprocessing/storage)


Thank you