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

1 Introduction: Heaps of pieces and trace monoids

2 Simulating Bernoulli distributions

3 Step-by-step simulation and pyramids

4 Conclusion
Heaps of pieces and trace monoids

Heap of pieces

- Pieces:
  - a
  - b
  - c
  - d

Trace monoid

- Alphabet:
  \[
  \Sigma = \{a, b, c, d\}
  \]
Heaps of pieces and trace monoids

**Heap of pieces**

- **Pieces:**
  - $a$, $b$, $c$, $d$

- **Horizontal layout:**
  - $a$
  - $b$
  - $c$
  - $d$

- **Vertical heaps:**
  - $a$
  - $b$
  - $c$
  - $d$

**Trace monoid**

- **Alphabet:**
  - $\Sigma = \{a, b, c, d\}$

**Dependency graph**

Samy Abbes & Vincent Jugé  
Uniform generation in infinite concurrent runs – The case of trace monoids
Heaps of pieces and trace monoids

Heap of pieces

- Pieces:
  - a
  - b
  - c
  - d

- Horizontal layout:
  - a
  - 1
  - 2
  - 3
  - 4
  - 5

  b
  - 1
  - 2
  - 3
  - 4
  - 5

  c
  - 1
  - 2
  - 3
  - 4
  - 5

  d
  - 1
  - 2
  - 3
  - 4
  - 5

- Vertical heaps:
  - a
  - c
  - a
  - d

  b
  - c
  - d
  - a

Trace monoid

- Alphabet:
  \[ \Sigma = \{ a, b, c, d \} \]
Heaps of pieces and trace monoids

**Heap of pieces**

- **Pieces:**
  - $a$
  - $b$
  - $c$
  - $d$

- **Horizontal layout:**
  - $a$
  - $b$
  - $c$
  - $d$

- **Vertical heaps:**

**Trace monoid**

- **Alphabet:**
  \[ \Sigma = \{a, b, c, d\} \]

- **Dependence relation:**
  \[ D = \{\{a, b\}, \{b, c\}, \{c, d\}\} \]

- **Trace monoid:**
  \[ \mathcal{M} = \langle a, b, c, d | ac = ca, ad = da, bd = db \rangle^+ \]

---

Samy Abbes & Vincent Jugé

Uniform generation in infinite concurrent runs – The case of trace monoids
Heaps of pieces and dependency graph

Dependency graph

Heap of pieces
Heaps of pieces and dependency graph

Dependency graph

Heap of (disconnected) pieces

Samy Abbes & Vincent Jugé  Uniform generation in infinite concurrent runs – The case of trace monoids
Heaps of pieces and left divisibility

Heap of pieces

Can you pick one infinite heap uniformly at random?

Define your preferred notion of heap length

Study uniform distributions on traces of length $k$: what if $k \neq 8$?

Theorem (Abbes & Mairesse 2015) — not constructive!

If $\mu_k$ is the uniform measure on $M_k$: $|\xi| = k$ exists and is the critical Bernoulli distribution of $M$. 

Samy Abbes & Vincent Jugé
Uniform generation in infinite concurrent runs – The case of trace monoids
Heaps of pieces and left divisibility

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

Can you pick one infinite heap uniformly at random?

1. Heap length = #pieces in the heap
2. Weak convergence of distributions:
   \[ \mu_k \to \nu \iff (\forall x \in M^+, \mathbb{P}_{\mu_k}[x \leq \xi] \to \mathbb{P}_{\nu}[x \leq \xi]) \]
Heaps of pieces and left divisibility

Heaps of pieces

Can you pick one infinite heap uniformly at random?

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

\[ \mu_k \rightarrow \nu \iff (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leq \xi] \rightarrow \mathbb{P}_{\nu}[x \leq \xi]) \]

Theorem (Abbes & Mairesse 2015)

If \( \mu_k \) is the uniform 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:
   \[
   \mu_k \rightarrow \nu \iff (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leq \xi] \rightarrow \mathbb{P}_{\nu}[x \leq \xi])
   \]

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

If \( \mu_k \) is the uniform 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 $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, P_\mu[x \sigma \leq \zeta | x \leq \zeta] = P_\mu[\sigma \leq \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $P_\mu[\sigma \leq \zeta] = p$
Bernoulli distributions

Definition

A probability measure $\mu$ on $M$ is:

- **Bernoulli** if $\forall x \in M, \forall \sigma \in \Sigma, P_{\mu}[x \sigma \leq \zeta | x \leq \zeta] = P_{\mu}[\sigma \leq \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $P_{\mu}[\sigma \leq \zeta] = p$

Which are the possible values of the parameter $p$?
Bernoulli distributions

Definition

A probability measure $\mu$ on $\mathcal{M}$ is:
- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x \sigma \leq \zeta | x \leq \zeta] = \mathbb{P}_\mu[\sigma \leq \zeta]$
- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leq \zeta] = p$

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

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

Definition

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

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

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

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

Definition

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

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

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

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

$p^{1/\mathcal{H}(p)}$.

![Generating function $G(p)$](image1)

![Möbius polynomial $\mathcal{H}(p)$](image2)

Samy Abbes & Vincent Jugé

Uniform generation in infinite concurrent runs – The case of trace monoids
Bernoulli distributions

Definition

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

- **Bernoulli** if $\forall x \in \mathcal{M}, \forall \sigma \in \Sigma, \mathbb{P}_\mu[x \sigma \leq \zeta | x \leq \zeta] = \mathbb{P}_\mu[\sigma \leq \zeta]$

- **uniform Bernoulli** of parameter $p$ if, furthermore, $\mathbb{P}_\mu[\sigma \leq \zeta] = p$

- **critical Bernoulli** if $p = p_c$ (requires infinite heaps)

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

$0 \leq p \leq p_c$

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

---

Samy Abbes & Vincent Jugé
Uniform generation in infinite concurrent runs – The case of trace monoids
Infinite heaps

Heap of pieces

\[ \begin{array}{cc}
  a & c \\
 b  & a & d \\
 a & c \\
\end{array} \]

Sets of interest

- Finite heaps: \( B \)
- Infinite heaps: \( M \)
- All heaps: \( \text{all} \)

\( M \) is the compactification of \( \text{all} \) for the topology induced by: \( t \in \zeta \), \( x \in \zeta u \)

Reminder:

\( \mu : k \in \nu \to (m, p : \mu k r x \in \zeta) s \to (n, q : \nu r x \in \zeta) \)

Fact #1 (Abbes & Mairesse 2015)

The limit \( \nu \) is a distribution on the set \( M \) with support \( B M \).

Samy Abbes & Vincent Jugé
Uniform generation in infinite concurrent runs – The case of trace monoids
Infinite heaps

Heap of pieces

\[ \cdots \]
\[ \begin{array}{cc}
  a & c \\
  b \\
  a & d \\
  a & c \\
\end{array} \]

Sets of interest

\( \overline{\mathcal{M}} \)

finite heaps

\( \partial \mathcal{M} \)

infinite heaps

all heaps

\( \overline{\mathcal{M}} \) is the compactification of \( \mathcal{M} \) for the topology induced by \( \uparrow x = \{ \zeta : x \leq \zeta \} \)

Reminder: \( \mu_k \rightarrow \nu \iff (\forall x \in \mathcal{M}^+, \mathbb{P}_{\mu_k}[x \leq \xi] \rightarrow \mathbb{P}_\nu[x \leq \xi]) \)
Infinite heaps

Heap of pieces

Sets of interest

\( \overline{M} \) is the **compactification** of \( M \) for the topology induced by \( \uparrow x = \{ \zeta : x \leq \zeta \} \)

**Reminder:** \( \mu_k \rightarrow \nu \iff (\forall x \in M^+, \mathbb{P}_{\mu_k}[x \leq \xi] \rightarrow \mathbb{P}_{\nu}[x \leq \xi]) \)

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

The limit \( \nu \) is a distribution **on the set** \( \overline{M} \) with **support** \( \partial M \).
Contents

1  Introduction: Heaps of pieces and trace monoids

2  Simulating Bernoulli distributions

3  Step-by-step simulation and pyramids

4  Conclusion
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

$$\mu_1(\uparrow a \cup \uparrow c) = \frac{2}{3}$$

$$\mu_2(\uparrow a \cup \uparrow c) = \frac{5}{8}$$
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
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

![Diagram of floors and doors]

**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
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

![Diagram of floors with symbols](image)

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

1 Introduction: Heaps of pieces and trace monoids

2 Simulating Bernoulli distributions

3 Step-by-step simulation and pyramids

4 Conclusion
Simulating the limit \( \nu \) piece by piece

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

Example: Recursive decomposition using \( b \)-pyramids, then \( a \)-pyramids...
Simulating the limit $\nu$ piece by piece

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

<table>
<thead>
<tr>
<th>a-pyramid</th>
<th>b-pyramid</th>
<th>c-pyramid</th>
<th>d-pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$c$</td>
<td>$b$</td>
<td>$d$</td>
</tr>
<tr>
<td>$a$</td>
<td>$d$</td>
<td>$a$</td>
<td>$c$</td>
</tr>
<tr>
<td>$a$</td>
<td>$c$</td>
<td>$a$</td>
<td>$c$</td>
</tr>
<tr>
<td>$a$</td>
<td>$c$</td>
<td>$a$</td>
<td>$c$</td>
</tr>
</tbody>
</table>

Example: Recursive decomposition using $b$-pyramids, then $a$-pyramids...
Simulating the limit $\nu$ piece by piece

Idea #3: Decompose heaps recursively by using pyramids

Example: Recursive decomposition using $b$-pyramids
Simulating the limit $\nu$ piece by piece

Idea #3: Decompose heaps recursively by using pyramids

Example: Recursive decomposition using $b$-pyramids
Simulating the limit $\nu$ piece by piece

Idea #3: Decompose heaps recursively by using pyramids

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

\[
\text{Heap} = a\text{-pyramid} \times \text{Heap} + a\text{-free heap}
\]

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

\[
\text{Heap} = \text{a-pyramid} \times \text{Heap} + \text{a-free heap}
\]

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

\[
\text{a-pyramid} = \times \quad A
\]

\[
S(\xi) = \{\sigma \in \Sigma : \xi \in M^+ \cdot \sigma\}
\]

\[
D(a) = \{\sigma \in \Sigma : \sigma \cdot a \neq a \cdot \sigma\}
\]

\[
S(\xi) \subseteq D(a)
\]
Simulating the limit $\nu$ piece by piece

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

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

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

\[
\sum_{k=1}^{\infty} k \times \text{Heap } \xi \text{ s.t. } S(\xi) \subseteq X = \text{a-pyramid } \times \text{a-free heap } \xi \text{ s.t. } S(a \cdot \xi) \subseteq X + \text{a-free heap } \xi \text{ s.t. } S(\xi) \subseteq X
\]

where right-hand side random variables are independent.

\[
\text{a-pyramid } \times \text{a-free heap } \xi \text{ s.t. } S(\xi) \subseteq D(a)
\]

\[
\bullet \ S(\xi) = \{ \sigma \in \Sigma : \xi \in \mathcal{M}^+ \cdot \sigma \}
\]

\[
\bullet \ D(a) = \{ \sigma \in \Sigma : \sigma \cdot a \neq a \cdot \sigma \}
\]
Our algorithm: Generating heaps distributed according to $\nu$

Remarks:
- if $\nu$ has support $\partial M$, $\xi = \text{infinite sequence of } a\text{-pyramids}$ \hspace{1cm} ($k \leftarrow \infty$)
- $a\text{-pyramid} = (a\text{-free sequence } \xi \text{ with } S(\xi) \subseteq D(a)) \cdot a$
- $S(a \cdot \zeta) \subseteq X \Rightarrow S(\zeta) \subseteq X$
Our algorithm: Generating heaps distributed according to $\nu$

Remarks:
- if $\nu$ has support $\partial M$, $\xi = \text{infinite sequence of a-pyramids}$ \hspace{1cm} ($k \leftarrow \infty$)
- a-pyramid = ( a-free sequence $\xi$ with $S(\xi) \subseteq D(a)$ ) · a
- $S(a \cdot \zeta) \subseteq X \Rightarrow S(\zeta) \subseteq X$

Auxiliary goal: Generate a heap $\xi \in M$ with $S(\xi) \subseteq X$:
1. Generate an a-free heap $\zeta \in M$ with $S(\zeta) \subseteq X$
2. How many a should $\xi$ contain? \hspace{1cm} ($k \leftarrow \text{Geometric law}$)
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 \( S(\xi) \subseteq D(a) \) ) \cdot a
- \( S(a \cdot \zeta) \subseteq X \Rightarrow S(\zeta) \subseteq X \)

**Auxiliary goal:** Generate a heap \( \xi \in \mathcal{M} \) with \( S(\xi) \subseteq X \):
1. Generate an a-free heap \( \zeta \in \mathcal{M} \) with \( S(\zeta) \subseteq X \)
2. How many a should \( \xi \) contain? \( (k \leftarrow \text{Geometric law}) \)
   - if \( k = 0 \): output \( \xi = \zeta \)
Our algorithm: Generating heaps distributed according to $\nu$

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

Auxiliary goal: Generate a heap $\xi \in M$ with $S(\xi) \subseteq X$:

1. Generate an $a$-free heap $\zeta \in M$ with $S(\zeta) \subseteq X$
2. How many $a$ should $\xi$ contain?
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geq 1$ and $S(a \cdot \zeta) \nsubseteq X$: go back to step #1 (anticipated rejection)
Our algorithm: Generating heaps distributed according to $\nu$

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

Auxiliary goal: Generate a heap $\xi \in \mathcal{M}$ with $S(\xi) \subseteq X$:
1. Generate an a-free heap $\zeta \in \mathcal{M}$ with $S(\zeta) \subseteq X$
2. How many a should $\xi$ contain? ($k \leftarrow \text{Geometric law}$)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geq 1$ and $S(a \cdot \zeta) \not\subseteq X$: go back to step $\#1$ (anticipated rejection)
   - if $k \geq 1$ and $S(a \cdot \zeta) \subseteq X$: 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 $\partial \mathcal{M}$, $\xi = \text{infinite sequence of } a\text{-pyramids}$ ($k \leftarrow \infty$)
- $a\text{-pyramid} = (a\text{-free sequence } \xi \text{ with } S(\xi) \subseteq D(a)) \cdot a$
- $S(a \cdot \zeta) \subseteq X \Rightarrow S(\zeta) \subseteq X$ — if $a \notin X$, $S(a \cdot \zeta) \subseteq X \Leftrightarrow S(\zeta) \subseteq X$

Auxiliary goal: Generate a heap $\xi \in \mathcal{M}$ with $S(\xi) \subseteq X$:
1. Generate an $a\text{-free heap } \zeta \in \mathcal{M}$ with $S(\zeta) \subseteq X$
2. How many $a$ should $\xi$ contain? ($k \leftarrow \text{Geometric law}$)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geq 1$ and $S(a \cdot \zeta) \not\subseteq X$: go back to step #1 (anticipated rejection)
   - if $k \geq 1$ and $S(a \cdot \zeta) \subseteq X$: generate $a\text{-pyramids } \zeta_1, \ldots, \zeta_k$
     and output $\xi = \zeta_1 \cdot \zeta_2 \cdot \ldots \cdot \zeta_k \cdot \zeta$

Variant: Choose $a \in X$ and avoid rejection
Our algorithm: Generating heaps distributed according to $\nu$

Remarks:
- if $\nu$ has support $\partial M$, $\xi = \text{infinite sequence of } a\text{-pyramids}$ \hspace{1cm} ($k \leftarrow \infty$)
- $a\text{-pyramid} = ( \text{a-free sequence } \xi \text{ with } S(\xi) \subseteq D(a) ) \cdot a$
- $S(a \cdot \zeta) \subseteq X \Rightarrow S(\zeta) \subseteq X \quad \text{if } a \in X, S(a \cdot \zeta) \subseteq X \iff S(\zeta) \subseteq X$

Auxiliary goal: Generate a heap $\xi \in M$ with $S(\xi) \subseteq X$:
1. Generate an $a$-free heap $\zeta \in M$ with $S(\zeta) \subseteq X$
2. How many $a$ should $\xi$ contain? \hspace{1cm} ($k \leftarrow \text{Geometric law}$)
   - if $k = 0$: output $\xi = \zeta$
   - if $k \geq 1$ and $S(a \cdot \zeta) \not\subseteq X$: go back to step #1 (anticipated rejection)
   - if $k \geq 1$ and $S(a \cdot \zeta) \subseteq X$: generate $a$-pyramids $\zeta_1, \ldots, \zeta_k$
     and output $\xi = \zeta_1 \cdot \zeta_2 \cdots \zeta_k \cdot \zeta$

Variant: Choose $a \in X$ and avoid rejection

Running time/piece: $n q$ or $n$ \hspace{1cm} Read-only memory usage: $n$ or $2^n$
where $M' = M(\Sigma \setminus \{a\})$ and $q = G_{M'}(p_c) \leq 1/p_c^n$ \hspace{1cm} ($q = n^{\Theta(n)}$ is possible)
Contents

1. Introduction: Heaps of pieces and trace monoids
2. Simulating Bernoulli distributions
3. Step-by-step simulation and pyramids
4. Conclusion
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)
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)
A distributed simulation algorithm

Algorithm based on:
- precomputing and storing \textit{Möbius polynomials} of sub-monoids
- decomposing heaps into \textit{independent} pyramids/heaps in sub-monoids
- outputting pieces \textit{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