Enumeration and Random Generation of Concurrent Computations

O. Bodini\textsuperscript{1} A. Genitrini\textsuperscript{2} F. Peschanski\textsuperscript{2}

\textsuperscript{1}Université Paris 13 – LIPN
\textsuperscript{2}UPMC Paris – LIP6

Aléa – March, 2012
1 Motivations
   - Concurrent computations
   - Related works

2 Shuffle trees and their typical shape
   - Recursive construction
   - Quantitative analysis

3 Algorithms
   - Probability of a concurrent run prefix
   - Uniform random generation of a run
When analyzing concurrent processes, the shuffle operator is the main source of combinatorial explosion. [Mi80], [ClGrPe99]
In concurrency theory, one manipulates:

- syntactic objects $\Rightarrow$ Process trees
- their semantic interpretation $\Rightarrow$ Shuffle trees
In concurrency theory, one manipulates:
- syntactic objects $\Rightarrow$ Process trees
- their semantic interpretation $\Rightarrow$ Shuffle trees

Ideas
- to consider these objects as combinatorial structures
- to use analytic combinatorics for quantitative studies
A process tree is a specification of events with precedence constraints:

```
  a
 / \
|   |
|   c|
|   |
|  b|
|  d|
```

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Process trees and shuffle trees

A **process tree** is a specification of **events with precedence constraints**:

![Process tree diagram]

The induced **shuffle tree** lists **all admissible concurrent runs** by sharing prefixes, as in a trie:

![Shuffle tree diagram]
Related works

[BrWi91]
[At90]

Poset Theory

linear extensions
Related works

- Poset Theory
- Algebraic Combinatorics
- Linear extensions
- Partly commutative algebras

References:
- [BrWi91]
- [At90]
- [DuHiNoTh11]
- [BoFeLaRe11]
- [Mi80]
- [ClGrPe99]
Related works

- Poset Theory
  - [BrWi91]
  - [At90]
- Algebraic Combinatorics
  - [DuHiNoTh11]
  - [BoFéLaRe11]
- Concurrency Theory
  - [Mi80]
  - [ClGrPe99]
- Linear extensions
- Partly commutative algebras
- Shuffle trees

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Outline

1️⃣ Motivations

2️⃣ Shuffle trees and their typical shape

3️⃣ Algorithms
Definition: Child contraction

Let $T$ be a tree with children $T_1, \ldots, T_r$ whose root-events are $\ell_1, \ldots, \ell_r$ ($r \in \mathbb{N}^*$). The $i$-contraction of $T$ is the tree $T \triangleleft i$ with root $\ell_i$ and children $T_1, \ldots, T_{i-1}, T_i, T_{i+1}, \ldots, T_r$ where $T_{i_1}, \ldots, T_{i_m}$ are the children of $T_i$.

Example

$T = \quad \Rightarrow \quad T \triangleleft 2 =$

```
    a
   /|
  b  c
 |  /|
|  e f
|   /|
|  g h
|   /|
|  i j
|   /|
|  k l
|   /|
|  m n
```

```
    c
   /|
  b  e
 |  /|
|  f
|   /|
|  g
|   /|
|  h
|   /|
|  i
```

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Recursive definition

Let $T$ be a tree. Its **shuffle tree** $\text{Shuf}(T)$ is defined inductively as:
- if $T$ is a leaf, then $\text{Shuf}(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $\text{Shuf}(T)$ is the tree with root-event $\ell$ and children $\text{Shuf}(T \triangleleft 1), \ldots, \text{Shuf}(T \triangleleft r)$
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its shuffle tree $\text{Shuf}(T)$ is defined inductively as:
- if $T$ is a leaf, then $\text{Shuf}(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $\text{Shuf}(T)$ is the tree with root-event $\ell$ and children $\text{Shuf}(T \triangleleft 1), \ldots, \text{Shuf}(T \triangleleft r)$

Example (Shuffle / Contraction):

```
   a
  / \      
b /   \     
c   d
  \ / \   /   \\
  e --- f
```
Building shuffle trees (2)

Recursive definition

Let \( T \) be a tree. Its **shuffle tree** \( \text{Shuf}(T) \) is defined inductively as:
- if \( T \) is a leaf, then \( \text{Shuf}(T) := T \)
- if \( T \) has root-event \( \ell \) and children \( T_1, \ldots, T_r \) \( (r \in \mathbb{N}^*) \)
  then \( \text{Shuf}(T) \) is the tree with root-event \( \ell \)
  and children \( \text{Shuf}(T \triangleleft 1), \ldots, \text{Shuf}(T \triangleleft r) \)

**Example (Shuffle / Contraction):**

```
    a
   / \
  a   \
   /    \
  b    c
     \   \
      \  d
       \  \
        e  f
```
Recursive definition

Let $T$ be a tree. Its **shuffle tree** $Shuf(T)$ is defined inductively as:

- if $T$ is a leaf, then $Shuf(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $Shuf(T)$ is the tree with root-event $\ell$ and children $Shuf(T \triangleleft 1), \ldots, Shuf(T \triangleleft r)$

**Example (Shuffle / Contraction):**

```
   a
  / \  
 b   c
   \ /  
    d  e
     \  
      f
```
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its **shuffle tree** $\text{Shuf}(T)$ is defined inductively as:

- if $T$ is a leaf, then $\text{Shuf}(T) := T$

- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $\text{Shuf}(T)$ is the tree with root-event $\ell$ and children $\text{Shuf}(T \triangleleft 1), \ldots, \text{Shuf}(T \triangleleft r)$

Example (Shuffle / Contraction):

```
     a
    / \
   b   b
  /   / \
 c   d   c   d
 / \ / \ / \ / \
 e f e f
```
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its **shuffle tree** $Shuf(T)$ is defined inductively as:
- if $T$ is a leaf, then $Shuf(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $Shuf(T)$ is the tree with root-event $\ell$ and children $Shuf(T \triangleleft 1), \ldots, Shuf(T \triangleleft r)$

Example (Shuffle / Contraction):

```
        a
         |
         b
        /   \
      c     d
     / |    /   |
    e  d  c   e  f
```

O. Bodini, A. Genitrini, F. Peschanski  Concurrent Computations
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its **shuffle tree** $Shuf(T)$ is defined inductively as:
- if $T$ is a leaf, then $Shuf(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r \ (r \in \mathbb{N}^*)$ then $Shuf(T)$ is the tree with root-event $\ell$ and children $Shuf(T \triangleleft 1), \ldots, Shuf(T \triangleleft r)$

Example (Shuffle / Contraction):

```
  a
 /  \
 b   
   /  \
  c   d
 /  \
 d   e
 /   \n e   f
```
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its **shuffle tree** $Shuf(T)$ is defined inductively as:

- if $T$ is a leaf, then $Shuf(T) := T$

- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r$ ($r \in \mathbb{N}^*$) then $Shuf(T)$ is the tree with root-event $\ell$ and children $Shuf(T \triangleleft 1), \ldots, Shuf(T \triangleleft r)$

**Example (Shuffle / Contraction):**

```
   a
   |
  --b
  |
 c   d
  |
 d   c
  |
 e   e
  |
 f
```
Building shuffle trees (2)

Recursive definition

Let $T$ be a tree. Its **shuffle tree** $Shuf(T)$ is defined inductively as:

- if $T$ is a leaf, then $Shuf(T) := T$
- if $T$ has root-event $\ell$ and children $T_1, \ldots, T_r \ (r \in \mathbb{N}^*)$
  then $Shuf(T)$ is the tree with root-event $\ell$
  and children $Shuf(T_1), \ldots, Shuf(T_r)$

Example (Shuffle / Contraction):

```
 a
  |   b
 c---
  |   d
  |   |
  d   c
  |   |
  |   |
  e   f
  |   |
  |   |
  f   e
```

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Observation

Information is extremely redundant in shuffle trees:
One can recover the process tree by traversing a single branch of
the shuffle tree.
In order to analyze the combinatorial explosion of shuffle trees, we want to answer the following questions:

- What is the number of runs for a given process tree $T$?
  $\Rightarrow$ the number of leaves in $Shuf(T)$

- What is the size of the shuffle tree induced by $T$?
  $\Rightarrow$ no correlation known with the number of runs (sharing)
Main results

**Theorem**
The typical shape of a shuffle tree built on a process tree of size $n$:

$$\Theta(\sqrt{n})$$

- Motivations
- Shuffle trees and their typical shape
- Algorithms

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Main results

Theorem

The typical shape of a shuffle tree built on a process tree of size $n$:

$$\Theta(\sqrt{n})$$

$$n - 1$$

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Main results

Theorem

The typical shape of a shuffle tree built on a process tree of size $n$:

$$\Theta(\sqrt{n})$$

$$\sim \frac{n!}{2^{n-1}}$$
Main results

Theorem

The typical shape of a shuffle tree built on a process tree of size $n$:

$$\Theta(\sqrt{n})$$

$$\sim e^{\frac{n!}{2^{2n-1}}}$$

$$\sim \frac{n!}{2^{2n-1}}$$
Concurrent runs and increasing trees (1)

Definition: Increasing tree

An *increasing tree* is a labelled plane tree such that the sequence of labels along any branch starting at the root is increasing.
Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

Motivations

Shuffle trees and their typical shape

Algorithms

Concurrent runs and increasing trees (2)
Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

O. Bodini, A. Genitrini, F. Peschanski
**Lemma: Bijection**

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$.

---

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

\[
\begin{align*}
\text{Diagram:} \\
\text{Tree}(T) & \\
\text{Increasing tree:} &
\end{align*}
\]
Concurrent runs and increasing trees (2)

Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

O. Bodini, A. Genitrini, F. Peschanski
Concurrent runs and increasing trees (2)

Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 
Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

O. Bodini, A. Genitrini, F. Peschanski
Lemma: Bijection

Let $T$ be a process tree. The number of runs associated to $T$ corresponds to the number of increasing trees whose structure is the unlabelled tree $T$. 

O. Bodini, A. Genitrini, F. Peschanski
Theorem: Hook length in trees [Kn73]

Let $T$ be a unlabelled tree.

The number of increasing trees built on $T$ equals:

$$\ell_T = \frac{|T|!}{\prod_{R \text{ subtree of } T} |R|}.$$ 

This corresponds equivalently to the number of runs induced by $T$. 

\[ \ell_T = \frac{6!}{6 \cdot 5 \cdot 1 \cdot 3 \cdot 1 \cdot 1} = 8. \]
Proposition

The *arithmetic* mean number of runs built on trees of size $n$ is:

$$\bar{\ell}_n \sim_{n \to \infty} \frac{n!}{2^{n-1}} \sim 2\sqrt{2\pi n} \left(\frac{n}{2e}\right)^n.$$
Mean number of runs and mean growth

Proposition

The arithmetic mean number of runs built on trees of size $n$ is:

$$\overline{\ell}_n \sim_{n \to \infty} \frac{n!}{2^{n-1}} \sim 2\sqrt{2\pi n} \left(\frac{n}{2e}\right)^n.$$

Proposition

The geometric mean growth between trees of size $n$ and their number of runs is:

$$\overline{\Gamma}_n \sim_{n \to \infty} \sqrt{2\pi} n^{n-1} \exp\left(-\left(1 + 2L\left(\frac{1}{4}\right)\right)n + \sqrt{\pi n} + L\left(\frac{1}{4}\right)\right),$$

with $L\left(\frac{1}{4}\right) = \sum_{n>1} \log n \cdot \text{Cat}_n \cdot 4^{-n} \approx 0.579043921 \pm 5 \cdot 10^{-9}$. 

O. Bodini, A. Genitrini, F. Peschanski
Size of shuffle trees: substructures

**Definition**

Let $T$ be a process tree. We define a *substructure* of $T$ a tree obtained by removing some subtrees of $T$.

**Example**

```
    a
   / \  
  b   c
     /   
    d     
   /       
  e       f
```
Size of shuffle trees: substructures

**Definition**

Let $T$ be a process tree. We define a *substructure* of $T$ a tree obtained by removing some subtrees of $T$.

**Example**

```
  a
 /|
 b / \
 c    d
 / | \
 e   f
```

```
  a
 /|
 b / \
 c    d
 / | \
 e   f
```

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
## Size of shuffle trees: substructures

### Definition
Let $T$ be a process tree. We define a *substructure* of $T$ a tree obtained by removing some subtrees of $T$.

### Example

```
     a
    /|
   / \
  b   c
 /    |
|     d |
|       |
|       e
|       f
```

```
     a
    /|
   / \
  b   c
 /    |
|     d |
|       |
|       f
```

O. Bodini, A. Genitrini, F. Peschanski
Size of shuffle trees: substructures

Definition
Let $T$ be a process tree. We define a substructure of $T$ a tree obtained by removing some subtrees of $T$.

Example
\[\begin{align*}
\text{Left:} & \quad a \quad \begin{array}{c} b \\ c \quad d \\ e \quad f \end{array} \\
\text{Right:} & \quad a \quad \begin{array}{c} b \\ d \\ e \quad f \end{array}
\end{align*}\]
Definition

Let $T$ be a process tree. We define a substructure of $T$ a tree obtained by removing some subtrees of $T$.

Example

- For the tree on the left, removing $d$, $c$, and $e$ yields the tree on the right.
- For the tree on the right, removing $c$ and $d$ yields the tree on the left.
Size of shuffle trees: substructures

**Definition**

Let \( T \) be a process tree. We define a *substructure* of \( T \) a tree obtained by removing some subtrees of \( T \).

**Example**

```
        a
       /|
      /  \
     b    \
    /     \
   c      d
  /       /|
 d       e  f
```

```
        a
       /|
      /  \
     b    \
    /     \
   d      e
```
Size of shuffle trees: substructures

Definition
Let $T$ be a process tree. We define a substructure of $T$ a tree obtained by removing some subtrees of $T$.

Example
```
    a
   /|
  b  c
 /|
d  e
 /|
f
```

```
    a
   /|
  b  d
 /|
f
```

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Size of shuffle trees: substructures

**Definition**

Let $T$ be a process tree. We define a *substructure* of $T$ a tree obtained by removing some subtrees of $T$.

**Example**

```
         a
        /|
       / ||
      b  c d
     /    /|
    e    f  
```

```
         a
        /|
       / ||
      b  d
     /    |
    e     
```
Size of shuffle trees: substructures

Definition
Let $T$ be a process tree. We define a substructure of $T$ a tree obtained by removing some subtrees of $T$.

Example

```
  a
 /|
/  |
b   d
 |  /|
|  /  |
eg  e  f
  c   
```

```
  a
 /|
/  |
b   c
```

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Size of shuffle trees: substructures

**Definition**

Let $T$ be a process tree. We define a *substructure* of $T$ a tree obtained by removing some subtrees of $T$.

**Example**

```
          a
         /
        b   a
       /    /
      c    b
     /     /   
    e   d   
   /   /    
  f   
```
**Size of shuffle trees: substructures**

**Definition**
Let \( T \) be a process tree. We define a *substructure* of \( T \) a tree obtained by removing some subtrees of \( T \).

**Example**

```
    a
   /|
  /  |
 b  c  d
   /    |
  e  f   a
```
Size of a shuffle tree

Proposition

The size of the shuffle tree built on $T$ satisfies:

$$n_T = \sum_{R \text{ substructure of } T} \ell_R.$$
Size of a shuffle tree

**Proposition**

The size of the shuffle tree built on $T$ satisfies:

$$n_T = \sum_{R \text{ substructure of } T} \ell_R.$$
Size of a shuffle tree

Proposition

The size of the shuffle tree built on $T$ satisfies:

$$n_T = \sum_{R \text{ substructure of } T} \ell_R.$$

Example

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Proposition

The size of the shuffle tree built on $T$ satisfies:

$$n_T = \sum_{R \text{ substructure of } T} \ell_R.$$
The mean size $\bar{s}_n$ of a shuffle tree induced by a tree of size $n$ follows a \textit{P-recurrence} and satisfies:

$$
\bar{s}_n \sim_{n \to \infty} e \frac{n!}{2^{n-1}} \sim 2e\sqrt{2\pi}n \left(\frac{n}{2e}\right)^n.
$$
Outline of the proof (1)

First step:
The generating function of the cumulative size of shuffle trees.
Outline of the proof (1)

First step:
The generating function of the cumulative size of shuffle trees.

\[ C = \mathbb{Z} \times \text{Seq} \mathcal{C} \quad \mathcal{M} = \mathcal{U} \times \mathbb{Z} \times \text{Seq}(\mathcal{M} \cup \mathcal{C}) \]
Outline of the proof (1)

First step:

The generating function of the cumulative size of shuffle trees.

\[
S = U^u \star Z \times \text{Seq}(S \cup C)
\]
Outline of the proof (1)

First step:
The generating function of the cumulative size of shuffle trees.

\[ S = \mathcal{U}^\Box u \star \mathcal{Z} \times \text{Seq}(S \cup C) \]

\[ S(z, u) = \int_{v=0}^{\infty} \frac{z}{1 - S(z, v) - C(z)} dv = \sum_{n,k \in \mathbb{N}} S_{n,k} \cdot z^n \cdot \frac{u^k}{k!} \]

where \( S_{n,k} \) is

\[ \sum_{\substack{T \\ |T| = n}} \sum_{S \text{ substructure of size } k \text{ of } T} \ell_S. \]
Outline of the proof (1)

First step:
The generating function of the cumulative size of shuffle trees.

\[ S = U^u \star Z \times \text{Seq}(S \cup C) \]

\[ S(z, u) = \int_{v=0}^{\infty} \frac{z}{1 - S(z, v) - C(z)} dv = \sum_{n,k \in \mathbb{N}} S_{n,k} \cdot z^n \cdot \frac{u^k}{k!} \]

where \( S_{n,k} \) is

\[ \sum_{|T|=n} \sum_{\ell_S} S \text{ substructure of size } k \text{ of } T \]

By substituting \( u^k \) by \( k! \) (Gamma transformation) we obtain the generating function \( S(z) \) for the size of the shuffle trees.

\[ S(z) = \int_{u=0}^{\infty} S(z, u) \exp(-u) du. \]
Outline of the proof (2)

Second step: Assisted proof using gfun.
Outline of the proof (2)

Second step: Assisted proof using gfun.

- As $S(z, u)$ is algebraic, it is holonomic.

- As $S(z, u)$ is holonomic, its Laplace transform is holonomic:

$$\hat{S}(z, u) = \int_{v=0}^{\infty} S(z, uv) \exp(-v)dv.$$  

- Using the holonomic stability under partial evaluation, $S(z)$ is holonomic.

- As $S(z)$ is holonomic, its coefficients $s_n$ follow a P-reccurence.
Computer assisted?

Motivations

Shuffle trees and their typical shape

Algorithms

Computer assisted?

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Outline of the proof (3)

Third step: Asymptotic behaviour of the coefficients of $\bar{s}_n$. 
Outline of the proof (3)

Third step: Asymptotic behaviour of the coefficients of $\bar{s}_n$.

- Classical method gives:
  \[ \bar{s}_n \cdot \frac{2^{n-1}}{n!} = \theta(1). \]

- Some more work is necessary to obtain the constant.

- Finally,
  \[ \bar{s}_n \sim_{n \to \infty} e \frac{n!}{2^{n-1}}. \]
Outline

1. Motivations
2. Shuffle trees and their typical shape
3. Algorithms
Probability of a run prefix

Data: $T$: a weighted process tree of size $n$
Data: $\sigma := \langle \alpha_1, \ldots, \alpha_p \rangle$: a run prefix of length $p \leq n$
Result: $\rho_\sigma$: the probability of $\sigma$ in the shuffle of $T$

$$\rho_\sigma := 1$$
$$i := 1$$

for $i$ from 1 to $p - 1$ do
  $$\rho_\sigma := \rho_\sigma \times \frac{|T(\alpha_{i+1})|}{n-i}$$
  $$i := i + 1$$

return $\rho_\sigma$

Directly deduced from the hook length formula.

Proposition

The number of runs of a process tree $T$ of size $n$ can be computed in $O(n)$ operations.

[At90] gave a quadratic complexity algorithm.
Uniform random generation example

\[
\{1..11\}
\]

\[
\begin{array}{c}
a^{11} \\
\mid \\
\vdots \\
\vdots \\
b^{10} \\
\mid \\
c^3 \\
\mid \\
d^1 \\
e^1 \\
g^1 \\
f^6 \\
h^4 \\
j^1 \\
k^1 \\
im^1 \\
empty \\
n \end{array}
\]

\text{run} = []
Uniform random generation example

construct
\{1..11\}

\[ a | 0,11,0 | L \]

run = []
Uniform random generation example

random choice; search

$5 \in \{1..11\}$

$a \mid 0, 11, 0 \mid L$

run = []
Shuffle trees and their typical shape

Uniform random generation example

$5 \in \{1..11\}$

$O. Bodini, A. Genitrini, F. Peschanski$

Concurrent Computations
Uniform random generation example

\[
\begin{align*}
\text{run} &= [a] \\
\text{swap} &\{1..10\} \\
b &\mid 0, 10, 0 \mid L
\end{align*}
\]
Uniform random generation example

random choice; search

7 ∈ \{1..10\}

\[ b \mid 0, 10, 0 \mid L \]

run = [a]
Uniform random generation example

take

\[ 7 \in \{1..10\} \]

\[ b \mid 0, 10, 0 \mid L \]

\[ \text{run} = [a, b] \]
Uniform random generation example

Motivations
Shuffle trees and their typical shape
Algorithms

\[
\text{swap } \{1..9\}
\]

\[
c \mid 0, 3, 0 \mid L
\]

\[
\begin{array}{c}
\text{run} = [a, b]\\
\end{array}
\]
Uniform random generation example

-run = [a, b]

\textit{construct; invert bit} 
\{1..9\}

\begin{align*}
&c \mid 0, 3, 0 \mid R \\
&f \mid 0, 6, 0 \mid L
\end{align*}

Motivations

Shuffle trees and their typical shape

Algorithms

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

```
update
{1..9}
```

```
c | 6, 3, 0 | R
f | 0, 6, 0 | L
```

run = \([a, b]\)
Uniform random generation example

random choice; search

\[ 8 \in \{1..9\} \]

\[
\begin{array}{c}
\text{c | 6, 3, 0 | R} \\
\text{f | 0, 6, 0 | L} \\
\text{empty}
\end{array}
\]

run = \([a, b]\)
Uniform random generation example

take

$8 \in \{1..9\}$

$c \mid 6, 3, 0 \mid R$

$f \mid 0, 6, 0 \mid L$

run = [$a, b, c$]
Uniform random generation example

\[
\text{swap} \quad \{1..8\}
\]

\[
d \mid 6, 1, 0 \mid R
\]

\[
f \mid 0, 6, 0 \mid L
\]

run = \[a, b, c]\]

\[O. \, Bodini, \, A. \, Genitrini, \, F. \, Peschanski\]

\[Concurrent Computations\]
Uniform random generation example

\[
\begin{array}{c}
\text{construct; invert bit} \\
\{1..8\}
\end{array}
\]

\[
\begin{array}{c}
d | 6, 1, 0 | L \\
f | 0, 6, 0 | L \\
e | 0, 1, 0 | L \\
\end{array}
\]

\[d^1 \mid e^1 \mid f^6 \mid g^1 \mid h^4 \mid i^1 \mid j^1 \mid k^1\]

\[\text{empty}\]

\[\text{run} = [a, b, c]\]
Uniform random generation example

run = [a, b, c]
Uniform random generation example

Random choice; search

\[ 8 \in \{1..8\} \]

\[ d \mid 6, 1, 1 \mid L \]

\[ f \mid 0, 6, 0 \mid L \]

\[ e \mid 0, 1, 0 \mid L \]

Run = \[ a, b, c \]
Uniform random generation example

\[ 8 - (6 + 1) = 1 \in \{1..1\} \]

\[
\begin{align*}
\text{search} \\
\quad d & | 6, 1, 1 | L \\
\quad f & | 0, 6, 0 | L \\
\quad e & | 0, 1, 0 | L \\
\text{empty}
\end{align*}
\]

run = \[a, b, c\]

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

take \{1..7\}

\[
\begin{align*}
d &| 6, 1, 1 \mid L \\
 f &| 0, 6, 0 \mid L \\
 e &| 0, 1, 0 \mid L \\
 &\text{empty}
\end{align*}
\]

\[
\begin{center}
\begin{tikzpicture}
\node at (0,0) {a\textsuperscript{11}}; \\
\node at (0,-1) {b\textsuperscript{10}}; \\
\node at (0,-2) {c\textsuperscript{3}}; \\
\node at (0,-3) {d\textsuperscript{1}}; \\
\node at (0,-4) {e\textsuperscript{1}}; \\
\node at (0,-5) {f\textsuperscript{6}}; \\
\node at (0,-6) {g\textsuperscript{1}}; \\
\node at (0,-7) {h\textsuperscript{4}}; \\
\node at (0,-8) {i\textsuperscript{1}}; \\
\node at (0,-9) {j\textsuperscript{1}}; \\
\node at (0,-10) {k\textsuperscript{1}}; \\
\end{tikzpicture}
\end{center}
\]

run = [a, b, c, e]
Uniform random generation example

\[
\text{run} = [a, b, c, e]
\]

O. Bodini, A. Genitrini, F. Peschanski  
Concurrent Computations
Uniform random generation example

Motivations
Shuffle trees and their typical shape
Algorithms

update
\{1..7\}

d | 6, 1, 0 | L

f | 0, 6, 0 | L

\epsilon | 0, 0, 0 | L

empty

run = [a, b, c, e]

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Uniform random generation example

random choice; search

2 ∈ {1..7}

\( d \mid 6, 1, 0 \mid L \)

\( f \mid 0, 6, 0 \mid L \)

\( \epsilon \mid 0, 0, 0 \mid L \)

empty

run = \([a, b, c, e]\)
Motivations

Shuffle trees and their typical shape

Algorithms

Uniform random generation example

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

take \{1..6\}

d | 6, 1, 0 | L
f | 0, 6, 0 | L
ε | 0, 0, 0 | L

\[
\text{run} = [a, b, c, e, f]
\]
Uniform random generation example

\[ \text{run} = [a, b, c, e, f] \]
Uniform random generation example

\[ \text{run} = [a, b, c, e, f] \]
Uniform random generation example

\[ \text{run} = [a, b, c, e, f] \]
Uniform random generation example

\[ \text{run} = [a, b, c, e, f] \]

*: *update* *

\[ \{1..6\} \]

**Shuffle trees and their typical shape**
Uniform random generation example

Algorithm: random choice; search

\[ 1 \in \{1..6\} \]

\[ d \mid 1, 1, 4 \mid L \]
\[ g \mid 0, 1, 0 \mid L \]
\[ h \mid 0, 4, 0 \mid L \]

Run: \[ [a, b, c, e, f] \]
Uniform random generation example

1 ∈ {1..1}

d | 1, 1, 4 | L

g | 0, 1, 0 | L

h | 0, 4, 0 | L

e | empty

run = [a, b, c, e, f]
Uniform random generation example

\[
\text{take} \quad \{1..5\}
\]

\[
d \mid 1, 1, 4 \mid L
\]

\[
g \mid 0, 1, 0 \mid L
\]

\[
h \mid 0, 4, 0 \mid L
\]

run = \[a, b, c, e, f, g\]
Uniform random generation example

```
run = [a, b, c, e, f, g]
```

```
Motivations
Shuffle trees and their typical shape
Algorithms
O. Bodini, A. Genitrini, F. Peschanski
```
Uniform random generation example

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

random choice; search

$$3 \in \{1..5\}$$

$$d \mid 0, 1, 4 \mid L$$

$$\epsilon \mid 0, 0, 0 \mid L$$

$$h \mid 0, 4, 0 \mid L$$

empty

run = [a, b, c, e, f, g]

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

Motivations

Shuffle trees and their typical shape

Algorithms

Uniform random generation example

search

2 ∈ \{1..4\}

d | 0, 1, 4 | L

ε | 0, 0, 0 | L

h | 0, 4, 0 | L

empty

run = [a, b, c, e, f, g]
Uniform random generation example

\[
\text{take } \{1..4\}
\]

\[
\begin{align*}
\text{d} & | 0, 1, 4 | L \\
\text{\epsilon} & | 0, 0, 0 | L \\
\text{h} & | 0, 4, 0 | L \\
\text{run} &= [a, b, c, e, f, g, h]
\end{align*}
\]
Motivations

Shuffle trees and their typical shape

Algorithms

Uniform random generation example

\[ \text{run} = [a, b, c, e, f, g, h] \]

\[ \text{swap} \quad \{1..4\} \]

\[ d \mid 0, 1, 4 \mid L \]

\[ \epsilon \mid 0, 0, 0 \mid L \]

\[ i \mid 0, 1, 0 \mid L \]

empty

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

Motivations
Shuffle trees and their typical shape
Algorithms

update
{1..4}

d | 0, 1, 1 | L

\[\epsilon | 0, 0, 0 | L \quad i | 0, 1, 0 | L\]

empty

run = [a, b, c, e, f, g, h]
Uniform random generation example

\[ \text{run} = [a, b, c, e, f, g, h] \]

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Uniform random generation example

update
\{1..4\}

\[\begin{array}{c}
\text{update} \\
\{1..4\}
\end{array}\]

\[\begin{array}{c}
d \mid 1,1,1 \mid L \\
\text{empty}
\end{array}\]

\[\begin{array}{c}
j \mid 0,1,0 \mid L \\
i \mid 0,1,0 \mid L
\end{array}\]

\[\begin{array}{c}
r\text{un} = [a, b, c, e, f, g, h]
\end{array}\]
Uniform random generation example

construct; invert bits

\{1..4\}

\begin{align*}
\text{run} &= [a, b, c, e, f, g, h] \\
\end{align*}
Uniform random generation example

update
\{1..4\}

d | 2, 1, 1 | R

j | 1, 1, 0 | R

i | 0, 1, 0 | L

k | 0, 1, 0 | L

empty

run = [a, b, c, e, f, g, h]
Uniform random generation example

random choice; search

\[ 2 \in \{1..4\} \]

\[ d \mid 2,1,1 \mid R \]
\[ j \mid 1,1,0 \mid R \]
\[ i \mid 0,1,0 \mid L \]
\[ k \mid 0,1,0 \mid L \]

run = \([a, b, c, e, f, g, h]\)

O. Bodini, A. Genitrini, F. Peschanski
Uniform random generation example

Motivations
Shuffle trees and their typical shape
Algorithms

search

2 ∈ \{1..2\}

\begin{align*}
d & \mid 2,1,1 \mid R \\
j & \mid 1,1,0 \mid R \\
i & \mid 0,1,0 \mid L \\
k & \mid 0,1,0 \mid L \\
\end{align*}

run = [a, b, c, e, f, g, h]
Uniform random generation example

take \{1..3\}

\[
\begin{align*}
d & | 2,1,1 | R \\
j & | 1,1,0 | R \\
i & | 0,1,0 | L \\
k & | 0,1,0 | L
\end{align*}
\]

run = \[ a, b, c, e, f, g, h, j \]
Uniform random generation example

run = \[a, b, c, e, f, g, h, j\]
Uniform random generation example

update
\{1..3\}

d \mid 1, 1, 1 \mid R

\epsilon \mid 1, 0, 0 \mid R

i \mid 0, 1, 0 \mid L

k \mid 0, 1, 0 \mid L

empty

run = [a, b, c, e, f, g, h, j]
Uniform random generation example

random choice; search

$3 \in \{1..3\}$

run = [a, b, c, e, f, g, h, j]

O. Bodini, A. Genitrini, F. Peschanski
Uniform random generation example

Motivations
Shuffle trees and their typical shape
Algorithms

search
1 ∈ \{1..1\}

d | 1, 1, 1 | R

\[\epsilon | 1, 0, 0 | R\]

\[i | 0, 1, 0 | L\]

\[k | 0, 1, 0 | L\]

eempty

run = [a, b, c, e, f, g, h, j]

O. Bodini, A. Genitrini, F. Peschanski

Concurrent Computations
Uniform random generation example

```
run = [a, b, c, e, f, g, h, j, i]
```
Uniform random generation example

run = [a, b, c, e, f, g, h, j, i]

O. Bodini, A. Genitrini, F. Peschanski
Concurrent Computations
Uniform random generation example

`run = [a, b, c, e, f, g, h, j, i]`
Uniform random generation example

random choice; search

$2 \in \{1..2\}$

$d \mid 1,1,0 \mid R$

$\epsilon \mid 1,0,0 \mid R$

$\epsilon \mid 0,0,0 \mid L$

$k \mid 0,1,0 \mid L$

empty

run = [a, b, c, e, f, g, h, j, i]
Uniform random generation example

take
\{1..1\}

run = [a, b, c, e, f, g, h, j, i, d]
Uniform random generation example

\[
\begin{align*}
\text{run} &= [a, b, c, e, f, g, h, j, i, d] \\
\end{align*}
\]
Uniform random generation example

random choice; search

$1 \in \{1..1\}$

$\epsilon \mid 1,0,0 \mid R$

$\epsilon \mid 0,0,0 \mid L$

$k \mid 0,1,0 \mid L$

$\epsilon \mid 1,0,0 \mid R$

empty

run = $[a, b, c, e, f, g, h, j, i, d]$
Uniform random generation example

Motivations

Shuffle trees and their typical shape

Algorithms

Uniform random generation example

search

$1 \in \{1..1\}$

$\epsilon | 1, 0, 0 | R$

$\epsilon | 0, 0, 0 | L$

$k | 0, 1, 0 | L$

$\epsilon | 1, 0, 0 | R$

empty

run = $[a, b, c, e, f, g, h, j, i, d]$
Uniform random generation example

\[ \text{run} = [a, b, c, e, f, g, h, j, i, d] \]
Uniform random generation example

run = [a, b, c, e, f, g, h, j, i, d, k]
Conclusion and perspectives

First step for the quantitative analysis of concurrent theory objects, . . .
Conclusion and perspectives

First step for the quantitative analysis of concurrent theory objects, ...

```
  a
   |
  b
 / \
c   d
  \
  e   f
```
Conclusion and perspectives

First step for the quantitative analysis of concurrent theory objects, ...
Conclusion and perspectives

First step for the quantitative analysis of concurrent theory objects, ...
Conclusion and perspectives

First step for the quantitative analysis of concurrent theory objects, ...

\[
\begin{aligned}
&\quad a \\
&\quad \quad b \\
&\quad \quad \quad + \\
&\quad \quad c \quad d \\
&\quad \quad \quad \quad e \quad f \\
&\quad \quad \quad \quad \quad \quad \quad a \\
&\quad \quad \quad \quad \quad \quad \quad \quad b \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad c \quad d \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad e \quad f \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad f \quad e
\end{aligned}
\]