Entropic Uniform Sampling of Linear Extensions in Series-Parallel Posets

Olivier Bodini[†], Matthieu Dien[‡],
Antoine Genitrini[‡] and Frédéric Peschanski[‡]

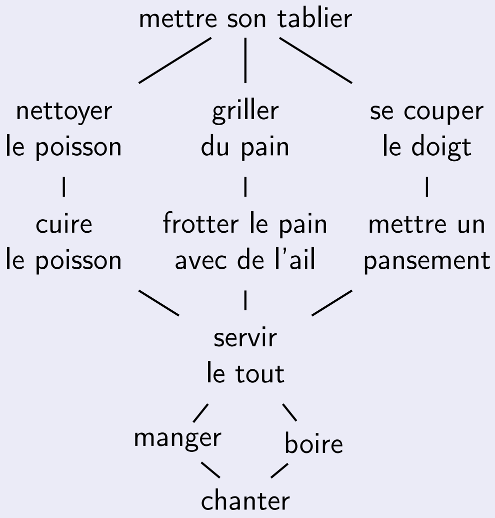
†: Université Paris XIII, LIPN, Villetaneuse
‡: Université Pierre et Marie Curie, LIP6, Paris

March 24th, 2017

Outline

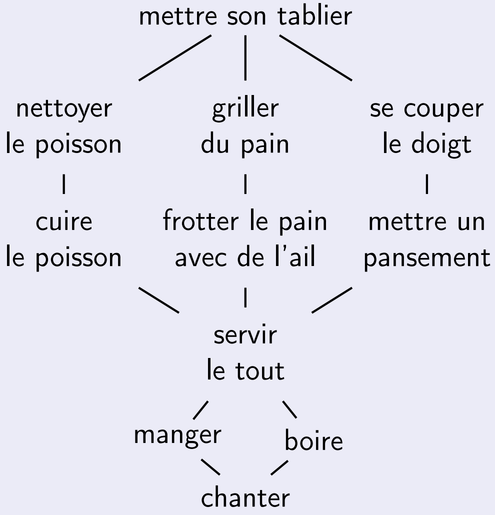
Problem

The *MakeBouillabaisse* program (poset)

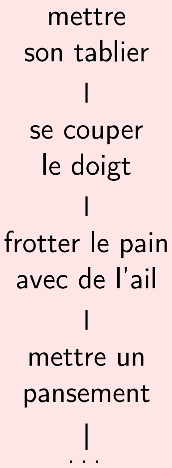


Problem

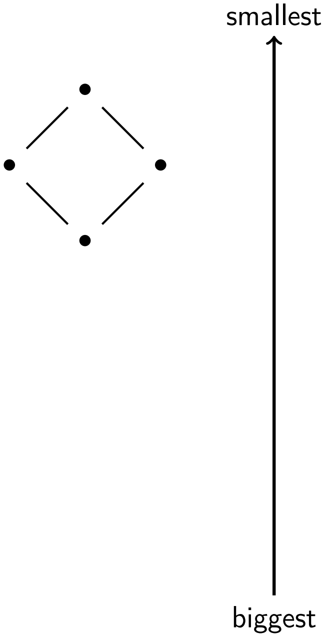
The *MakeBouillabaisse* program (poset)



A dangerous run (linear extension)



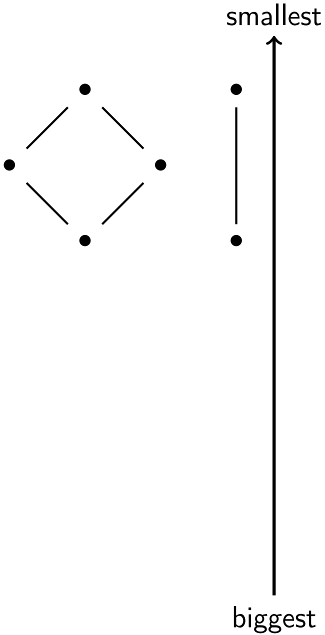
Series-Parallel posets



Definition

The smallest class of posets containing the singleton poset and closed under parallel

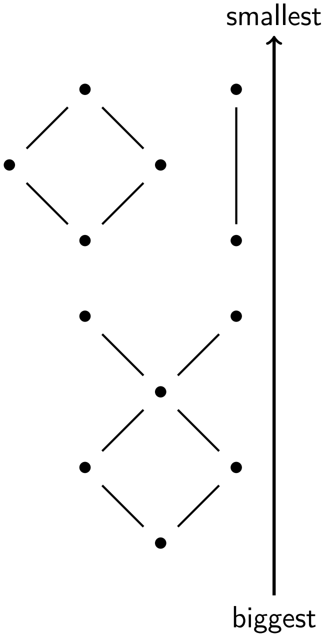
Series-Parallel posets



Definition

The smallest class of posets containing the singleton poset and closed under parallel

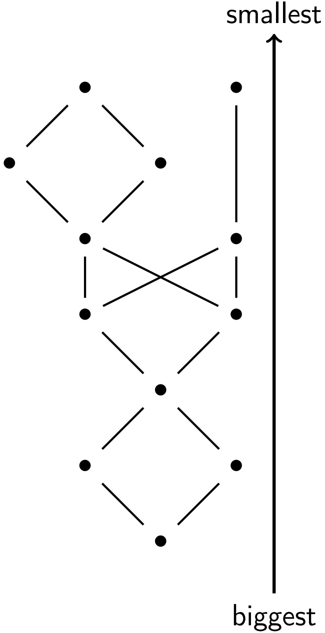
Series-Parallel posets



Definition

The smallest class of posets containing the singleton poset and closed under parallel and series composition.

Series-Parallel posets



Definition

The smallest class of posets containing the singleton poset and closed under parallel and series composition.

Posets to DAGs

Remark

- Posets are sets of order relations betweens elements

Posets to DAGs

Remark

- Posets are sets of order relations betweens elements $\Rightarrow$ but it is not a convenient way to represent it (from an algorithmic point of view)

Posets to DAGs

Remark

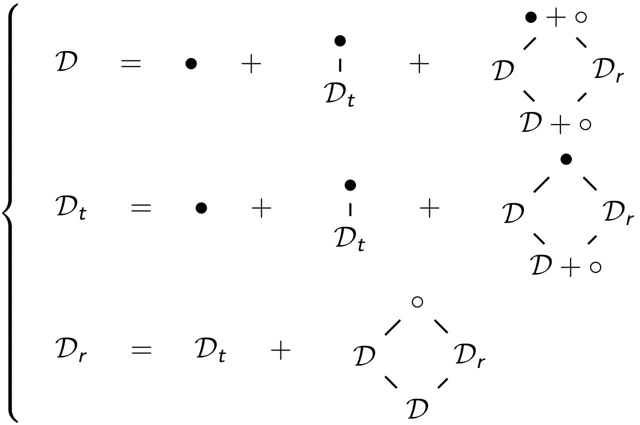
- Posets are sets of order relations betweens elements $\Rightarrow$ but it is not a convenient way to represent it (from an algorithmic point of view)
- Intransitive or covering DAGs (Directed Acyclic Graphs) are isomorphic to posets

Posets to DAGs

Remark

- Posets are sets of order relations betweens elements $\Rightarrow$ but it is not a convenient way to represent it (from an algorithmic point of view)
- Intransitive or covering DAGs (Directed Acyclic Graphs) are isomorphic to posets $\Rightarrow$ linear extensions of a poset are isomorphic to increasing labelings of its correponding DAG

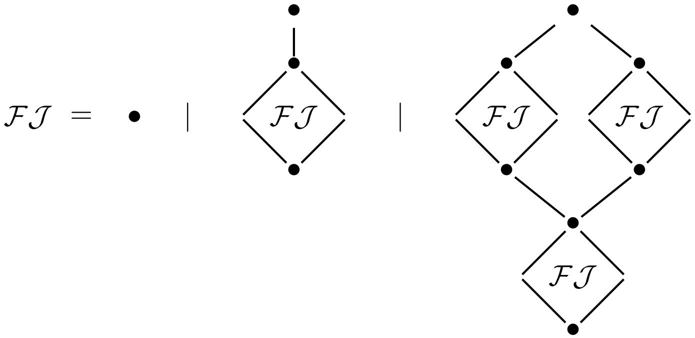
Fork-Join graphs



Bijection

The bicolored Fork-Join graphs $\mathcal{D}$ are in bijection with the combinatorial embeddings of covering DAGs of Series Parallel posets.

Algorithm



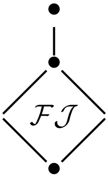
$\mathcal{F}\mathcal{J}$ =
$\mathcal{F}\mathcal{J}$
$\mathcal{F}\mathcal{J}$
$\mathcal{F}\mathcal{J}$
$\mathcal{F}\mathcal{J}$

Algorithm

partial order	$\Rightarrow$	linear extension
$\bullet$	$\Rightarrow$	return (1)

Algorithm

partial order $\Rightarrow$ linear extension

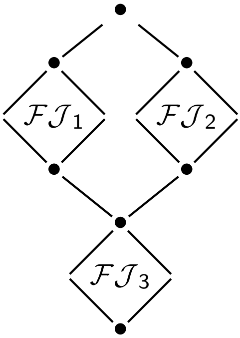


$\Rightarrow$

$(x_2, \ldots x_n) :=$ draw_lin_ext($\mathcal{FJ}$)
return $(1, x_2 + 1, \ldots, x_n + 1)$

Algorithm

partial order $\Rightarrow$ linear extension



$\Rightarrow$

$\mathbf{x} := \text{draw\_lin\_ext}(\mathcal{F}\mathcal{J}_1)$
$\mathbf{y} := \text{draw\_lin\_ext}(\mathcal{F}\mathcal{J}_2)$
$\mathbf{z} := \text{draw\_lin\_ext}(\mathcal{F}\mathcal{J}_3)$
$\mathbf{t} := \text{Shuffle}(\mathbf{x}, \mathbf{y}) \qquad |\mathbf{t}| = |\mathbf{x}| + |\mathbf{y}|$
return $(1,\ \mathbf{t} + \mathbb{1},\ \mathbf{z} + \mathbb{1} \cdot (|\mathbf{t}| + 1))$

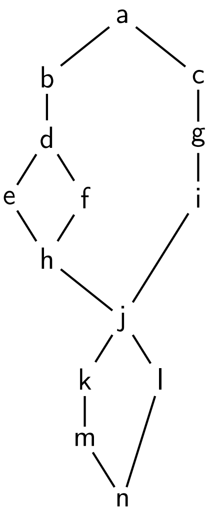
Average Complexity

The average complexity of the algorithm `draw_lin_ext` in memory writes is $\mathcal{O}(n\sqrt{n})$.

Example

draw_lin_ext(P)

$P =$

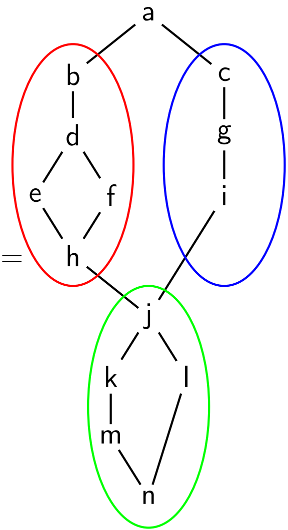


Example

draw_lin_ext(P) $\rightarrow$

- draw_lin_ext(●)
- draw_lin_ext(●)
- draw_lin_ext(●)

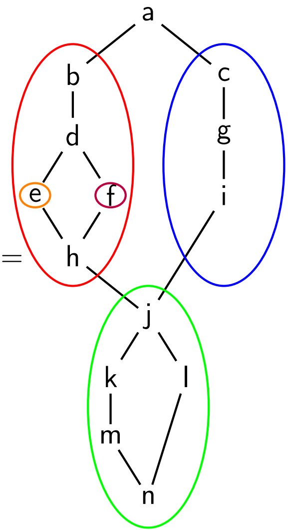
$P =$



Example

draw_lin_ext(P) $\rightarrow$

- draw_lin_ext(●)
  - draw_lin_ext(●)
  - draw_lin_ext(●)
- draw_lin_ext(●)
- draw_lin_ext(●)

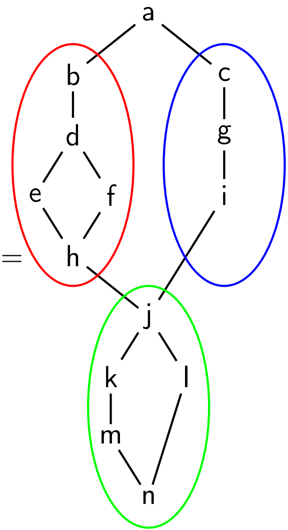
$P =$ 

Example

draw_lin_ext(P) $\rightarrow$

- draw_lin_ext(●)
  $\rightarrow$ [b, d, e, f, h]
- draw_lin_ext(●)
- draw_lin_ext(●)

$P =$

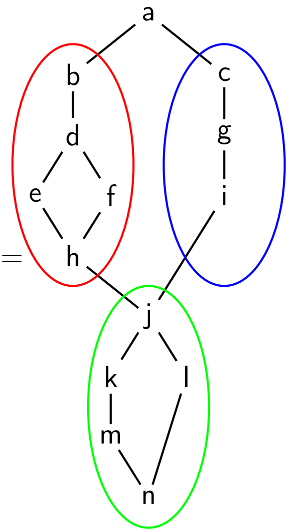


Example

draw_lin_ext(P) $\rightarrow$

- draw_lin_ext(●) $\rightarrow$ [b, d, e, f, h]
- draw_lin_ext(●) $\rightarrow$ [c, g, i]
- draw_lin_ext(●) $\rightarrow$ [k, l, m]

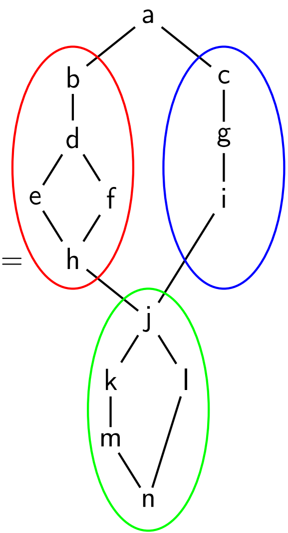
$P =$



Example

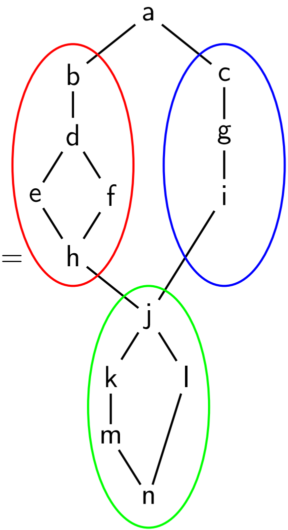
draw_lin_ext(P) $\rightarrow$

- Shuffle(●, ●)
  $\rightarrow$ [b, d, c, e, g, i, f, h]
- draw_lin_ext(●)
  $\rightarrow$ [k, l, m]

$P =$ 

Example

draw_lin_ext(P) $\rightarrow$
$\Rightarrow$ [a, b, d, c, e, g, i, f, h, k, l, m]

$P =$ 

Shuffle

Algorithm 1 Algorithm of uniform random shuffling

```
function Shuffle(ℓ, r)
    t := [ ]
    v := RandomCombination(|ℓ|, |r|)
    for all e ∈ v do
        if e then
            append pop(ℓ) to t
        else
            append pop(r) to t
    return t
```

RandomCombination(a, b) samples a combination of a `True` and b `False`.

Entropy

Definition

Let A be an algorithm sampling an element of a finite set S at random accoding to a probability distribution μ. We say that A is *entropic* if the average number of random bits n_e it uses to sample one element $e \in S$ is proportionnal to the entropy of μ, in the sense of Shannon's entropy:

$$\exists K > 0, \forall e \in S, n_e \leq K \cdot \sum_{x \in S} -\mu(x) \log_2(\mu(x))$$

RandomCombination

Algorithm 2 Algorithm of uniform random generation of combination

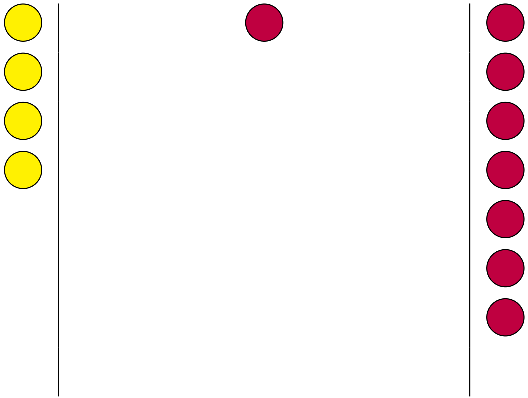
```
function RandomCombination(p, q)                          ▷ Suppose q < p
    ℓ := [ ]
    if q > log(p)^2 then
        while ♯True <= p ∧ ♯False <= q do
            if Bernoulli(p/(p+q)) then ℓ := cons(True, ℓ)
            else  ℓ := cons(False, ℓ)
        remaining := ¬pop(ℓ)
    else
        ℓ := a list of q times False
        remaining := True
    for i := ♯True + ♯False − 1 to p + q − 1 do
        j := uniformRandomInt[0 . . . i]
        insert remaining at position j in ℓ
    return ℓ
```

Example

RandomCombination(4, 8)

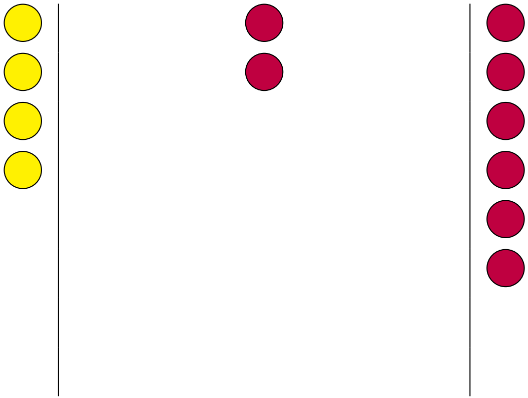
Example

Bernoulli($\frac{2}{3}$) $\Rightarrow$ Head



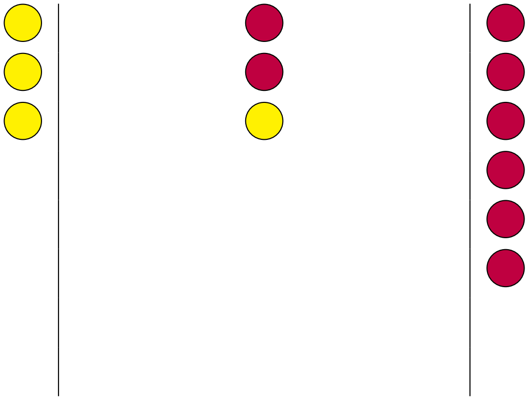
Example

Bernoulli($\frac{2}{3}$) $\Rightarrow$ Head



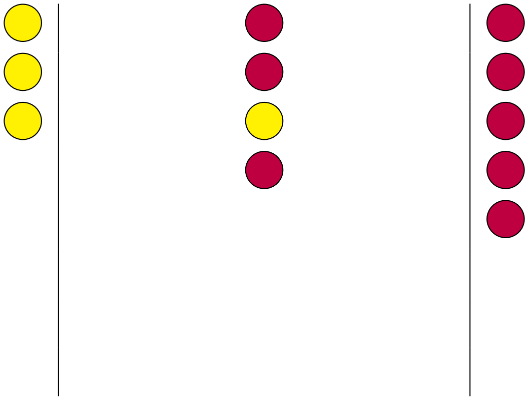
Example

Bernoulli($\frac{2}{3}$) $\Rightarrow$ Tail



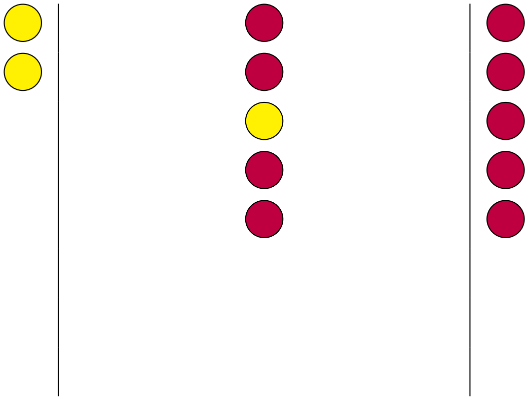
Example

Bernoulli$(\frac{2}{3}) \Rightarrow$ Head



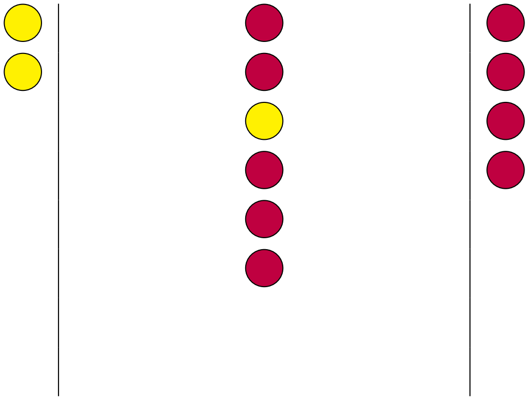
Example

Bernoulli($\frac{2}{3}$) $\Rightarrow$ Head



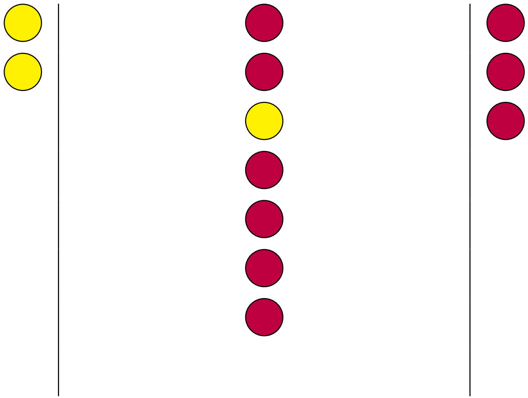
Example

$\text{Bernoulli}(\frac{2}{3}) \Rightarrow$ Head



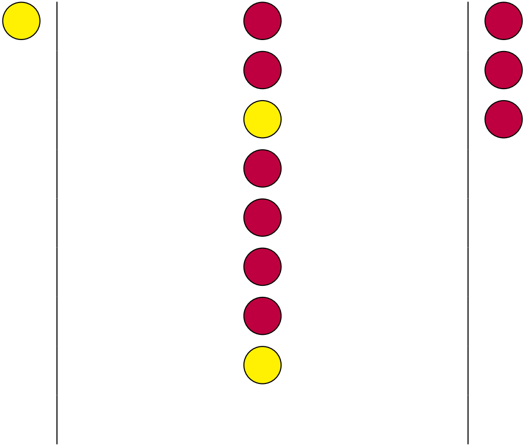
Example

Bernoulli$(\frac{2}{3}) \Rightarrow$ Head



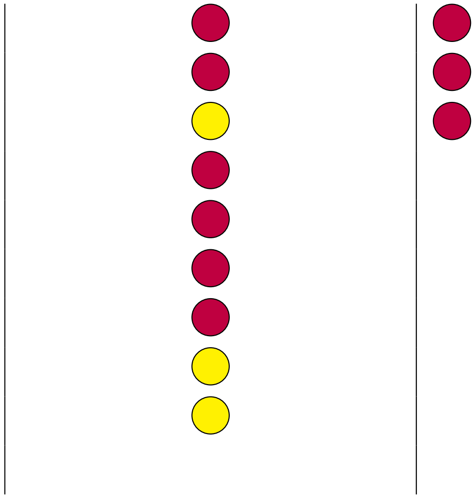
Example

$\text{Bernoulli}(\frac{2}{3}) \Rightarrow$ Tail



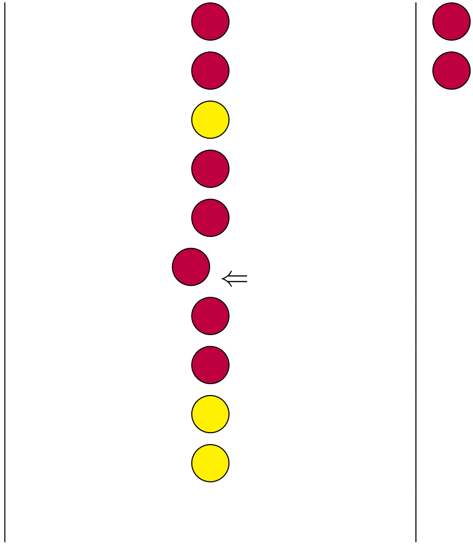
Example

$\text{Bernoulli}(\frac{2}{3}) \Rightarrow$ Tail



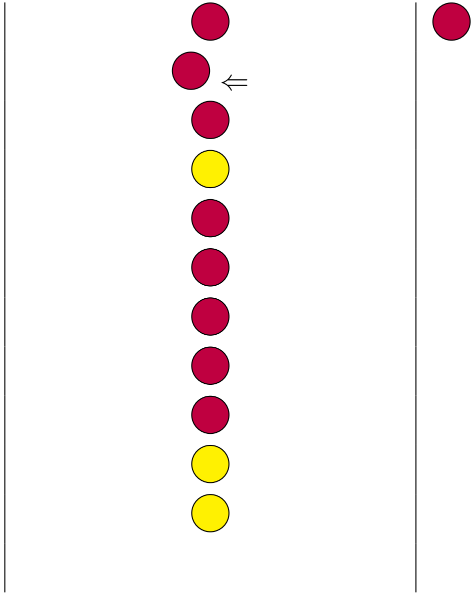
Example

$\text{Uniform}([0,9]) \Rightarrow\ 5$



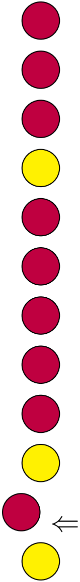
Example

$\text{Uniform}([0,10]) \Rightarrow\ 1$



Example

$$\text{Uniform}([0,11]) \Rightarrow \ 10$$



Bernoulli sampling

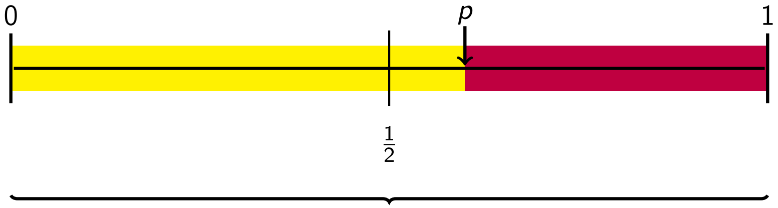
Algorithm 3 Sampling of Bernoulli random variable

```
function Bernoulli(p)                                ▷ p is less than 1
    function RecBernoulli(a, b, p)
        if RandomBit() = 0 then
            if m ≥ p then return False
            else RecBernoulli((a+b)/2, b, p)
        else
            if m < p then return True
            else RecBernoulli(a, (a+b)/2, p)
    return RecBernoulli(0, 1, p)
```

Complexity

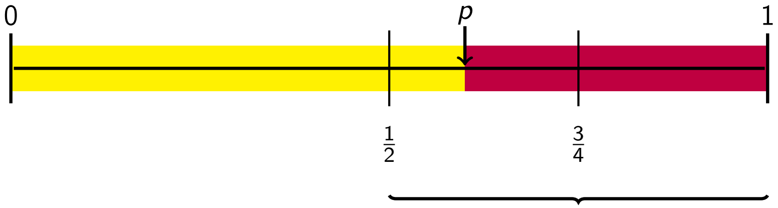
The algorithm uses 2 random bits in average.

Example



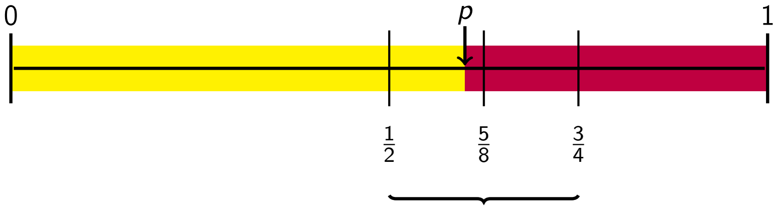
Example

Toss a fair coin → Head



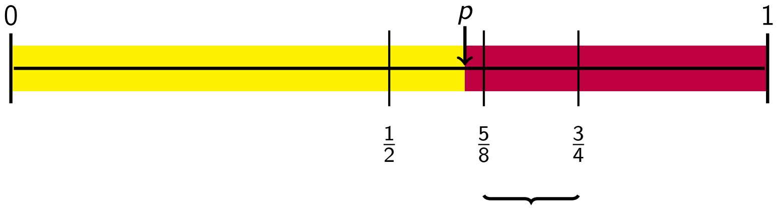
Example

Toss a fair coin $\rightarrow$ Head
Toss a fair coin $\rightarrow$ Tail



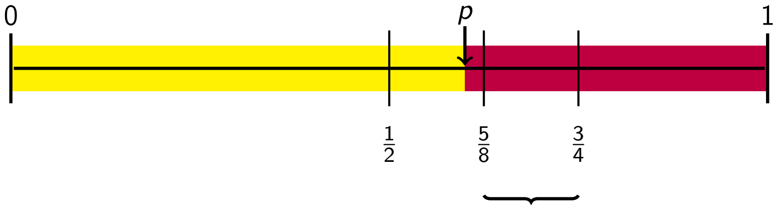
Example

Toss a fair coin → Head
Toss a fair coin → Tail
Toss a fair coin → Head



Example

Toss a fair coin → Head
Toss a fair coin → Tail
Toss a fair coin → Head



The drawn color is **purple**.

k-Bernoulli sampling

Algorithm 4 Sampling of k Bernoulli random variables

function k-Bernoulli(p) ▷ p is less than 1

  function k-BernoulliAux(p)

    $k := \left\lfloor \frac{\log \frac{1}{2}}{\log p} \right\rfloor, \; i := 0$

    while $\neg$Bernoulli($\sum_{\ell=0}^{i} \binom{k}{\ell} p^{k-\ell}(1-p)^{\ell}$) **do** $i := i + 1$

    $v :=$ a vector of $k - i$ times `True`

    for $\ell = 0$ to i **do**

      $j :=$ **uniformRandomInt**($[0 \ldots k - i]$)

      insert `False` at position j

    return v

  if $p < \frac{1}{2}$ **then return negate**(k-BernoulliAux($1 - p$))

  else return k-BernoulliAux(p)

Theorem: k-Bernoulli is entropic.

Example

We want to draws Bernoulli r.v. of parameter $p := 0.9$

- $k := \left\lfloor \frac{\log \frac{1}{2}}{\log 0.9} \right\rfloor = 6$

Example

We want to draws Bernoulli r.v. of parameter $p := 0.9$

- $k := \left\lfloor \frac{\log \frac{1}{2}}{\log 0.9} \right\rfloor = 6$
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k \simeq 0.53$. It fails $\Rightarrow$ at least **one** of the k experiences fail

Example

We want to draws Bernoulli r.v. of parameter $p := 0.9$

- $k := \left\lfloor \frac{\log \frac{1}{2}}{\log 0.9} \right\rfloor = 6$
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k \simeq 0.53$. It fails $\Rightarrow$ at least **one** of the k experiences fail
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k + k \cdot p^{k-1}(1-p) \simeq 0.89$. It fails $\Rightarrow$ at least **two** of the k experiences fail

Example

We want to draws Bernoulli r.v. of parameter $p := 0.9$

- $k := \left\lfloor \frac{\log \frac{1}{2}}{\log 0.9} \right\rfloor = 6$
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k \simeq 0.53$. It fails $\Rightarrow$ at least **one** of the k experiences fail
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k + k \cdot p^{k-1}(1-p) \simeq 0.89$. It fails $\Rightarrow$ at least **two** of the k experiences fail
- use Bernoulli algorithm to draw a Bernoulli r.v. of parameter $p^k + k \cdot p^{k-1}(1-p) + \binom{k}{2} \cdot p^{k-2}(1-p)^2 \simeq 0.98$. Success ! $\Rightarrow$ uniformly insert two "Fail" among four "Success": [Fail, Success, Success, Fail, Success, Success]

RandomCombination optimality

Facts

A Bernoullli r.v. of parameter $\frac{p}{p+q}$ has entropy

$$B_{p,q} = -p \log\left(\frac{p}{p+q}\right) - q\left(\frac{q}{p+q}\right).$$

A uniform r.v. over the set of p, q-combinations has entropy

$$E_{p,q} = (p+q)\ln(p+q) - p\ln(p) - q\ln(q).$$

RandomCombination optimality

Facts

A Bernoullli r.v. of parameter $\frac{p}{p+q}$ has entropy

$$B_{p,q} = -p \log\left(\frac{p}{p+q}\right) - q\left(\frac{q}{p+q}\right).$$

A uniform r.v. over the set of p, q-combinations has entropy

$$E_{p,q} = (p+q)\ln(p+q) - p\ln(p) - q\ln(q).$$

Theorem

The algorithm RandomCombination(p, q) uniformy samples a list of p `True` and q `False` using

$$k \cdot B_{p,q} \sim E_{p,q} \text{ random bits,}$$

when p and q tends to ∞, in average. It is entropic.

Proof sketch

- if q is very small ($q < \log(p+q)^{2+\epsilon}$):
- the algorithm inserts uniformly q `False`

$\Rightarrow$ it uses $q \cdot \log(p+q) \sim E_{p,q}$ random bits

- else, let T be the r.v. of the number of random bits used
$$\mathbb{P}(T = t) = \binom{t}{p} \left(\frac{p}{p+q}\right)^{p+1} \left(\frac{q}{p+q}\right)^{t-p} + \binom{t}{q} \left(\frac{p}{p+q}\right)^{t-q} \left(\frac{q}{p+q}\right)^{q+1}$$
- prove the convergence in law of T to the sum of two half-Gaussian laws
- compute the expectation $\mathbb{E}[T]$:

$$\mathbb{E}(T) \sim p + q - \frac{(p+q)^{3/2}}{\sqrt{2pq\pi}},$$

when p and q tends to infinity.

$\Rightarrow$ $(p+q) - \frac{(p+q)^{3/2}}{\sqrt{2pq\pi}}$ Bernoulli r.v. are drawn using $B_{p,q}$ random bits (single calls to Bernoulli algorithm) and $\frac{(p+q)^{3/2}}{\sqrt{2pq\pi}}$ are drawn using uniform insertions, so a negligible number of bits.

Merci !

(La dorade était très bonne aussi)