Chaînes de Markov, Génération Aléatoire, et Couplage Depuis le Passé (10)

Petite École de Combinatoire

P. Duchon

LaBRI

Résumé (ultra-)rapide

- Génération aléatoire exacte par couplage venu du passé (CFTP): $G_n(x)$ si G_n est constante

Résumé (ultra-)rapide

- Génération aléatoire exacte par couplage venu du passé (CFTP): $G_n(x)$ si G_n est constante
- Réalisation par composition de fonctions aléatoires :

$$G_n = F_{-1} \circ F_{-2} \circ \ldots \circ F_{-n}$$

Résumé (ultra-)rapide

- Génération aléatoire exacte par couplage venu du passé (CFTP): $G_n(x)$ si G_n est constante
- Réalisation par composition de fonctions aléatoires :

$$G_n \quad = \quad F_{-1} \circ F_{-2} \circ \ldots \circ F_{-n}$$

- Difficulté : cela implique de composer les fonctions F_i dans l'ordre inverse (appliquer la plus récemment choisie d'abord), et donc on est *a priori* obligé de se souvenir des fonctions F_i déjà utilisées (ou des tirages aléatoires effectués, ou des graines de générateurs pseudo-aléatoires).

Résumé (ultra-)rapide

- Génération aléatoire exacte par couplage venu du passé (CFTP): $G_n(x)$ si G_n est constante
- Réalisation par composition de fonctions aléatoires :

$$G_n \quad = \quad F_{-1} \circ F_{-2} \circ \ldots \circ F_{-n}$$

- Difficulté : cela implique de composer les fonctions F_i dans l'ordre inverse (appliquer la plus récemment choisie d'abord), et donc on est *a priori* obligé de se souvenir des fonctions F_i déjà utilisées (ou des tirages aléatoires effectués, ou des graines de générateurs pseudo-aléatoires).
- Conséquence : sans utiliser une grande quantité de mémoire, on ne peut pas utiliser un “vrai” générateur aléatoire à lecture unique.

CFTP classique

- $F \leftarrow$ FonctionIdentité
- Tant que on n'est pas sûr que Image(F) est un singleton
  - $F \leftarrow F \circ$ FonctionAléatoire$()$
- Retourner l'unique valeur prise par F

CFTP classique

- $F \leftarrow$ FonctionIdentité
- Tant que on n'est pas sûr que Image(F) est un singleton
  - $F \leftarrow F \circ$ FonctionAléatoire$()$
- Retourner l'unique valeur prise par F

Contrainte de simulation de la chaîne de Markov :

$$\mathbb{P}(\text{FonctionAléatoire}()(u) = v) = p_{u,v}$$

CFTP classique

- $F \leftarrow$ FonctionIdentité
- Tant que on n'est pas sûr que Image(F) est un singleton
  - $F \leftarrow F \circ$ FonctionAléatoire$()$
- Retourner l'unique valeur prise par F

Contrainte de simulation de la chaîne de Markov :

$$\mathbb{P}(\text{FonctionAléatoire}()(u) = v) = p_{u,v}$$

Il n'est pas indispensable de s'arrêter dès que la fonction F devient constante : on peut se contenter d'essayer de détecter la coalescence.

Couplage venu du passé à lecture unique

[Wilson, 00] Read-once CFTP
Hypothèses sur l'utilisation du CFTP :

Couplage venu du passé à lecture unique

[Wilson, 00] Read-once CFTP
Hypothèses sur l'utilisation du CFTP :

- les fonctions de mise à jour sont indépendantes, de même loi;

Couplage venu du passé à lecture unique

[Wilson, 00] Read-once CFTP
Hypothèses sur l'utilisation du CFTP :

- les fonctions de mise à jour sont indépendantes, de même loi;
- le résultat du calcul est l'objet complet, avec précision absolue, après composition d'un nombre fini de fonctions de mise à jour;

Couplage venu du passé à lecture unique

[Wilson, 00] Read-once CFTP
Hypothèses sur l'utilisation du CFTP :

- les fonctions de mise à jour sont indépendantes, de même loi;
- le résultat du calcul est l'objet complet, avec précision absolue, après composition d'un nombre fini de fonctions de mise à jour;
- le fait d'évaluer les fonctions de mise à jour en un point quelconque, n'affecte pas la détection de coalescence.

Couplage venu du passé à lecture unique

[Wilson, 00] Read-once CFTP
Hypothèses sur l'utilisation du CFTP :

- les fonctions de mise à jour sont indépendantes, de même loi;
- le résultat du calcul est l'objet complet, avec précision absolue, après composition d'un nombre fini de fonctions de mise à jour;
- le fait d'évaluer les fonctions de mise à jour en un point quelconque, n'affecte pas la détection de coalescence.

Alors, on peut transformer l'algorithme de CFTP en RO-CFTP; RO-CFTP ne compose les fonctions que vers le futur, n'a pas besoin de les stocker, et a une complexité du même ordre de grandeur.

AppliquerFonctionAléatoire

La fonction AppliquerFonctionAléatoire(u) :

- $F \leftarrow$ NouvelleFonctionAléatoire$()$ (tire une fonction aléatoire F, selon une “bonne” loi)
- essaie de déterminer si F est ou non constante (officiellement coalescente);
- calcule $F(u)$;
- renvoie (OfficiellementCoalescente$(F), F(u), F)$

AppliquerFonctionAléatoire

La fonction AppliquerFonctionAléatoire(u) :

- $F \leftarrow$ NouvelleFonctionAléatoire$()$ (tire une fonction aléatoire F, selon une "bonne" loi)
- essaie de déterminer si F est ou non constante (officiellement coalescente);
- calcule $F(u)$;
- renvoie $($OfficiellementCoalescente$(F), F(u), F)$

Les contraintes sur la loi de NouvelleFonctionAléatoire$()$:

AppliquerFonctionAléatoire

La fonction AppliquerFonctionAléatoire(u) :

- $F \leftarrow$ NouvelleFonctionAléatoire$()$ (tire une fonction aléatoire F, selon une “bonne” loi)
- essaie de déterminer si F est ou non constante (officiellement coalescente);
- calcule $F(u)$;
- renvoie $($OfficiellementCoalescente$(F), F(u), F)$

Les contraintes sur la loi de NouvelleFonctionAléatoire$()$:

- simulation de la chaîne de Markov : $\mathbb{P}(F(u) = v) = p_{u,v}$;

AppliquerFonctionAléatoire

La fonction AppliquerFonctionAléatoire(u) :

- $F \leftarrow$ NouvelleFonctionAléatoire$()$ (tire une fonction aléatoire F, selon une “bonne” loi)
- essaie de déterminer si F est ou non constante (officiellement coalescente);
- calcule $F(u)$;
- renvoie (OfficiellementCoalescente$(F), F(u), F)$

Les contraintes sur la loi de NouvelleFonctionAléatoire$()$:

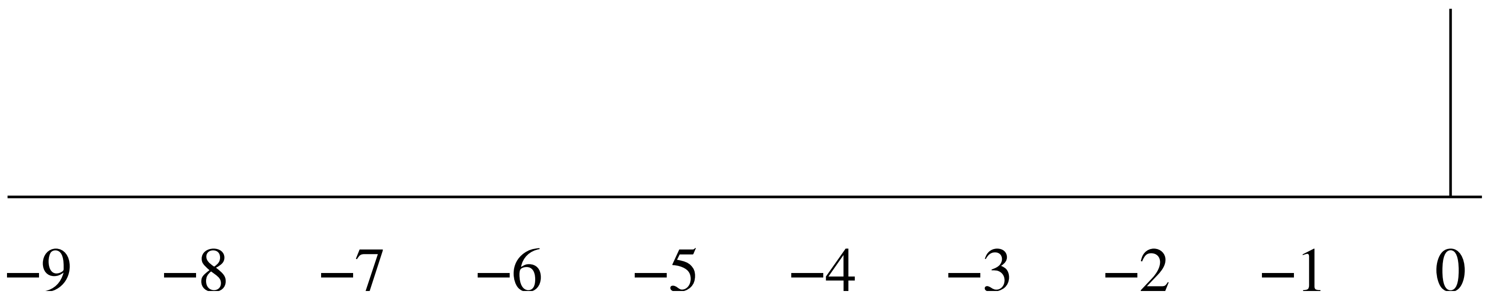
- simulation de la chaîne de Markov : $\mathbb{P}(F(u) = v) = p_{u,v}$;
- les fonctions renvoyées par NouvelleFonctionAleatoire$()$ ont une probabilité raisonnable d'être officiellement coalescentes (par exemple, au moins $1/2$)

Méga-CFTP

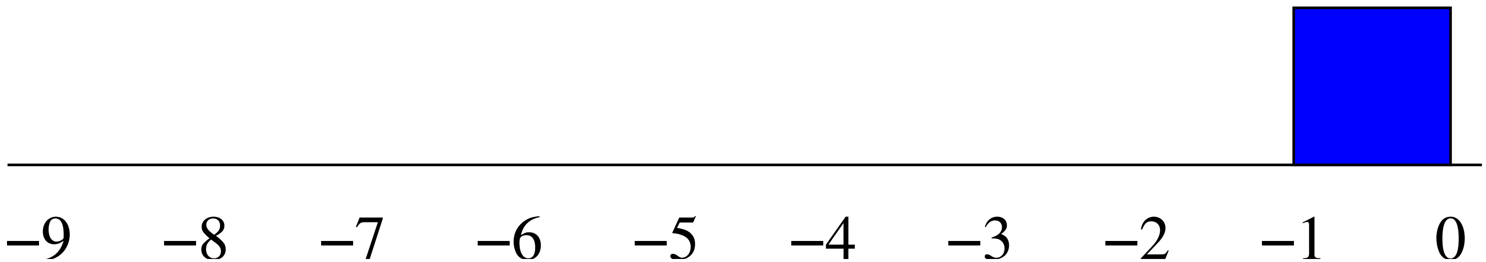
La fonction AppliquerFonctionAléatoire$()$ permet de faire du CFTP : on va engendrer une suite de fonctions et les composer en ordre inverse, jusqu'à être sûr que la composée soit constante.
En particulier, dès que l'une des fonctions est officiellement coalescente, la composée est constante et on peut s'arrêter.

- $(b, u, F) \leftarrow ($Faux$, u_0,$ Identité$)$;
- Tant que b est faux
  - $(b, u, G) \leftarrow$ AppliquerFonctionAléatoire(u);
  - $F \leftarrow F \circ G$;
- Retourner $F(u_0)$

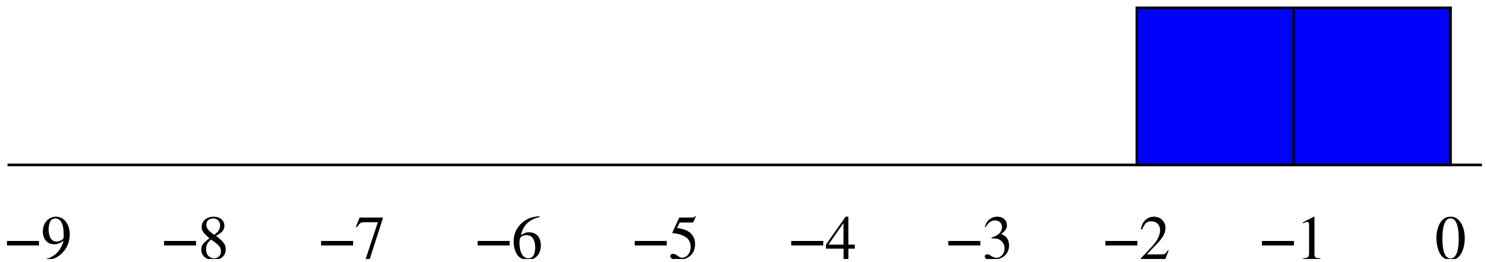
Schématiquement. . .



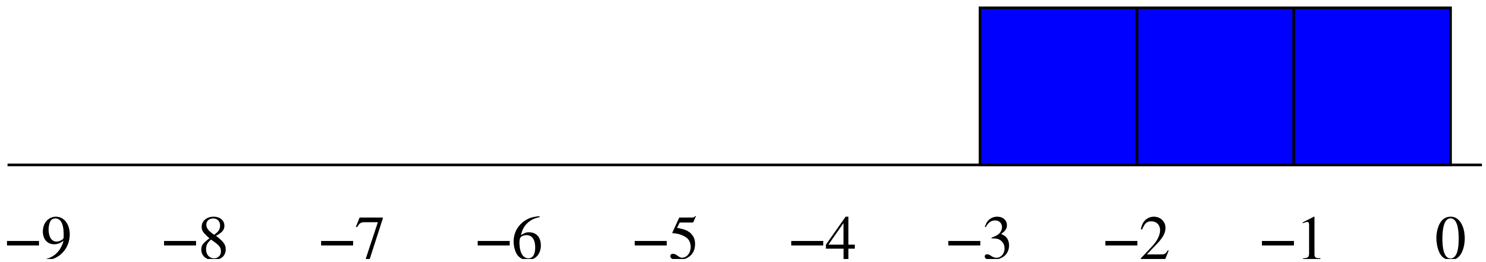
Schématiquement…



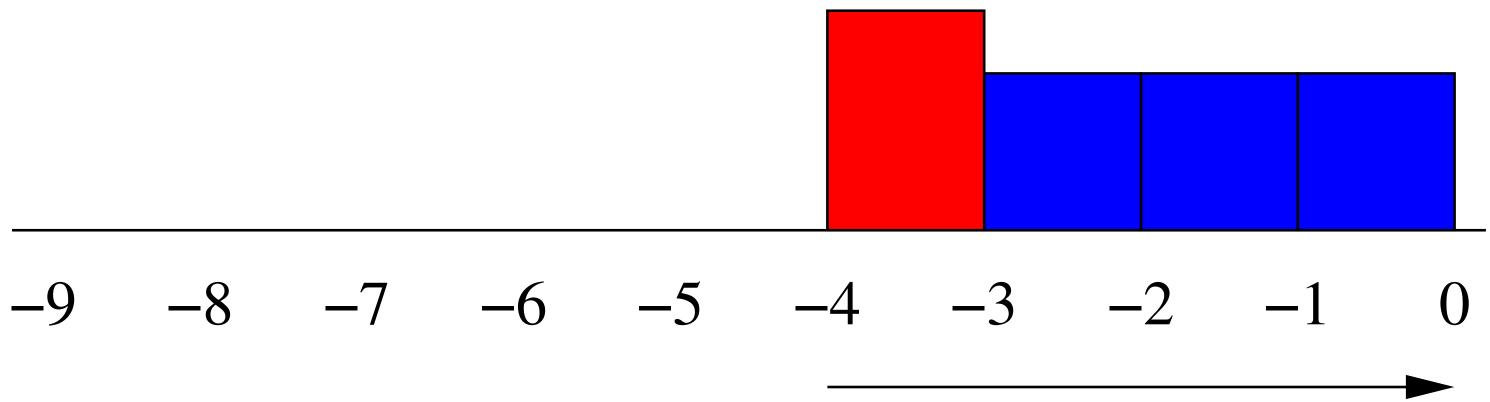
Schématiquement. . .



Schématiquement. . .

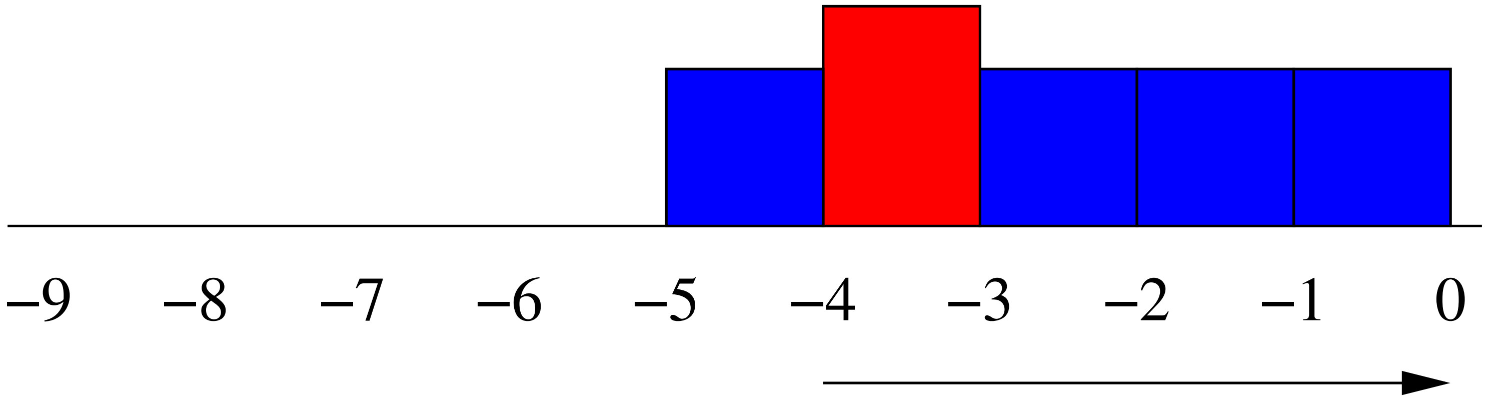


Schématiquement...



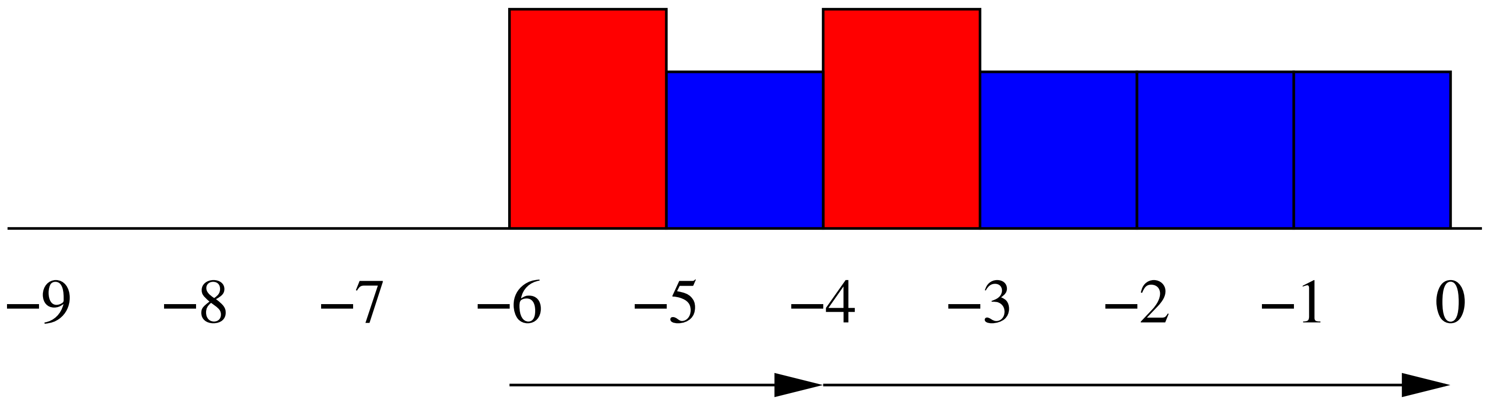
- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;

Schématiquement…



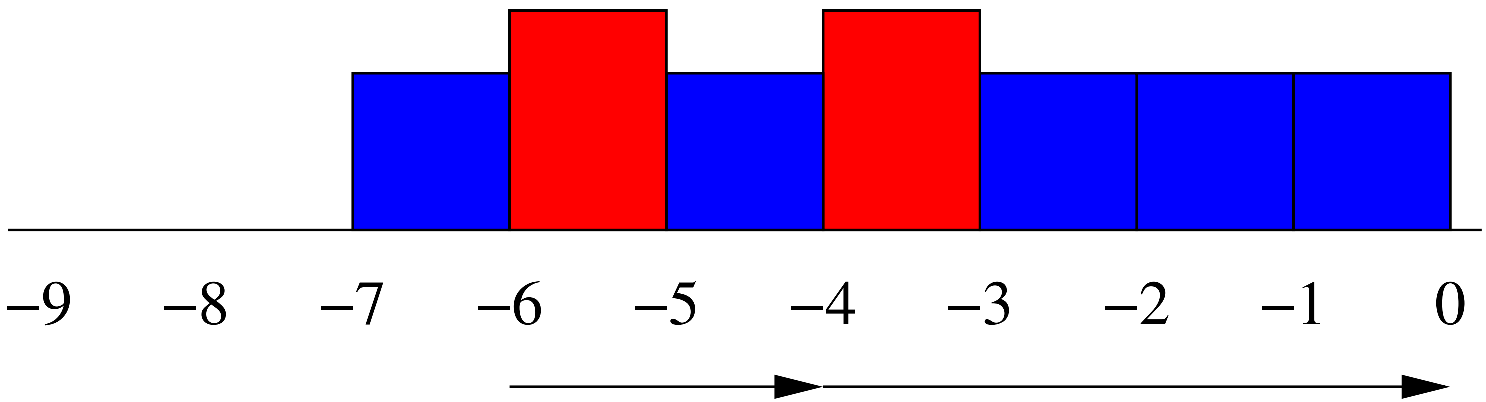
- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;

Schématiquement…



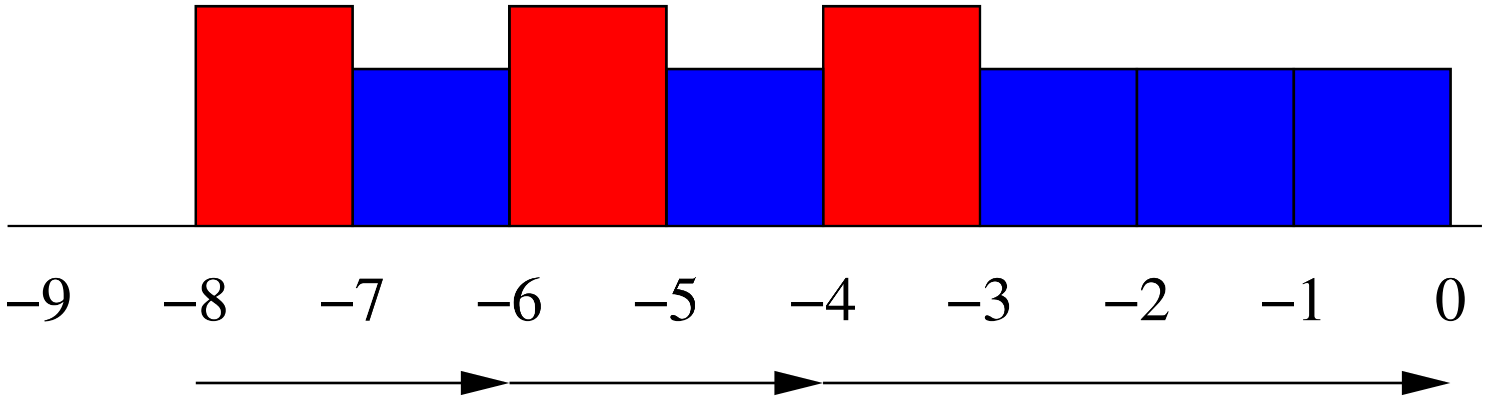
- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;
- $U_2 = F_{-4} \circ F_{-5} \circ F_{-6}()$;

Schématiquement. . .



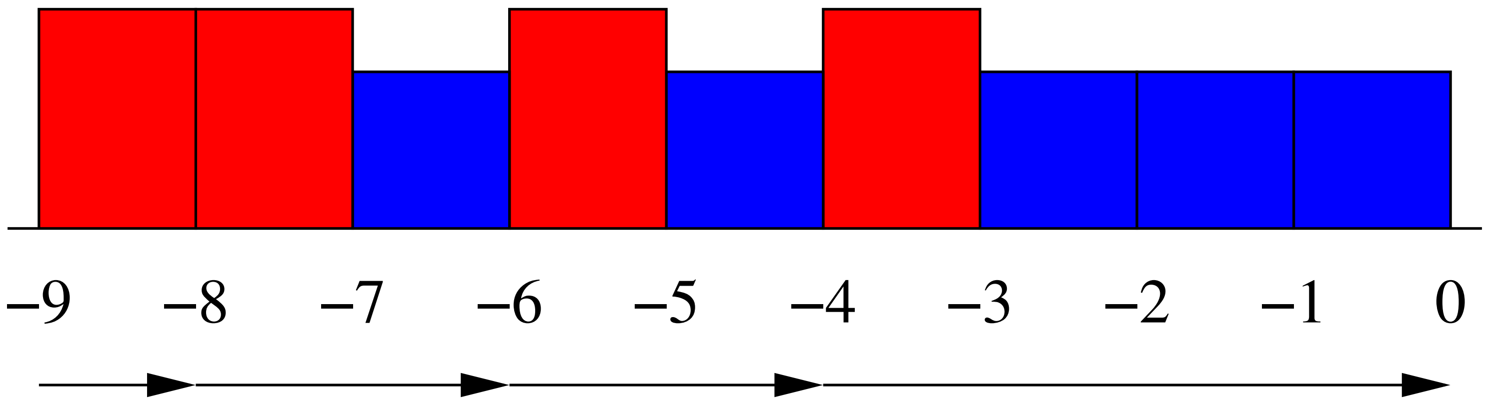
- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;
- $U_2 = F_{-4} \circ F_{-5} \circ F_{-6}()$;

Schématiquement...



- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;
- $U_2 = F_{-4} \circ F_{-5} \circ F_{-6}()$;
- $U_3 = F_{-7} \circ F_{-8}()$;

Schématiquement...



- $U_1 = F_{-1} \circ F_{-2} \circ F_{-3} \circ F_{-4}()$;
- $U_2 = F_{-4} \circ F_{-5} \circ F_{-6}()$;
- $U_3 = F_{-7} \circ F_{-8}()$;
- $U_4 = F_{-9}()$.

Vers le futur!

On peut réaliser la même chose en composant vers le futur :

- Tirer des fonctions jusqu'à en trouver une qui soit officiellement coalescente; retenir son image v;
- Répéter
  - $u \leftarrow v$;
  - $(b, v, .) \leftarrow$ AppliquerFonctionAléatoire(u)

  tant que b est faux
- Retourner u.

(On compose une fonction officiellement coalescente, avec toutes les fonctions non officiellement coalescentes qui la suivent)

Obtention de k échantillons

Lors de l'obtention d'un échantillon, la dernière fonction tirée (officiellement coalescente) n'est pas utilisée pour la génération aléatoire (sauf pour indiquer la fin de la séquence); on peut donc la réutiliser pour l'échantillon suivant si on veut plusieurs échantillons et non un seul (on économise l'étape initiale).

NouvelleFonctionAléatoire?

Comment, à partir de FonctionAléatoire$()$, peut-on construire NouvelleFonctionAléatoire$()$ (avec la contrainte que la probabilité d'être officiellement coalescente soit strictement positive) ?

NouvelleFonctionAléatoire?

Comment, à partir de FonctionAléatoire$()$, peut-on construire NouvelleFonctionAléatoire$()$ (avec la contrainte que la probabilité d'être officiellement coalescente soit strictement positive) ?

Solution : composer deux séries de FonctionAléatoire$()$, indépendantes, vers le futur, jusqu'à ce que la première soit (officiellement) coalescente, et retourner la deuxième. . .

NouvelleFonctionAléatoire?

Comment, à partir de FonctionAléatoire$()$, peut-on construire NouvelleFonctionAléatoire$()$ (avec la contrainte que la probabilité d'être officiellement coalescente soit strictement positive) ?

Solution : composer deux séries de FonctionAléatoire$()$, indépendantes, vers le futur, jusqu'à ce que la première soit (officiellement) coalescente, et retourner la deuxième. . .

Les deux séries étant indépendantes, la propriété de "simulation de la chaîne" de la deuxième série, se conserve pour la composition arrêtée à un temps qui ne dépend que de la première série.

NouvelleFonctionAléatoire?

Comment, à partir de FonctionAléatoire$()$, peut-on construire NouvelleFonctionAléatoire$()$ (avec la contrainte que la probabilité d'être officiellement coalescente soit strictement positive) ?

Solution : composer deux séries de FonctionAléatoire$()$, indépendantes, vers le futur, jusqu'à ce que la première soit (officiellement) coalescente, et retourner la deuxième...

Les deux séries étant indépendantes, la propriété de "simulation de la chaîne" de la deuxième série, se conserve pour la composition arrêtée à un temps qui ne dépend que de la première série.

(Chacune des séries a autant de chances que l'autre de converger la première, donc la probabilité que la deuxième série soit officiellement coalescente est d'au moins $1/2$)

NouvelleFonctionAléatoire!

- $F \leftarrow$ Identité, $G \leftarrow$ Identité
- Tant que G n'est pas officiellement coalescente
  - $G \leftarrow$ FonctionAléatoire$() \circ G$
  - $F \leftarrow$ FonctionAléatoire$() \circ F$
- $b \leftarrow$ OfficiellementCoalescente(F)
- Retourner (b, u, F)

NouvelleFonctionAléatoire!

Version monotone :

- $m_G \leftarrow \min$, $M_G \leftarrow \max$, $m_F \leftarrow \min$, $M_F \leftarrow \max$
- Tant que $m_G < M_G$
  - $G \leftarrow$ FonctionAléatoire(), $m_G \leftarrow G(m_G)$, $M_G \leftarrow G(M_G)$
  - $F \leftarrow$ FonctionAléatoire(), $u \leftarrow F(u)$, $m_F \leftarrow F(m_F)$, $M_F \leftarrow F(M_F)$
- $b \leftarrow (m_F = M_F)$
- Retourner (b, u)

Note : Dans la version où on ne compose que vers le futur, il devient inutile que AppliquerFonctionAléatoire() renvoie la fonction

Complexité…

Soit T l'espérance du nombre de fonctions FonctionAléatoire$()$ à composer pour en obtenir une qui soit (officiellement) coalescente.

- Le nombre moyen d'appels à FonctionAléatoire$()$ de AppliquerFonctionAléatoire$()$ est $2T$;
- Le nombre moyen d'appels à AppliquerFonctionAléatoire$()$ pour obtenir un échantillon est 3 pour le premier échantillon, 2 pour chaque suivant.
- On a *a priori* un facteur entre 4 et 6 avec le CFTP, mais le binary-backoff CFTP (où on double la longueur des runs jusqu'à avoir coalescence) peut être amené à faire 2 fois trop d'appels à FonctionAléatoire$()$, et à évaluer chaque fonction en moyenne 2 fois.
- Au total, la complexité est du même ordre.