Tirage aléatoire uniforme dans une collection dynamique de chemins

Frédéric Voisin, Marie-Claude Gaudel

VALS - Test Club - Janvier 2020

Résumé des épisodes précédents

Test de programmes qui ont de très nombreux chemins (faisables ou non) :

- Tirage aléatoire de chemins d'exécution dans des Graphes de Flot de Contrôle (C.F.G.)
- Couverture uniforme de chemins de taille maximale donnée
- Application à la génération de tests:
  - Sélectionner une collection de chemins qui satisfait le critère de couverture choisi
  - Pour chaque chemin de la collection :
    Calculer par exécution symbolique son "prédicat de cheminement"
    Vérifier la faisabilité avec un solveur SMT et en déduire des valeurs d'entrée.
- Exploration de grands modèles ou ensembles de données représentés par des graphes

Problème : les chemins d'un C.F.G. ne sont pas forcément tous faisables

- Il est très fréquent qu'un programme ait des chemins infaisables. Certains programmes ont même une proportion énorme de chemins infaisables.
- Les solveurs SMT ont des limitations; le test de faisabilité peut être coûteux
- "Folks knowledge" : plus le chemin est long, moins il a de chance d'être faisable.
  Notre objectif n'est donc pas la sélection de très longs chemins .

Exemple

```
Function gcd(int x, int y)
1   |  let a = x;
2   |  let b = y;
3   |  while a ≠ b do
4   |  |  while a > b do
5   |  |  |  a ← a − b;
6   |  |  while b > a do
7   |  |  |  b ← b − a;
8   |  return a;
```

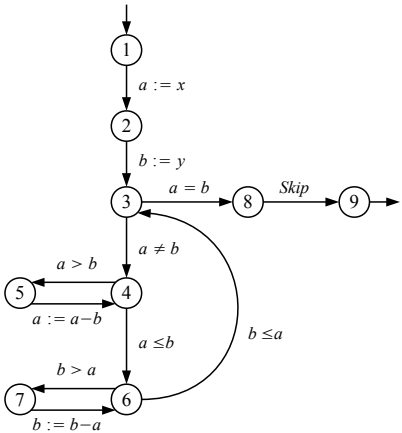
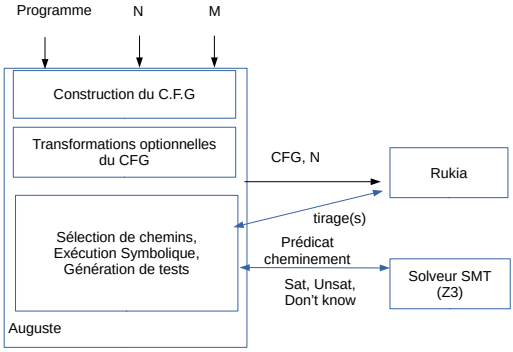


Figure: The gcd program and its associated CFG

- Dépendances (simples) entre les 3 boucles
- Le CFG contient 15478 chemins de longueur maximale $l = 30$ dont 792 sont faisables (5%)
- Le ratio de chemins faisables diminue avec l (0.003 et 0.000003, pour $l = 50$ et 100 resp.)

Auguste et Rukia



Rukia (J. Oudinet):

- Une librairie C++ construite à partir de Boost;
- Implémente une famille d'algorithmes de tirage aléatoire (**méthode récursive**, générateur de Boltzmann, marches aléatoires isotropiques)
- Indépendant du domaine d'application

Auguste:

- Elimination statique de certains chemins infaisables (thèse de R. Aïssat)
- Actuellement applicable à un sous-ensemble restreint du langage C

Nos objectifs

- Exclure du tirage certaines catégories de chemins : chemins infaisables, chemins déjà tirés ("tirage sans remise"), etc.
- Pas de connaissance a priori sur les chemins à exclure : décider de la "faisabilité" d'un chemin relève d'une procédure extérieure au tirage
- Basé sur les préfixes : on définit incrémentalement un ensemble des préfixes "indésirables"
- Par exécution symbolique on construit naturellement les préfixes infaisables de longueurs minimales. Notion de "préfixe interdit minimal", indépendante du tirage.

- Le "tirage sans remise", ou en excluant certains mots, est un problème qui intéresse aussi les combinatoriciens.

- Spécialisation de la méthode récursive de génération aléatoire de Structures Combinatoires [Wilf, Flajolet, ...].
- Engendrer aléatoirement et uniformément des chemins de longueur n dans un graphe $\mathcal{G}$ de racine s_0 avec un sommet final s_f (s_0 sans arc entrant et s_f sans arc sortant).
- Pré-calculer une table $f(s,l)$ où s est un sommet et l une longueur :
  $f(s,l)$ est le nombre de chemins de longueur l de s à s_f ;
  en particulier $f(s_0,n)$ est le nombre de chemins de longueur n de s_0 à s_f.

On définit f par les relations suivantes :

$$f(s_i,j) = \sum_{s_i \to s_k \in \mathcal{G}} f(s_k, j-1), \qquad f(s,0) = 0 \text{ for } s \neq s_f, \qquad f(s_f,0) = 1 \tag{1}$$

Soit $\mathcal{G}, n$ and f, l'algorithme **??** tire uniformément un chemin p de longueur n entre s_0 et s_f.

Algorithm 1 : tirage aléatoire uniforme d'un chemin p de longueur n

$s = s_0$; $p = s_0$; $l = n$;
while $(l > 0)$ {
  tirer s' parmi les successeurs s_k de s avec probabilité $f(s_k, l-1)/f(s,l)$;
  $s = s'$; $p = p.s'$; $l = l - 1$;
}

Pour tirer des chemins de longueur au plus n de s_0 à s_f, on ajoute un arc de s_f sur lui-même.

Contributions

Situation de départ:

- Rukia peut tirer efficacement des chemins avec des centaines d'arcs dans des graphes avec des milliards de sommets.
- Quand un chemin tiré est infaisable, on l'écarte et on en tire un autre, en croisant les doigts... Pas de mémoire des infaisables d'un tirage à l'autre.
- Actuellement, ni la taille de la table de comptage, ni le temps de tirage ne sont des facteurs limitatifs.

Contributions:

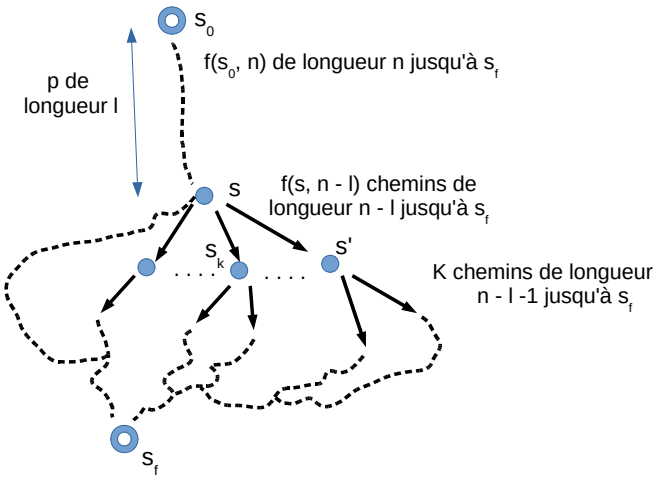
- Soit un ensemble $\mathcal{F}$ de préfixes infaisables, on exclut des futurs tirages toutes les extensions de ces préfixes.
- $\mathcal{F}$ croît incrémentalement en fonction des tirages.
- Tirage aléatoire uniforme sur l'ensemble courant des chemins sans préfixe infaisable connu.

Le tirage prend en compte les infaisables au fur et à mesure qu'on les détecte.

Comptage et tirage à partir de préfixes

Soit un chemin de longueur n avec un préfixe infaisable $p.s.s'$ (sans préfixe strict infaisable) : tous les chemins de préfixe $p.s.s'$ doivent être exclus des futurs tirages.

Soit l la longueur de $p.s$ et $K = f(s', n - l - 1)$.

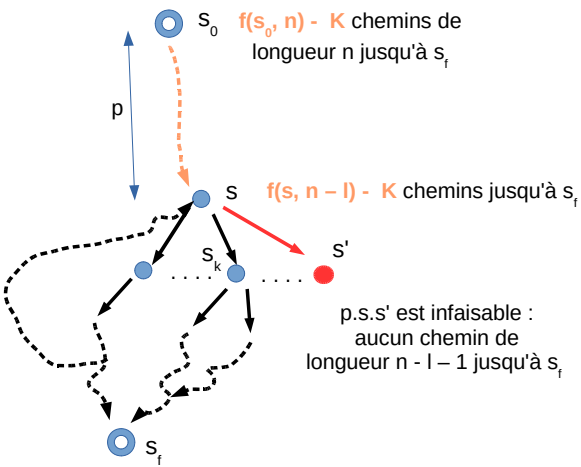


Comptage et tirage à partir de préfixes

Soit un chemin de longueur n avec un préfixe infaisable $p.s.s'$ (sans préfixe strict infaisable) : tous les chemins de préfixe $p.s.s'$ doivent être exclus des futurs tirages.

Soit l la longueur de $p.s$ et $K = f(s', n - l - 1)$.

- Poser $f(s', n - l - 1) = 0$ éviterait au tirage de choisir s' pour prolonger $p.s$.
- $f(s, n - l)$ doit être décrémenté de K, de même que tous les sommets le long de $p.s$, en mettant à jour la table f jusqu'à la racine s_0.

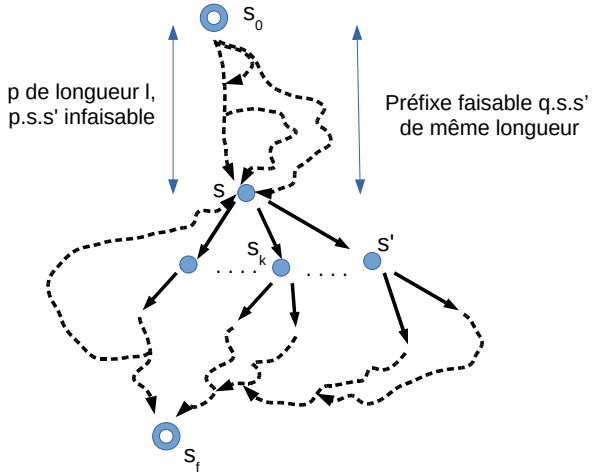


Comptage et tirage à partir de préfixes

Soit un chemin de longueur n avec un préfixe infaisable $p.s.s'$ (sans préfixe strict infaisable) : tous les chemins de préfixe $p.s.s'$ doivent être exclus des futurs tirages.

Soit l la longueur de $p.s$ et $K = f(s', n-l-1)$.

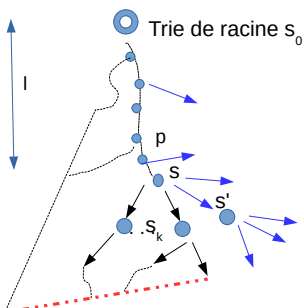
- Poser $f(s', n-l-1) = 0$ éviterait au tirage de choisir s' pour prolonger $p.s$.
- $f(s, n-l)$ doit être décrémenté de K, de même que tous les sommets le long de $p.s$, en mettant à jour la table f jusqu'à la racine s_0.
- Pour les préfixes faisables $q.s.s'$ de même longueur, f donnerait des valeurs incorrectes ! **Le comptage doit dépendre du préfixe**.



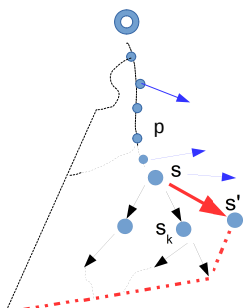
Implémenter efficacement le comptage par les préfixes

Principes: Soit $\mathcal{F}$ l'ensemble des préfixes infaisables

- Généraliser f en une nouvelle table de comptage $f_{\mathcal{F}}$ indicée par des préfixes
- Revoir l'Algo. **??** pour utiliser $f_{\mathcal{F}}$ et construire incrémentalement les préfixes en partant de s_0
- **Construire $f_{\mathcal{F}}$ de façon paresseuse** en ne la définissant que pour (**tous**) les préfixes qui mènent à des préfixes infaisables
  Utiliser f pour les préfixes faisables : soit $r.x \notin \mathcal{F}$ et l sa longueur, $f_{\mathcal{F}}(r.x, n-l) = f(x, n-l)$
  Stocker $f_{\mathcal{F}}$ pour les préfixes infaisables dans une structure de **trie** $\mathcal{C}_{\mathcal{F}}$: les clefs sont les préfixes et la valeur associée à un préfixe r est $f_{\mathcal{F}}(r, n-|r|)$.



La partie bleue n'est pas dans le trie



Le trie après traitement de p.s.s'

- Les clefs dans $\mathcal{C}_{\mathcal{F}}$ sont les préfixes infaisables et tous leurs sous-préfixes;
- Les éléments de $\mathcal{F}$ sont aux feuilles du trie, avec 0 comme valeur associée.

Tirage à partir de $\mathcal{C}_{\mathcal{F}}$

Soit $\mathcal{F}$ l'ensemble des préfixes infaisables, $\mathcal{C}_{\mathcal{F}}$ le trie associé, $f_{\mathcal{F}}$ et f les tables de comptage.

Algorithm 2 : Tirage aléatoire uniforme d'un chemin p de longueur n avec $f_{\mathcal{F}}$

let $count(p.x, l) =$ if $p.x \in \mathcal{C}_{\mathcal{F}}$ then return $\mathcal{F}(p.x, l)$ else return $f(x, l)$

$s = s_0$; $p = \varepsilon$; $l = n$;
while $(l > 0)$ {
  tirer s' parmi les successeurs s_k de s avec probabilité $count(p.s.s_k, l-1))/count(p.s, l)$
  $s = s'$; $p = p.s'$; $l = l - 1$;
}

Remarque 1 : On démarre avec $\mathcal{C}_{\mathcal{F}}$ un trie réduit à la clef s_0 associé à la valeur $f(s_0, n)$.

Remarque 2 : Tant que le préfixe en construction reste à l'intérieur de $\mathcal{C}_{\mathcal{F}}$ il est faisable !
Rukia renvoie maintenant un chemin et la longueur de son préfixe maximal dans $\mathcal{C}_{\mathcal{F}}$.

Mise à jour du trie

Soit $\mathcal{F}$ l'ensemble des préfixes interdits, $\mathcal{C}_{\mathcal{F}}$ le trie associé, $f_{\mathcal{F}}$ et f les tables de comptage,
soit r le préfixe d'un chemin tiré à partir de $\mathcal{C}_{\mathcal{F}}$ qu'on détecte comme infaisable, $\mathcal{F}' = \mathcal{F} \cup \{r\}$ et $\mathcal{C}_{\mathcal{F}'}$ est le trie associé,

---

Algorithm 3 : étendre $\mathcal{C}_{\mathcal{F}}$ en $\mathcal{C}_{\mathcal{F}'}$

---

let $K = f_{\mathcal{F}}(r, n - |r|)$;
Ajouter dans $\mathcal{C}_{\mathcal{F}}$ une branche étiquetée par les sommets de r en utilisant f pour les valeurs des sommets qui ne sont pas déjà dans $\mathcal{C}_{\mathcal{F}}$;
Pour r et tous ses sous-préfixes, soustraire K de leur valeur dans $\mathcal{C}_{\mathcal{F}}$.

Par construction r a maintenant 0 comme valeur dans $\mathcal{C}_{\mathcal{F}'}$.

L'algorithme de tirage garantit que ni r ni un de ses préfixes est dans $\mathcal{F}$.

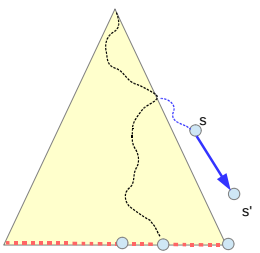
Mise à jour du trie

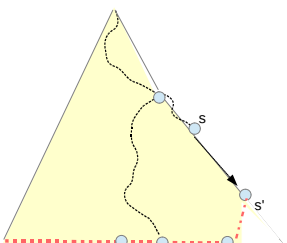
Soit $\mathcal{F}$ l'ensemble des préfixes interdits, $\mathcal{C}_\mathcal{F}$ le trie associé, $f_\mathcal{F}$ et f les tables de comptage,
soit r le préfixe d'un chemin tiré à partir de $\mathcal{C}_\mathcal{F}$ qu'on détecte comme infaisable, $\mathcal{F}' = \mathcal{F} \cup \{r\}$ et $\mathcal{C}_{\mathcal{F}'}$ est le trie associé,

Algorithm 3 : étendre $\mathcal{C}_\mathcal{F}$ en $\mathcal{C}_{\mathcal{F}'}$

let $K = f_\mathcal{F}(r, n - |r|)$;
Ajouter dans $\mathcal{C}_\mathcal{F}$ une branche étiquetée par les sommets de r en utilisant f pour les valeurs des sommets qui ne sont pas déjà dans $\mathcal{C}_\mathcal{F}$;
Pour r et tous ses sous-préfixes, soustraire K de leur valeur dans $\mathcal{C}_\mathcal{F}$.

Tous les sous-préfixes de $p.s.s'$ sont dans le trie et s' devient une feuille du trie.

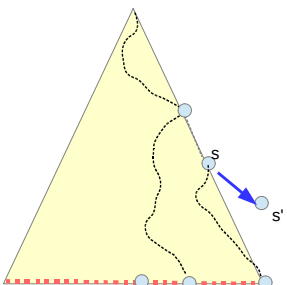


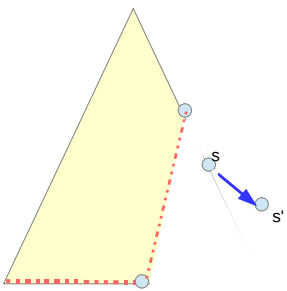


Élagage du trie

Complétion implicite de $\mathcal{F}'$:
La propagation dans $\mathcal{C}_{\mathcal{F}'}$ aux sous-préfixes de r peut leur associer la valeur 0
Ces préfixes ne sont plus tirables bien qu'ils ne soient pas dans $\mathcal{F}'$: ce sont des "impasses".
Le tirage les traite naturellement comme des préfixes infaisables.

Élagage de $\mathcal{C}_{\mathcal{F}'}$:
Tout sous-arbre de $\mathcal{C}$ associé à la valeur 0 peut être élagué pour ne conserver comme feuille que le premier sommet de valeur 0.





Remarque: Si, en plus des infaisables, on retire les chemins faisables ("tirage sans remise"), on obtient un algorithme de **génération exhaustive** des chemins faisables.

Le nouveau prototype

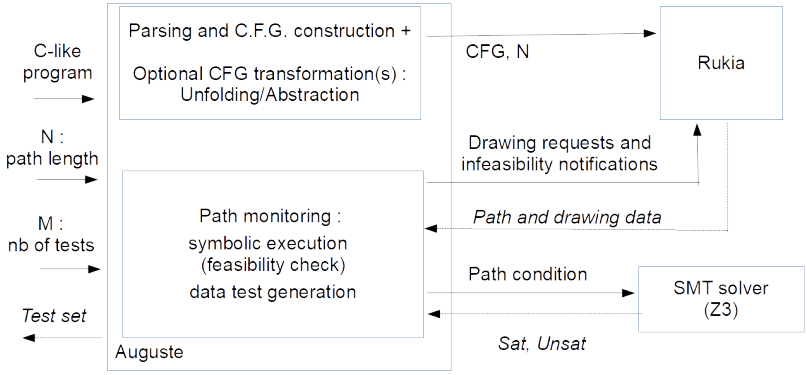


Figure: The `Auguste` prototype for program testing

Expériences

- Dans certains exemples, on peut obtenir un CFG sans chemin infaisable mais dont la taille (nombre de sommets) est beaucoup plus grande que le CFG de départ.
- Plus généralement les expériences sont faites :
  - Dans le CFG de départ
  - Dans un CFG “déplié” jusqu'à une certaine profondeur choisie avant la phase de tirage
  - La longueur des chemins tirés reste non contrainte : on peut tirer au-delà de la profondeur de dépliage.
  - Soit on tire un nombre donné de chemins faisables, soit on lance une recherche exhaustive

Remarques :

- Habituellement, on ne connaît ni la borne pour un dépliage optimal (qui n'existe pas toujours), ni la proportion ou la forme des chemins infaisables.
- La recherche exhaustive des chemins faisables sert ici de méthode de stress de la méthode

Résultats pour gcd

- On peut obtenir statiquement un CGF sans chemin infaisable
- Dans ce CFG optimal il existe 792 chemins de longueur $l \leq 30$

	l	$\|G\|$	paths	drawn	K	size	% SMT saved
initial CFG	30	10	15478	1152	4672	1796 - 1880	80.1- 80.4 %
optimal CFG	30	24	792	792	1	1358 - 1444	76.0 - 76.2 %

Table: Results for the gcd example

$K =$ "killing score" (chemins éliminés par un seul préfixe)
size = taille maximale du trie (nombre de sommets)
"SMT saved" : ratio d'appels au SMT économisés.

Expériences : Tcas

Contexte:

- Fait partie du classique “Siemens benchmark for testers”
- La documentation mentionne la présence d’un chemin infaisable
- Plusieurs fonctions qui n’étaient constituées que d’une unique expression ont été “in-linées”
- Pas de boucle mais de nombreux opérateurs booléens (paresseux), induisant un CFG complexe
- Le CFG peut être déplié statiquement en un “arbre d’exécution symbolique” contenant exactement 123 chemins faisables de longueur au plus 47.

Experiences : Recherche de chemins faisables

- Tirage dans le CFG initial : tous les faisables de longueur 47
- Tirage dans l’arbre d’exécution symbolique (dépliage optimal) : idem
- Tirage dans un CFG partiellement déplié (40 au lieu de 47) et chemins de longueur 50.

Le nombre de chemins faisables est le même dans tous les cas.

Tcas: résultats

Nouvel algorithme : élimination des infaisables *et* des doubles. 123 chemins faisables à trouver

	l	$\|G\|$	chemins	tirage	K	taille trie	% SMT
tcas-opt	47	1677	123	123	1	282 - 322	58.6 - 60.0 %
tcas-CFG	47	88	179720	752 - 758	7562	1729 - 1812	87.4 - 87.9 %
tcas-40	50	1232	386	298	4	1017 - 1174	79.0 - 80.1 %
tcas-40 (60)	50	1232	386	161 - 171	4	1051 - 1149	68.1 - 71.7 %
CFG-40 (60)	50	88	181512	579 - 601	7562	1697 - 1787	85.7 - 85.9 %

Table: Résultats pour `tcas`

Version précédente (pas d'élimination des infaisables ou des faisables dupliqués):

- tcas-opt: Tirages: 491 – 906;
- tcas-CFG: Après 130 400 tirages: 62 sur 123
- tcas-40 (60): tirer 60 chemins faisables parmi 386; Tirages: 194;
- CFG-40 (60): tirer 60 chemins faisables parmi 181 512; Tirages: 83 301;

“Binary search”

Context:

- Inspiré d’un exemple de la "Gallery" en ligne de Pathcrawler
- Version originale: recherche dichotomique dans un tableau trié
- Notre version effectue une sorte de recherche dichotomique dans un tableau non trié, entre deux bornes passées en paramètres.
- Pas de dépliage qui donnerait un CFG sans chemin infaisable.

Expériences: Tirage de chemins faisables de longueur 50

- Dans le CFG initial
- Dans un CFG déplié jusqu’à une profondeur 30
- Génération exhaustive de tous les chemins faisables ? Leur nombre est inconnu

“Binary search”: les résultats

Nouvel algorithme: élimination des infaisables et des doubles. Tirage de **300** ou **3000** faisables

	l	$\|G\|$	chemins	tirages	K	trie	Eco. SMT
CFG-300	50	17	21247	831 - 883	4607	6495 - 6632	73.5 - 74.6 %
b30-300	50	855	8148	738 - 817	31	6061 - 6523	71.8 - 73.2 %
CFG-3000	50	17	21247	4883	4607	10841 - 11022	85.7- 85.9 %
b30-3000	50	855	8148	4787	31	10732 - 10950	85.6 - 85.8 %

Table: Résultats pour `bsearch`

Remarque: le système énumère 2594 chemins faisables et détecte qu'il n'y en a plus d'autres.

Version précédente (pas d'élimination des infaisables ou des faisables dupliqués):

- CFG-300: 300 parmi 21 247; Tirage: 2726 (2406 infaisables);
- b30-300: 300 parmi 8 148; Tirage: 944 (632 infaisables);
- CFG-3000: **2594** parmi 21 247; Crash après 130 300 tirages (2590 trouvés)
- b30-3000: **2594** parmi 8 148; Tirage: 83 369 (56 854 infaisables)

Conclusion

État courant:

- Prototype basé sur une version modifiée de Rukia
- Expériences prometteuses :
  - Les chemins infaisables sont des obstacles pour les techniques d'analyse statique ;
  - Se baser sur les plus courts préfixes infaisables est une méthode naturelle d'élimination ;
  - Amélioration spectaculaire par rapport à la méthode de tirage avec rejet et remise.
- D'autres expériences sont nécessaires pour évaluer le passage à l'échelle
- L'efficacité dépend en pratique du "score d'élimination" de chaque préfixe : entre une poignée de chemins et plusieurs milliers
- Le tirage exhaustif des chemins faisables par notre algorithme n'est pas un objectif en soi.

Principe adaptable aussi à la génération par la méthode de Boltzmann

Travaux en cours:

- Extension du langage pris en charge et intégration à Frama-C
- Optimisation de la taille du trie (actuellement pas un problème)
- Génération non uniforme de chemins, pour privilégier certaines partie du programme
- Combinaison de notre méthode par "élimination de préfixes" avec des méthodes de génération aléatoire basées sur la sémantique du programme (schémas de chemins infaisables).