Random generation of labelled combinatorial structure

Yassine H., Fatemeh H.

Sorbonne Université, UFR Ingénierie
Frédéric PESCHANSKI

Novembre 2019

Sommaire

Introduction

Un des intérêts de générer des structures décomposables de façon aléatoire est de **comparer** leurs caractéristiques à celles obtenues de structures issues de données réelles. Un autre serait de générer des structures à des **fins de tests**.

Dans cet article nous allons voir comment :

- Définir une spécification d'une structure
- Standardiser une spécification
- Générer la structure qui correspond à une spécification standardisée

Rappels

Classe combinatoire : Une collection C d'objets définis de manière similaire dotés d'une notions de taille de manière à ce qu'il y ait qu'un nombre fini d'objet de chaque taille.
Objet étiqueté : Un objet, qui peut être vu comme un graphe, dont certains noeuds sont étiquetés par des entiers distincts.
Taille d'un objet : Le nombre de noeuds étiquetés.

Définition d'une spécification

Une spécification peut posséder 2 éléments de base et une suite d'opérations :

- un élément de taille 0 sans label désigné par 1
- une élément de taille 1 qui possède un noeud étiqueté , désigné par le symbole Z
- des opérations : $+$, ., sequence(), set(), cycle()
- une spécification peut aussi composer d'autres objets possédant eux même une spécification
- une spécification peut aussi être récursive

Les opérations de base

- Le $+$: opérateur binaire qui représente l'union disjointe entre deux objets
- Le $.$: opérateur binaire qui représente le produit cartésien entre les éléments de deux objets
- La **séquence** : opérateur unaire qui représente une séquence d'éléments de A
- Le **set** : opérateur unaire qui représente un ensemble non ordonné d'éléments de A
- Le **cycle** : opérateur unaire qui définit une séquence d'éléments qui forme un cycle

Spécification

Soit $T = (T_0, T_1 ... T_m)$ un $(m+1)$ tuplet de classe de structures combinatoires. Une spécification de T est un ensemble de m+1 avec la i-ème équation étant de la forme suivante :

$$T_i = \Psi_i(T_0, T_1 ... T_m)$$

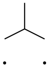
où $\Psi(i)$ est un terme construit a partir de 1,Z et des T_j utilisant les constructions standards listées en (2).

Exemple de spécification de structure de base

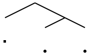
Spécification des arbres binaires : $B = Z + B.B$

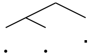
Un objet de taille 1 .

Un objet de taille 2



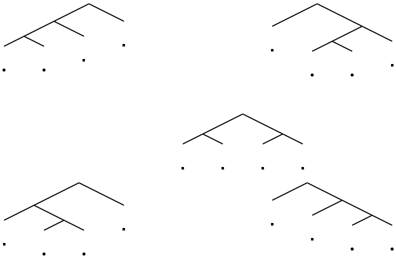
Deux objets de taille 3 :





Exemple de spécification de structure de base

Cinq objets de taille 4 :



Les spécifications de 11 structures basiques

Table 1
Eleven basic combinatorial structures and their specifications

Specification	Objects
$A = Z \cdot \mathtt{set}(A)$	Non plane trees
$B = Z + B \cdot B$	Plane binary trees
$C = Z \cdot \mathtt{sequence}(C)$	Plane general trees
$D = \mathtt{set}(\mathtt{cycle})(Z)$	Permutations
$E = \mathtt{set}(\mathtt{cycle}(A))$	Functional graphs
$F = \mathtt{set}(\mathtt{set}(Z, \text{card} \geqslant 1))$	Set partitions
$G = Z + Z \cdot \mathtt{set}(G, \text{card} = 3)$	Non plane ternary trees
$H = Z + \mathtt{set}(H, \text{card} \geqslant 2)$	Hierarchies
$K = \mathtt{set}(\mathtt{cycle}(Z \cdot \mathtt{set}(G, \text{card} = 2)))$	3-constrained functional graphs
$L = \mathtt{set}(\mathtt{set}(\mathtt{set}(Z, \text{card} \geqslant 1), \text{card} \geqslant 1))$	3-balanced hierarchies
$M = \mathtt{sequence}(\mathtt{set}(Z, \text{card} \geqslant 1))$	Surjections

Fonction génératrice

Fonctions génératrice : La séquence qui compte le nombre d'éléments d'une taille donnée pour un objet, est définie par une série dite génératrice dont les coefficients correspondent à cette séquence.

$$\mathrm{C(z)}=\sum_{n=0}^{\infty}\mathrm{C}_n \ \frac{z^n}{n!}$$

où C_n est le nombre d'objets C de taille n.
On peut écrire donc le nombre $c_n = \frac{C_n}{n!}$
la normalisation de C_n (selon les conventions d'écritures)

$$c_n=[z^n]\mathrm{C(z)}$$

Fonction génératrice

Théorème Folk : Soit une spécification S pour une classe C, on obtient un ensemble d'équations pour les fonctions génératrices correspondantes.

$$C = A + B \implies C(z) = A(z) + B(z)$$
$$C = A.B \implies C(z) = A(z).B(z)$$
$$C = sequence(A) \implies C(z) = (1 - A(z))^{-}1 \ldots$$

Exemple de fonction génératrice

Spécification des arbres binaires : B = Z + B.B

$$B^2(z) - B(z) + z = 0 \tag{1}$$

on résout (1) et on choisit

$$\mathrm{B(z)} = \frac{1-\sqrt{1-4z}}{2}$$
$$\mathrm{B(z)} = \frac{1}{2} - \frac{(1-4z)^{\frac{1}{2}}}{2}$$

on utilise le binome de newton :

$$\mathrm{B(z)} = \frac{1}{2} - \sum_{n=0}^{\infty} \binom{\frac{1}{2}}{k}(-4z)^n$$
$$b_n = [z^n]\mathrm{B(z)} = \frac{1}{n} \binom{2n-2}{n-1}$$

L'opérateur Θ(*Marquage*)

$$\Theta(A) = \bigcup_{n=1}^{\infty} (A_n x[1...n])$$

où A_n est la sous-classe des objets de A de taille n et [1...n] est l'intervalle d'entier {I, 2,...,n}.
En d'autres termes, un objet de la classe Θ(A) peut être considéré comme un objet de A avec la propriété supplémentaire que l'une des étiquettes, correspondant au champ dans [1...n], est distinguée.
Et donc par définition pour $C = \Theta(A)$ $c_n = na_n$

Standardisation

Soit T $= (T_0, T_1, ..., T_m)$ un tuple de classes de structures combinatoires. Une spécification standard de T est un ensemble de m $+$ 1 équations, la i-eme équation est de l'une des formes suivantes :

$$T_i = 1\,;\; T_i = Z_i\,;\; T_i = U_j + U_k\,;\; T_i = U_j.U_k\,; \Theta Ti = U_j.U_k$$

où chaque U$_j \in \{1, Z, T_0, ..., T_m, \Theta T_0, ..., \Theta T_m\}$

Exemple de standardisation de spécification

Arbre binaire :

$$\mathrm{B} = \mathrm{Z} + \mathrm{B.B} \rightarrow B = Z + \mathrm{U}_i : U_1 = \mathrm{B.B}$$

Arbre plan :

- $\mathrm{C} = \mathrm{Z.sequence(C)}$
- $\mathrm{C=Z.(1+Z.C)}$
- $\mathrm{C} = \mathrm{Z}.U_1$, $U_1 = 1 + U_2$, $U_2 = \mathrm{Z}.U_1$

Hierarchie :

$$\mathrm{H} = \mathrm{Z} + \mathrm{set(H, card} \geq 2) \rightarrow$$
$$\mathrm{H} = \mathrm{Z} + U_1\ \Theta \mathrm{U}_1 = U_2.\Theta H, \Theta U2 = \mathrm{U}_3.\ \Theta H, \Theta \mathrm{U}_3 = U_3\ .\ \Theta H.$$

Génération aléatoire à partir de la spécification standard

Avec la spécification standardisée on va maintenant pouvoir générer la structure correspondante :

- Chaque opérateur défini dans la spécification standard correspond à une règle de génération
- Ne nécessite qu'une itération sur la spécification standardisée
- En revanche besoin de pré-calcule de toute les énumérations des structures présentes dans la spécification en $O(n^2)$ en temps et $O(n)$ en espace,

Exemple de template de génération d'éléments

Case : C = 1
begin
    gC := procedure (n :integer)
    if $n = 0$ **then**
        return 1 ;
    end
end

Algorithm 1: C = 1

Exemple de template de génération d'éléments

Case : $C = Z$
begin
  gC := procedure (n :integer)
  if $n = 1$ **then**
    return Z ;
  end
end

Algorithm 2: $C = 1$

Exemple de template de génération d'éléments

Case : C = A + B
begin
  gC := procedure (n :integer)
  U := Uniform([0,1])
  if $U < (a_n/c_n)$ **then**
    return g(A(n)
  end
  else
    return g(B(n)
  end
end

Algorithm 3: C = A + B

Exemple de template de génération d'éléments

```
Case : C = A . B
begin
    gC := procedure (n :integer)
    U := Uniform([0,1])
    K :=0 ; S :=(a_0.b_n)/c_n
    while U > S do
        K := K + 1
        S :=S + (a_k.b_{n-k})/c_n
    end
    return [gA(K),gB(n - k)]
end
```

Algorithm 4: C = A . B

Exemple d'exécution de l'algorithme

On va prendre l'exemple des arbres binaires
Spécification standard :

$$B{=}Z{+}U_i \ {:}U_1{=}B.B$$

On souhaite générer un arbre de taille 4
On considère avoir pré-calculés

les b_n de 0 à 4 : 0,1,1,2,5
et nb_n de 0 à 4 : 0,1,2,6,20

On commence le parcours de la spécification :
gB(4) tombe dans le template A+B où A vaut Z et B U_1 :

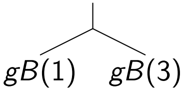
n tire U uniformément entre 0 et 1
$0 \leq$ U donc on retourne gU_1(4)

Exemple d'exécution de l'algorithme

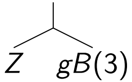
gU_1(4) où U_1=B.B, suit le template A.B :

On tire U = 0.3, K=0,S=0
K=1, S=2/5, S > U on retourne gB(1), gB(3)

cela donne l'arbre



On continue sur gB(1), U < 1, on retourne donc gZ(1) qui retourne Z
On a donc l'arbre



Conclusion

- La génération aléatoire d'une grande partie des structures étiquetées peut être automatisée en utilisant des systèmes de manipulation symboliques.
- Complexité : O(n^2) en pré-calcul et génération en O(n), optimisable avec des heuristiques en $\frac{1}{2}$nlog(n)
- Des structures de taille de quelques centaines d'éléments sont générées avec une distribution uniforme exacte en quelques secondes de temps d'ordinateur.