

Éléments d'algorithmique : les arbres généraux

Cours 10

29 novembre 2021

Structures arborescentes, arbres généraux

Motivations : en pratique on doit souvent manipuler des structures arborescentes, pas forcément binaires

- ▶ Table des matières : volume, chapitre, section, sous-section
- ▶ Classification des espèces en sous-espèce
- ▶ Arbres phylogénétiques : arbre de l'évolution par mutation d'un ensemble de gènes
- ▶ les expressions fonctionnelles : $1 - \max(5, -(3+2), 7)$

Par ailleurs en algorithmique on arrive naturellement à des arbres ordonnées lorsqu'on fait par exemples de parcours de graphes (en largeur, en profondeur).

Arbres généraux étiquetés (ou valués)

Soit T un type.

Opérations :

- ▶ $\text{Nœud} : T \times (\text{AV}(T))^* \rightarrow \text{AV}(T)$
- ▶ $\text{Forêt} : \text{AV}(T) \rightarrow (\text{AV}(T))^*$
- ▶ $\text{Val} : \text{AV}(T) \rightarrow T$


Axiomes :


- ▶ $\text{Forêt}(\text{Nœud}(v, [t_1, \dots, t_n])) = [t_1, \dots, t_n]$
- ▶ $\text{Val}(\text{Nœud}(v, [t_1, \dots, t_n])) = v$
- ▶ $\text{Nœud}(\text{Val}(t), \text{Forêt}(t)) = t$.

- ▶ $(\text{AV}(T))^*$ représente l'ensemble des listes d'arbres (qu'on appelle aussi **forêts**).
- ▶ On remarque qu'avec un tel formalisme, il n'y a pas d'arbre vide.
- ▶ Si $\text{Forêt}(t) = []$, on dit que t est une **feuille**.
- ▶ Les parcours préfixe, et postfixe (ou suffixe) sont définis de la même manière que pour les arbres binaires, en revanche, le parcours infixé n'est plus défini.

Arbres généraux : exemples

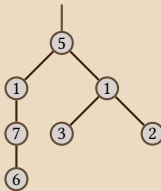
- Exemples -

► $\text{Nœud}(5, []) =$ 

► $\text{Nœud}(6, [\text{Nœud}(5, [])]) =$ 

► $\text{Val}\left(\begin{array}{c} \text{1} \\ / \quad \backslash \\ \text{3} \quad \text{2} \end{array}\right) = 1$

► Forêt $\left(\begin{array}{c} \text{5} \\ / \quad \backslash \\ \text{1} \quad \text{1} \\ / \quad \backslash \quad / \quad \backslash \\ \text{7} \quad \text{3} \quad \text{2} \quad \text{6} \end{array} \right) = \left[\begin{array}{c} \text{1} \\ | \\ \text{7} \\ | \\ \text{6} \end{array}, \begin{array}{c} \text{1} \\ / \quad \backslash \\ \text{3} \quad \text{2} \end{array} \right]$

► $\text{Nœud}\left(5, \left[\begin{array}{c} \text{1} \\ | \\ \text{7} \\ | \\ \text{6} \end{array}, \begin{array}{c} \text{1} \\ / \quad \backslash \\ \text{3} \quad \text{2} \end{array} \right] \right) =$ 

Code préfixe

Problème : le codage d'un arbre général par pointeurs (comme pour les listes, les piles, les files et les arbres binaires) est faisable mais complexe. Un arbre général serait en effet composée d'une valeur et d'une liste d'arbres binaires généraux (la forêt).

⇒ On cherche un codage plus simple et compact.

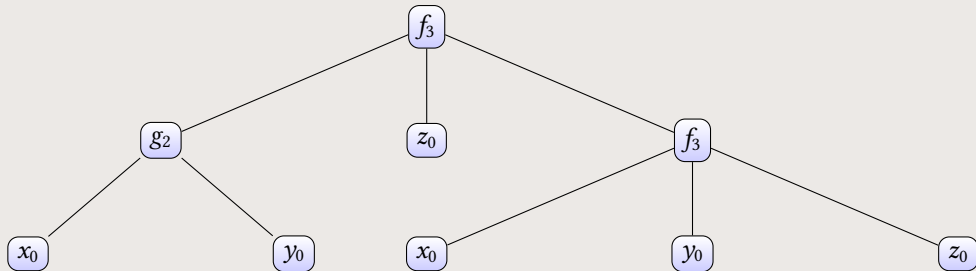
Algorithme de codage préfixe

- ▶ Entrée : un arbre général t à n nœuds.
- ▶ Sortie : la liste des noeuds avec leur degré, dans l'ordre préfixe.

```
liste code_prefixe (arbre t)
  T <- Val(t)
  F <- Foret(t)
  d <- Longueur (Foret(t))
  L <- [(T,d)]
  pour i de 1 à d
    L <- Concatene (L, Code_Prefixe( F[i] ))
  renvoyer L
```

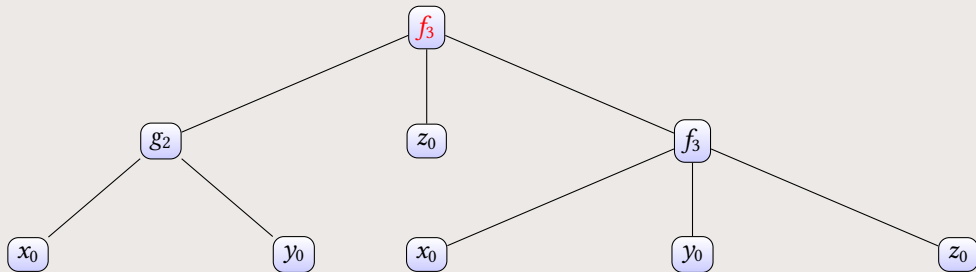
Exemple d'un codage préfixe

[]



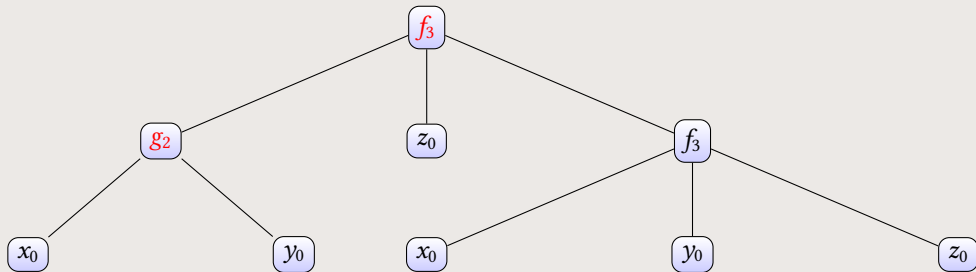
Exemple d'un codage préfixe

$[f_3]$



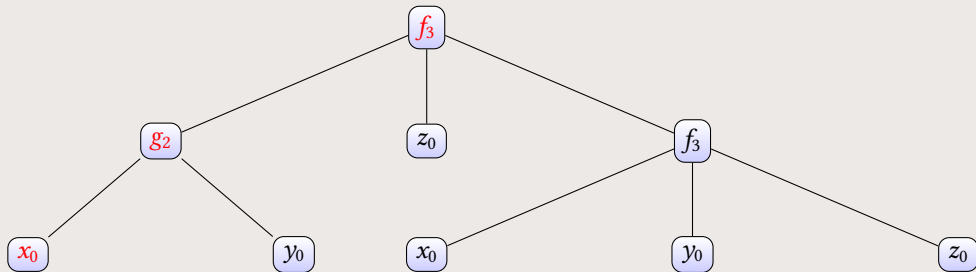
Exemple d'un codage préfixe

$[f_3, g_2]$



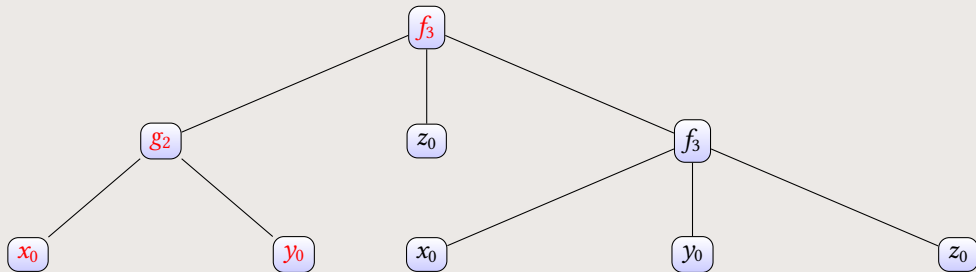
Exemple d'un codage préfixe

$[f_3, g_2, x_0]$



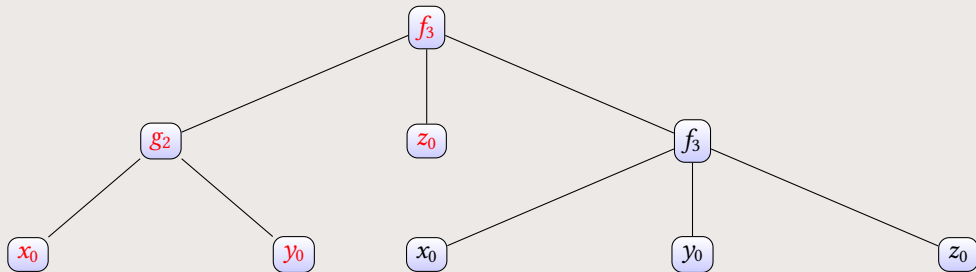
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0]$



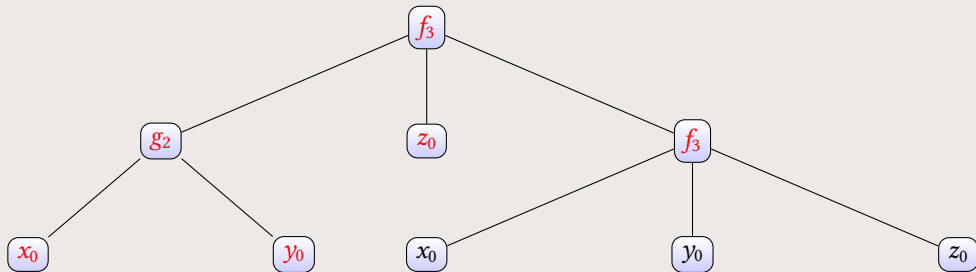
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0]$



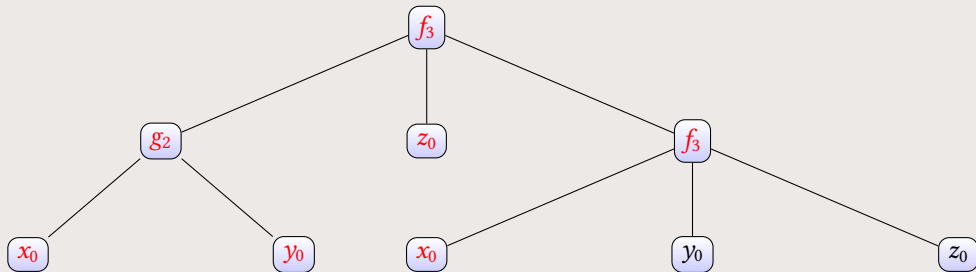
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3]$



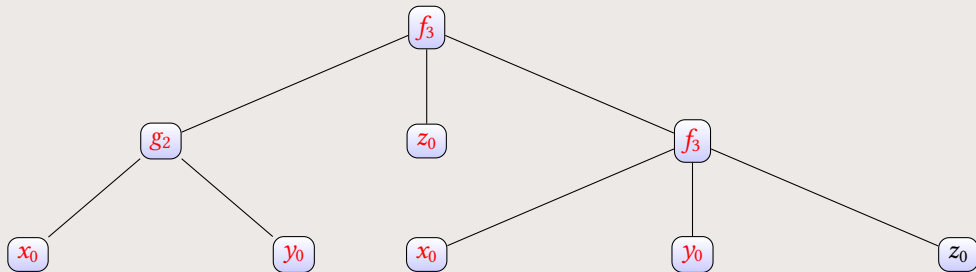
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0]$



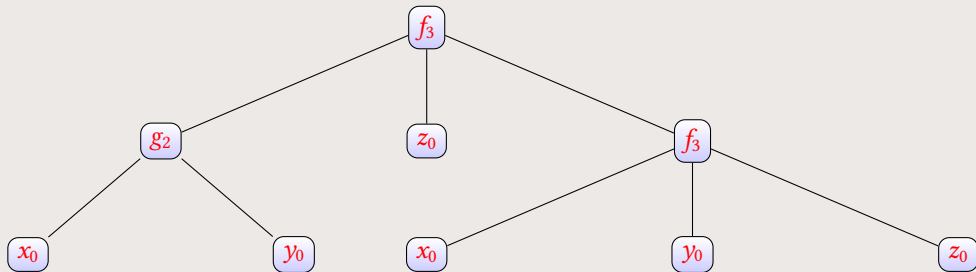
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0]$



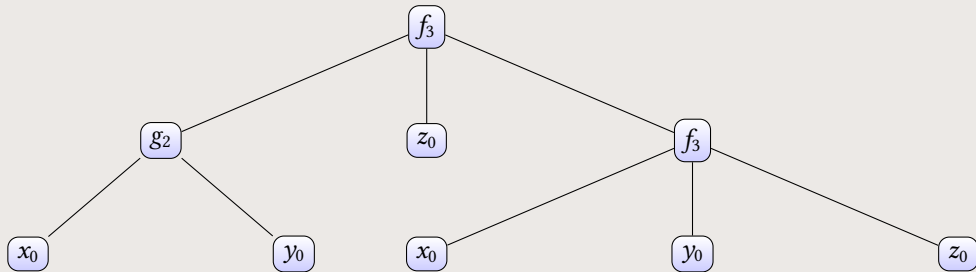
Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$



Exemple d'un codage préfixe

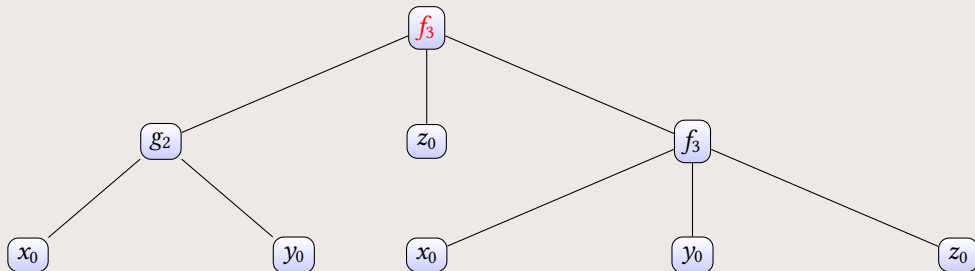
$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$



Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$

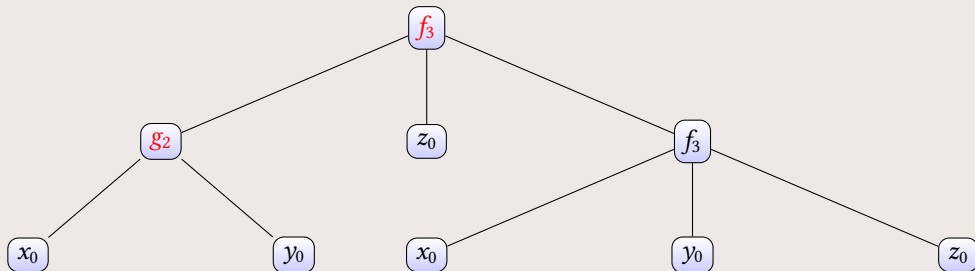


Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Oui, il suffit de relire le mot de gauche à droite et replacer les noeuds.

Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$

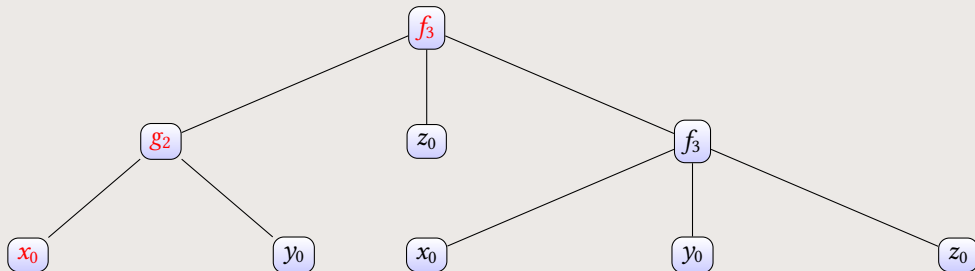


Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Oui, il suffit de relire le mot de gauche à droite et replacer les noeuds.

Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$

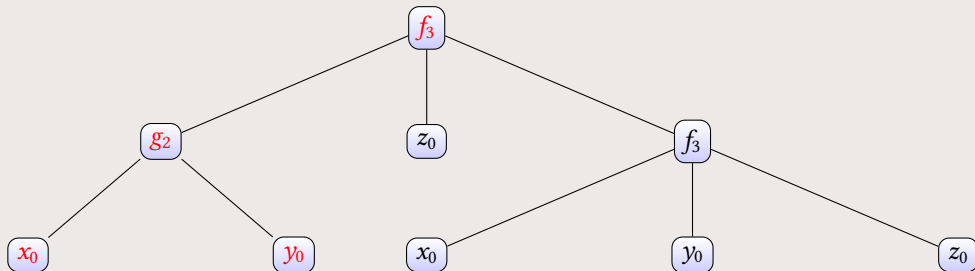


Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Oui, il suffit de relire le mot de gauche à droite et replacer les noeuds.

Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$

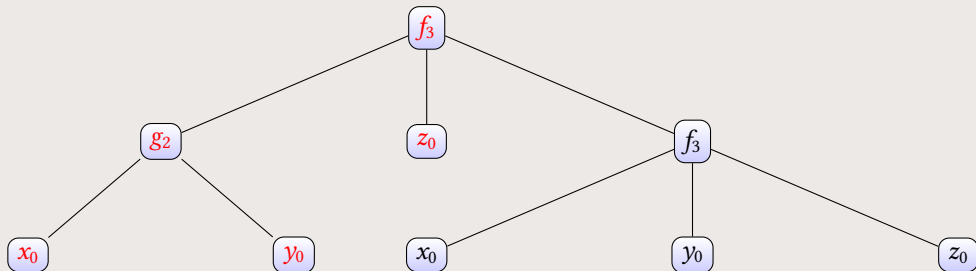


Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Oui, il suffit de relire le mot de gauche à droite et replacer les noeuds.

Exemple d'un codage préfixe

$[f_3, g_2, x_0, y_0, z_0, f_3, x_0, y_0, z_0]$



Question : Peut-on reconstruire l'arbre général à partir du code préfixe ?

Oui, il suffit de relire le mot de gauche à droite et replacer les noeuds.

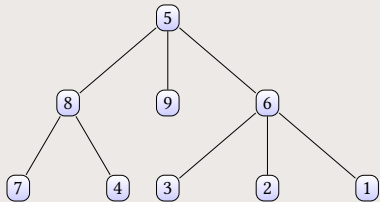
Transformation d'un arbre général en arbre binaire

Objectif : on veut construire une bijection f entre l'ensemble des arbres généraux et l'ensemble des arbres binaires.

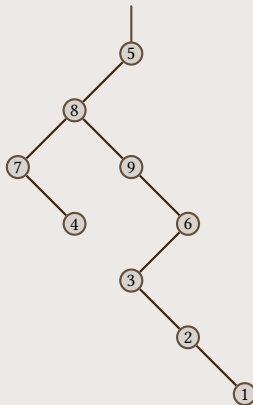
Soit t un arbre général, et soient x et y deux nœuds de t . On construit l'arbre binaire s correspondant de la façon suivante :

1. si y est **fil** de x dans t alors y est le **fil gauche** de x dans s ;
 2. si y est **frère droit** de x (frère immédiat) dans t alors y est le **fil droit** de x dans s .
- Il s'agit d'une **bijection** entre l'ensemble des arbres généraux et l'ensemble des arbres binaires sans sous-arbre droit ;
 - la **racine** des deux arbres est la même.

Transformation d'un arbre général en arbre binaire : exemple



\xrightarrow{f}



Avantages de la bijection f

⇒ Soit s un arbre binaire sans sous-arbre droit. On peut reconstruire l'arbre général correspondant en changeant les implications de la construction arbres généraux \rightarrow arbres binaires sans sous-arbre droit.

⇒ L'arbre binaire obtenu est plus compact dans le sens où on n'a pas besoin de liste pour représenter les forêts.

⇒ Pour tous arbres généraux t , on a préservation par f du parcours préfixe :

$$\text{parcours-préfixe}(t) = \text{parcours-préfixe}(f(t)).$$

⇒ Pour tous arbres généraux t , le résultat de parcours suffixe (postfixe) de t par f est le même que le résultat du parcours infixe de $f(t)$:

$$\text{parcours-suffixe}(t) = \text{parcours-infixe}(f(t)).$$