1 Arbres binaires : rappels

1.1 Définition

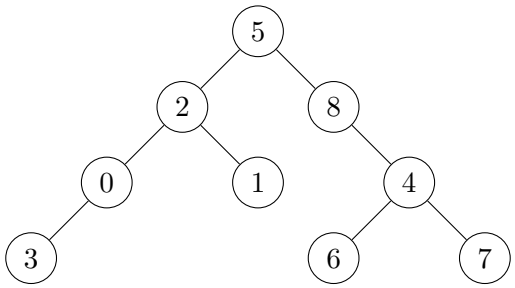
Soit E un ensemble et Nil un objet n'appartenant pas à E.

On définit récursivement l'ensemble $\mathscr{B}(E)$ des arbres binaires sur E :

— l'arbre vide Nil appartient à $\mathscr{B}(E)$;

— si $a \in E$ et $(B, B') \in \mathscr{B}(E)^2$, alors $(a, (B, B')) \in \mathscr{B}(E)$.

Tout arbre binaire (sauf l'arbre vide) est de la forme $(a, (B, B'))$: a est appelé la racine de l'arbre, B est appelé le fils gauche de a et B' est appelé le fils droits de a.



Par convention, on ne représente pas l'arbre vide dans ces représentations.

1.2 Feuilles et nœuds

On définit récursivement les feuilles $F(B)$ et les nœuds (internes) $N(B)$ d'un arbre B (listes car il peut y avoir des répétitions) :

— si B est vide, alors $F(B) = [\,]$ et $N(B) = [\,]$;

— si $B = (a, (B', B''))$, alors

  — si B' et B'' sont vides, alors $F(B) = [B]$ et $N(B) = [\,]$;

  — sinon $F(B) = F(B') \bullet F(B'')$ et $N(B) = [B] \bullet N(B') \bullet N(B'')$.

($\bullet$ désigne la concaténation)

Un sommet de l'arbre est soit une feuille, soit un nœud (interne). Les feuilles sont donc les sommets qui ont deux fils vides (on dit en général par abus de langage qu'ils n'ont pas de fils), les nœuds (internes) sont les sommets qui ont au moins un fils non vide.

Tout sommet étant un arbre non vide, il est de la forme $(a, (B', B''))$: a est appelé l'étiquette du sommet.

Remarque.

— La terminologie est relativement mouvante : certains auteurs appellent nœud ce qui est appelé ici sommet, les nœuds étant internes ou externes (= feuilles). En fonction du contexte, il est facile de s'adapter...

— En général, lorsqu'on parle des feuilles ou des nœuds, on identifie le sommet avec son étiquette. Avec l'arbre donné en exemple, on se permettra de parler de la feuille 7, ou du nœud 8.

Proposition 1 *Si B est un arbre possédant n nœuds (internes), alors B possède au plus $n+1$ feuilles.*

Définition. Un arbre binaire est dit entier s'il est non vide et si tous ses sommets ont 0 ou 2 fils.

Proposition 2 *Un arbre binaire ayant n nœuds (internes) est entier si et seulement si il possède $n+1$ feuilles.*

1.3 Hauteur

Définition. On définit là encore récursivement la hauteur d'un arbre B :
— si B est vide, on pose $h(B) = -1$;
— si $B = (a, (B', B''))$, alors $h(B) = 1 + \max(h(B'), h(B''))$.

La hauteur d'un arbre est donc le nombre d'arêtes du plus long chemin menant de la racine à une feuille.

On parle aussi de la profondeur d'un sommet (ou d'un nœud) comme la longueur du chemin menant de la racine à ce sommet.

Proposition 3 *Si B est un arbre de hauteur h et possédant n sommets, alors*

$$h + 1 \leqslant n \leqslant 2^{h+1} - 1$$

ou encore

$$\lfloor \log n \rfloor \leqslant h \leqslant n - 1$$

2 Arbres en CAML

Les arbres binaires ne font pas partie des types natifs de CAML. Mais il est facile de les créer en utilisant les types abstraits.

2.1 Implémentation homogène

Tous les sommets contiennent le même type d'objet.

Sans distinguer les feuilles des nœuds (internes) :

```
type 'a arbre = V | S of 'a * 'a arbre * 'a arbre;;

let ar = S(5, S(2, V, V), S(8, V, S(4, V, V)));;

let fg ar =
  match ar with
  | V -> failwith "non défini"
  | S(_,g,d) -> g;;

let feuilles ar =
  match ar with
  | V -> []
  | S(a,V,V) -> [a]
  | S(a,g,d) -> feuilles g @ feuilles d;;
```

En distinguant les feuilles des nœuds (internes) :

```
type 'a arbre = V | F of 'a | N of 'a * 'a arbre * 'a arbre;;

let ar = N(5, F(2), N(8, V, F(4)));;

let feuilles ar =
  match ar with
  | V -> []
  | F(a) -> [a]
  | N(a,g,d) -> feuilles g @ feuilles d;;
```

2.2 Implémentation hétérogène

Pour les arbres « hétérogènes », on a deux solutions :
- soit on crée un type somme *ad hoc* pour réunir les types différents en un seul type ;
- soit on crée un type arbre qui prend en compte l'hétérogénéité des types.

```
type ('f,'n) arbre = V | F of 'f | N of 'n * ('f,'n) arbre * ('f,'n) arbre;;

let a = N("+", F(2), F(7));;
```

3 Parcours d'arbres

3.1 Parcours en profondeur

Il y a trois parcours récursifs de l'arbre, suivant l'ordre dans lequel on traite chaque sommet :
- parcours préfixe : racine, puis arbre gauche, puis arbre droit ;
- parcours infixe : arbre gauche, puis racine, puis arbre droit ;
- parcours postfixe (ou suffixe) : arbre gauche, puis arbre droit, puis racine.

Sur l'exemple du début du chapitre, cela donne :
- préfixe :
- infixe :
- postfixe :

3.2 Parcours en largeur

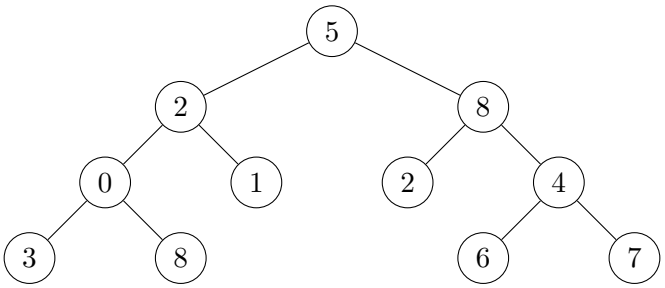
Cela consiste à parcourir l'arbre du haut vers le bas et sur chaque niveau, de gauche à droite.
Sur l'exemple précédent, cela donne :

3.3 Linéarisation d'un arbre entier

Pour enregistrer un arbre entier dans un fichier, on le transforme en un tableau ou une liste grâce à un parcours préfixe ou suffixe.

Proposition 4 *Si on distingue dans un parcours préfixe ou suffixe les feuilles des nœuds internes, alors il y a injection de l'ensemble des arbres binaires* ***entiers*** *dans l'ensemble des suites de feuilles ou de nœuds.*
Autrement dit, un type de parcours étant fixé, à toute suite de feuilles ou de nœuds correspond au plus un arbre entier.

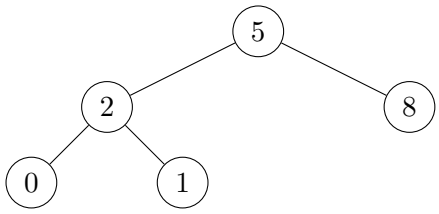
Connaissant la représentation linéaire de l'arbre et la nature du parcours (préfixe ou suffixe), on peut alors reconstituer l'arbre entier.
Exemple.



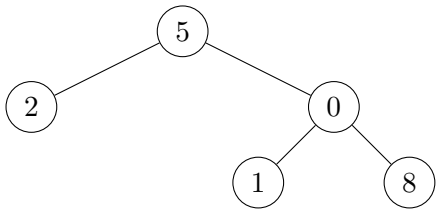
Le parcours préfixe avec différenciation donne la suite suivante :
N(5), N(2), N(0), F(3), F(8), F(1), N(8), F(2), N(4), F(6), F(7)

Dans l'autre sens,

— si on donne la suite : N(5), N(2), F(0), F(1), F(8)
alors on peut reconstituer l'arbre



— et si on donne la suite : N(5), F(2), N(0), F(1), F(8)
alors on obtient l'arbre



On notera que sans distinction des feuilles et des nœuds, ces deux derniers arbres ont la même représentation linéaire préfixe : $5, 2, 0, 1, 8$.

Remarque. Pour linéariser un arbre binaire non entier, il suffit d'ajouter des feuilles fictives aux noeuds ayant un seul fils : il devient alors entier.