1 Arbres binaires

1.1 Définition mathématique

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

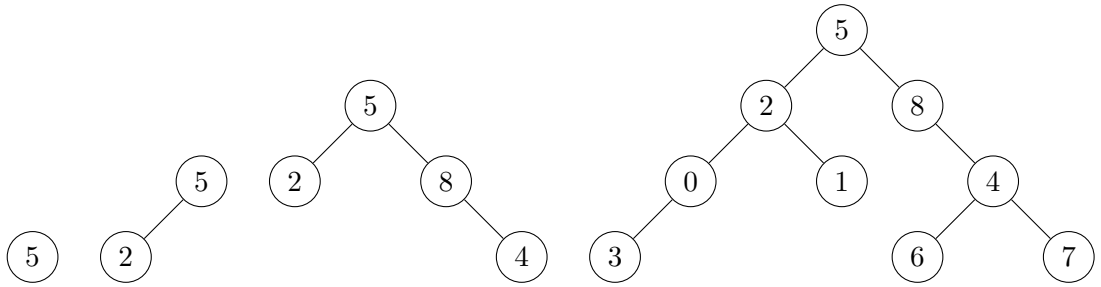
On définit récursivement l'ensemble $\mathscr{B}(E)$ des arbres binaires sur E :

— 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.

On représente les arbres de la façon suivante (dans le monde informatique, les arbres poussent vers le bas !) :



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

Habituellement, les éléments de E apparaissant dans l'arbre sont appelés les étiquettes des sommets.

1.2 Feuilles et noeuds

On définit récursivement les feuilles $F(B)$ et les noeuds $N(B)$ 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 noeud. 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 noeuds sont les sommets qui ont au moins un fils non vide.

Sur les représentations précédentes, les feuilles sont tout en bas de l'arbre.

Proposition 1 *Si B est un arbre possédant n noeuds, alors B possède au plus $n+1$ feuilles.*

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

1.3 Hauteur

On définit là encore récursivement la hauteur (ou profondeur) 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.

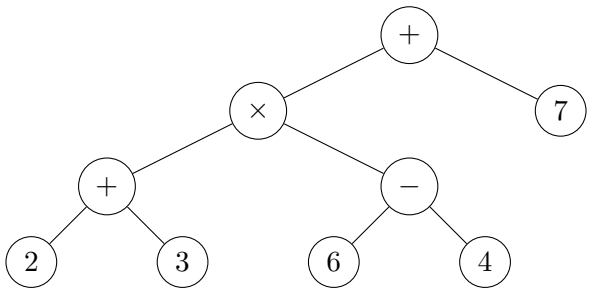
On parle aussi de la hauteur d'un sommet comme la longueur du chemin menant de la racine à ce sommet.

Proposition 2 *Si B est un arbre de hauteur h et possédant n noeuds, alors $h \leqslant n \leqslant 2^h - 1$.*

2 Exemples

2.1 Représentation d'une expression arithmétique

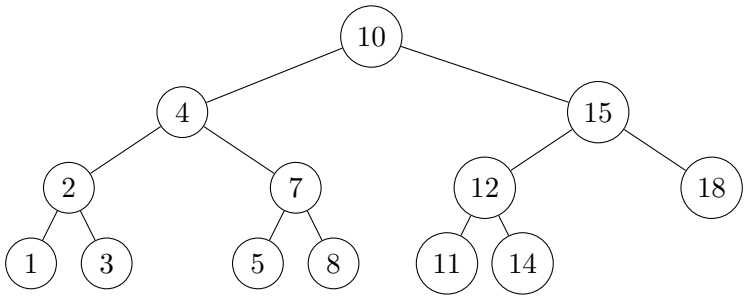
On peut représenter une expression arithmétique telle que $(2+3)\times(6-4)+7$ par un arbre dont les noeuds sont les opérateurs et les feuilles les nombres :



2.2 Arbres binaires de recherche

Un arbre binaire de recherche est un arbre tel que ses sommets soient dans un ensemble totalement ordonné et qui vérifie la propriété suivante :

pour tout noeud n de fils gauche G et fils droit D, les sommets de G sont inférieurs ou égaux à n et ceux de D sont strictement supérieurs à n.



Pour savoir si un objet appartient à un tel arbre, il suffit à chaque fois de comparer l'objet avec le sommet en cours : s'il est plus petit, on part à gauche, s'il est plus grand, on part à droite, jusqu'à le trouver ou arriver sur une feuille différente, auquel cas il n'est pas dans l'arbre.

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 : arbre gauche, puis arbre droit, puis racine.

Sur le quatrième arbre des exemples précédents, cela donne :

- préfixe : $5, 2, 0, 3, 1, 8, 4, 6, 7$;
- infixe : $3, 0, 2, 1, 5, 8, 6, 4, 7$;
- postfixe : $3, 0, 1, 2, 6, 7, 4, 8, 5$.

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 : $5, 2, 8, 0, 1, 4, 3, 6, 7$.

La réalisation d'un tel parcours se fait à l'aide d'une file, ce qui sera vu plus tard.

4 Représentation informatique

Un arbre binaire est une suite non contigüe de triplets de cases-mémoires : la première contient la donnée (l'élément de E), les deux suivantes contiennent les adresses des fils gauches et droits. Un arbre est finalement une sorte de liste à deux dimensions. Et réciproquement, une liste peut être vue comme un arbre dont tous les fils droits sont vides.

4.1 Implémentation homogène 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.

```
type 'a arbre = Vide | Sommet of 'a * 'a arbre * 'a arbre;;

let ar = Sommet(5,Sommet(2,Vide,Vide),Sommet(8,Vide,Sommet(4,Vide,Vide)));;

let fg ar =
  match ar with
  | Vide -> failwith "non defini"
  | Sommet(_,g,d) -> g;;

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

L'ennui de cette représentation est que les feuilles et les nœuds contiennent le même type d'objet. Or il arrive souvent que les objets attachés aux nœuds sont de type différent de ceux attachés aux feuilles.

4.2 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.

Par exemple, les arbres représentant les expressions arithmétiques peuvent être définis des façons suivantes :

```
type objet = Ent of int | Op of string;;

type arbre = Vide | Sommet of objet * arbre * arbre;;

let a = Sommet(Op("+"),
               Sommet(Ent(2), Vide, Vide),
               Sommet(Ent(7), Vide, Vide))
        );;
```

ou

```
type arbre = Vide | Feuille of int | Noeud of string * arbre * arbre;;

let a = Noeud( "+", Feuille(2), Feuille(7));;
```