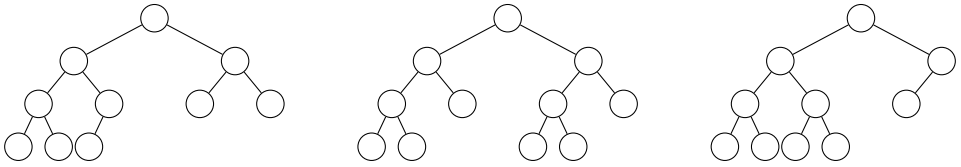
Outils logiques et algorithmiques – TD 9 – Arbres binaires

Exercise 1 (Arbres quasi-parfaits) Définitions :

- un arbre binaire parfait est un arbre binaire dont chaque niveau est rempli au maximum,
- un arbre binaire quasi-parfait est un arbre binaire dont chaque niveau sauf éventuellement le dernier est rempli au maximum, et dans lequel tous les nœuds du dernier niveau sont le plus à gauche possible.

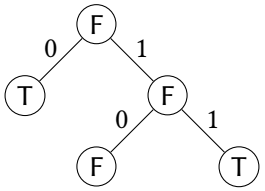
Ainsi, ci-dessous l'arbre de gauche est un arbre quasi-parfait de hauteur 4, mais l'arbre du milieu et l'arbre de droite ne sont pas quasi-parfaits.



1. Dessiner tous les arbres quasi-parfaits de hauteurs 0, 1, 2 et 3.
2. Combien y a-t-il d'arbres quasi-parfaits de hauteur k ?
3. Dans un arbre quasi-parfait de hauteur k, combien a-t-on de nœuds au minimum ? au maximum ?
4. Dans un arbre quasi-parfait de hauteur k, combien peut-on trouver de nœuds ayant un seul fils ? Donner un nombre minimum et un nombre maximum.
5. Dans un arbre quasi-parfait de hauteur k, combien a-t-on de feuilles au minimum ? au maximum ?

□

Exercise 2 (Arbres préfixes) Un arbre préfixe est un arbre binaire qui permet de représenter de manière compacte certains ensembles de mots sur l'alphabet $\{0, 1\}$. Pour cela, un mot binaire $a_1 \dots a_n$ est interprété comme une position dans l'arbre de la manière suivante : la lecture d'un 0 fait passer au fils gauche, et la lecture d'un 1 fait passer au fils droit. Chaque nœud de l'arbre est ensuite étiqueté par un booléen indiquant si le mot correspondant à sa position appartient ou non à l'ensemble. Ainsi, en notant T pour vrai et F pour faux, l'arbre suivant représente l'ensemble $\{0,\ 11\}$.



On adapte la représentation déjà vue des arbres binaires : E représente l'arbre vide, et $\mathsf{N}(a_1, b, a_2)$ représente le nœud étiqueté par le booléen b, et dont les sous-arbres gauche et droit sont a_1 et a_2. On définit par les équations récursives suivantes une fonction nbNœuds donnant le nombre de nœuds d'un arbre préfixe.

$$\begin{aligned} \mathsf{nbNœuds}(\mathsf{E}) &= 0 \\ \mathsf{nbNœuds}(\mathsf{N}(g, b, d)) &= 1 + \mathsf{nbNœuds}(g) + \mathsf{nbNœuds}(d) \end{aligned}$$

1. Dessiner un arbre préfixe représentant l'ensemble $\{\epsilon,\ 01,\ 10,\ 11,\ 010,\ 0111\}$.
2. Donner des équations récursives pour définir une fonction cardinal donnant le nombre de mots dans l'ensemble représenté par un arbre préfixe.
3. Donner le principe de raisonnement par récurrence associé aux arbres préfixes.
4. Démontrer que pour tout arbre préfixe p, $\mathsf{cardinal}(p) \leq \mathsf{nbNœuds}(p)$.

5. On propose le raisonnement suivant pour démontrer que pour tout arbre préfixe p, cardinal(p) < somme(p), où somme(p) désigne la somme des longueurs des mots contenus dans l'ensemble représenté par p :

   Soient g et d vérifiant les hypothèses de récurrence $cardinal(g) < somme(g)$ *et* $cardinal(d) < somme(d)$. *Par définition de cardinal on a*

   $$cardinal(N(g, b, d)) \leq 1 + cardinal(g) + cardinal(d)$$

   Comme $A < B$ *est équivalent à* $A \leq B-1$, *avec les deux premières hypothèses on en déduit* $cardinal(N(g, b, d)) \leq 1 + somme(g) - 1 + somme(d) - 1$
   c'est-à-dire $cardinal(N(g, b, d)) \leq somme(g) + somme(d) - 1$
   et donc $cardinal(N(g, b, d)) < somme(g) + somme(d)$
   Enfin $somme(g) + somme(d) \leq somme(N(g, b, d))$, *donc*

   $$cardinal(N(g, b, d)) < somme(N(g, b, d))$$

   Ainsi, par principe de récurrence on a bien $cardinal(p) < somme(p)$ *pour tout p.*

   Que penser de cette démonstration ? Que penser de la propriété démontrée ? Justifiez vos réponses.
6. Donner la définition d'un type Caml `prf` représentant les arbres préfixes.
7. Définir une fonction Caml `appartient : bool list -> prf -> bool` telle que l'appel `appartient` m a renvoie `true` si l'arbre préfixe a contient le mot représenté par la liste m.
8. Définir une fonction Caml `ajoute : bool list -> prf -> prf` telle que `ajoute` m a renvoie un arbre préfixe contenant le mot m en plus des mots déjà contenus par a.

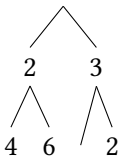
□

Exercise 3 (Arbres binaires de recherche) On rappelle la représentation des arbres binaires :

— E représente l'arbre vide

— N(a_1, n, a_2) représente le nœud contenant l'entier n, et ayant pour fils gauche l'arbre a_1 et pour fils droit l'arbre a_2.

On note $\mathcal{A}$ l'ensemble des arbres binaires.

1. Donner des équations pour définir une relation in(n, t) qui représente le fait que l'entier n apparaît dans un nœud de l'arbre t.
2. Donner les équations pour définir une fonction récursive infix qui étant donné un arbre binaire a, renvoie la séquence composée des entiers contenus dans l'arbre, parcouru de manière infixe : on parcourt d'abord le sous-arbre gauche, puis on ajoute l'entier du nœud puis on parcourt le sous-arbre droit.
   Par exemple, l'arbre 

   donnera la séquence : 426132.

On s'intéresse aux *arbres binaires de recherche*, définis par la caractérisation suivante :

— l'arbre vide E est un arbre binaire de recherche

— si a_1 et a_2 sont deux arbres binaires de recherche et un n un entier tel que tous les entiers de a_1 sont inférieurs ou égaux à n et tous les entiers de a_2 sont supérieurs ou égaux à n, alors N(a_1, n, a_2) est un arbre binaire de recherche.

On note $\mathcal{A}_r$ l'ensemble des arbres binaires de recherche.

3. On note $\mathsf{S}(n)$ l'arbre $\mathsf{N}(\mathsf{E}, n, \mathsf{E})$ qui contient un seul entier n. Est-ce un arbre binaire de recherche ? Les arbres suivants sont-ils des arbres binaires de recherche ?

   $\mathsf{N}(\mathsf{S}(1), 2, \mathsf{S}(3))$ $\mathsf{N}(\mathsf{E}, 2, \mathsf{N}(\mathsf{S}(1), 3, \mathsf{E}))$

4. Donner deux arbres binaires de recherche différents contenant la même suite de valeurs 1, 2, 3 et 4. Donner la valeur de la fonction infix pour ces arbres.
5. Démontrer par récurrence sur a que si l'arbre a est un arbre binaire de recherche alors la séquence $\mathsf{infix}(a)$ est triée, c'est-à-dire que les entiers apparaissent en ordre croissant.

La structure d'arbre binaire de recherche permet de chercher efficacement un élément.

6. Lorsque l'on recherche un élément n dans un arbre binaire de recherche de la forme $\mathsf{N}(a_1, x, a_2)$, est-il utile de fouiller l'ensemble de l'arbre ? dans quel cas faut-il chercher dans a_1 et dans quel cas dans a_2 ?
7. Proposer des équations récursives pour une fonction inabr telle que $\mathsf{inabr}(n, a)$ teste si l'entier n est présent dans l'arbre a, en supposant que a est un arbre binaire de recherche.
8. Montrer par récurrence sur a que

   $$\forall a \in \mathcal{A},\ \forall n \in \mathbb{N},\ \mathsf{inabr}(n, a) \implies \mathsf{in}(n, a)$$

9. Donner un contre-exemple à la proposition

   $$\forall a \in \mathcal{A},\ \forall n \in \mathbb{N},\ \mathsf{in}(n, a) \implies \mathsf{inabr}(n, a)$$

10. Montrer par récurrence sur a que

    $$\forall a \in \mathcal{A}_r,\ \forall n \in \mathbb{N},\ \mathsf{in}(n, a) \implies \mathsf{inabr}(n, a)$$

On peut également définir une procédure efficace pour vérifier qu'un arbre binaire est bien un arbre binaire de recherche.

11. Donner des équations récursives pour définir une fonction verif telle que $\mathsf{verif}(min, max, a)$ renvoie vrai si et seulement si a est un arbre binaire de recherche tel que tous les entiers x dans a vérifient $min \leq x \leq max$.
12. Comment faut-il choisir min et max pour que $\mathsf{verif}(min, max, a)$ teste si a est un arbre binaire de recherche ?
13. Démontrer par récurrence sur a que la fonction verif est correcte, c'est-à-dire que

    $$\forall a \in \mathcal{A},\ \forall m_1, m_2 \in \mathbb{N},\ \mathsf{verif}(m_1, m_2, a) \implies (a \in \mathcal{A}_r \wedge (\forall x,\ \mathsf{in}(x, a) \implies m_1 \leq x \leq m_2))$$

□