

LIFPF – Programmation fonctionnelle

TD3 – Structures inductives et types

Licence informatique UCBL – Printemps 2025

Exercice 1 : Arbres binaires

1. Rappeler la définition des arbres binaires d'entiers (`int`) avec données dans les nœuds. Donner des exemples d'arbres contenant 0, 1, 2 et 3 `int`.
2. On rappelle que les arbres binaires de recherche sont des arbres binaires tels que :
 - Le fils gauche ne contient que des éléments *plus petits* que le nœud courant.
 - Le fils droit ne contient que des éléments *plus grands* que le nœud courant.Est-il possible de définir les arbres binaires de recherche uniquement avec la déclaration `type` ? Si oui, donner la définition, sinon expliquer pourquoi et comment contourner le problème.
3. Discuter des différences entre fonctions et constructeurs de type.

Exercice 2 : Généralisation des arbres binaires

1. Définir un type paramétré `'a abr` pour des arbres binaires. Le paramètre de type correspondra au type des éléments.
2. Définir la fonction `renverse` qui renverse un arbre c'est-à-dire qui inverse les fils gauche et droit de chacun des nœuds de l'arbre. On prendra soin de bien écrire le type des paramètres et du résultat de la fonction. Ces types devront être *les plus généraux possibles*.
3. Pourquoi n'a-t-on pas que des paramètres de type différents pour chaque type dans la fonction précédente ?
4. On regarde à présent le cas particulier où le paramètre de type `'a` est une paire (`'b * 'c`). Donner une fonction `assoc` qui prend un élément `e` de type `'b` et renvoie *la liste des éléments* de type `'c` associés à `e` dans l'arbre.
5. Contrairement à la fonction `renverse`, la fonction `assoc` accède à la valeur des éléments. Pourtant son type reste paramétré, pourquoi ?
6. De même qu'en λ -calcul, OCaml peut prendre des fonctions en arguments. Donner le code de la fonction `insere` qui insère un élément dans un arbre binaire de recherche. Cette fonction prendra en argument *la fonction qui permettra de comparer les éléments de l'arbre*. Pour obtenir le bon type pour `insere`, il faudra réfléchir au type de cette fonction de comparaison par rapport au type des autres arguments de `insere`.
7. Donner une version alternative de `assoc` avec les hypothèses suivantes :
 - l'arbre est un arbre de recherche ordonné selon les clés (de type `'b`) ;
 - on a *au plus* une occurrence de chaque clé dans l'arbre ;
 - le résultat n'est plus une liste mais une `'c option`.