TME 4 : Tri par arbre binaire

Dans ce TME, on s'intéresse à définir un tri sur les listes d'entiers en utilisant la structure d'arbre binaire de recherche. Un arbre binaire de recherche est un arbre où, pour chaque noeud, les clés de l'enfant gauche sont plus petites que la clé du noeud courant, et les clés de l'enfant droit strictement plus grandes.

1 Arbres binaires

1. Définir un prédicat inductif `forall_btree {A:Set} (cond: A -> Prop) : btree A -> Prop` vérifiant que `cond` est vérifiée sur chaque clé de l'arbre binaire.
2. Définir une fonction d'insertion dans un arbre binaire de recherche.
3. Montrer que, sous certaines conditions, la fonction d'insertion préserve le prédicat défini précédemment.
4. En utilisant le prédicat `forall_btree`, définir un prédicat inductif `bstree` caractérisant les arbres binaires de recherche.
5. Comparer les principes d'induction définis par Coq pour les arbres binaires et les arbres binaires de recherche, via les commandes `Check btree_ind.` et `Check bstree_ind.`

   Dans la suite, si `H : bstree bt`, il est possible de faire une induction sur la structure des arbres binaires de recherche via la tactique `induction H.`
6. Montrer que l'insertion d'un élément dans un arbre binaire de recherche donne un arbre binaire de recherche.

2 Export vers les listes

1. Définir une fonction `list_of_btree` transformant un arbre binaire de recherche en une liste, via un parcours infixe.
2. Définir un prédicat inductif `forall_list`, similaire à celui défini sur les arbres binaires précédemment.
3. Montrer que si une condition est vraie sur toutes clés d'un arbre binaire, alors cette condition est encore vraie sur la liste créée par un appel à `list_of_btree`. Il pourra être intéressant d'introduire un lemme auxiliaire [1].
4. On réutilise le prédicat inductif `Sorted` caractérisant les listes triées et défini au TME précédent. Étant donné deux listes triées `l1` et `l2`, et un entier `x`, quelles conditions faut-il énoncer (en utilisant `forall_list`), pour montrer que la liste `l1 ++ x :: l2` est triée ? Le prouver. La tactique `lia` pourra être utilisée pour résoudre les buts arithmétiques.
5. Déduire des résultats précédents une preuve que `list_of_btree` renvoie bien une liste triée lorsqu'un arbre binaire de recherche lui est passé en argument.

3 Import de listes et conclusion

1. Définir une fonction `bst_of_list`.
2. Montrer que `bst_of_list` créée un arbre binaire de recherche.
3. Définir une fonction de tri utilisant `list_of_btree` et `bst_of_list`. Montrer que cette fonction renvoie une liste triée.
4. Que reste-t-il à prouver pour montrer que ce tri est correct ?

1. `Lemma forall_list_app : forall (A: Set) (cond: A -> Prop) (l l': list A), forall_list cond l -> forall_list cond l' -> forall_list cond (l ++ l').`