Projet Arbre 2-3-4 (brouillon !)

## Objectif

Ce projet a pour but d’implémenter en Coq une structure informatique : les arbres 2-3-4. L’implémentation inclura les spécifications de la structure en elle-même ainsi que les opérations de base de la structure (l’ajout / retrait d’un élément, tests d’appartenance…). Dans un second temps, l’objectif sera de formaliser et prouver des prédicats correspondant notamment aux propriétés de cette structure.

## Specification

### Structure

La structure des arbres 2-3-4 est une structure construite par induction avec 4 types d’élément :

Les feuilles, vides, les binodes contenant un élément et deux sous arbres. Les trinodes avec 2 éléments et 3 sous arbres et les quadnodes avec 3 éléments et 4 sous-arbres

Inductive tree (A: Type ) : Type :=

leaf : tree A

|binode: A -> tree A -> tree A -> tree A

|trinode : A -> A -> tree A -> tree A -> tree A -> tree A

|quadnode : A -> A -> A -> tree A -> tree A -> tree A -> tree A -> tree A.

### Opérations

* L’existence : exist (a:nat)(T:tree nat): bool retourne vrai si l’élément a est dans l’arbre T, faux sinon. Parcours de l’arbre de manière descendante jusqu’à trouver l’élément a en faisant des tests d’ordre sur les différents élémetn des nœuds.
* L’ajout add (a:nat) (T : tree nat): tree nat retourne un arbre qui contient tous les éléments de T et l’élément a. Parcours de l’arbre de manière descendante. Si on rencontre un quadnode on effectue un éclatement.
* Le retrait delete (a:nat)(T: tree nat): tree nat retourne un arbre qui contient tous les éléments de T sauf a. On utilise une fonction intermédiaire qui transforme l’arbre en liste puis on supprime l’élément de cette liste qu’on transforme ensuite en arbre.
* Test de l’équilibre

## Preuves

Les lemmes que nous pouvons démontrer à partir de cette spécification vont correspondre aux différentes propriétés des arbres 2-3-4, comme la hauteur équilibré et l’ordonancement des élément, et vont aussi correspondre aux « post  condition » des fonctions. Par exemple que si on ajoute un élément e dans un arbre T, e existe dans T après l’opération.

### Propriétés

La principale propriété des arbres 2-3-4 réside dans l’équilibre et l’ordonnancement des éléments.

* Un arbre reste équilibré par sa construction, lorsqu’on ajoute un élément :

forall T: tree nat ,forall X: nat, (is\_balanced\_height(T)=true) -> (is\_balanced\_height(add X T)=true).

* (OTHERS ?)

Pour démontrer ces propriétés difficiles il faut démontrer de plus petits lemme :

* Propriété arithmétique des hauteurs, le père d’un nœud dont la hauteur vaut h a une hauteur qui vaut h+1. Nous avons décomposé les preuves en fonctions du type de nœuds

forall T2 T1 T3:tree nat,forall n:nat,forall x: nat,(T1 = binode x T2 T3) /\ (n = (max (hauteurMax T2) (hauteurMax T3)) -> (hauteurMax T1) = (S n).

* Propriété de la structure, si un Arbre est équilibré alors ses sous arbres sont équilibrés

forall T2 T1 T3:tree nat,forall x: nat,(T1 = binode x T2 T3) /\ is\_balanced\_height(T1)=true -> is\_balanced\_height(T2)=true.

* (OTHERS)

### Post Conditions des opérations structurelles

* Opération add : forall T: tree nat ,forall X: nat, exist X (add X T) = true.
* Opération delete : forall T : tree nat, forall X : nat, exist X (delete X T) = false