Verificação Formal (2025/26)

Rocq (4)

Recorde a formalização de Binary Bearch Trees (BST) trabalhada na aula.

1. Uma das propriedades demonstradas foi que a função `insert`, que acrescenta um par chave-valor a uma árvore binária de procura, preserva o invariante de tipo das BST. No entanto, esta propriedade, só por si, não garante a correção da função.

   Para assegurar a correção da função `insert` temos que provar mais duas propriedades, que podem ser enunciadas pelos teoremas abaixo indicados. Prove cada um desses teoremas.

   Nota: Estas provas podem ser bastante longas e repetitivas (sucessivos "destructs" das guardas dos "ifs", seguidas da utilização das táticas `lia` ou `auto`). Para agilizar a prova utiliza a táctica `dall` definida na aula.

   (a)
   ```
   Theorem insert_BST_lookup_eq :
           forall (A : Type) (k : key) (d a : A) (t : tree A),
                                   BST t -> lookup d k (insert k a t) = a.
   ```

   (b)
   ```
   Theorem insert_BST_lookup_diff :
           forall (A : Type) (k k': key) (d a : A) (t : tree A),
                                BST t -> k <> k' -> bound k' t = true ->
                                lookup d k' (insert k a t) = lookup d k' t.
   ```

2. Pretende-se agora provar que toda a chave que aparece na BST pertence à lista gerada pela função de travessia.

   A biblioteca `List` já disponibiliza o predicado `In`, definido como uma função, juntamente com vários lemas úteis sobre esse predicado.

   ```
   Print In.
   (*           In =
                fun A : Type =>
                fix In (a : A) (l : list A) {struct l} : Prop :=
                  match l with
                  | [] => False
                  | b :: l' => b = a \/ In a l'
                  end
                      : forall [A : Type], A -> list A -> Prop              *)
   Search In.
   (*
   ...
   in_app_iff: forall [A : Type] (l l' : list A) (a : A), In a (l ++ l') <-> In a l \/ In a l'
   ...
   *)
   ```

   Prove o seguinte teorema:

   ```
   Theorem In_BST_inorderKeys : forall (A : Type) (k : key) (t : tree A),
                            BST t  ->  bound k t = true  ->  In k (inorderKeys t).
   ```