Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

SS 2021
11. 6. 2021

Functional Data Structures

Exercise Sheet 9

Exercise 9.1 Indicate Unchanged by Option

Write an insert function for red-black trees that either inserts the element and returns a new tree, or returns None if the element was already in the tree.

fun *ins'* :: *"'a::linorder ⇒ 'a rbt ⇒ 'a rbt option"*
lemma *"invc t ⟹ case ins' x t of None ⇒ ins x t = t | Some t' ⇒ ins x t = t'"*

Exercise 9.2 Joining 2-3-Trees

Write a join function for complete 2-3-trees: The function shall take two 2-3-trees l and r and an element x, and return a new 2-3-tree with the inorder-traversal $l\ x\ r$.

Write two functions, one for the height of l being greater, the other for the height of r being greater. The result should also be a complete tree, with height equal to the greater height of l and r.

height r greater:

fun *joinL* :: *"'a tree23 ⇒ 'a ⇒ 'a tree23 ⇒ 'a upI"*
lemma *complete_joinL*: *"⟦ complete l; complete r; height l < height r ⟧
⟹ complete (treeI (joinL l x r)) ∧ hI (joinL l x r) = height r"*

lemma *inorder_joinL*: *"⟦ complete l; complete r; height l < height r ⟧
⟹ inorder (treeI (joinL l x r)) = inorder l @x # inorder r"*

height l greater:

fun *joinR* :: *"'a tree23 ⇒ 'a ⇒ 'a tree23 ⇒ 'a upI"*
lemma *complete_joinR*: *"⟦ complete l; complete r; height l > height r ⟧ ⟹
complete (treeI (joinR l x r)) ∧ hI(joinR l x r) = height l"*

lemma *inorder_joinR*: *"⟦ complete l; complete r; height l > height r ⟧ ⟹ inorder (treeI (joinR l x r)) = inorder l @x # inorder r"*

Combine both functions.

fun *join* :: "'a tree23 ⇒ 'a ⇒ 'a tree23 ⇒ 'a tree23"
lemma "⟦ complete l; complete r ⟧ ⟹ complete (join l x r)"

lemma "⟦ complete l; complete r ⟧ ⟹ inorder (join l x r) = inorder l @x # inorder r"

Homework 9.1 List to RBT

Submission until Thursday, June 17, 23:59pm.

In this task you are to define a function *list_to_rbt* which constructs a red-black tree that contains the members of a given list.

Hint:

This function could be constructed by composing two functions. The first is a function that constructs an almost complete binary tree from a list (see the function *balance_list* in *HOL−Data_Structures.Balance*) – a tree is almost complete if its minimum height and its height differ by at most 1 (see *acomplete* in the file *HOL−Library.Tree*)

The second function, which is *mk_rbt*, constructs the equivalent red-black tree to a given almost complete binary tree:

```
fun mk_rbt :: "'a tree ⇒ 'a rbt" where
  "mk_rbt ⟨⟩ = ⟨⟩"
| "mk_rbt ⟨l, a, r⟩ = (let
    l'=mk_rbt l;
    r'=mk_rbt r
  in
    if min_height l > min_height r then
      B (paint Red l') a r'
    else if min_height l < min_height r then
      B l' a (paint Red r')
    else
      B l' a r'
  )"
```

fun *list_to_rbt* :: "'a list ⇒ 'a rbt"

Hint: If you follow the hint above and construct the function *list_to_rbt* by composing the functions *mk_rbt* and *balance_list*, then a good idea to prove the theorems required below is to prove lemmas about *mk_rbt* applied to almost complete trees, and then leverage the results to get the theorems about *list_to_rbt*

Warmup

Show the following alternative characterization of almost complete:

lemma *acomplete_alt*:

"*acomplete t* $\longleftrightarrow$ *height t* = *min_height t* $\vee$ *height t* = *min_height t* + *1*"

The Easy Parts

Show that the inorder traversal of the tree constructed by *list_to_rbt* is the same as the given list:

lemma *mk_rbt_inorder*: "*Tree2.inorder* (*list_to_rbt xs*) = *xs*"

Show that the color of the root node is always black:

lemma *mk_rbt_color*: "*color* (*list_to_rbt xs*) = *Black*"

Medium Complex Parts

Show that the returned tree satisfies the height invariant.

lemma *mk_rbt_invh*: "*invh* (*list_to_rbt xs*)"

Hint: Use Isar to have better control on when to unfold with *acomplete_alt*, and when to use (e.g. to discharge the premises of the IH). Also, a useful lemma to prove is *acomplete ?t* $\Longrightarrow$ *bheight* (*mk_rbt ?t*) = *min_height ?t*.

The Hard Part (Bonus, 5 points)

Show that the returned tree satisfies the color invariant.

lemma *mk_rbt_invc*: "*invc* (*list_to_rbt t*)"

Hint: A useful lemma is *acomplete ?t* $\Longrightarrow$ *invc* (*mk_rbt ?t*). To prove it, combine case splitting, automation and manual proof (Isar, aux-lemmas), in order to deal with the multiple cases without a combinatorial explosion of the proofs.