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

SS 23
20. 6. 2023

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 2-3 Tree to Red-Black Tree

Submission until Monday, June 26, 23:59pm.
In this task you are to define a function *mk_rbt* which constructs a red-black tree that contains the members of a given 2-3 tree.

fun *mk_rbt* :: *"'a tree23 ⇒ 'a rbt"*

Show that the inorder traversal of the constructed tree is the same as the original:

lemma *mk_rbt_inorder_btree*: *"Tree2.inorder (mk_rbt t) = Tree23.inorder t"*

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

lemma *mk_rbt_color_btree*: *"color (mk_rbt t) = Black"*

Show that the returned tree satisfies the height invariant:

lemma *mk_rbt_invh_btree*: *"Tree23.complete t ⟹ invh (mk_rbt t)"*

Show that the returned tree satisfies the color invariant.

lemma *mk_rbt_invc_btree*: *"invc (mk_rbt t)"*

Homework 9.2 Red-Black Tree Property

Submission until Monday, June 26, 23:59pm.
In a red-black tree, all paths from a root to any leaf traverse the same number of black nodes. In this exercise you are required to prove that. Consider the following function:
bhs ⟨⟩ = {0}
bhs ⟨l, (uu, c), r⟩ = (let H = bhs l ∪ bhs r in if c = Black then Suc ‘ H else H)

Note that *f ‘ s* denotes the image of a function *f* on a set *s*. With that in mind, the above function encodes the set of numbers of black nodes traversed in all paths from the root to any of the leaves. Prove the following lemma, which formalises the fact that all paths starting at the root and ending at a leaf have the same number of black nodes.

theorem *invh_iff_bhs*: *"invh t ⟷ bhs t = {bheight t}"*