Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Simon Roßkopf

SS 24
10. 5. 2024

Functional Data Structures

Exercise Sheet 4

Exercise 4.1 List Elements in Interval

Write a function to in-order list all elements of a BST in a given interval. I.e. *in_range t u v* shall list all elements x with $u \le x \le v$. Write a recursive function that does not descend into subtrees that definitely contain no elements in the given range.

fun *in_range* :: *"'a::linorder tree ⇒ 'a ⇒ 'a ⇒ 'a list"*

Show that you list the right set of elements

lemma *"bst t ⟹ set (in_range t u v) = {x∈set_tree t. u≤x ∧ x≤v}"*

Show that your list is actually in-order

lemma *"bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"*

Exercise 4.2 Enumeration of Trees

Write a function that generates the set of all trees up to a given height. Show that only trees up to the specified height are contained.

fun *enum* :: *"nat ⇒ unit tree set"*
lemma *enum_sound*: *"t ∈ enum n ⟹ height t ≤ n"*

Show the other direction, i.e. that all trees of the specified height are contained.

lemma *enum_complete*: *"height t ≤ n ⟹ t ∈ enum n"*

lemma *enum_correct*: *"enum h = {t. height t ≤ h}"*

Homework 4.1 Min Annotated Trees

Submission until Thursday, 16 May, 23:59pm.

In this homework, we will develop an augmented binary search tree that stores the minimum element in the tree at the root. This auxiliary information enables the implementation of more efficient membership queries.

datatype *'a mtree = Leaf | Node "'a mtree" (minimum: 'a) (element: 'a) "'a mtree"*

Define a function to return the set of elements in such a tree

fun *set_mtree2* **where**

Define a recursive function that charcterises the invariant on the tree: the binary search tree property and the correct minimum node labels. Note: you should not use the function *Min*.

fun *mbst* :: *"'a::{linorder,zero} mtree ⇒ bool"* **where**

To confirm that the invariant characterises what it is supposed to, define a function which computes the minimum value in an ordered tree. This function should be recursive on the given tree. You can assume the tree is an ordered tree.

fun *min_val* :: *"'a::{linorder,zero} mtree ⇒ 'a"* **where**

Show that this function returns the label of the root of a given tree, if the tree is *mbst*.

lemma *mbst_minval*: *"mbst (Node l m a r) ⟹ min_val (Node l m a r) = m"*

Define the insert function for this tree. Note: it has to correctly update the node with the correct minimum labels.

fun *mins* :: *"'a::{linorder,zero} ⇒ 'a mtree ⇒ 'a mtree"* **where**

Now show that *mins* preserves the invariant. Hint: you will need a lemma showing that *mins* actually inserts the element in the set of elements in the tree.

lemma *mbst_mins*: *"mbst t ⟹ mbst (mins x t)"*

Define the membership query function and show it correct. Note: the function has to exploit the augmented minimum value!

fun *misin* :: *"'a::linorder ⇒ 'a mtree ⇒ bool"* **where**

lemma *misin_set*: *"mbst t ⟹ misin x t ⟷ x∈set_mtree2 t"*

Specify a function that lists the elements within a given range in a given augmented tree and show that it lists the right elements. Again, the function must exploit the augmented minimum values.

fun *mtree_in_range* :: "$'a$::*linorder mtree* $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ *list*"

Show that the function lists the right set of elements

lemma *mbst_range*: "*mbst t* $\Longrightarrow$ *set* (*mtree_in_range t u v*) $= \{x \in$ *set_mtree2 t*. $u \leq x \wedge x \leq v\}$"