אביגיל דאנש

אופק ימיני

2 עבודה עם עצים-ב: הפונקציה שלנו, בצורה בה אנו כתבנו אותה, אינה זקוקה לשום שינוי על מנת שתעבוד לעצי חיפושבינארים שכל אבריהם הם מספרים של נקודה צפה. זאת כיוון שרוב החישוב התבצע ע"י הפונקציה filter (נלמדה בתרגול) של ספריית list, שהיא קיבלה את הפונק <,> שעובדים גם לfloat.

.1

Base: for x=Empty -> len(x) = 0 From the definition of len function.

Len(x app y) = len(y) from def of append function in case x is empty.

In conclusion:

Len(x)+len(y) = 0 + len(y) = len(y) = Len(x app y)

finally we prove the base the induction.

Step:

We assume that len(x) + len(y) = len(x app y). we will prove that:

Len(cons(h,x) app y) = len(cons(h,x))+ln(y)

Len(cons(h,x) app y) = len(cons(h, (x app y))) = 1+len(x app y) = 1+ len(x) +len(y)

First step is from app definition, sec step is from len definition and third is from the induction assume.

len(cons(h,x))+ln(y) = 1+len(x)+len(y) according to len definition.

So finally we can see that Len(cons(h,x) app y) = len(cons(h,x))+ln(y) like we wanted.

we proved the induction step.

3.

Base: for t=Empty we get that the height of t is 0 (def of height). We have no root or leaves so the claim is right in empty way.

Step: lets call the root of the tree k, lets assume that the left and right child of k are maintain the claim, now lets proof it for node(k,left,right) = t.

Height(t) = height(node(k,left,right)) = 1+max(height left)(height right) >= 1+ the longest track from k child's to the leaf. (step 1 height defintion, step 2 from max definition & induction assumption)

rl = longest track from the roof to the leaf.

kl = the longest track from k child's to the leaf.

Now lets prove that 1 + kl = rl.

Its true because the distance between k to its child is 1, so rl = kl+1 and we already have kl from the induction assume.

So height(t) >= longest path from the root to the leaf = rl.

we proved the induction step.

4.

a. disproof, let exp=not(x) and then

num_of_vars(exp) =   num_of_vars(not(x)) = num_of_vars(x) = 1

num_of_connectives(exp) + 1 = num_of_connectives(not(x)) + 1 = num_of_connectives(x) + 1 + 1 = 0 + 1 + 1 = 2

2=!1

b. num_of_vars(exp) = num_of_connectives(exp) + 1

base: let assume that x is Var type

num_of_vars(x) = 1 = num_of_connectives(x) + 1 = 0+1=1

Step: we assume that the claim works for every exp a, b and we will prove that the claim works for a&b,a|b.

num_of_vars(and a b) = (num_of_vars a) + (num_of_vars b) = num_of_connectives (a) + 1 + num_of_connectives(b) + 1 = 2 + num_of_connectives (a) + num_of_connectives (b)

(first step is the num_of _var def, sec is from the induction assume)

num_of_connectives(and a b) + 1 = (num_of_connectives x) + (num_of_connectives y) +2

(def of num_of_connectives)

num_of_vars(or a b) = (num_of_vars a) + (num_of_vars b) = num_of_connectives (a) + 1 + num_of_connectives(b) + 1 = 2 + num_of_connectives (a) + num_of_connectives (b)

(first step is the num_of _var def, sec is from the induction assume)

num_of_connectives(or a b) + 1 = (num_of_connectives x) + (num_of_connectives y) + 2