Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

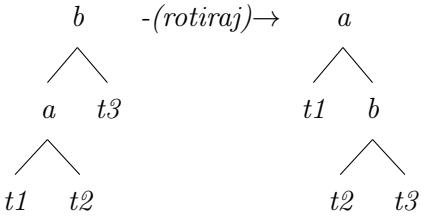
Šifra predmeta: R265
27.05.2024.

Uvod u interaktivno dokazivanje teorema

Vežbe 12

Zadatak 1 *Desni linearni lanac*

Data je funkcija *rotiraj* koja rotira drvo na sledeći način:



fun *rotiraj* :: *'a tree* ⇒ *'a tree* **where**
rotiraj ⟨⟨*t1*, *a*, *t2*⟩, *b*, *t3*⟩ = ⟨*t1*, *a*, ⟨*t2*, *b*, *t3*⟩⟩
| *rotiraj x* = *x*

thm *rotiraj.simps*

1. *(2p)*

   Definisati funkciju *bps* :: (*'a*::*linorder*) *tree* ⇒ *bool* koja proverava da li je stablo binarno pretraživačko. Za relaciju poretka koristiti <, a za konstruisanje skupa elemenata stabla koristiti *set-tree*.

   Sledeća dva izraza moraju da se evaluiraju u *True*:

   value *bps* ⟨⟨⟨⟩, *1*::*nat*, ⟨⟩⟩, *2*, ⟨⟩⟩, *5*, ⟨⟨⟩, *6*, ⟨⟩⟩⟩

   value ¬ *bps* ⟨⟨⟨⟩, *3*::*nat*, ⟨⟩⟩, *2*, ⟨⟩⟩, *5*, ⟨⟨⟩, *6*, ⟨⟩⟩⟩

2. *(2p)*

   Pokazati da je funkcija *rotiraj* korektna, tj. da funkcija *rotiraj* ne ruši svostvo binarnog pretraživačkog stabla i da skup čvorova ostaje nepromenjen nakon primene funkcije *rotiraj*.

   lemma *bps t* ⟹ *bps* (*rotiraj t*)

   lemma *set-tree* (*rotiraj t*) = *set-tree t*

3. *(3p)*

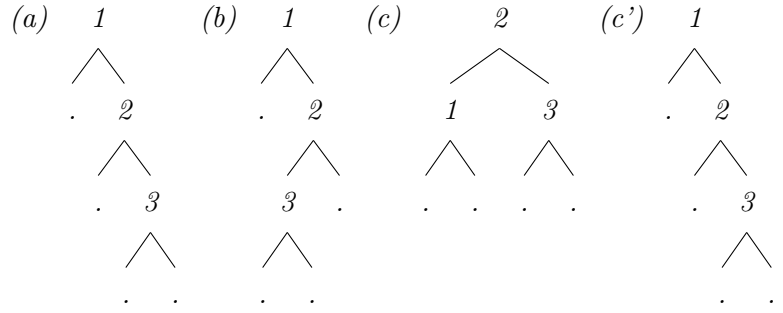
   Binarno stablo ima svojstvo *desnog linearnog lanaca* kada svaki unutrašnji čvor ima samo desnog potomka, tj. binarno stablo koje predstavlja listu duž desne strane. Definisati funkciju *dll* :: *'a tree* ⇒ *bool* koja proverava da li je binarno stablo desni linearni lanac. *Savet*: Izbegavati *if−then−else* izraze, moguće je koristiti *pattern matching*.

   Sledeći izrazi moraju se evaluirati u *True* (pogledati i odgovarajuće slike).

value *dll* $\langle\langle\rangle, 1::nat, \langle\langle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle\rangle$ — (slika (a))

value $\neg$ *dll* $\langle\langle\rangle, 1::nat, \langle\langle\rangle, 3, \langle\rangle\rangle, 2, \langle\rangle\rangle\rangle$ — (slika (b))

value *dll* (*rotiraj* $\langle\langle\rangle, 1::nat, \langle\rangle\rangle, 2, \langle\langle\rangle, 3, \langle\rangle\rangle\rangle\rangle$) — (slika (c) — pre rotiranja; slika (c') – nakon rotiranja).



4. (2p)

Definisati funkciju *rotiraj1* :: *'a tree* $\Rightarrow$ *'a tree* koja obilazi stablo i vrši prvu moguću rotaciju.

Sledeći izraz mora da se evaluira u *True* (nakon 3 primene *rotiraj1* nema promena u stablu):

value (*rotiraj1* ^^ *3*) $\langle\langle\rangle, 3::nat, \langle\langle\rangle, 5::nat, \langle\langle\rangle, 6::nat, \langle\rangle\rangle\rangle\rangle, 1::nat, \langle\langle\rangle, 2::nat, \langle\rangle\rangle\rangle$
$= (rotiraj1\ \hat{}\hat{}\ 10)\ \langle\langle\rangle, 3::nat, \langle\langle\rangle, 5::nat, \langle\langle\rangle, 6::nat, \langle\rangle\rangle\rangle\rangle, 1::nat, \langle\langle\rangle, 2::nat, \langle\rangle\rangle\rangle$

5. (2p)

Želimo da dokažemo činjenicu da je najviše potrebno *size t* rotacija da bi binarno stablo postalo desni linearni lanac.

Kako bi ovo pokazali moramo definisati meru zaustavljanja, tj. funkciju koja linearno opada u odnosu na broj primena funkcije *rotiraj1* i dostiže *0* kada binarno stablo postane desni linearni lanac. Jedna takva funkcija može biti definisana kao:

fun *mera* :: *'a tree* $\Rightarrow$ *nat* **where**

mera $\langle\rangle$ = *0*

| *mera* $\langle l, a, r\rangle$ = *size l* + *mera r*

Pokazati da stablo *t* koje je desni linearni lanac uzima vrednost mere zaustavljanja 0.

lemma *mera-0*: *dll t* $\longleftrightarrow$ *mera t* = *0*

6. (2p)

Pokazati da se mera smanjuje za 1 nakon primene funkcije *rotiraj1*.

lemma *mera-rotiraj-1*: *mera* (*rotiraj1 t*) = *mera t* − *1*

Pokazati da se mera smanjuje za *n* nakon *n* primena funkcija *rotiraj1*.

lemma *mera-rotiraj-n*: *mera* ((*rotiraj1* ^^ *n*) *t*) = *mera t* − *n*

7. *(2p)*

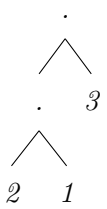
   Konačno pokazati da:

   theorem *dll-rotiraj*: $\exists\, n \leq size\ t.\ dll\ ((rotiraj1\ \hat{}\hat{}\ n)\ t)$

Zadatak 2 *Fold-ovanje nad stablima*

1. *(2p)* ——— / 10 p.

   Definisati tip *'a ldrvo* koji predstavlja binarno stablo koje čuva podatke samo u listovima. Definisati instancu *test-drvo* :: *nat ldrvo* tako da predstavlja sledeće drvo



2. *(1p)*

   Definisati funkciju *lkd* :: *'a ldrvo* $\Rightarrow$ *'a list* koja vraća listu elemenata obilaskom stabla metodom levo-koren-desno.

   value *lkd test-drvo* = [*2*,*1*,*3*] — (Ovaj izraz mora da se evaluira u *True*)

3. *(2p)*

   Jedan način foldovanja nad elementima ovog stabla je *fold f* (*lkd d*) *acc*. Definisati funkciju *fold-ldrvo* :: (*'a* $\Rightarrow$ *'b* $\Rightarrow$ *'b*) $\Rightarrow$ *'a ldrvo* $\Rightarrow$ *'b* $\Rightarrow$ *'b* rekurzivno po strukturi drveta koja vraća isti rezultat kao i *fold f* (*lkd d*) *acc*.

   Napomena: Definisanje funkcije *fold-ldrvo* preko *fold* funkcije na iznad naveden način se ne priznaje kao tačno rešenje, već funckiju *fold-ldrvo* morate definisati primitivnom rekurzijom.

   value *fold-ldrvo* (+) *test-drvo 0* = *6* — (Ovaj izraz mora da se evaluira u *True*)

4. *(2p)*

   Pokazati da je *fold f* (*lkd d*) *acc* = *fold-ldrvo f d acc*.

5. *(1p)*

   Definisati funkciju *obrni* :: *'a ldrvo* $\Rightarrow$ *'a ldrvo* koja obrće stablo kao u ogledalu.

   value *ldk* (*obrni test-drvo*) = [*3*,*1*,*2*] — (Ovaj izraz mora da se evaluira u *True*)

6. *(2p)*

   Pokazati da je lkd poredak obrnutog drveta isto što i obrnuti lkd poredak početnog drveta.

Zadatak 3 *Odsecanje liste*

1. *(1p)*
   Definisati funkciju *sadrzi* :: *'a* ⇒ *'a list* ⇒ *bool* koja ispituje da li se element nalazi u listi.

   Sledeća dva izraza se moraju evaluirati u *True*:

   value *sadrzi 5* [*1*,*6*,*2*,*5*,*3*,*4*::*nat*] = *True* — (Sadrži se)
   value *sadrzi 8* [*1*,*6*,*2*,*5*,*3*,*4*::*nat*] = *False* — (Ne sadrži se)

2. *(2p)*

   Pokazati da je funkcija *sadrzi* invarijantna u odnosu na obrtanje liste.

   Savet: Definisati pomoćnu lemu koja opisuje da li se element nalazi u listi kojoj znamo početak i poslednji element.

3. *(1p)*
   Definisati funkciju *razliciti* :: *'a list* ⇒ *bool* koja ispituje da li su svi elementi liste međusobno različiti.

   Sledeća dva izraza se moraju evaluirati u *True*:

   value *razliciti* [*1*,*6*,*2*,*5*,*3*,*4*::*nat*] = *True* — (Svi različiti)
   value *razliciti* [*1*,*4*,*2*,*5*,*2*,*4*::*nat*] = *False* — (Ima istih)

4. *(2p)*

   Pokazati da je funkcija *razliciti* invarijantna na obrtanje liste.

5. *(3p)*
   Pomoću funkcija *fold* i *foldr* definisati funkcije *duzina-fold* i *duzina-foldr*, respektivno, koje računaju dužinu liste.

   Sledeći izrazi se moraju evaluirati u *True*.

   value *duzina-fold* [] = *0* — (Prazna lista)
   value *duzina-fold* [*1*,*2*,*3*::*nat*] = *3* — (Lista dužine 3)
   value *duzina-foldr* [] = *0* — (Prazna lista)
   value *duzina-foldr* [*1*,*2*,*3*::*nat*] = *3* — (Lista dužine 3)

6. *(4p)*

   Pokazati da su *duzina-fold* i *duzina-foldr* korektne, tj. da daju isti rezultat kao i funkcija *length*.

7. *(3p)*
   Definisati funkciju *iseci* :: *'a list* ⇒ *nat* ⇒ *nat* ⇒ *'a list*. Za listu $xs = [x_0, \ldots, x_n]$, poziv *iseci xs s l* vraća listu: $[x_s, \ldots, x_{s+l-1}]$. Ako su vrednosti s i l van opsega funkcija *iseci* vraća kraću ili praznu listu.

   Savet: Koristite opštu rekurziju i *pattern matching*, umesto *if−then−else* izraza. Npr. umesto *f s* = (*if s* = *0 then e1 else e2*) koristiti *f 0* = *e1* i *f s* = *e2*.

   Sledeći izrazi se moraju evaluirati u *True*.

value *iseci* [*0*,*1*,*2*,*3*,*4*,*5*,*6*::*int*] *2 3* = [*2*,*3*,*4*] — (U opsegu)

value *iseci* [*0*,*1*,*2*,*3*,*4*,*5*,*6*::*int*] *2 10* = [*2*,*3*,*4*,*5*,*6*] — (Dužina van opsega)

value *iseci* [*0*,*1*,*2*,*3*,*4*,*5*,*6*::*int*] *10 10* = [] — (Početni indeks van opsega)

8. (2p)

Pokazati da važi: *iseci xs s l1* @ *iseci xs* (*s* + *l1*) *l2* = *iseci xs s* (*l1* + *l2*)

9. (2p)

Pokazati da odsecanje liste zadržava invarijantnost o različitim elementima, tj. ako su elementi liste različiti, onda će biti različiti i elementi isečka.