Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
13.05.2024.

Uvod u interaktivno dokazivanje teorema

Vežbe 10

Zadatak 1 *Tip: 'a drvo*

Definisati algebarski tip *'a drvo* koji predstavlja binarno drvo.

datatype *'a drvo = List*
  | Cvor 'a drvo 'a 'a drvo

Definisati funkciju *zbir* :: *nat drvo* ⇒ *nat* primitivnom rekurzijom koja računa zbir elemenata drveta tipa *nat drvo*. Da li je moguće definisati ovu funkciju nad tipom *'a drvo*?

primrec *zbir* :: *nat drvo* ⇒ *nat* **where**
  zbir List = 0
| *zbir (Cvor lt x rt) = zbir lt + x + zbir rt*

Definisati bilo koju instancu *test-drvo* tipa *nat drvo*. Proveriti da li funkcija *zbir* daje dobar rezultat kada se primeni na *test-drvo*.

definition *test-drvo* :: *nat drvo* **where**
  test-drvo ≡ Cvor (Cvor List 1 List) 3 (Cvor (Cvor List 4 List) 2 (Cvor List 3 List))

value *zbir test-drvo*

Definisati funkciju *sadrzi* :: *'a drvo* ⇒ *'a* ⇒ *bool* primitivnom rekurzijom koja proverava da li se dati element nalazi u drvetu. Takođe, testirati funkciju nad instancom *test-drvo*.

primrec *sadrzi* :: *'a drvo* ⇒ *'a* ⇒ *bool* **where**
  sadrzi List a ⟷ False
| *sadrzi (Cvor ld x dd) a ⟷ sadrzi ld a ∨ x = a ∨ sadrzi dd a*

value *sadrzi test-drvo 3*
value *sadrzi test-drvo 5*

Definisati funkciju *skup* :: *'a drvo* ⇒ *'a set* primitivnom rekurzijom koja proverava da li se dati element nalazi u drvetu. Takođe, testirati funkciju nad instancom *test-drvo*.
Pronaći vezu između funkcija *skup* i *sadrzi*. Formulisati i dokazati tu lemu.

primrec *skup* :: *'a drvo* ⇒ *'a set* **where**
  skup List = {}
| *skup (Cvor ld x dd) = skup ld ∪ {x} ∪ skup dd*

value *skup test-drvo*

lemma *pripada-skup-sadrzi*:
  shows *a ∈ skup d ⟷ sadrzi d a*
  by (*induction d*) *auto*

Zadatak 2 *Obilazak stabla*

Definisati funkciju *infiks* koja vraća listu čvorova stabla u infiksnom poretku.

primrec *infiks* :: $'a$ *drvo* $\Rightarrow$ $'a$ *list* **where**
infiks List = []
| *infiks* (*Cvor ld x dd*) = (*infiks ld*) @ [*x*] @ (*infiks dd*)

Pokazati korektnost ove funkcije. Dve invarijante:

1. *Skup elemenata infiksnog obilaska drveta i skup elemenata drveta ostaju isti.*
2. *Multiskup elemanata infiksnog obilaska drveta i skupa elemenata drveta ostaju isti.*

Savet: Tip multiskupa: $'a$ *multiset*, prazan multiskup se definiše kao {#}, multiskup sa jednim elementom {#*x*#}, unija multiskupova je operator +.

lemma *set-infiks-skup*[*simp*]:
shows *set* (*infiks d*) = *skup d*
by (*induction d*) *auto*

primrec *multiskup* :: $'a$ *drvo* $\Rightarrow$ $'a$ *multiset* **where**
multiskup List = {#}
| *multiskup* (*Cvor ld x dd*) = *multiskup ld* + {#*x*#} + *multiskup dd*

lemma *mset-indiks-multiskup* [*simp*]:
shows *mset* (*infiks d*) = *multiskup d*
by (*induction d*) *auto*

Definisati efikasnu implementaciju infiksnog obilaska drveta *infiks-opt* i pokazati da je ekvivalentna funkciju *infiks*.

primrec *infiks-opt'* :: $'a$ *drvo* $\Rightarrow$ $'a$ *list* $\Rightarrow$ $'a$ *list* **where**
infiks-opt' List xs = *xs*
| *infiks-opt'* (*Cvor ld x dd*) *xs* = *infiks-opt' ld* (*x* # *infiks-opt' dd xs*)

definition *infiks-opt* :: $'a$ *drvo* $\Rightarrow$ $'a$ *list* **where**
infiks-opt xs = *infiks-opt' xs* []

value *infiks-opt test-drvo*

lemma *infiks-opt'-append*:
shows *infiks-opt' d xs* @ *ys* = *infiks-opt' d* (*xs* @ *ys*)
by (*induction d arbitrary*: *xs*) *auto*

lemma *infiks-infiks-opt*:
shows *infiks d* = *infiks-opt d*
unfolding *infiks-opt-def*
by (*induction d*) (*auto simp add*: *infiks-opt'-append*)

Zadatak 3 *Binarno pretraživačko stablo.*

Definisati predikat *sortirano* nad binarnim stablom tipa ($'a$::*linorder*) *drvo* koji ukazuje na to da li je stablo pretraživačko ili nije. Definisati instancu *test-drvo-sortirano* nad tipom *nat drvo* koja predstavlja binarno pretraživačko stablo. Testirati funkciju *sortirano* nad instancom *test-drvo* i *test-drvo-sortirano*. Zapisati i dokazati vezu između funkcije *sortirano* i *infiks*.

primrec *sortirano* :: ($'a$::*linorder*) *drvo* $\Rightarrow$ *bool* **where**

$sortirano\ List \longleftrightarrow True$
$|\ sortirano\ (Cvor\ ld\ x\ dd) \longleftrightarrow (\forall\ a \in skup\ ld.\ a \leq x)$
$\wedge (\forall\ a \in skup\ dd.\ x \leq a)$
$\wedge\ sortirano\ ld$
$\wedge\ sortirano\ dd$

value *infiks test-drvo*
value *sortirano test-drvo*

definition *test-drvo-sortirano* :: *nat drvo* **where**
test-drvo-sortirano = Cvor (Cvor List 1 (Cvor List 2 List)) 3 (Cvor (Cvor List 3 List) 4 List)

value *infiks test-drvo-sortirano*
value *sortirano test-drvo-sortirano*

lemma *sortirano-sorted-infiks*:
shows $sortirano\ d \Longrightarrow sorted\ (infiks\ d)$
by *(induction d) (auto simp add: sorted-append order-trans)*

Primitivnom rekurzijom definisati funkciju $ubaci :: 'a::linorder \Rightarrow 'a\ drvo \Rightarrow 'a\ drvo$ koja ubaciju element u binarno pretraživačko drvo.

Pokazati da važe invarijante:

1. *Element će se nalaziti u drvetu nakon što se ubaci.*
2. *Skup elemenata drveta nakon ubacivanja elementa se proširuje za taj element.*
3. *Multiskup elemenata drveta nakon ubacivanja elementa se proširuje za taj element.*
4. *Zbir elemenata drveta nakon ubacivanja elementa se povećava za njegovu vrednost.*
5. *Nakon ubacivanja elementa u pretraživačko drvo, drvo ostaje pretraživačko.*

primrec $ubaci :: 'a::linorder \Rightarrow 'a\ drvo \Rightarrow 'a\ drvo$ **where**
$ubaci\ a\ List = (Cvor\ List\ a\ List)$
$|\ ubaci\ a\ (Cvor\ ld\ x\ dd) =$
$(if\ a \leq x\ then\ Cvor\ (ubaci\ a\ ld)\ x\ dd$
$else\ Cvor\ ld\ x\ (ubaci\ a\ dd))$

lemma *sadrzi-ubaci* [*simp*]:
shows *sadrzi (ubaci x d) x*
by *(induction d) auto*

lemma *skup-ubaci* [*simp*]:
shows $skup\ (ubaci\ x\ d) = \{x\} \cup skup\ d$
by *(induction d) auto*

lemma *multiskup-ubaci*[*simp*]:
shows $multiskup\ (ubaci\ x\ d) = \{\#x\#\} + multiskup\ d$
by *(induction d) auto*

lemma *zbir-ubaci* [*simp*]:

shows *zbir* (*ubaci x d*) = *x* + *zbir d*
by (*induction d*) *auto*

lemma *sortirano-ubaci* [*simp*]:
shows *sortirano d* $\Longrightarrow$ *sortirano* (*ubaci x d*)
by (*induction d*) *auto*

Definisati funkciju *listaUDrvo* :: (*'a*::*linorder*) *list* $\Rightarrow$ *'a drvo* koja od liste elemenata gradi binarno pretraživačko drvo.

primrec *listaUDrvo* :: (*'a*::*linorder*) *list* $\Rightarrow$ *'a drvo* **where**
listaUDrvo [] = *List*
| *listaUDrvo* (*x* # *xs*) = *ubaci x* (*listaUDrvo xs*)

Pokazazati sledeće osobine funkcije *listaUDrvo*:

1. *listaUDrvo održava skup elemenata.*
2. *listaUDrvo održava multiskup elemenata.*
3. *listaUDrvo gradi binarno pretraživačko drvo.*

lemma [*simp*]: *skup* (*listaUDrvo xs*) = *set xs*
by (*induction xs*) *auto*

lemma [*simp*]: *multiskup* (*listaUDrvo xs*) = *mset xs*
by (*induction xs*) *auto*

lemma [*simp*]: *sortirano* (*listaUDrvo xs*)
by (*induction xs*) *auto*

Definisati funkciju koja sortira elemente liste pomoću stabla:

definition *sortiraj* :: *nat list* $\Rightarrow$ *nat list* **where**
sortiraj xs = *infiks* (*listaUDrvo xs*)

Pokazati korektnost ove funkcije

1. *Nakon primene funkcije lista je sortirana.*
2. *Skup elemenata sortirane liste i početne liste ostaje isit.*
3. *Multiskup elemenata sortirane liste i početne liste ostaje isti.*

theorem *sorted* (*sortiraj xs*)
unfolding *sortiraj-def*
by (*induction xs*) (*auto simp add*: *sortirano-sorted-infiks*)

theorem *set* (*sortiraj xs*) = *set xs*
unfolding *sortiraj-def*
by *auto*

theorem *mset* (*sortiraj xs*) = *mset xs*
unfolding *sortiraj-def*
by *auto*