Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
03.05.2023.

Uvod u interaktivno dokazivanje teorema

Vežbe 10

Zadatak 1 *Tip: list.*

Diskutovati o sledećim termovima i vrednostima.

term []
term *1* # *2* # []
term (*1*::*nat*) # *2* # []
term [*1*, *2*]
term [*1*::*nat*, *2*]

value [*1..5*]
value [*1..<5*]

term *sum-list*
value *sum-list* [*1..<5*]

term *map*
term λ *x*. *f x*
value *map* (λ *x*. *x*^*2*) [*1..<5*]
value *sum-list* (*map* (λ *x*. *x*^*2*) [*1..<5*])

value ∑ *x* ← [*1..<5*]. *x*^*2*

Zadatak 2 *Sumiranje nizova preko listi.*

Pokazati da važi: $1 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

primrec *zbir-kvadrata* :: *nat* ⇒ *nat* **where**
zbir-kvadrata 0 = *0*
| *zbir-kvadrata* (*Suc n*) = *zbir-kvadrata n* + (*Suc n*) ^ *2*

Definisati funkciju *zbir-kvadrata′* :: *nat* ⇒ *nat* preko definicije, koja računa levu stranu jednakosti pomoću liste i funkcijama nad listama.

definition *zbir-kvadrata′* :: *nat* ⇒ *nat* **where**
zbir-kvadrata′ n = *undefined*

Pokazati da su ove dve funkcije ekvivalentne.

lemma *zbir-kvadrata n* = *zbir-kvadrata′ n*

Pokazati automatski da je *zbir-kvadrata n* = *n* ∗ (*n* + *1*) ∗ (*2* ∗ *n* + *1*) *div 6*.
Savet: Razmotriti leme koje se koriste u Isar verziji dokaza i dodati ih u *simp*.

lemma *zbir-kvadrata n* = *n* ∗ (*n* + *1*) ∗ (*2* ∗ *n* + *1*) *div 6*

Zadatak 3 *Algebarski tip podataka: lista.*

Definisati polimorfan algebarski tip podataka *'a lista* koji predstavlja listu elemenata polimorfong tipa *'a*.

datatype *'a lista* = *Prazna*
| *Dodaj 'a 'a lista*

term *Dodaj* (*1*::*nat*) (*Dodaj 2* (*Dodaj 3 Prazna*))

Definisati funkcije *duzina'* :: *'a lista* ⇒ *nat*, *nadovezi'* :: *'a lista* ⇒ *'a lista* ⇒ *'a lista*, *obrni'* :: *'a lista* ⇒ *'a lista* primitivnom rekurzijom koje računaju dužinu liste, nadoveziju i obrću liste tipa *'a lista*.

Definisati funkciju *duzina* :: *'a list* ⇒ *nat* primitivnom rekurzijom koja računa dužinu liste tipa *'a list*. Ta pokazati da su *duzina* i *length* ekvivalentne funkcije.

primrec *duzina* :: *'a list* ⇒ *nat* **where**
duzina [] = *undefined*
| *duzina* (*x* # *xs*) = *undefined*

lemma *duzina-length*:
shows *duzina xs* = *length xs*

Definisati funkciju *prebroj* :: (*'a*::*equal*) ⇒ *'a list* ⇒ *nat* primitivnom rekurzijom koja računa koliko se puta javlja element tipa *'a*::*equal* u listi tipa (*'a*::*equal*) *list*. Ta pokazati da je *prebroj a xs* ≤ *length xs*.

Definisati funkcicju *sadrzi* :: (*'a*::*equal*) ⇒ *'a list* ⇒ *bool* primitivnom rekurzijom koja ispituje da li se element tipa *'a*::*equal* javlja u listi tipa (*'a*::*equal*) *list*. Ta pokazati da je *sadrzi a xs* = *a* ∈ *set xs*

Definisati funkciju *skup* :: *'a list* ⇒ *'a set* primitivnom rekurzijom koja vraća skup tipa *'a set* koji je sačinjen od elemenata liste tipa *'a list*. Ta pokazati da je *skup xs* = *set xs*.

primrec *skup* :: *'a list* ⇒ *'a set* **where**
skup [] = *undefined*
| *skup* (*x* # *xs*) = *undefined*

lemma *skup-set*:
shows *skup xs* = *set xs*

Definisati funkciju *nadovezi* :: *'a list* ⇒ *'a list* ⇒ *'a list* primitivnom rekurzijom koja nadovezuje jednu listu na drugu tipa *'a list*. Ta pokazati da je ekvivalentna ugrađenoj funkciji *append* ili infiksom operatoru @.

primrec *nadovezi* :: *'a list* ⇒ *'a list* ⇒ *'a list* **where**
nadovezi [] = *undefined*
| *nadovezi* (*x* # *xs*) = *undefined*

Formulisati i pokazati da je dužina dve nedovezane liste, zbir dužina pojedinačnih listi.
Orediti i dokazati osobine za funkcije *skup* i *nadovezi*, kao i za *sadrzi* i *nadovezi*.

Definisati funkcicju *obrni* :: *'a list* ⇒ *'a list* primitivnom rekurzijom koja obrće listu tipa *'a list*. Ta pokazati da funkcija je *obrni* ekvivalentna funkciji *rev*. Nakon toga pokazati da je dvostruko

obrnuta lista ekvivalentna početnoj listi.
Napomena: Pri definisanju funkcije *obrni* nije dozvoljeno koristiti operator nadovezivanje @.
Savet: Potrebno je definisati pomoćne leme.

primrec *obrni* :: *'a list* ⇒ *'a list* **where**
  obrni [] = *undefined*
| *obrni* (*x* # *xs*) = *undefined*

lemma *obrni-rev*:
  shows *obrni xs* = *rev xs*

lemma *obrni-obrni-id*: *obrni* (*obrni xs*) = *xs*

Definisati funkciju *snoc* :: *'a* ⇒ *'a list* ⇒ *'a list* koja dodaje element na kraj liste, i funkciju *rev-snoc* :: *'a list* ⇒ *'a list* koja uz pomoć funkcije *snoc* obrće elemente liste. Da li *rev-snoc* popravlja složenost obrtanja liste?

primrec *snoc* :: *'a* ⇒ *'a list* ⇒ *'a list* **where**
  snoc a [] = *undefined*
| *snoc a* (*x* # *xs*) = *undefined*

primrec *rev-snoc* :: *'a list* ⇒ *'a list* **where**
  rev-snoc [] = *undefined*
| *rev-snoc* (*x* # *xs*) = *undefined*

Definisati funkciju *itrev* koja obrće listu iterativno.
Savet: Koristiti pomoćnu listu.

Pokazati da je funkcija *itrev* ekvivalentna ugrađenoj funkciji *rev*, kada je inicijalna pomoćna lista prazna.

Pomoću funkcije *fold* opisati obrtanje liste. Pokazati ekvivalentnost funkciji *itrev* sa obrtanjem liste preko *fold*-a.

term *fold*