Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
22.04.2024.

Uvod u interaktivno dokazivanje teorema

Vežbe 09

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

term fold
term foldr
term foldl
value fold (λ x acc. x^2 + acc) [1..<5] (0::nat)
value foldr (λ x acc. x^2 + acc) [1..<5] (0::nat)
value foldl (λ acc x. x^2 + acc) (0::nat) [1..<5]

term filter
value filter (λ x. x > 2) [1..<5]
```

Zadatak 2 *Algebarski tip podataka: lista.*

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

```
datatype 'a lista = undef

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*.

Zadatak 3 *Osnovne funkcije nad listama.*

Definisati funkciju *duzina* :: *'a list* ⇒ *nat* primitivnom rekurzijom koja računa dužinu liste tipa *'a list*. 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*. Pokazati da je *prebroj a xs* ≤ *length xs*.

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

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

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*.

Zadatak 4 *Obrtanje liste.*

Definisati funkicju *obrni* :: *'a list* ⇒ *'a list* primitivnom rekurzijom koja obrće listu tipa *'a list*. 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.

Zadatak 5 *Zamena u listi.*

Definisati funkicju *zameni* :: *'a* ⇒ *'a* ⇒ *'a list* ⇒ *'a list* primitivnom rekurzijom, tako da *zameni a b xs* u listi *xs* zamenjuje sva pojavljivanja elementa *a* sa elementom *b*. Pokazati da je funkcija *zameni* korektna preko zadatih lema.

lemma *zameni-length*: *length* (*zameni a b xs*) = *length xs*

lemma *zameni-set*: *a* ≠ *b* ⟹ *a* ∉ *set* (*zameni a b xs*)

lemma *zameni-set2*: *b* ∈ *set xs* ⟹ *b* ∈ *set* (*zameni a b xs*)

Definisati funkciju *zameni'* koja u listi zamenjuje određeni element drugim elementom. Funkcija *zameni'* treba da bude repno rekurzivna.
Savet: Kao u primeru *itrev* koristiti pomoćnu listu.

lemma *zameni'-len*: *length* (*zameni' a b xs* []) = *length xs*

lemma *zameni'-set*: *a* ≠ *b* ⟹ *a* ∉ *set* (*zameni' a b xs* [])

lemma *zameni'-set2*: *b* ∈ *set xs* ⟹ *b* ∈ *set* (*zameni' a b xs* [])