Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
19.02.2024.

Uvod u interaktivno dokazivanje teorema

Vežbe 01

Zadatak 1 *Primer jednostavne teorije*

(a) Pokazati da važi komutativnost i asocijativnost operacije (+) :: *nat* ⇒ *nat* ⇒ *nat*.

lemma $(x{::}nat) + y = y + x$
  by *simp*

lemma $((x{::}nat) + y) + z = x + (y + z)$
  by *simp*

(b) Definisati funkciju *sledbenik* :: *nat* ⇒ *nat* i pokazati da važi $sledbenik\ (sledbenik\ x) = x + 2$.

definition *sledbenik* :: *nat* ⇒ *nat* **where**
  $sledbenik\ x = x + 1$

lemma $sledbenik\ (sledbenik\ x) = x + 2$
  unfolding *sledbenik-def*
  by *simp*

(c) Pokazati da ako važi $x > 0$ onda $sledbenik\ x > 1$. Te pokazati da ako važi $x < 5$ onda $sledbenik\ x < 6$.

lemma $x > 0 \longrightarrow sledbenik\ x > 1$
  unfolding *sledbenik-def*
  by *simp*

lemma $x < 5 \longrightarrow sledbenik\ x < 6$
  unfolding *sledbenik-def*
  by *simp*

(d) Prethodna dva tvrđenja uopštiti u opšte tvrđenje o ograničenosti sledbenika.

lemma *ogranicenost-sledbenika*:
  fixes $a\ b :: nat$
  assumes $a < x\ x < b$
  shows $a + 1 < sledbenik\ x \wedge sledbenik\ x < b + 1$
  unfolding *sledbenik-def*
  using *assms*
  by *simp*

(e) Definisati funkciju *kvadrat* :: *nat* ⇒ *nat* i pokazati da važi $kvadrat\ (x + 1) = kvadrat\ x + 2 * x + 1$.

abbreviation *kvadrat* :: *nat* ⇒ *nat* **where**
  $kvadrat\ x \equiv x * x$

lemma $kvadrat\ (x + 1) = kvadrat\ x + 2 * x + 1$

by *simp*

(f) Definisati rekurzivnu funkciju *sum* :: *nat list* ⇒ *nat* koja računa sumu liste prirodnih brojeva. Pokazati da se *sum xs* ponaša isto kao i *foldr* primenjen na odgovarajuću funkciju, listu *xs*, i odgovarajuću početnu vrenodst akomulatora. Nako toga pokazati sledeće svojstvo *sum* (*xs* @ *ys*) = *sum xs* + *sum ys*.

fun *sum* :: *nat list* ⇒ *nat* **where**
sum [] = *0*
| *sum* (*x* # *xs*) = *x* + *sum xs*

lemma *sum xs* = *foldr* (+) *xs 0*
by (*induction xs*) *auto*

lemma *sum* (*xs* @ *ys*) = *sum xs* + *sum ys*
by (*induction xs*) *auto*

(g) Definisati rekurzivnu funkciju *len* :: *nat list* ⇒ *nat* koja računa dužinu liste prirodnih brojeva. Pokazati da se *len xs* ponaša isto kao i *foldr* primenjen na odgovarajuću funkciju, listu *xs*, i odgovarajuću početnu vrednost akumulatora (Savet: Zgodno je koristiti lambda funkciju (λ *x y*. *f x y*) za definisanje funkcije koju prima *foldr*). Nako toga pokazati sledeće svojstvo *len* (*xs* @ *ys*) = *len xs* + *len ys*.

fun *len* :: *nat list* ⇒ *nat* **where**
len [] = *0*
| *len* (*x* # *xs*) = *1* + *len xs*

lemma *len xs* = *foldr* (λ *x*. (+) *1*) *xs 0*
by (*induction xs*) *auto*

lemma *len* (*xs* @ *ys*) = *len xs* + *len ys*
by (*induction xs*) *auto*

Zadatak 2 *Zapisivanje logičkih formula*

(a) Zapisati nekoliko logičkih formula koje uključuju logičke konstante *True* i *False*, logičke veznike ¬, ∧, ∨, ⟶, i ⟷/=, i univerzalne i egzistencionalne kvantifikatore ∀ i ∃

lemma *A* ∧ *B* ⟶ *A* ∨ *B*
by *simp*

lemma *A* ∧ *A* ⟷ *A*
by *simp*

lemma *A* ∨ ¬ *A* ⟷ *True*
by *simp*

lemma ∀ *x*. *P x* ⟶ *Q x*
nitpick
oops

lemma (∀ *x*. *P x* ⟶ *Q x*) ∧ (∃ *x*. *P x*) ⟶ (∃ *x*. *Q x*)
sledgehammer
by *blast*

(b) Zapisati sledeće rečenice u logici prvog reda i dokazati njihovu ispravnost.

(b.1) Ako onaj ko laže taj i krade i ako bar neko laže, onda neko i krade.

lemma
$(\forall\ x.\ Laze\ x \longrightarrow Krade\ x) \wedge$
$(\exists\ x.\ Laze\ x) \longrightarrow$
$(\exists\ x.\ Krade\ x)$
by *auto*

(b.2) Ako "ko radi taj ima ili troši" i "ko ima taj peva" i "ko troši taj peva", onda "ko radi taj peva"

lemma
$(\forall\ x.\ Radi\ x \longrightarrow Ima\ x \vee Trosi\ x) \wedge$
$(\forall\ x.\ Ima\ x \longrightarrow Peva\ x) \wedge$
$(\forall\ x.\ Trosi\ x \longrightarrow Peva\ x) \longrightarrow$
$(\forall\ x.\ Radi\ x \longrightarrow Peva\ x)$
by *auto*

(c) Zapisati sledeći skup rečenica u logici prvog reda i dokazati njihovu nezadovoljivost.

(c.1) Ako je X prijatelj osobe Y, onda je i Y prijatelj osobe X.

(c.2) Ako je X prijatelj osobe Y, onda X voli Y.

(c.3) Ne postoji neko ko je povredio osobu koju voli.

(c.4) Osoba Y je povredila svog prijatelja X.

lemma
$(\forall\ x\ y.\ Prijetelj\ x\ y \longrightarrow Prijatelj\ y\ x) \wedge$
$(\forall\ x\ y.\ Prijatelj\ x\ y \longrightarrow Voli\ x\ y) \wedge$
$(\neg\ (\exists\ x\ y.\ Voli\ x\ y \wedge Povredio\ x\ y)) \wedge$
$(\exists\ y\ x.\ Prijatelj\ y\ x \wedge Povredio\ y\ x) \longrightarrow$
$False$
by *auto*