Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
18.04.2023.

Uvod u interaktivno dokazivanje teorema

Vežbe 9

Zadatak 1 *Zasnivanje prirodnih brojeva.*

Definisati algebarski tip podataka *prirodni* koji predstavlja prirodni broj.

datatype *prirodni* =
  Nula (**0**)
  | *Sled prirodni*

Diskutovati o tipu *prirodni* i sledećim termovima.

typ *prirodni*

term *Nula*
term *Sled Nula*
term *Sled* (*Sled Nula*)

Definisati skraćenice za prirodne brojeve **1**, **2**, **3**.

abbreviation *jedan* :: *prirodni* (**1**) **where**
  1 ≡ *Sled* **0**

abbreviation *dva* :: *prirodni* (**2**) **where**
  2 ≡ *Sled* **1**

abbreviation *tri* :: *prirodni* (**3**) **where**
  3 ≡ *Sled* **2**

Primitivnom rekurzijom definisati operaciju sabiranja. Uvesti levo asocijativni operator ⊕ za operaciju sabiranja.

primrec *saberi* :: *prirodni* ⇒ *prirodni* ⇒ *prirodni* (**infixl** ⊕ *100*)**where**
  0 ⊕ *b* = *b*
| (*Sled a*) ⊕ *b* = *Sled* (*a* ⊕ *b*)

Testirati funkciju sabiranjem nekih skraćenica za prirodne brojeve.

value 1 ⊕ **2** ≠ **3**

Pokazati da je sabiranje asocijativno.

lemma *saberi-asoc*:
  shows *a* ⊕ (*b* ⊕ *c*) = *a* ⊕ *b* ⊕ *c*
  by (*induction a*) *auto*

Pokazati da je sabiranje komutativno.
Savet: Potrebno je pokazati pomoćne lemu.

lemma *saberi-Nula-desno*[*simp*]:
  shows *a* ⊕ **0** = *a*
  by (*induction a*) *auto*

```
lemma saberi-Sled-desno[simp]:
  shows a ⊕ Sled b = Sled (a ⊕ b)
  by (induction a) auto

lemma saberi-kom:
  shows a ⊕ b = b ⊕ a
  by (induction a) auto

lemma saberi-kom-isar:
  shows a ⊕ b = b ⊕ a
proof (induction a)
  case Nula
  have 0 ⊕ b = b by (rule saberi.simps(1))
  also have b = b ⊕ 0 by (rule saberi-Nula-desno[symmetric])
  finally show ?case .
next
  case (Sled a)
  have Sled a ⊕ b = Sled (a ⊕ b) by (rule saberi.simps(2))
  also have ... = Sled (b ⊕ a) by (subst Sled, rule refl)
  also have ... = b ⊕ Sled a by (rule saberi-Sled-desno[symmetric])
  finally show ?case .
qed
```

Primitivnom rekurzijom definisati operaciju množenja. Uvesti levo asocijativni operator ⊗ za operaciju množenja.

```
primrec pomnozi :: prirodni ⇒ prirodni ⇒ prirodni (infixl ⊗ 101) where
  pomnozi 0 b = 0
| pomnozi (Sled a) b = a ⊗ b ⊕ b
```

Pokazati komutativnost množenja.
Savet: Pokazati pomoćne lemme.

```
lemma pomnozi-Nula-desno[simp]:
  shows a ⊗ 0 = 0
  by (induction a) auto

lemma pomnozi-Sled-desno[simp]:
  shows a ⊗ Sled b = a ⊕ a ⊗ b
  by (induction a) (auto simp add: saberi-asoc)

lemma pomnozi-kom:
  shows a ⊗ b = b ⊗ a
  by (induction a) (auto simp add: saberi-kom)
```

Pokazati da je množenje asocijativno.

```
lemma saberi-pomnozi-distrib-desno:
  shows (a ⊕ b) ⊗ c = a ⊗ c ⊕ b ⊗ c
  by (induction a) (auto simp add: pomnozi-kom saberi-asoc)

lemma pomnozi-asoc:
  shows a ⊗ (b ⊗ c) = a ⊗ b ⊗ c
  by (induction a) (auto simp add: saberi-pomnozi-distrib-desno)
```

Primitivnom rekurzijom definisati operaciju stepenovanja. Uvesti levo asocijativni operator $\frown$ za operaciju stepenovanja.

primrec *stepenuj* :: *prirodni* ⇒ *prirodni* ⇒ *prirodni* (**infixl** $\frown$ *102*) **where**
  $a \frown \mathbf{0} = \mathbf{1}$
| $a \frown (Sled\ n) = a \otimes a \frown n$

value $\mathbf{2} \frown \mathbf{0}$
value $\mathbf{2} \frown \mathbf{2}$

Pokazati da važi: $a^1 = a$.

lemma *stepenuj-jedan*:
  shows $a \frown \mathbf{1} = a$
  by *auto*

Pokazati da važi: $a^{(n+m)} = a^n b^m$.

lemma *stepenuj-na-zbir*[*simp*]:
  shows $a \frown (n \oplus m) = a \frown n \otimes a \frown m$
  by (*induction n*) (*auto simp add*: *pomnozi-asoc*)

Pokazati da važi: $a^{nm} = a^{n^m}$.

lemma *stepenuj-jedinicu*[*simp*]:
  shows $\mathbf{1} \frown n = \mathbf{1}$
  by (*induction n*) *auto*

lemma *stepenuj-proizvod*[*simp*]:
  shows $(a \otimes b) \frown n = a \frown n \otimes b \frown n$
  by (*induction n*) (*auto*, *metis pomnozi-asoc pomnozi-kom*)

lemma *stepenuj-na-proizvod*:
  shows $a \frown (n \otimes m) = a \frown n \frown m$
  by (*induction n*) (*auto simp add*: *pomnozi-kom*)

Zadatak 2 *Dodatni primeri.*

Pokazati sledeće teoreme u Isar-u. Kao dodatan izazov, dozvoljeno je korišćenje samo primenjivanje pravila *rule* i *subst* za dokazivanje među koraka, tj. bilo kakva automatizacija (*simp*, *auto*, *metis*, *blast*, *force*, *fastforce*, *sladgehammer*, ...) je zabranjena.

lemma $a \oplus \mathbf{0} = a$
proof (*induction a*)
  case *Nula*
  have $\mathbf{0} \oplus \mathbf{0} = \mathbf{0}$ **by** (*rule saberi.simps*(*1*))
  then show *?case* **.**
next
  case (*Sled a*)
  have $Sled\ a \oplus \mathbf{0} = Sled\ (a \oplus \mathbf{0})$ **by** (*rule saberi.simps*(*2*))
  also have ... = *Sled a* **by** (*subst saberi-Nula-desno*, *rule refl*)
  finally show *?case* **.**
qed

lemma $a \otimes (Sled\ b) = a \otimes b \oplus a$
proof (*induction a*)

case *Nula*
have $\mathbf{0} \otimes \mathit{Sled}\ b = \mathbf{0}$ **by** (*rule pomnozi.simps*(*1*))
also have ... $= \mathbf{0} \otimes b$ **by** (*rule pomnozi.simps*(*1*)[*symmetric*])
also have ... $= \mathbf{0} \oplus \mathbf{0} \otimes b$ **by** (*rule saberi.simps*(*1*)[*symmetric*])
also have ... $= \mathbf{0} \otimes b \oplus \mathbf{0}$ **by** (*rule saberi-kom*)
finally show *?case* **.**
next
case (*Sled a*)
thm *pomnozi.simps*(*2*)
have $\mathit{Sled}\ a \otimes \mathit{Sled}\ b = \mathit{Sled}\ b \otimes \mathit{Sled}\ a$ **by** (*rule pomnozi-kom*)
also have ... $= b \otimes \mathit{Sled}\ a \oplus \mathit{Sled}\ a$ **by** (*rule pomnozi.simps*(*2*))
also have ... $= \mathit{Sled}\ a \otimes b \oplus \mathit{Sled}\ a$ **by** (*subst pomnozi-kom*, *rule refl*)
finally show *?case* **.**
qed

lemma $(a \oplus b) \otimes c = a \otimes c \oplus b \otimes c$
proof (*induction a*)
case *Nula*
have $(\mathbf{0} \oplus b) \otimes c = b \otimes c$ **by** (*subst saberi.simps*(*1*), *rule refl*)
thm *saberi.simps*(*1*)[*symmetric*]
also have ... $= \mathbf{0} \oplus b \otimes c$ **by** (*rule saberi.simps*(*1*)[*symmetric*])
also have ... $= \mathbf{0} \otimes c \oplus b \otimes c$ **by** (*subst pomnozi.simps*(*1*)[*symmetric*], *rule refl*)
finally show *?case* **.**
next
case (*Sled a*)
have $(\mathit{Sled}\ a \oplus b) \otimes c = \mathit{Sled}\ (a \oplus b) \otimes c$ **by** (*subst saberi.simps*(*2*), *rule refl*)
also have ... $= (a \oplus b) \otimes c \oplus c$ **by** (*rule pomnozi.simps*(*2*))
also have ... $= a \otimes c \oplus b \otimes c \oplus c$ **by** (*subst Sled*, *rule refl*)
also have ... $= a \otimes c \oplus (b \otimes c \oplus c)$ **by** (*rule saberi-asoc*[*symmetric*])
also have ... $= b \otimes c \oplus c \oplus a \otimes c$ **by** (*rule saberi-kom*)
also have ... $= b \otimes c \oplus (c \oplus a \otimes c)$ **by** (*rule saberi-asoc*[*symmetric*])
also have ... $= c \oplus a \otimes c \oplus b \otimes c$ **by** (*rule saberi-kom*)
also have ... $= a \otimes c \oplus c \oplus b \otimes c$ **by** (*subst saberi-kom*, *rule refl*)
also have ... $= \mathit{Sled}\ a \otimes c \oplus b \otimes c$ **by** (*subst pomnozi.simps*(*2*)[*symmetric*], *rule refl*)
finally show *?case* **.**
qed

lemma $a \otimes b \otimes c = a \otimes (b \otimes c)$
proof (*induction a*)
case *Nula*
thm *pomnozi.simps*(*1*)
have $\mathbf{0} \otimes b \otimes c = \mathbf{0} \otimes c$ **by** (*subst pomnozi.simps*(*1*), *rule refl*)
also have ... $= \mathbf{0}$ **by** (*rule pomnozi.simps*(*1*))
also have ... $= \mathbf{0} \otimes (b \otimes c)$ **by** (*rule pomnozi.simps*(*1*)[*symmetric*])
finally show *?case* **.**
next
case (*Sled a*)
have $\mathit{Sled}\ a \otimes b \otimes c = (a \otimes b \oplus b) \otimes c$ **by** (*subst pomnozi.simps*(*2*), *rule refl*)
also have ... $= a \otimes b \otimes c \oplus b \otimes c$ **by** (*rule saberi-pomnozi-distrib-desno*)
also have ... $= a \otimes (b \otimes c) \oplus b \otimes c$ **by** (*subst Sled*, *rule refl*)
also have ... $= \mathit{Sled}\ a \otimes (b \otimes c)$ **by** (*rule pomnozi.simps*(*2*)[*symmetric*])

```
  finally show ?case .
qed

lemma a ⊗ b = b ⊗ a
proof (induction a)
  case Nula
  have 0 ⊗ b = 0 by (rule pomnozi.simps(1))
  also have ... = b ⊗ 0 by (rule pomnozi-Nula-desno[symmetric])
  finally show ?case .
next
  case (Sled a)
  have Sled a ⊗ b = a ⊗ b ⊕ b by (rule pomnozi.simps(2))
  also have ... = b ⊗ a ⊕ b by (subst Sled, rule refl)
  also have ... = b ⊕ b ⊗ a by (rule saberi-kom)
  also have ... = b ⊗ Sled a by (rule pomnozi-Sled-desno[symmetric])
  finally show ?case .
qed

lemma a ⊗ (b ⊕ c) = a ⊗ b ⊕ a ⊗ c
proof (induction a)
  case Nula
  have 0 ⊗ (b ⊕ c) = 0 by (rule pomnozi.simps(1))
  also have ... = 0 ⊗ c by (rule pomnozi.simps(1)[symmetric])
  also have ... = 0 ⊕ 0 ⊗ c by (rule saberi.simps(1)[symmetric])
  also have ... = 0 ⊗ b ⊕ 0 ⊗ c by (subst pomnozi.simps(1)[symmetric], rule refl)
  finally show ?case .
next
  case (Sled a)
  have Sled a ⊗ (b ⊕ c) = a ⊗ (b ⊕ c) ⊕ (b ⊕ c) by (rule pomnozi.simps(2))
  also have ... = a ⊗ b ⊕ a ⊗ c ⊕ (b ⊕ c) by (subst Sled, rule refl)
  also have ... = a ⊗ b ⊕ a ⊗ c ⊕ b ⊕ c by (rule saberi-asoc)
  also have ... = a ⊗ c ⊕ a ⊗ b ⊕ b ⊕ c by (subst saberi-kom, rule refl)
  also have ... = a ⊗ c ⊕ (a ⊗ b ⊕ b) ⊕ c by (subst saberi-asoc, rule refl)
  also have ... = a ⊗ b ⊕ b ⊕ a ⊗ c ⊕ c by (subst saberi-kom, rule refl)
  also have ... = Sled a ⊗ b ⊕ a ⊗ c ⊕ c by (subst pomnozi.simps(2)[symmetric], rule refl)
  also have ... = Sled a ⊗ b ⊕ (a ⊗ c ⊕ c) by (subst saberi-asoc, rule refl)
  also have ... = a ⊗ c ⊕ c ⊕ Sled a ⊗ b by (subst saberi-kom, rule refl)
  also have ... = Sled a ⊗ c ⊕ Sled a ⊗ b by (subst pomnozi.simps(2)[symmetric], rule refl)
  also have ... = Sled a ⊗ b ⊕ Sled a ⊗ c by (rule saberi-kom)
  finally show ?case .
qed
```