

Uvod u interaktivno dokazivanje teorema

Vežbe 11

Zadatak 1 *stek mašina*.

Definisati algebarski tip podataka *izraz* koji predstavlja izraz koga čine konstante koje su prirodni brojevi, i tri binarne operacije plus, minus, i puta nad izrazom.

```
datatype izraz =  
  Const nat  
  | Plus izraz izraz  
  | Minus izraz izraz  
  | Put izraz izraz
```

Konstruisati proizvoljnu instancu tipa *izraz* i proveriti njenu ispravnost pomoću ključne reči *term*.

```
term Plus (Const 2) (Const 3)
```

Definisati funkciju *vrednost* :: *izraz* ⇒ *nat* koja računa vrednost izraza.

```
primrec vrednost :: izraz ⇒ nat where  
  vrednost (Const x) = x  
  | vrednost (Plus i1 i2) = vrednost i1 + vrednost i2  
  | vrednost (Minus i1 i2) = vrednost i1 - vrednost i2  
  | vrednost (Put i1 i2) = vrednost i1 * vrednost i2
```

Definisati izraze x_1 , x_2 , i x_3 , gde je $x_1 \equiv 2 + 3$, $x_2 \equiv 3 \cdot (5 - 2)$, i $x_3 \equiv 3 \cdot 4 \cdot (5 - 2)$. Izračunati vrednosti ovih izraza.

```
definition x1 :: izraz where  
  x1 ≡ Plus (Const 2) (Const 3)
```

```
definition x2 :: izraz where  
  x2 ≡ Put (Const 3) (Minus (Const 5) (Const 2))
```

```
definition x3 :: izraz where  
  x3 ≡ Put (Plus (Const 3) (Const 4)) (Minus (Const 5) (Const 2))
```

```
value vrednost x1  
value vrednost x2  
value vrednost x3
```

Definisati tip *stek* kao listu prirodnih brojeva. Dodavanje na vrh steka podrazumeva operaciju *Cons* (dodavanje na početak liste).

```
type-synonym stek = nat list
```

Definisati algebarski tip *operacija* koji predstavlja moguće operacije koje će se izvršavati nad stekom. Nad stekom je moguće primeniti operaciju za plus, minus, puta i dodavanje nogov elementa na stek.

```

datatype operacija =
  OpPlus
| OpMinus
| OpPutata
| OpPush nat

```

Definisati funkciju $izvrsiOp :: operacija \Rightarrow stek \Rightarrow stek$ koja izvršava datu operaciju nad stekom i vraća novo stanje steka.

```

fun izvrsiOp :: operacija  $\Rightarrow$  stek  $\Rightarrow$  stek where
  izvrsiOp (OpPush x) xs = x # xs
| izvrsiOp OpPlus (a # b # xs) = (a + b) # xs
| izvrsiOp OpMinus (a # b # xs) = (a - b) # xs
| izvrsiOp OpPutata (a # b # xs) = (a * b) # xs

```

Definisati tip *program* kao listu operacija.

```

type-synonym program = operacija list

```

Definisati funkciju $prevedi :: izraz \Rightarrow program$ koja data izraz prevodi u listu operacija, tj. program. Primeniti ovu funkciju nad izrazima $x1$, $x2$, i $x3$.

```

primrec prevedi :: izraz  $\Rightarrow$  program where
  prevedi (Const x) = [OpPush x]
| prevedi (Plus a b) = OpPlus # (prevedi a) @ (prevedi b)
| prevedi (Minus a b) = OpMinus # (prevedi a) @ (prevedi b)
| prevedi (Putata a b) = OpPutata # (prevedi a) @ (prevedi b)

```

```

value x1
value prevedi x1
value x2
value prevedi x2
value x3
value prevedi x3

```

Definisati funkciju $izvrsiProgram :: program \Rightarrow stek \Rightarrow stek$ koja primenjuje listu operacija, tj. program nad stekom. Izračunati vrednost ove funkcije kada se primeni nad programe (koji se dobiju prevođenjem izraza $x1$, $x2$, i $x3$) i praznim stekom.

```

primrec izvrsiProgram :: program  $\Rightarrow$  stek  $\Rightarrow$  stek where
  izvrsiProgram [] stek = stek
| izvrsiProgram (op # program) stek = izvrsiOp op (izvrsiProgram program stek)

```

```

value prevedi x1
value izvrsiProgram (prevedi x1) []
value prevedi x2
value izvrsiProgram (prevedi x2) []
value prevedi x3
value izvrsiProgram (prevedi x3) []

```

Dodatno, definisati funkciju $racunar :: izraz \Rightarrow nat$ koja prevodi program izvršava program (koji se dobija prevođenjem izraza) nad praznim stekom. Takođe, testirati ovu funkciju nad izrazima $x1$, $x2$, i $x3$.

```

definition racunar :: izraz  $\Rightarrow$  nat where
  racunar x = hd (izvrsiProgram (prevedi x) [])

```

```
value x1
value racunar x1
value x2
value racunar x2
value x3
value racunar x3
```

Pokazati da računar korektno izračunava vrednost izraza, tj. da su funkcije *racunar* i *vrednost* ekvivalentne. *Savet*: Potrebno je definisati pomoćne leme generalizacijom.

```
lemma izvrsiprogram-append [simp]:
  shows izvrsiprogram (p1 @ p2) s = izvrsiprogram p1 (izvrsiprogram p2 s)
  by (induction p1) auto
```

```
lemma izvrsiprogram-prevedi [simp]:
  shows izvrsiprogram (prevedi x) s = vrednost x # s
  by (induction x arbitrary: s) auto
```

```
lemma hd-racunar-vrednost:
  shows racunar x = vrednost x
  unfolding racunar-def
  by auto
```