Funkcijsko programiranje

Tema: Tipovi i klase.

17. listopada 2018.

predavač: doc. dr. sc. Slobodan Jelić

asistent: Luka Borozan, mag. math.

Osnovni koncepti I

Definicija

Tip je kolekcija/skup međusobno povezanih vrijednosti.

- u sljedećem primjeru navodimo tipove i pripadne vrijednosti u oznaci $v :: T$ gdje je v oznaka vrijednosti, a T oznaka tipa

Primjer

$$\begin{array}{lll} False & :: & Bool \\ True & :: & Bool \\ \neg & :: & Bool \to Bool \end{array}$$

- moguće je koristiti i oznaku $e :: T$ gdje e označava izraz koji će se evaluirati u vrijednost tipa T

Osnovni koncepti II

Primjer

$$\begin{array}{lcl} \neg False & :: & Bool \\ \neg True & :: & Bool \\ \neg(\neg False) & :: & Bool \end{array}$$

- Haskell koristi proces *zaključivanja o tipovima (eng. type inference)* koji se ”poziva” prije evaluacije samog izraza:

$$\frac{f :: A \to B \quad e :: A}{f\ e :: B}$$

Osnovni koncepti III

Primjer

$$\neg False :: Bool$$

Haskell zaključuje koristeći pravilo

$$\frac{\neg :: Bool \to Bool \quad False :: Bool}{\neg\ False :: Bool}$$

- Haskell zaključiti o tipu izraza $\neg 3$. Zašto?
- U Haskellu *zaključivanje o tipu izraza prethodi evaluaciji izraza*
- Haskell izrazi su *tipski sigurni*
- iako se zaključivanjem o tipovima uklanja veliki broj grešaka, postoje i dalje greške koje ne možemo eliminirati, kao primjerice `1 'div' 0`

Osnovni koncepti IV

Primjer

if $True$ ***then*** 1 ***else*** $False$

- prethodni izraz bit će odbijen zbog zaključivanja o tipovima jer tipovi vrijednosti koje se vraćaju nisu isti u svim granama
- uočite da ovakav izraz, iako bi imao smisla, producirat će grešku, što se može smatrati *nedostatkom zaključivanja o tipovima*

Osnovni tipovi

Oznaka	Kratak opis	Detaljan opis
`Bool`	logičke vrijednosti	Tip sadrži dvije vrijednosti $False$ i $True$
`Char`	pojedinačni znakovi	Tip sadrži znakove koji se mogu unijeti preko standardnog unosa kao i kontrolne znakove. Treba naglasiti da je znakove, kao i u ostalim programskim jezicima, potrebno staviti unutar navodnika.

Osnovni tipovi

Oznaka	Kratak opis	Detaljan opis
`String` (`[Char]`)	znakovni nizovi	Tip sadrži sve znakovne nizove. Primjerice `"abc"`, `"1+2+3"` kao i prazan string `""`. Znakovni nizovi moraju se navoditi unutar `""`
`Int`	cijeli broj **fiksne** preciznosti, $[-2^{63}, 2^{63}-1]$	Tip sadrži cijele brojeve za koje je fiksirana količina memorije
`Integer`	cijeli broj **proizvoljne** preciznosti	Tip sadrži brojeve za koje se koristi proizvoljna (raspoloživa) količina memorije

Osnovni tipovi

Oznaka	Kratak opis	Detaljan opis
`Float`, `Double`	brojevi s pomičnom točkom jednostruke i dvostruke preciznosti	Tip sadrži brojeve s decimalnom točkom i koristi fiksnu količinu memorije za njihovo spremanje. Broj mjesta iza decimalne točke ovisi o fiksiranoj količini memorije za spremanje broja.

Liste

Definicija

Liste je niz elemenata istog tipa koji se stavljaju unutar zagrada i odvajaju zarezom. Tip za listu u kojoj su elementi tipa T označavamo s $[T]$.

Primjer

```
[False, True, False]      :: [Bool]
['a', 'b', 'c', 'd']      :: [Char]
["One", "Two", "Three"]   :: [String]
```

- $[]$ je lista duljine 0, koja se zove prazna lista
- Koja je razlika između $[]$ i $[[]]$?
- Koji je tip označen s $[[Char]]$?
- na osnovu definicije liste nemamo informacije o njezinoj duljini
- u Haskellu su dozvoljene *liste beskonačne duljine* što omogućava "lazy" evaluacija

Torke I

Definicija

Torka je **konačan** niz elemenata **različitih** tipova koji se stavljaju unutar oblih zagrada i odvajaju zarezom.

Primjer

`(False, True)`	::	$(Bool, Bool)$
`(False,'a', True)`	::	$(Bool, Char, bool)$
`["Yes", True, 'a']`	::	$(String, Bool, Char)$

- s obzirom na broj elemenata razlikujemo prazne liste, parove, uređene trojke, itd.
- nisu dozvoljene liste s jednim elementom zbog kolizije s izrazima koji koriste obične zagrade

Torke II

- ne postoji ograničenje na tip elemenata torki

Primjer

`('a', (False,'b'))`	::	$(Char, (Bool, Char))$
`(['a','b'],[False,True])`	::	$([Char], [Bool])$
`[('a', False,),('b',True)]`	::	$[(Char, Bool)]$

- torke moraju imati konačan broj elemenata kako bi se o tipovima torki zaključivalo prije evaluacije

Funkcije I

Definicija

Funkcija je preslikavanje koje vrijednostima (argumentima) jednog tipa pridružuje rezultat drugog tipa. S $T1 \to T2$ označavamo tip funkcija koje argumentu tipa $T1$ pridružuju rezultat tipa $T2$.

Primjer

$$\begin{array}{lcl} \neg & :: & Bool \to Bool \\ isDigit & :: & Char \to Bool \end{array}$$

- nema ograničenja tipove

Funkcije II

Primjer

$$
\begin{array}{lcl}
add & :: & (Int, Int) \to Int \\
add(x, y) & = & x + y \\
zeroto & :: & Int \to [Int] \\
zeroton & = & [0..n]
\end{array}
$$

Kaskadne funkcije I

- moguće je da funkcija vrati drugu funkciju kao rezultat (eng. curried function)
- na taj način mogu se zaobići funkcije koje primaju više argumenata.

Primjer

$$\begin{array}{lll} add' & :: & Int \to (Int \to Int) \\ add'\ x\ y & = & x + y \end{array}$$

Primjer

$$\begin{array}{lll} mult & :: & Int \to (Int \to (Int \to Int)) \\ mult\ x\ y\ z & = & x * y * z \end{array}$$

Kaskadne funkcije II

- kaskadne funkcije mogu biti korisnije od funkcija kojima proslijeđujemo torke

Primjer

Moguće je primijeniti funkciju add' *na parcijalnu listu argumenata. Primjerice, pomoću funkcije* add' *moguće je dobiti funkciju koja će povećati broj za* 1*:*

$$add'\ 1 \quad :: \quad Int \to Int$$

- umjesto $Int \to (Int \to (Int \to Int))$ možemo koristiti $Int \to Int \to Int \to Int$
- sve funkcije koje primaju više argumenata u Haskellu su definirane kao kaskadne funkcije

Polimorfni tipovi I

- funkcija $length$ koja vraća duljinu **bilo koje** liste trebala bi biti sljedećeg tipa: $length \quad :: \quad [a] \to Int$
- kažemo da je a varijabilan tip

Definicija

Tip koji sadrži jedan ili više varijabilnih tipova naziva se **polimorfni tip**.

Primjer

$$
\begin{array}{lll}
fst & :: & (a, b) \to a \\
head & :: & [a] \to a \\
take & :: & Int \to [a] \to [a] \\
zip & :: & [a] \to [b] \to [(a, b)] \\
id & :: & a \to a
\end{array}
$$

Preopterećeni tipovi I

- operator zbrajanja $+$ možemo primijeniti na različite numeričke tipove
- to postižemo uvođenjem **ograničenja na klasu** u obliku Ca

$$(+) \quad :: \quad Num\ a \Rightarrow a \to a \to a$$

Definicija

Tipovi koji sadrže jedan ili više ograničenja na klasu nazivaju se **preopterećeni tipovi**, dok se funkcije, koji na njih djeluju, nazivaju **preopterećene funkcije**.

Preopterećeni tipovi II

Primjer

$$
\begin{array}{lcl}
(-) & :: & Num\ a \Rightarrow a \to a \to a \\
(*) & :: & Num\ a \Rightarrow a \to a \to a \\
negate & :: & Num\ a \Rightarrow a \to a \\
abs & :: & Num\ a \Rightarrow a \to a \\
signum & :: & Num\ a \Rightarrow a \to a
\end{array}
$$

- brojevi su preopterećeni tipovi
- $3 :: Num\ a \Rightarrow a$ znači da je za bilo koji numerički tip a, broj 3 tipa a

Osnovne klase I

Definicija

Klasa je skup tipova koji podržavaju određene preopterećene operacije koje se nazivaju **metode**.

- Eq - tipovi čije se vrijednosti mogu uspoređivati prema jednakosti.
  - Metode ove klase su:
    $(==) \quad :: \quad a \to a \to Bool$
    $(\neq) \quad :: \quad a \to a \to Bool$
  - $Bool, Char, String, Int, Integer$ i $Float$ su instance klase Eq, kao i liste i torke čiji su elementi tipa iz Eq klase
  - funkcijski tipovi nisu u klasi Eq
- Ord - tipovi iz klase Eq čije nad čijim vrijednostima se može definirati potpuni (linearni) uređaj.

Osnovne klase II

- Metode ove klase su:

$$
\begin{array}{lll}
(<) & :: & a \to a \to Bool \\
(\leq) & :: & a \to a \to Bool \\
(>) & :: & a \to a \to Bool \\
(\geq) & :: & a \to a \to Bool \\
min & :: & a \to a \to a \\
max & :: & a \to a \to a
\end{array}
$$

- $Bool, Char, String, Int, Integer$ i $Float$ su instance klase Ord, kao i liste i torke čiji su elementi tipa iz Ord klase
- $String$, liste i torke su uređene **leksikografski**
- **leksikografski poredak $<_l$:** razmotrimo dva para (a_1, b_1) i (a_2, b_2)
- Kada je $(a_1, b_1) <_l (a_2, b_2)$?
- Ako je $a_1 <_l a_2$ ili ($a_1 = a_2$ i $b_1 <_l b_2$)

Osnovne klase III

Primjer

$False < True$	`True`
$min\ 'a'\ 'b'$	`'a'`
`"fakultet" > "faks"`	`True`
$[1,2,3] < [1,2]$	`False`
$('a',2) < ('b',1)$	`True`

- $Show$ - tipovi čije se vrijednosti mogu konvertirati u vrijednost tipa $String$ pomoću metode:
  $show \quad :: \quad a \to String$

Osnovne klase IV

- $Bool, Char, String, Int, Integer$ i $Float$ su instance klase $Show$, kao i liste i torke čiji su elementi tipa iz $Show$ klase

Primjer

$show\ False$ `"False"`
$show\ 'a'$ `"'a'"`
$show\ [1,2,3]$ `"[1,2,3]"`

- $Read$ - ova klasa je dualna klasi $Show$ koja sadrži tipove čije vrijednosti možemo dobiti konverzijom stringova

$$read :: String \to Bool$$

  - $Bool, Char, String, Int, Integer, Float$ i $Double$ su instance klase $Read$, kao i liste i torke čiji su elementi tipa iz $Read$ klase

Osnovne klase V

- Num - ova klasa sadrži numeričke tipove nad čijim vrijednostima možemo provesti sljedeće operacije:

$$
\begin{array}{lll}
(+) & :: & a \to a \to a \\
(-) & :: & a \to a \to a \\
(*) & :: & a \to a \to a \\
(negate) & :: & a \to a \\
abs & :: & a \to a \\
signum & :: & a \to a
\end{array}
$$

  - Int, $Integer$, $Float$ i $Double$ su instance klase Num

- $Integral$ - ova klasa sadrži numeričke tipove koji su instance klase Num, a koji podržavaju operaciju cjelobrojnog djeljenja s ostatkom

$$
\begin{array}{lll}
div & :: & a \to a \to a \\
mod & :: & a \to a \to a
\end{array}
$$

  - Int i $Integer$ su instance klase $Integral$

Osnovne klase VI

- $Fractional$ - ova klasa sadrži numeričke tipove koji su instance klase Num, koji ne podržavaju operaciju cjelobrojnog djeljenja s ostatkom već operaciju djeljenja i računanja inverza (recipročnog broja) na skupu racionalnih brojeva:

$$(/) \quad :: \quad a \to a \to a$$
$$recip \quad :: \quad a \to a \to a$$

  - $Float$ i $Double$ su instance klase $Fractional$