Haskell in razredi tipov

Danes bomo spoznali programski jezik Haskell in razrede tipov (angl. type classes), ki so še en način za organizacijo kode in funkcionalnosti. Haskell je deklarativni jezik, ki se odlikuje po tem, da je tudi **čist** (angl. pure), kar pomeni, da v osnovi nima računskih učinkov (stanje, I/O, izjeme). O računskih učinkih in o tem, kako so implementirani z monadami, bomo govorili naslednjič.

Haskell lahko opredelimo kot programski jezik, ki je:

- deklarativni jezik z leno evaluacijo
- je funkcijski jezik: funkcije so vrednosti
- ima koinduktivne tipe in zapise
- ima parametrični polimorfizem in izpeljavo tipov
- čist jezik: vsi računski učinki so eksplicitno zavedeni v tipu izraza
- ima razrede tipov

Osnovno o Haskellu

Mnoge koncepte, ki nastopaj v programskem jeziku Haskell, že poznamo. Danes bomo spoznali

Konkretna sintaksa

Zamikanje

V Haskellu je treba kodo pravilno zamikati, podobno kot v Pythonu. V nekaterih primerih se lahko zamikanju izognemo z uporabo alternativne sintakse `{ ... ; ... ; ... }`. Primer bomo videli kasneje.

Imena

- Imena spremenljivk se pišejo z malo začetnico: `x`, `y`, ...
- Imena tipov se pišejo z veliko začetnico: `Int`, `Bool`, `Char`, ...
- Parametri v polimorfnih tipih se pišejo z malo začetnico: `a`, `b`, `c`, ...
- Imena konstruktorjev algebraičnih tipov se pišejo z veliko

Definicije

Vrednost `t` tipa `A` zapišemo

```
t :: A
t = ⟨defincija-t⟩
```

Pozor: zapis `t :: A` pomeni "`t` ima tipa `A`" zapis `x :: ℓ` pa pomeni seznam z glavo `x` in repom `ℓ` (ravno obratno kot v OCamlu).

Definicija ima lahko tudi več vrstic, na primer:

```
f :: Int -> Int
f 0 = 1
f 1 = 1
f n = n * f (n - 1)
```

Definicije so lahko rekurzivne.

Deklaracijo tip v definicijo lahko izpustimo in zapišemo samo

```
t = ⟨defincija-t⟩
```

V tem primeru bo Haskell izpeljal tip `t`. Vendar pa je v Haskellu navada, da se zapiše tip vrednosti, ki jo definiramo.

Lokalne definicije

Lokalno definicijo zapišemo

```
let x = e₁
in
  e₂
```

ali

```
e₂ where x = e₁
```

Seznami

Prazen seznam zapišemo z `[]`, stik glave `x` z repom ℓ zapišemo `x : ℓ`, elemente lahko tudi naštejemo z `[x₁, x₂, ..., xᵢ]`.

Seznam števil od `1` do `n` zapišemo `[1..n]` in podobno za interval `[a..b]`.

Haskell ima izpeljane sezname (angl. list comprehension), podobno kot Python:

- matematika: `{f(x) | x ∈ A}` je množica elementov `f(x)`, kjer je `x ∈ A`
- Python `[f(x) for x in ℓ]` je seznam elementov `f(x)`, kjer `x` preteče seznam ℓ
- Haskell `[f x | x <- ℓ]` je seznam elementov `f x`, kjer `x` preteče seznam ℓ

Podrobnosti o izpeljanih seznamih si preberite na zgornji povezavi.

Funkcije

Funkcijo `x ↦ e` zapišemo kot `\ x -> e`, kar nas spominja na `λ x . e`.

Naloga: Zakaj se ta programski jezik imenuje Haskell?

Izraz `case`

Izraz

```
case e of
p₁ -> e₁
p₂ -> e₂
   ⋮
pᵢ -> eᵢ
```

primerja `e` z vzorci `p₁, ..., pᵢ`. Vrednost izraza je enaka `eⱼ`, kjer je `pⱼ` prvi vzorec, ki se ujema. Torej je `case` podoben izrazu `match` iz OCaml.

Primeri morajo biti pravilno zamaknjeni, lahko pa uporabimo tudi sintakso

```
case e of { p1 -> e1 ; ⋯ ; pi -> ei }
```

ki ne zahteva zamikanja.

Tipi

Imena tipov pišemo z velikimi črkami

Osnovni konstrukturji

- Osnovni tipi so `Int`, `Char`, `Bool`, ...
- `a -> b` je tip funkcij iz `a` v `b`
- `(a, b)` je produktni tip, ki ga v Ocamlu pišemo `a * b`
- `[a]` je tip seznamov elementov tipa `a`, v OCamlu `a list`

Definicije tipov

Haskell poznam definicije tipov

```
type T = ...
```

in definicije koinduktivnih algebraičnih tipov

```
datatype T = ...
```

Definicija `type` uvaja okrajšavo, `datatype` pa nov podatkovni tip.

(Tipe lahko definiramo tudi z `newtype`, ki ga ta trenutek ne bomo obravnavali.)

Na primer, sezname lahko definiramo takole:

```
datatype List a =
    Nil
  | Cons (a, List a)
```

Tip `Maybe` je definiran z

```
datatype Maybe a =
    Nothing
  | Just a
```

To je pravzaprav tip `a option` iz OCaml.

Primer: AVL drevesa

Več podrobnosti bomo spoznali tako, da bomo iz OCaml prepisali implementacijo AVL dreves v Haskell, glej [Avl.hs](Avl.hs).

Razredi tipov

Kakšnega tipa je `42`?

Razred tipov

Primeri

`Monoid`

`Functor`

`Applicative`