

Numerické datové typy

IB016 Seminář z funkcionálního programování

Vladimír Štill, Martin Ukrop

Fakulta informatiky, Masarykova univerzita

Jaro 2015

Numerické typy

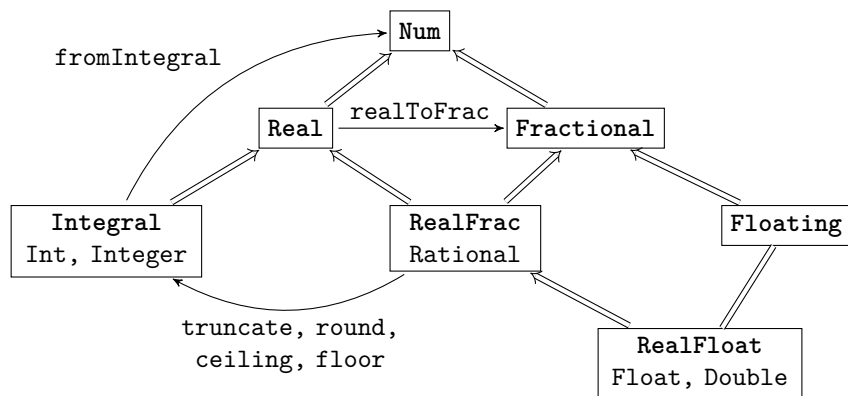
Základní

- Integer, Int: celočíselné typy se znamínkem, Int je omezený (32/64 bit typicky), Integer je neomezený
- Float, Double: s plovoucí desetinnou čárkou, platformě omezené (většinou 32, resp. 64 bit)
- Rational = Ration Integer: typ zlomků, s neomezenou přesností (konstruktor (%), $1.5 \equiv 3 \% 2$)

Další (většinou nejsou nutné)

- Int*, Word* z modulů Data.Int/Data.Word: celočíselné, znaménkové/neznaménkové, přesně daná bitová délka v názvu
- Complex (Data.Complex)
- Natural (Numeric.Natural): neomezená čísla bez znaménka (\geq GHC 7.8)
- Data.Fixed: obsahuje různé typy s pevnou desetinnou čárkou
- C* (Foreign.C.Types): typy pro volání z/do C

Numerické typové třídy a základní typy, některé konverze



Každý typ je vždy uveden v nejmenší třídě do níž patří (ale zároveň patří i do všech tříd nad ní).

Numerické typové třídy I

Num a

- všechny numerické typy
- (+), (-), (*), negate, abs, signum
- `fromInteger :: Integer -> a`: používá se pro konverzi celočíselných literálů – číselný literál 42 se interpretuje jako `fromInteger 42` a je tedy typu `Num a => a`

Real a

- `toRational :: a -> Rational` konvertuje reální číslo na typ `Rational` s maximální možnou přesností

Integral a

- `quot`, `rem`, `div`, `mod`, `quotRem`, `divMod`
- `toInteger :: a -> Integer`

Fractional a

- čísla podporující dělení
- (/), recip
- `fromRational :: Rational -> a`: používá se pro konverzi literálů s desetinnou čárkou – 1.5 se interpretuje jako `fromRational (3 % 4)`

Floating a

- typy umožňující trigonometrické, hyperbolické a příbuzné funkce (které nejde přesně vyjádřit na přesných typech)
- `pi`, `exp`, `log`, `sqrt`, `(**)`, `logbase`
- `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`

RealFrac a

- pro zaokrouhlování, extrakci komponent zlomků
- properFraction, truncate, round, ceiling, floor

RealFloat a (*spíše pro zajímavost*)

- pro efektivní extrakci komponent čísel v plovoucí desetinné čárce
- floatRadix, floatDigits, floatRange, decodeFloat, encodeFloat, exponent, significant, scaleFloat, isNaN, isInfinite, isDenormalized, isNegativeZero, isIEEE, atan2

Implicitní konverze, konstanty

- Haskell implicitní konverze hodnot při volání funkcí neprovádí
- nicméně odvozuje typy literálů zapsaných ve zdrojovém kódu podle jejich použití (literály jsou polymorfní)
- konstanty bez typu mohou dostat typ `Integer` (celočíselné) nebo `Double` (desetinné) pokud jsou nepoužité, jinak typ podle použití, polymorfní typy nutno uvádět explicitně¹

```
a = 42
```

```
b = 42
```

```
c :: Int
```

```
c = b + 1
```

```
f = 3.14
```

```
:t a ~> a :: Integer
```

```
:t b ~> b :: Int
```

```
:t f ~> f :: Double
```

¹Toto platí pokud je zapnuté `MonomorphismRestriction` v GHC, pokud ne jsou konstanty polymorfní (ghci `-XNoMonomorphismRestriction`)

Explicitní konverze

Zobecňovací funkce

- `fromIntegral :: (Integral i, Num n) => i -> n`
- `fromRational :: Fractional a => Rational -> a`

Zaokrouhlování desetinných čísel

- `truncate, round, ceiling, floor`
`:: (RealFrac f, Integral i) => f -> i`

Specializace

- `toRational :: Real a => a -> Rational`
- `toInteger :: Integral i => i -> Integer`

Další

- `realToFrac :: (Real a, Fractional b) => a -> b`

Explicitní konverze: příklady

- `Int` \rightsquigarrow `Float` – `fromIntegral`
- `Float` \rightsquigarrow `Int` – `round/ceil/...`
- `Rational` \rightsquigarrow `Integer` – `round/...`
- `Rational` \rightsquigarrow `Float` – `fromRational/realToFrac`

↓ from/to →	Integer	Int	Double	Rational
Integer	id	fromInteger		
Int	toInteger	id	fromIntegral	
Double	round		id	toRational
Rational	round		fromRational	id