Frege

@vijaykiran

FP-AMS/Dec 9, 2015

Friedrich Ludwig Gottlob Frege

/ˈfreɪɡə/

https://en.wikipedia.org/wiki/Gottlob_Frege

A <u>purely-functional</u>, <u>statically-typed</u>, <u>non-strict</u> language that compiles to Java.

In other words, a Haskell for the JVM.

History

- Created by Ingo Wechsung (@iweschu)
- **v1** : 2003/2004 - Written in Perl
- **v2**: 2006/2007 - Written in Java
- **v3**: 2011/Now - Self Hosted, Haskell Semantics

http://www.infoq.com/news/2015/08/frege-haskell-for-jvm

Algebraic Data Types

```
data Planet = Mercury
            | Venus
            | Earth
            | Mars
            | Jupiter
            | Saturn
            | Uranus
            | Neptune
            | Pluto
```

Algebraic Data Types

```
-- Point x y z
data Point = Point Double Double Double
```

Algebraic Data Types

```haskell
data Point = Point { x :: Double
                   , y :: Double
                   , z :: Double }
```

Algebraic Data Types

```
frege> Point 1 2 3
Point
frege> Point 1 2
Double -> Point
```

functions

```haskell
-- Increments argument
inc :: Int -> Int
inc x = x + 1
```

functions

```haskell
-- fns take only one argument
max :: Int -> Int -> Int
max x y
  | x > y     = x     -- Guards
  | otherwise = y
```

functions

```
frege> max 5 6
6
frege> max 5
Int -> Int
frege> maxOf5Or = max 5
function maxOf5Or :: Int -> Int
frege> maxOf5Or 7
7
```

functions

```
-- pattern matching
startsWithOne :: [Int] -> Bool
startsWithOne xs = case xs of
  (1:_)   -> True
  _       -> False
```

functions

```haskell
-- sugared pattern matching
startsWithOne :: [Int] -> Bool
startsWithOne (1:_)  = True
startsWithOne _      = False
```

Laziness

```haskell
allEvens :: [Int]
allEvens = map (*2) [1..]
```

Laziness

```
frege> take 10 allEvens
[2,4,6,8,10,12,14,16,18,20]
```

Laziness

```
frege> take 10 allEvens
[2,4,6,8,10,12,14,16,18,20]
```

Pure function

- Returns same result every time.
- Has no Side effects

Java Interop - Pure functions

```
data BigInt = native java.math.BigInteger where
    pure  native abs    :: Integer -> Integer
    pure  native negate :: Integer -> Integer
```

Java Interop - Impure functions

```
data Date = native java.util.Date where
    native new      :: () -> IO (MutableIO Date)
    native toString :: Mutable s Date -> ST s String
```

Java Interop - Exceptions

```
data NuPoEx = native java.lang.NullPointerException
derive Exceptional NuPoEx

frege> :t catch
JavaType γ => ST β α -> (γ -> ST β α) -> ST β α

frege> :t try
Monad γ => (β -> γ α) -> β -> γ α
```

Current State

- Frege compiler - Self Hosted
- Frege Repl and Online Repl
  - try.frege-lang.org
- Eclipse Plugin
- Standard Library*
- maven/gradle/sbt/leiningen Plugins

Contributors Welcome!

- Port Haskell Libraries
- Report Bugs and Fix Bugs :)
- Libraries
- More Haskell Compatibility

Contributors Welcome!

- Code : github.com/Frege
- IRC/Chat : gitter.im/Frege/frege
- Twitter : @fregelang