# Epigram

A Deep Dive into Dependent Types and Interactive Programming

By Ilyas Satik

# Agenda

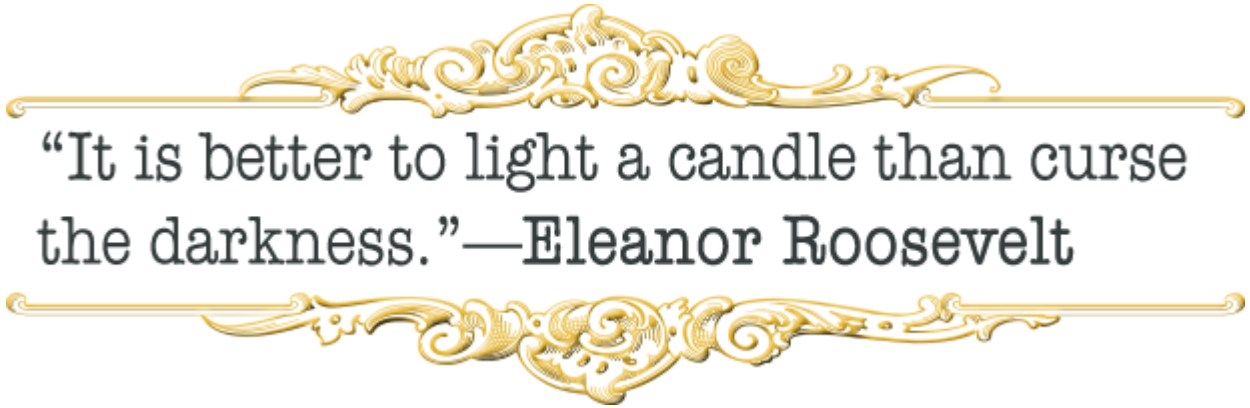
- Brief Overview & Importance of Topic
- Introduction to Dependent Types and Curry-Howard Correspondence
- Epigram and Interactive Programming
- Use Cases and Comparisons with Other Languages
- Demonstration
- Conclusion and personal experience
- Q&A

# Introduction

- What is an Epigram?

# Introduction

- What is an Epigram?

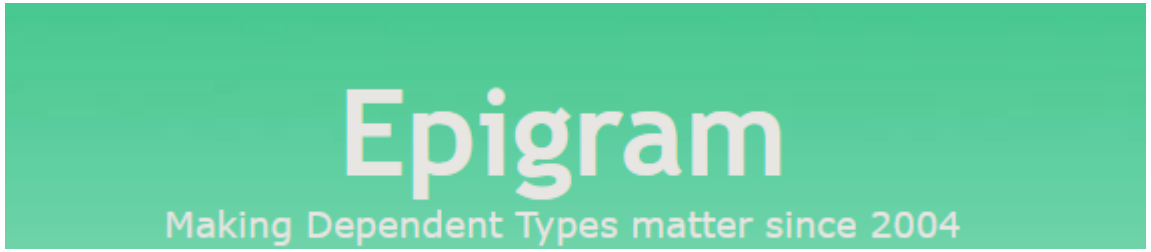



[1]

- „a short and witty statement, usually written in verse, that conveys a single thought or observation.“

# Introduction

- What is Epigram in computer Science?




[2]

```
------------------------------------------------------------------------------
                                        (   n : Nat   !
data (---------!  where (------------! ;  !-------------!
     ! Nat : * )        ! zero : Nat )    ! suc n : Nat )
------------------------------------------------------------------------------
     (   x, y : Nat   !
let  !----------------!
     ! plus x y : Nat )

     plus x y <= rec x
     { plus x y <= case x
       { plus zero y []
         plus (suc n) y => suc (plus n y)
       }
     }
------------------------------------------------------------------------------
inspect plus (suc (suc zero)) (suc (suc zero)) => suc (suc ?) : Nat
------------------------------------------------------------------------------
```

[3]

# Epigram in Computer Science

- Functional programming language
- Dependently typed
- Interactive programming
- Proof engine

# Functional programming

- We already know haskell:

```haskell
removeFirst :: Eq a => a -> [a] -> [a]
removeFirst _  [] = []
removeFirst x (x':xs)
    | x == x' = xs
    | otherwise = x' : removeFirst x xs
```

[4]

- Computation is the evaluation of mathematical functions
- Avoids changing state and mutable data
- Emphasizes 'what to solve' rather than 'how to solve'

# History of Dependent Types

BASED ON DEPENDENT TYPE THEORY

INVENTED IN EARLY 1970S BY MARTIN-LÖF

IDEA: TYPES ARE DEPENDENT ON THE VALUE OF OTHER TYPES

# Dependent Types

- Example:

$$\underline{\text{data}}\ \left(\frac{}{\mathsf{Nat} : \star}\right)\ \underline{\text{where}}\ \left(\frac{}{\mathsf{zero} : \mathsf{Nat}}\right)\ ;\ \left(\frac{n : \mathsf{Nat}}{\mathsf{suc}\ n : \mathsf{Nat}}\right)$$

$$\underline{\text{let}}\quad \left(\frac{x, y : \mathsf{Nat}}{\mathbf{plus}\ x\ y : \mathsf{Nat}}\right)$$

$$\begin{aligned}
&\mathbf{plus}\ x\ y \Leftarrow \underline{\text{rec}}\ x\ \{\\
&\quad \mathbf{plus}\ x\ y \Leftarrow \underline{\text{case}}\ x\ \{\\
&\qquad \mathbf{plus}\ \mathsf{zero}\ y \Rightarrow y\\
&\qquad \mathbf{plus}\ (\mathsf{suc}\ x)\ y \Rightarrow \mathsf{suc}\ (\mathbf{plus}\ x\ y)\ \}\}
\end{aligned}$$

Epigram doesn't build $\mathbf{plus} : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$ but rather a *proof*

$$\Diamond\mathbf{plus} : \forall x, y{:}\mathsf{Nat} \Rightarrow \langle \mathbf{plus}\ x\ y : \mathsf{Nat} \rangle$$

[5]

# Dependent Types

- Types that depend on values
- Allows more precise typing
- Enables proofs-as-programs

$$\underline{\text{data}} \quad \left(\frac{n : \mathsf{Nat} \;;\; X : \star}{\mathsf{Vec}\ n\ X\ :\ \star}\right) \quad \underline{\text{where}} \quad \left(\frac{}{\mathsf{vnil}\ :\ \mathsf{Vec}\ \mathsf{zero}\ X}\right)$$

$$\left(\frac{x\ :\ X\ ;\ \ xs\ :\ \mathsf{Vec}\ n\ X}{\mathsf{vcons}\ x\ xs\ :\ \mathsf{Vec}\ (\mathsf{suc}\ n)\ X}\right)$$

$$\begin{array}{ll}\underline{\text{data}} & \mathsf{Vec} : \mathsf{Nat} \to \star \to \star \\ \underline{\text{where}} & \mathsf{vnil} : \forall \_X : \star \Rightarrow \mathsf{Vec}\ \mathsf{zero}\ X \\ & \mathsf{vcons} : \forall \_X : \star \Rightarrow \forall \_n : \mathsf{Nat} \Rightarrow X \to \mathsf{Vec}\ n\ X \to \mathsf{Vec}\ (\mathsf{suc}\ n)\ X\end{array}$$

[6]

# The Curry-Howard correspondence

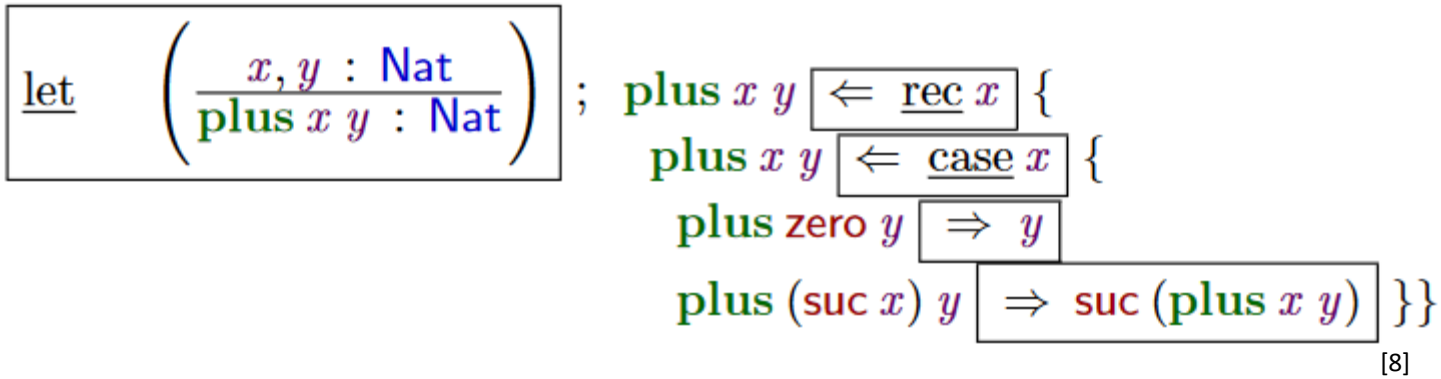
- Connection between programming and logic
- Proving a proposition = finding a program of a certain type.
- Enables theorem proving through programming

| Logic | Type Theory |
|---|---|
| proposition | type |
| proof | term |
| axiom | constant symbol |
| assumption | variable |
| provability | inhabitance |
| cut | substitution |
| normalization | reduction |
| ... | ... |

[7]

# Interactive Programming in Epigram

- Previous Example:




[8]

- Only ´Boxes´ need to be filled

# Interactive Programming in Epigram

- In Epigram:

```
-------------------------------------------------------------------------------
                                          (   n : Nat   !
data (---------!  where (------------! ;  !-------------!
     ! Nat : * )        ! zero : Nat )    ! suc n : Nat )
-------------------------------------------------------------------------------
     (   x, y : Nat   !
let  !---------------!
     ! plus x y : Nat )

     plus x y <= rec x
     { plus x y <= case x
       { plus zero y []
         plus (suc n) y => suc (plus n y)
       }
     }
-------------------------------------------------------------------------------
inspect plus (suc (suc zero)) (suc (suc zero)) => suc (suc ?) : Nat
-------------------------------------------------------------------------------
```

[9]

- Plus zero y is not ´filled in´
- ´Inspect´ already gives us information

# Use Cases of Epigram

## High correctness and precision

- Academia
- Critical Systems (banking software)
- Formal Verification

## Reality: hardly used at all

# Other Dependently Typed Languages

| | Coq | Agda | Idris | Epigram |
|---|---|---|---|---|
| Userbase | Large | Medium | Small | Small |
| Primary Use | Academic | Academic | Practical | Academic |
| Focus | Powerful | Complex Types | Practical | Clarity |
| Proof | Automation | Inductive | Effects | Interactive |

- Epigram shares many features with other dependently typed languages

# The Problems of Epigram

- Competition (e.g., coq)
- Learning Curve -> advanced concepts in type theory & formal logic.

- **Development Status:**
  - Few libraries, tools and learning resources
  - Not actively maintained (last official update in 2006)

# My Experience with Epigram

# My Experience with Epigram

Very difficult to install

No learning resources

Papers are very theoretical and complex

# Thanks for your attention

# Sources


- [1]: https://www.freelancewriting.com/creative-writing/using-the-epigram-as-a-literary-device/
- [2]: http://www.e-pig.org/
- [3], [9]: http://www.e-pig.org/downloads/epigram-system.pdf
- [4]: my own code
- [5], [6], [8]: http://www.e-pig.org/downloads/epigram-notes.pdf
- [7]: https://danielbmarkham.com/curry-howard-isomorphism/