Philosophy of (Functional) Programming

Attila Egri-Nagy
www.egri-nagy.hu
@EgriNagy

Akita International University, JAPAN

2018 @ClojuTRE
Ryoan-ji rock garden, zen buddhist temple, Kyoto, Japan
Enlightenment by watching pebbles (not moving at all)
My way: staring at code

Asking questions:

- What do I see?
- What do I see exactly? (removing hidden assumptions, unquestioned background knowledge)
- Is this a natural thing?
- or something that needs some mental force applied?
- What do I do when I ask these questions? (self-reflection)

Why???

Teaching beginners require constant re-evaluation of what I think about programming.
Poetry of Programming MAT245

Akita International University (Liberal Arts College), in the middle of nowhere in rural Japan

CODE IN CLOJURE

YOU MUST
Snowed in?
Time to learn some Clojure then!

egri-nagy.github.io/popbook
a path to enlightenment

Poetry of Programming
2018 Spring MAT245

egri-nagy.github.io/popbook/
(defn tower-of-hanoi
  [numofdisks source target spare]
  (if (= 1 numofdisks) (str " " source "->" target " ")
   (str
    (tower-of-hanoi (dec numofdisks) source spare target)
    (str " " source "->" target " ")
    (tower-of-hanoi (dec numofdisks) spare target source))))

(tower-of-hanoi 4 :1 :2 :3)
...or even simpler

(map inc [1 2 3 4])
Is this natural?

(map peel oranges)
(reduce cut empty-bowl fruits)

(reduce (fn [sum digit] (+ (* 10 sum) digit)) [5 2 1 7])
5217
(reduce (fn [sum digit] (+ (* 2 sum) digit)) [1 1 1 0])
14
What is going on when we read this line?

(map inc [1 2 3 4])
Black to move and live
Play it out in the head
Reading code

You play it out in your head.

Your brain is the runtime.

‘Something is something else’ is a very special relation.
Morphism – emulation, simulation, metaphor...

structure preserving (forgetting map)
When you play it out, what machine is in the head?

Notional machine:

“an abstract computer for executing programs of a particular kind”

Notional Machines and Introductory Programming Education by Juha Sorva, Aalto University
2013 ACM Transactions on Computing Education 13(2):8:1-8:31

assembly CPU registers, and memory locations
OOP communication network of objects
functional substitution model

Kind of bad news for hybrid languages.
Morphisms everywhere

Reading code  Our brain imitates machine execution.

Computer  A computer is something that emulates another computer.

Developing software  A computational process models a piece of reality.
What is programming?

We write text, which on a suitable computational device creates a process, whose outcome is some desirable output, or its dynamics is some required behaviour modeling a piece of reality.

smaller question: **What is coding?**

Describing an imagined computational process in a language.
What reality?

“...life and reality are at bottom amorphous, disordered, contradictory, inconsistent, non-rational, and non-objective.”

“Rational views of the universe are idealized models that only approximate reality.”

“The models are successful enough in predicting the behavior of things...”

Data & Reality – A timeless perspective on perceiving and managing information in our imprecise world by William Kent 1978 (3rd Edition 2012, only comments added)
Bit of encouragement…

“Programming is a highly unnatural activity.” Guy Steele

In ‘Coders at Work: Reflections on the Craft of Programming’ by Peter Seibel
Predicate functions – answering yes or no questions. Easy.

(filter orange? fruits)

(filter even? [1 2 3 4 5 6])
(2 4 6)
what makes filter great

TRUTHY

true
42
'("a\ b")
[1 2 3]
()
il?

FALSEY

false
nil
Filtering by a hash-set -- not so easy

```
(filter #{1} [1 2 3 2 3 1 1])
```

truthy-falsey – it is difficult to appreciate without knowing other programming languages
Example problem

Write a function replace-chars that takes a character-to-character hash-map and a string, and returns a string that has characters replaced according to the supplied lookup table.

```
(replace-chars \{\t \p, \p \t\} "pot")
"top"
(replace-chars \{\e \a, \l \p, \o \y\} "hello")
"happy"
```

Simple map doesn’t work.

```
(map \{\t \p, \p \t\} "pot")
(\t nil \p)
```
Solutions

Default value for lookup – nothing like this in everyday life.

(map (fn [c] (get {\t \p, \p \t} c c)) "pot")
(\t \o \p)

A student asked for help as it was not clear why this worked.

(defn replace-chars
    [m word]
    (apply str (map m word (seq word))))
Tradition vs. present time

Write a function `sum-of-digits` that returns the sum of the digits of a given number.

```
(sum-of-digits 1)
1
(sum-of-digits 12)
3
(sum-of-digits 123)
6
(sum-of-digits 999)
27
```

Traditionally: Use ASCII values.
Now: we can just use a hash-map.
Piece of code that is obviously wrong...

or is it?

(let [m {0 0, 1 1, 2 2, 3 3, 4 4,
      5 5, 6 6, 7 7, 8 8, 9 9}]
  (defn sum-of-digits
    [n]
    (apply + (map m (str n)))))
Types: a symptom

Write a function that Produces a sequence from a collection with all 42s removed.

Typical mistakes: careless naming and ‘double packaging’.

(defn f
  [x]
  (filter #{42} [x]))
Types: an argument

1. Categorizing things and processes into types is a defining feature of intelligence.
2. What is not written down, you have to keep it in your head.

Therefore: types!

Another tendency:

Programming languages getting closer to natural languages. Even in mathematics we don't use ‘static typing’.
Homomorphism

Definition

A homomorphism is a map \( \varphi : S \to T \) such that for all \( x, y \in S \)

\[
\varphi(x) \varphi(y) = \varphi(xy).
\]

In which semigroup is the composition done?

\( (S, \cdot), \ (T, \ast), \ x, y \in S, \ \varphi(x), \varphi(y) \in T \)

\[
\varphi(x) \ast \varphi(y) = \varphi(x \cdot y)
\]
The re-implementation game

The evil wizard took away map, filter and remove. But made a mistake and left reduce. Can we recover from the blow?

The enemy realized the mistake, and erased reduce as well. We only have recursion. Are we still ok?

In mathematics this is called searching for foundations.
Overarching statements and a practical advice

Mathematics $\sim$ Programming $\sim$ Philosophy $\sim$ Writing

Their unified essence is having a shared understanding of the world helped by off-loading our cognitive efforts to suitable languages.

Take-home message:

Pick a piece of code and meditate over it!
Conclusion

Programming is as difficult as understanding the world. Coding is easy.

Thank You!

https://egri-nagy.github.io/popbook/
www.egri-nagy.hu
github.com/egri-nagy/
twitter.com/EgriNagy