Functional Programming. Clojure

by Alexandru Gherega, Ph.D.
icslab.eu

Agenda

- FP short induction
- FP landscape
- Clojure overview

Definitions

Computer Programming

Activity of “writing” computer programs.

Computer Program

Series of computations on some input data with the goal of outputting some result.

Programming Language

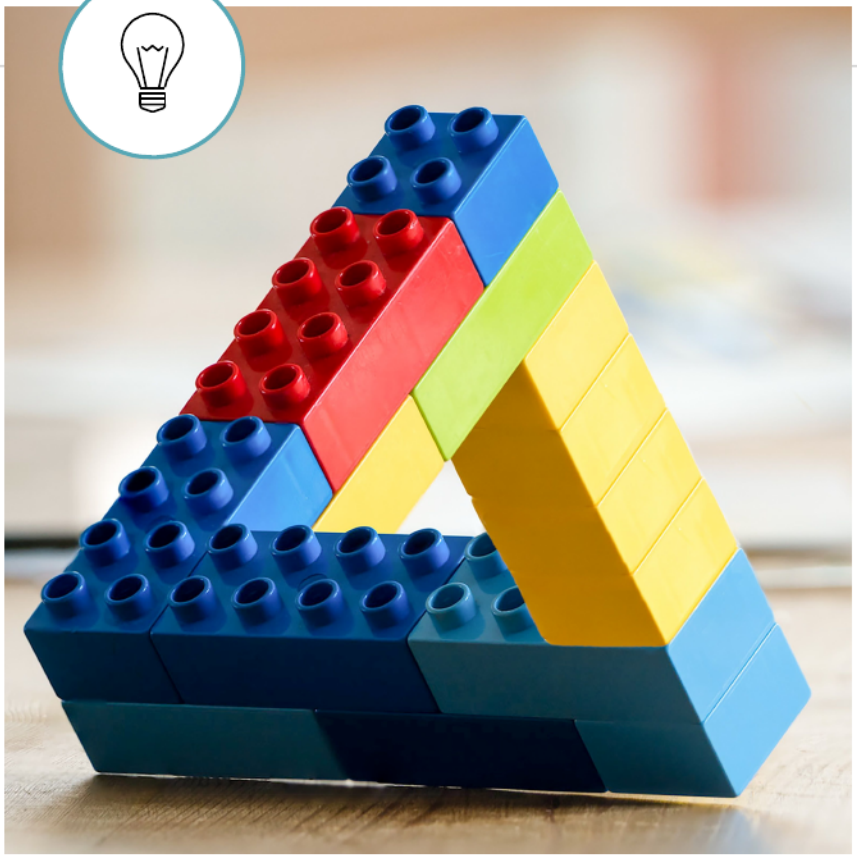
Set of symbols and keywords used to produce/describe/write a description of a computer program.

Let P be a set of $n \geq 3$ points in the plane, not all on a line. Then the set L of lines passing through at least two points contains at least n lines.

1

Programming paradigms

Machine, Assembly, High-level, System, Scripting, DSL, Visual Languages



Prove that π is irrational.

Programming Paradigms

Many have been proposed
and (in)formally studied

Usually you get a mix

Imperative programming
vs.
Declarative programming

Imperative programming

- imperative verb: do, execute, jump, goto
- statements linked to classical architecture
- control flow is **explicit**: *statements* ↔ *how*

Describes the solution/computer program in terms of instructions/commands – properly known as statements

Prove $1+\frac{1}{2}+\frac{1}{4}+...+\frac{1}{2^n}=1$ when $n \to \infty$

Imperative programming

- Take a sheet of paper, fold it in half along its height, holding it height-wise fold the upper right corner such that it meets the middle edge formed by the previous folding, ...

Structured Programming,
Object-Oriented Programming have imperative baked in

Declarative programming

- imperative verb is decoupled or hidden
- declarations linked to math abstractions
- control flow is **implicit**: *declarations* ↔ *what*

Describes the solution/computer program in terms of definitions/equations– properly known as declarations

How many functions are there of the form: $f: A \rightarrow B$ where $|A| = n$ and $|B| = k$?

Declarative programming

- Build “The Buzz” paper airplane glider

Functional Programming
Logic Programming have declarative baked in

2

Functional Programming

Powerful and beautiful formalism

“For twenty years programming languages have been steadily progressing toward their present condition of obesity”

“ John Backus - 1976

"... functional style of programming is founded on the use of combining forms for creating programs. [...]

Combining forms can use high level programs to build still higher level ones in a style not possible in conventional languages."

John Backus - 1976

Concept

“Programming in a functional language consists of building definitions and using the computer to evaluate expressions.” Philip Wadler & Richard Bird

Concept

Programmer: construct a function to solve a given problem. This function is expressed in notation that obeys normal mathematical principles.

Computer: act as an evaluator or calculator; its job is to evaluate expressions and print the results.

How many injective functions are there of the form: $f: A \rightarrow B$ where $|A| = n$ and $|B| = k$?

It's all about functions

Call
Fn call is uniquely determined by
Fn's arguments' values

If A and B are finite sets and |A| = |B| then $f: A \to B$ is injective iff f is surjective.

It's all about functions

Call
Fn call is uniquely determined by Fn's arguments' values

Assignments
NO!

If A and B are finite sets and |A| = |B| then $f: A \to B$ is injective iff f is surjective.

It's all about functions

Call
Fn call is uniquely determined by Fn's arguments' values

Assignments
NO!

Hence: No side-effects!

If A and B are finite sets and |A| = |B| then $f: A \to B$ is injective iff f is surjective.

the result of a function is solely determined by the value of its arguments **under all conditions**

There is an infinite number of primes.

Formalism

λ-calculus by Alonzo Church

Simple mathematical formalism gives a universal model of computation based on function application

Formalism

- Using λ-calculus one is enabled to formalize, study and investigate properties and notions common to all functional languages
- .. without being burdened by all the technicalities of actual languages, useless from a theoretical point of view

λ-calculus is *THE* functional language

Properties of pure FP languages

No side effects

Formal correctness proofs

Immutability

Functions are 1st class citizens

Recursion: powerful concept

Expressions are evaluated

The set R of irrational numbers is uncountable.

Benefits of pure FP languages

No side effects
Easy to reason about

Correctness proofs
Little or no need for tests

Immutability
über necessary in a world of multi-core devices and multi-threading programs

Functions are 1st class citizens
can construct formal concepts: you can be very expressive

Recursion: powerful concept
diff. From iteration
fib(x) = fib(x-1) + fib(x-2)

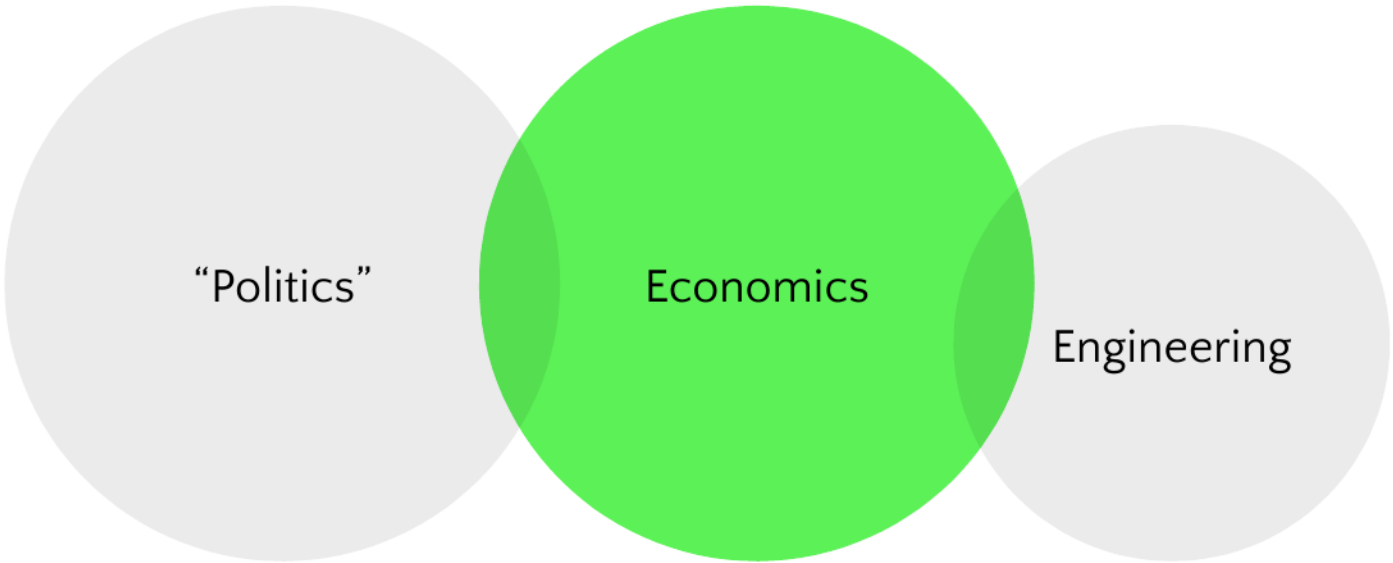
Evaluation
(diff. execution): when we eval an expression we are not changing anything

Drawbacks of pure FP languages

In the past
Efficiency (no longer an issue)

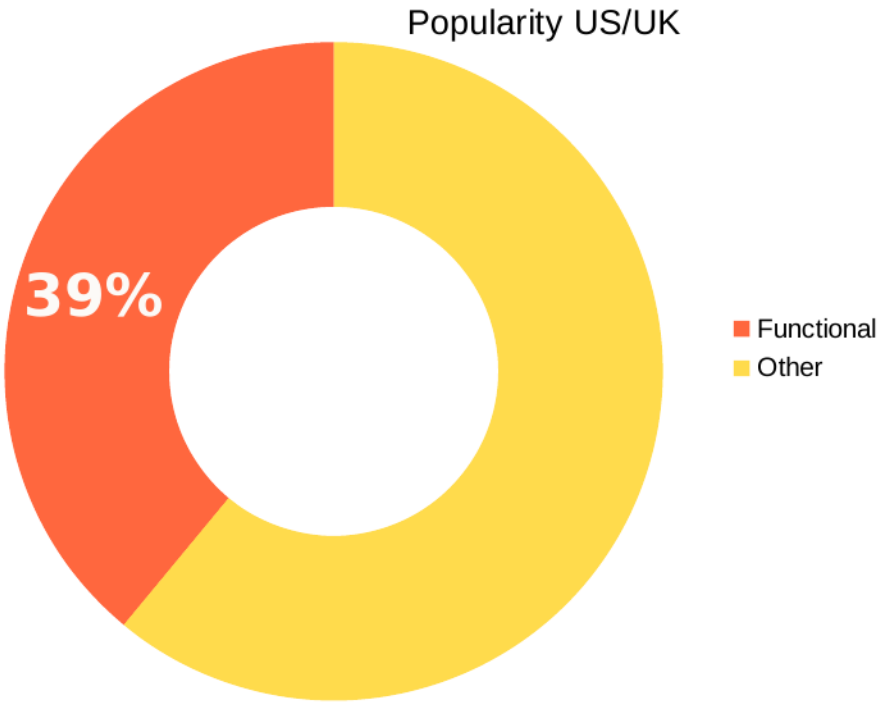
Suitability
For apps with a strong imperative nature – rarely the case in high-level languages

Why the comeback?



$$\forall\, m, n \text{ and } r \in \mathbb{N} \quad \binom{m+n}{r} = \sum_{j=0}^{r} \binom{m}{j}\binom{n}{r-j}$$

Landscape



Most used

Scala
Python
Clojure
JavaScript
Erlang
Haskell
Lisp/Scheme

$$\forall\, m, n \text{ and } r \in \mathbb{N} \quad \binom{m+n}{r} = \sum_{j=0}^{r} \binom{m}{j}\binom{n}{r-j}$$

Landscape

Beyond Tellerand

Twitter - Videos - Europe - Q4
An annual Programming conference in Berlin

BOB Conference

Twitter - Videos - Europe - Q4
A conference to discover the best tools available for so

3

Clojure

An opinionated functional language

[:w :w :w]

Rich Hickey

2007
because there was no Clo*j*ure language

Who?

Why?

Rationale
To have a Lisp FP language

Built for concurrency

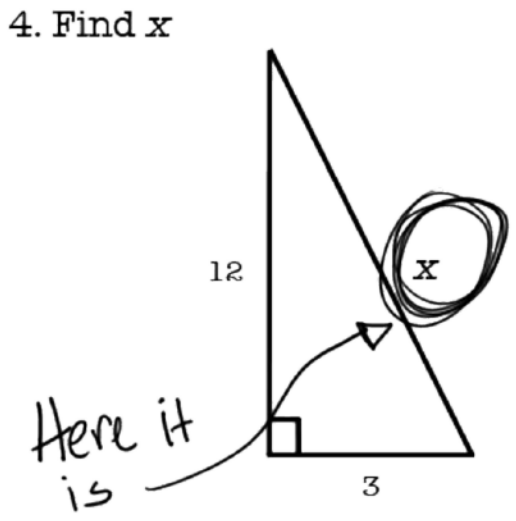
On-top of a powerful VM platform

... and
Couldn't find one!

Make money from nothing: Two persons rent a room that cost 15$ + 15$ = 30$. At reception a mistake is discovered: the room only costs 25$.

Clojure Philosophy

Simple



Expressive

Practical

The bellboy is asked to refund the 5$. He refunds 2$ to the first person and 2$ to the second and keeps 1$ to himself. So each person paid 13$ + 13$=26$.

Clojure Philosophy

Matrix transposition

$$M = \begin{pmatrix} 0 & 6 & 4 \\ 12 & -1 & 3 \\ -42 & 2 & 101 \end{pmatrix}$$

$$M^T = \begin{pmatrix} 0 & 12 & -42 \\ 6 & -1 & 2 \\ 4 & 3 & 101 \end{pmatrix}$$

Adding the 2$+2$ (the refund) we get 26$ + 4$=30$ .. and adding the 1$ the bellboy kept we get 31$. Where did the extra $ came from?

Clojure Philosophy

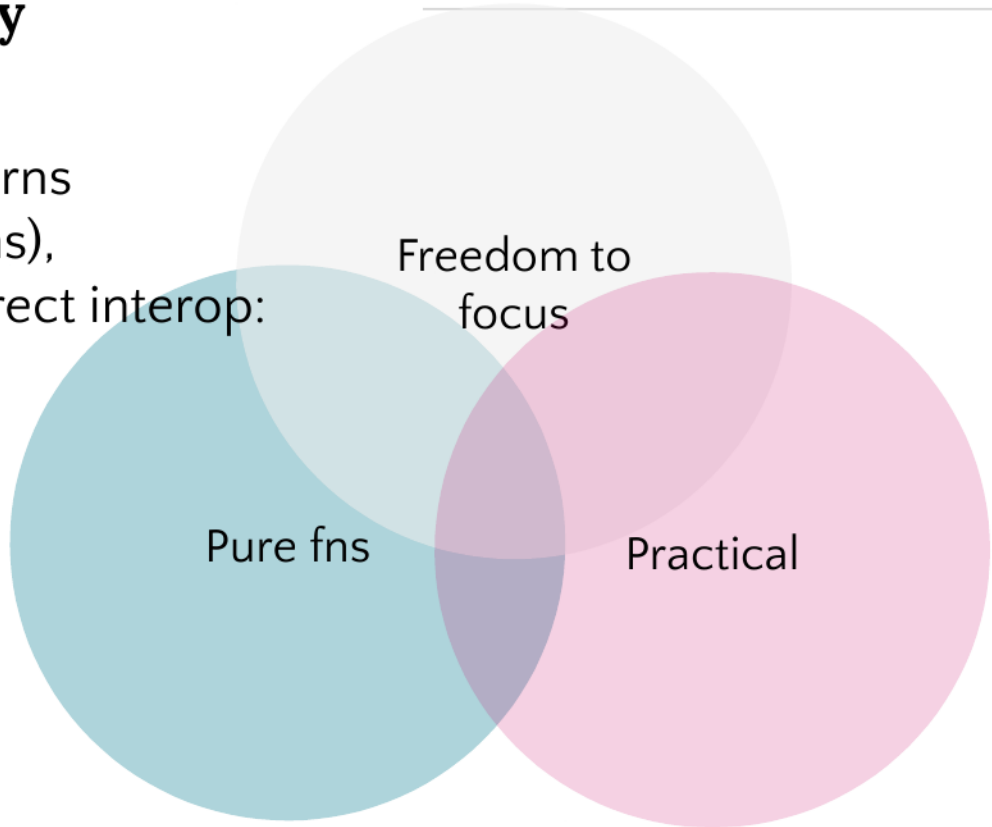
Matrix transposition

```
(map vector [ [:name :address :e-mail ]
              [“Alex” “secret” “icslabcrew@gmail.com”] ])
```

Adding the 2$+2$ (the refund) we get 26$ + 4$=30$ .. and adding the 1$ the bellboy kept we get 31$. Where did the extra $ came from?

Clojure Philosophy

Separation of concerns
(built on abstractions),
and concise, and direct interop:
you get clarity &
consistency



Clojure.core

- Syntax: prefix notation
- Homoiconic: code-as-data
- Evaluation is done to specific rules
- Dynamic typed
- Data is immutable (most of the time)

What sort of data? Literals, collections (persistent DS with shared structure)

Abstractions

Clojure is built on abstractions:
Sequence, Callability, Abstract transformations

User defined types & polymorphism

- All for polymorphism
- Hierarchies (done properly)
- Define Java classes from Clojure
- Complex *is-a* hierarchies
- Overloaded functions
- Multi-methods
- Protocols, Records, Deftypes

Concurrency

Built-in
Threads are safe due to immutability

Side-effects
You get 4 reference types and a powerful STM model

Vars, Refs, Agents, Atoms
(a)synchronous, (un)coordinated change of multiple/single locations or thread locally bound

Given $\sigma(n,k)=\sum_{n_1+....+n_k=n,n_i\in P}\binom{n}{n1,..,nk}$ we have the recursion $\sigma(n,k)=k(\sigma(n-1,k-1)+\sigma(n-1,k))$

A word about performance

Clojure has sipped that into its philosophy. If it's not practical it will have to go!

It uses powerful and fast DS, and adding concurrency, transients and the JVM to the mix and you get competitive (if not better) results.

Given $\sigma(n,k)=\sum_{n_1+\ldots+n_k=n,\, n_i\in P}\binom{n}{n1,\ldots,nk}$ we have the recursion $\sigma(n,k)=k(\sigma(n-1,k-1)+\sigma(n-1,k))$

Java interop

- Made seamlessly
- You get the beloved dot operator
- But it looks, feels and IS Clojure

You can integrate any Java API into Clojure and vice-versa.

Given $\sigma(n,k)=\sum_{n_1+\ldots+n_k=n,\, n_i\in P}\binom{n}{n1,\ldots,nk}$ we have the recursion $\sigma(n,k)=k(\sigma(n-1,k-1)+\sigma(n-1,k))$

4

Clojure ecosystem

Libs, communities and economics

Libs

You will find something for almost everything (if not you'll use some Java).

www.clojure-toolbox.com
dev.clojure.org
clojure.org
cognitect.com
clojars.org
clojuredocs.org
clojure-doc.org
github

Community

Several 10K users in US/UK alone

Keeps growing.
Very active.

Open: Everyone is welcome to contribute.

Once you can read Clojure you will understand everyone.

https://clojure.org/community

Economics

It's the same as fore every other mainstream Software Development technology.

You get ~80k £ /year to 500£ / day.

Not as many jobs as regular dev technologies but is steadily growing.

Sum up!

Abstractions
Formalisms are powerful!
If one desires efficiency and progress one needs to embrace the idea.

Functional Programming
Offers a good and pragmatic formalism for writing programs.

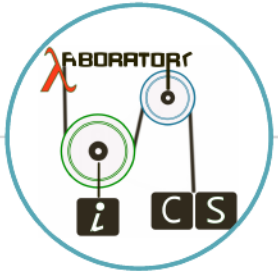
Clojure
One of the state of the art FP languages.

Be philosophical!
“Build solutions not futures”.

Research hard
We're still leaching on VonNeumann machines & whatever work was done in the last century.

Have fun
“...broken pencil”

The Art Gallery Theorem: For any museum with n walls, ⌊n/3⌋ guards suffice.



Thanks!

You can find me at
icslab.eu
alex.gherega@gmail.com