

Object Meets Function

Programming Paradigm

Steven Lolong

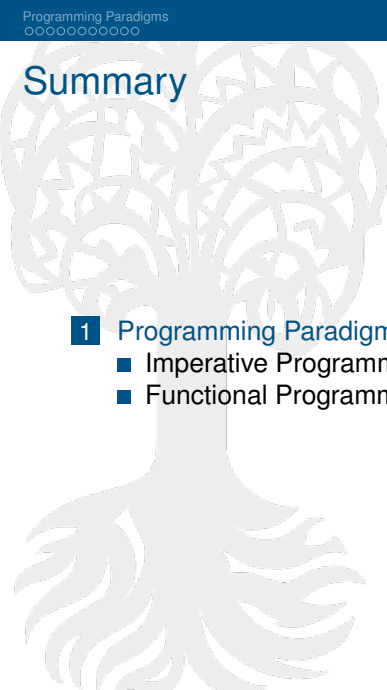
`steven.lolong(at)uni-tuebingen.de`

**Programming Language Research Group
Tuebingen University**

26 October 2022



Summary

- 
- 1 Programming Paradigms
 - Imperative Programming
 - Functional Programming

- Object Oriented Programming
- Logic Programming



Paradigms

“a Paradigm describes distinct concepts or thought patterns in some specific discipline.”

=Martin Odersky=

There are four Paradigms in PL:

- Imperative programming
- Functional programming (FP)
- Object-oriented programming
- ***Logic programming***



Imperative Programming

- Modifying mutable variable (change the content of variable)
- Using assignments (rather than binding)
- Control structures such as if-then-else, loops, break, continue, and return
- it comes from the concept of running instruction sequentially in the Von Neumann Machine (an old version of the computer)



Von Neumann Correspondence with Imperative

- Mutable variable \approx memory cells
- Variable dereferences \approx load instructions
- Variable assignments \approx store instructions
- Control structures \approx jump

Problem: Scaling up. How can we avoid conceptualizing programs word by word?
Martin Odersky



Imperative(s)

- C
- Javascript
- Please find the other examples on the internet based on the imperative language criteria.



Functional Programming

- Programming without mutable variable, assignments, loops, and other imperative control structure (in a restricted sense).
- Focus on function
- In particular, functions can be values that are produced, consumed, and composed



Functional Programming

In particular, functions in an FP language are first-class citizens. This means:

- They can be defined anywhere, including inside other functions
- Like a value, it can be passed as parameters to functions and returned as results (ex.: higher-order functions)
- A function can evaluate to another function (return a new function. ex.: higher-order function)
- As for other values, there exists a set of operators to compose functions



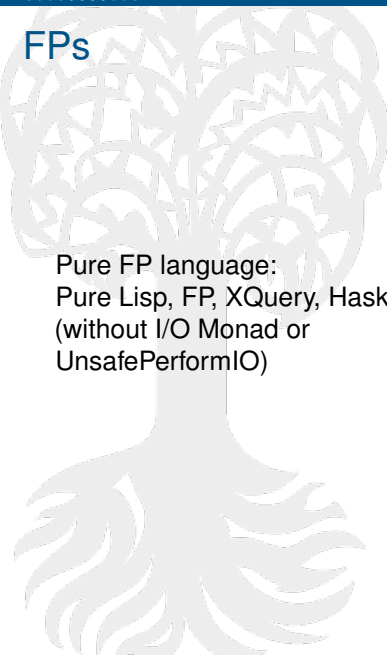
Functional Programming

In particular, functions in an FP language are first-class citizens. This means:

- They can be defined anywhere, including inside other functions
- Like a value, it can be passed as parameters to functions and returned as results (ex.: higher-order functions)
- A function can evaluate to another function (return a new function. ex.: higher-order function)
- As for other values, there exists a set of operators to compose functions



FPS



Pure FP language:
Pure Lisp, FP, XQuery, Haskell
(without I/O Monad or
UnsafePerformIO)

With additional features:
Lisp, Scheme, Racket (Dr.
Racket), Clojure SML, OCaml, F#,
Haskell, Scala, Javascript,
Smalltalk.



Object-oriented Programming (OOP)

Object = a group of attribute(s) [with] method(s).

- Abstraction (modeling)
- Encapsulation (visibility)
- Inheritance (reuse)



OOPs

Object = a group of attribute(s) [with] method(s).

- Smalltalk
- Java (Pure OOP)
- C++
- etc.



Logic Programming

ex.: **Prolog**

I will skip this concept!

