Introduction to Functional Programming Techniques

Luca Abeni

luca.abeni@santannapisa.it

About the Course

- The goal of this course is **not** to learn a specific functional programming language!
- So, what is the goal???
  - Understand what “functional programming” is
  - Learn how a program written using functional programming looks like
  - Be able to read and write programs developed according to FP
- Do not be afraid of functional programming!
  - It is not a solution for every problem, but can be **very** useful in many situations

Practical Details

- The course gives 2 credits $\rightarrow$ 20 hours of lesson
  - 8 lessons by 2.5 hours each
  - 1 lesson per week
- Try to have practical lessons (programming exercises) as much as possible
- Contacts: `luca.abeni@santannapisa.it`; office at TeCIP

Overview

- Why Functional Programming?
  - Functional Programming in real world
- Functional Programming as a paradigm
  - No side effects
  - No mutable state / no variables (???)
  - ...
- Functional Programming techniques
  - How to code iterations without mutable state
  - How to avoid mutable variables
  - ...

Functional Programming: What?

- What is *Functional Programming*?
  - Lots of definitions based on properties of functional programs
  - Usage of recursion, high-order functions / functions as values, anonymous functions, ...
- The real essence of functional programming is that *programs are obtained by combining pure functions*
  - Lots of interesting consequences
  - Programs are not executed, but evaluated
  - No side effects $\rightarrow$ no mutable state
  - ...

Functional Programming: Why?

- Why using functional programming? Which are the advantages?
  - Lot of theoretical discussions about coding what the program does and not how it does it...
  - ...Reusability of the code (functional programming is based on composing small/simple functions)...
  - ...More parallelizable code...
- But the big advantage of functional programming is that *it makes it easier to reason about the code*
  - Easier to formally prove the correctness of the code
  - Improved testability

Functional Programming: Drawbacks

- Sometimes, it is natural to see algorithms as sequences of actions...
  - ...Operating on mutable variables!
- The computers on which the program will run are based on the Von Neumann architecture...
  - ...So, the implementation of functional programs can be inefficient!
    - (actually, this issue can be solved...)

Programmers are generally not used to functional programming languages (and the mathematical theory behind them)

Some Fun about Functional Programming

- Functional programming languages can look strange

```
(defun factorial (n)
  (if (<= n 1)
    1
    (* n (factorial (- n 1)))))
```

  - "*Due to high costs caused by a post-war depletion of the strategic parentheses reserve, LISP never becomes popular*"
- The elegant mathematical theory behind functional programming looks scary
  - "*A monad is a monoid in the category of endofunctors, what's the problem?*"

Functional Programming: Academic Only???

- Strong mathematical foundations, lots of formality...
  - λ Calculus (untyped, typed, ...), type theory
  - Monads, Monoids, Functors, Endofunctors, Lenses, ...
- Pretty theoretical stuff!!! Only good for writing papers?
- No! FP is really used in practice!
  - The WhatsApp server is written in Erlang
  - Haskell is used in satellites (see https://github.com/erkmos/haskell-companies for more details)
  - Scala is used in Amazon
  - F# and Clojure are used in industry too...

What is Functional Programming

- Lof of different definitions around
  - Many of them are based on enumerating functionalities of FP languages
  - Many people tend to identify FP with Haskell, or Lisp, ...
- Let's start with what FP is not:
  - Functional Programming is not bound to a specific programming language!
- Functional Programming is a paradigm that can be applied using many different languages...

The Functional Programming Paradigm

- The key essence of functional programming is to write programs without using side effects
  - Programs as compositions of **pure functions**
  - Pure function: function without side effects!
- Variable assignments are side effects!
  - So, no mutable variables...
  - Notice: `for`/`while` loops need mutable variables... So, no loops???

Pure Functions: What?

- What is a pure function? A mathematical function
  - As in: $f(x) = x^2 + 2x + 1$
  - No side effects
  - When invoked multiple times with the same input value, it always returns the same result
- Different from “C-style functions” (which can do I/O, rely on global variabiles, ...)
- Different from OO methods (which can access the object’s state)
- Programs composed by pure functions can be executed by evaluating the functions
  - Replacing the invocation of a function with its return value!

Purity and its Consequences

- Example of “impure” functions:
  - Functions performing I/O, reading the current time, etc...
  - Functions using some kind of mutable state
- Pure functions **must** have a return value!
  - Otherwise, they are just useless
- When we invoke a pure function, it is not going to interfere with any other component of the application
  - Referential transparency: a function invocation can be replaced with the return value without changing the application’s behaviour

Mutable State

- So, the essence of Functional Programming is related to (the absence of) mutable state
  - But why is mutable state so important?
  - Controlling the evolution of the program state is important to understand the program's behaviour!
- Object Oriented programming controls the program complexity by encapsulating the mutable state in objects...
- ...Functional programming controls the program complexity by avoiding mutable state!
  - Or by confining state mutation in very specific (non-functional!) parts of the code

Working without a Mutable State

- Variable can be initialized, but other assignments are forbidden
  - Everything is "`const`"
- What to do when I need to update the value of a variable?
  - Define a new variable, initialized with the updated value?
  - Pass the updated value as an argument to a function? (the formal parameter can be seen as a "variable containing the updated value"...)
- Pattern: function invocation instead of assignment
- What about loops? They become recursive invocations...