Programming Languages – Functional Languages

Mahmoud Abdelsalam

Functional Programming - Intro

- A Turing machine is a mathematical model of computation that defines an abstract machine, which manipulates symbols on a strip of tape according to a table of rules. Despite the model's simplicity, given any computer algorithm, a Turing machine capable of simulating that algorithm's logic can be constructed.

Source: https://en.wikipedia.org/wiki/Turing_machine

Functional Programming - Intro

- Church’s thesis states that any real-world computation can be translated into an equivalent computation involving a Turing machine. In Church's original formulation (Church 1935, 1936), the thesis says that real-world calculation can be done using the lambda calculus, which is equivalent to using general recursive functions.

Source: https://mathworld.wolfram.com/Church-TuringThesis.html

Functional Programming - Intro

- Imperative languages (e.g., C, C++, Java, etc.) have all these components:
  - Name/variables
  - Variables = values (assignments)
  - If conditions
  - Iterations
  - Data types (strings, ints, booleans, arrays, classes, etc..)

Functional Programming - Intro

- Imperative languages (e.g., C, C++, Java, etc.) have all these components:
  - Name/variables
  - Variables = values (assignments)
  - If conditions
  - Iterations
  - Data types (strings, ints, booleans, arrays, classes, etc..)
- Throughout these components, you can:
  - Evaluate expressions
  - Change states (values of variables)

Functional Programming - Intro

- Imperative languages (e.g., C, C++, Java, etc.) have all these components:
  - Name/variables
  - Variables = values (assignments)
  - If conditions
  - Iterations
  - Data types (strings, ints, booleans, arrays, classes, etc..)
- Throughout these components, you can:
  - Evaluate expressions
  - Change states (values of variables)
- **What is wrong? Side effects**

Functional Programming - Intro

- Consider this C++ example:

  int x = 10;

  f(x); *(Corrected from video)*

  cout << x;

  – **Can you tell what is x?**

Functional Programming - Intro

- Consider this C++ example:

  int x = 10;

  f(x); *(Corrected from video)*

  cout << x;

  – **Can you tell what is x?**

  NO, unless you check the internal code of function “f” since it could have changed the value of “x”

Functional Programming - Intro

- Consider another example in Python:

  l = [1, 2, 3]

  f(l)

  print(l)

  - **Can you tell what is the values of “l”?**

Functional Programming - Intro

- Consider another example in Python:

  l = [1, 2, 3]

  f(l)

  print(l)

  – **Can you tell what is the values of "l"?**

  Again NO, but can't we simply check "f"!

Functional Programming - Intro

- Consider another example in Python:

  l = [1, 2, 3]

  f(l)

  print(l)

  - **Can you tell what is the values of "l"?**

  Again NO, but can't we simply check "f"!

  - But what if: *def f(l):*

    x = a(l) + aa(l) + aaa(l) + ...

    ...

Functional Programming - Intro

- Consider another example in Python:

  l = [1, 2, 3]

  f(l)

  print(l)

  – **Can you tell what is the values of "l"?**

  Again NO, but can't we simply check "f"!

  – But what if: *def f(l):*

  x = a(l) + aa(l) + aaa(l) + ...

  ...

  Sol: Check every function "a", "aa", …

Functional Programming - Intro

- Consider another example in Python:

  l = [1, 2, 3]

  f(l)

  print(l)

  – **Can you tell what is the values of "l"?**

  Again NO, but can't we simply check "f"!

  – But what if: *def f(l):*

  x = a(l) + aa(l) + aaa(l) + ...

  ...

  Sol: Check every function "a", "aa", …

  TOO COMPLICATED TO DEBUG

What is Functional Programming

- Functional languages treat programs as mathematical expressions (Church's thesis).
- Mathematical vs programming functions:

What is Functional Programming

- Functional languages treat programs as mathematical expressions (Church's thesis).
- Mathematical vs programming functions:

  - $f(x) = x * 2$

$1 \rightarrow 2$

$2 \rightarrow 4$ *(Corrected from video)*

…

  - $g(x, y) = f(x) + f(y)$
  - $f(x) = \begin{cases} 1, \text{ if } x \text{ is even} \\ 2, \text{ if } x \text{ is odd} \end{cases}$

What is Functional Programming

- Functional languages treat programs as mathematical expressions (Church's thesis).
- Mathematical vs programming functions:

  - $f(x) = x * 2$

  $1 \rightarrow 2$

  $2 \rightarrow 3$

  …

  - $g(x, y) = f(x) + f(y)$
  - $f(x) = \begin{cases} 1, \text{ if x is even} \\ 2, \text{ if x is odd} \end{cases}$

  Can do more:

  - Change value of x
  - Delete file
  - Update Database
  - Same input can give different result based on time

What is Functional Programming

- A paradigm trying to imitate mathematical functions, so we need to add **restrictions**.

What is Functional Programming

- A paradigm trying to imitate mathematical functions, so we need to add **restrictions**.
- Has to add concepts like:
  - Immutability (Can't change your variables once assigned.)

What is Functional Programming

- A paradigm trying to imitate mathematical functions, so we need to add **restrictions**.
- Has to add concepts like:
  - Immutability (Can't change your variables once assigned.)
  - No loops (e.g., for, while, etc..)

What is Functional Programming

- A paradigm trying to imitate mathematical functions, so we need to add **restrictions**.
- Has to add concepts like:
  - Immutability (Can't change your variables once assigned.)
  - No loops (e.g., for, while, etc..)
  - No side effects (e.g., updating database, writing to screen, etc..)

What is Functional Programming

- A paradigm trying to imitate mathematical functions, so we need to add **restrictions**.
- Has to add concepts like:
  - Immutability (Can't change your variables once assigned.)
  - No loops (e.g., for, while, etc..)
  - No side effects (e.g., updating database, writing to screen, etc..)
  - Control over the order of execution is of low importance.

What is Functional Programming

- Functional languages such as Lisp, Scheme, FP, ML, Miranda, and Haskell are an attempt to realize Church's lambda calculus in practical form as a programming language
- The key idea: do everything by composing functions
  - no mutable state
  - no side effects