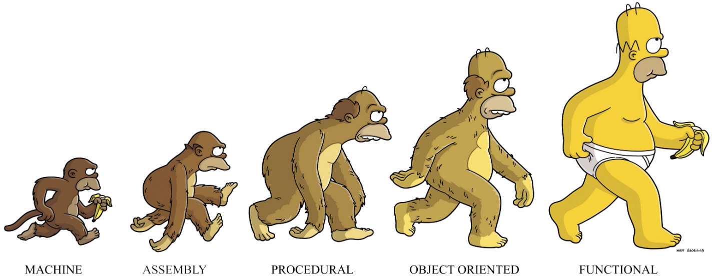
COMP302: Programming Languages and Paradigms

Prof. Brigitte Pientka (Sec 01)
bpientka@cs.mcgill.ca

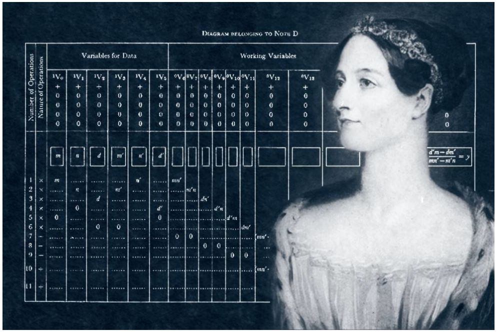
Francisco Ferreira (Sec 02)
fferre8@cs.mcgill.ca

School of Computer Science
McGill University

Week 5-1, Fall 2017



Functional Tidbit: Tuesday (10 Oct)



10 Dec 1815 – 27 Nov 1852
Inventor of the Analytic Engine

"The analytical engine weaves algebraic patterns just as the Jacquard loom weaves flowers and leaves."

Happy (belated) Ada Lovelace Day!

Functional Tidbit: !

"I find languages that support just one programming paradigm constraining."

- Bjarne Stroustroup

Computation and Effects

So far:

Expressions in OCaml have characteristics:

- An expression has a type
- An expression evaluates to a value
  (or diverges).

Today:

Expressions in OCaml may also have an *effect*.

Recall: Variable Bindings and Overshadowing

Example 1:

```
let (k : int) = 4;;
let (k : int) = 3 in  k * k ;;
k;;
```

Recall: Variable Bindings and Overshadowing

Example 1:

```
let (k : int) = 4;;
let (k : int) = 3 in  k * k ;;
k;;
```

Example 2:

```
let pi = 3.14 ;;
let area (r:float) = pi *. r *. r;;

let a2 = area (2.0)

let (pi : float) = 6.0;;

let b1 = area (2.0) = a2

let area (r:float) = pi *. r *. r;;
let b2  = area (2.0) = a2
```

Today

How to program with state?

Today

How to program with state?

– Demo –

How to program with state? – Allocate and Compare

- How to allocate state?

```
let x = ref 0
```

Allocates a reference cell with the name `x` in memory and initializes it with `0`.

How to program with state? – Allocate and Compare

- How to allocate state?

```
let x = ref 0
```

  Allocates a reference cell with the name `x` in memory and initializes it with `0`.

- How to compare two reference cells?

How to program with state? – Allocate and Compare

- How to allocate state?

```
let x = ref 0
```

  Allocates a reference cell with the name `x` in memory and initializes it with `0`.

- How to compare two reference cells?

  Compare their address: `r == s`

  Succeeds, if both `r` and `s` are names for the same location in memory

  Compare their content: `r = s`

  Succeeds, if both reference cells store the same value.

How to program with state? – Read and Write

- How to read value stored in a reference cell?

```
!x
```

Read value that is stored in the reference cell with name `x`.

```
let {contents = x} = r
```

Pattern match on value that is stored in the reference cell with name `x`.

- How to update the value stored in a reference cell?

```
x := 3
```

Writes the value in the reference cell with the name `x`
The previously stored value is overwritten.

Revisiting Variable Binding and Overshadowing

– Demo –

Imperative Programming in OCaml

```ocaml
let imperative_fact n =
  begin
    let result = ref 1 in
    let i = ref 0 in
    let rec loop () =
      if !i = n then ()
      else (i := !i + 1; result := !result * !i; loop ())
    in
    (loop (); !result)
  end
```

Imperative Programming in OCaml

```
let imperative_fact n =
  begin
    let result = ref 1 in
    let i = ref 0 in
    let rec loop () =
      if !i = n then ()
      else (i := !i + 1; result := !result * !i; loop ())
    in
    (loop (); !result)
  end
```

- More complicated than the purely functional version
- Considered bad style in a functional language
- Harder to reason about its correctnes

Good Uses of State

```
let counter = ref 0

(* newName () ===> a,  where a is a new name *)
(* Names are described by strings and an nat. *)
let newName () =
  (counter := !counter + 1;
   "a" ^ string_of_int (!counter))
```