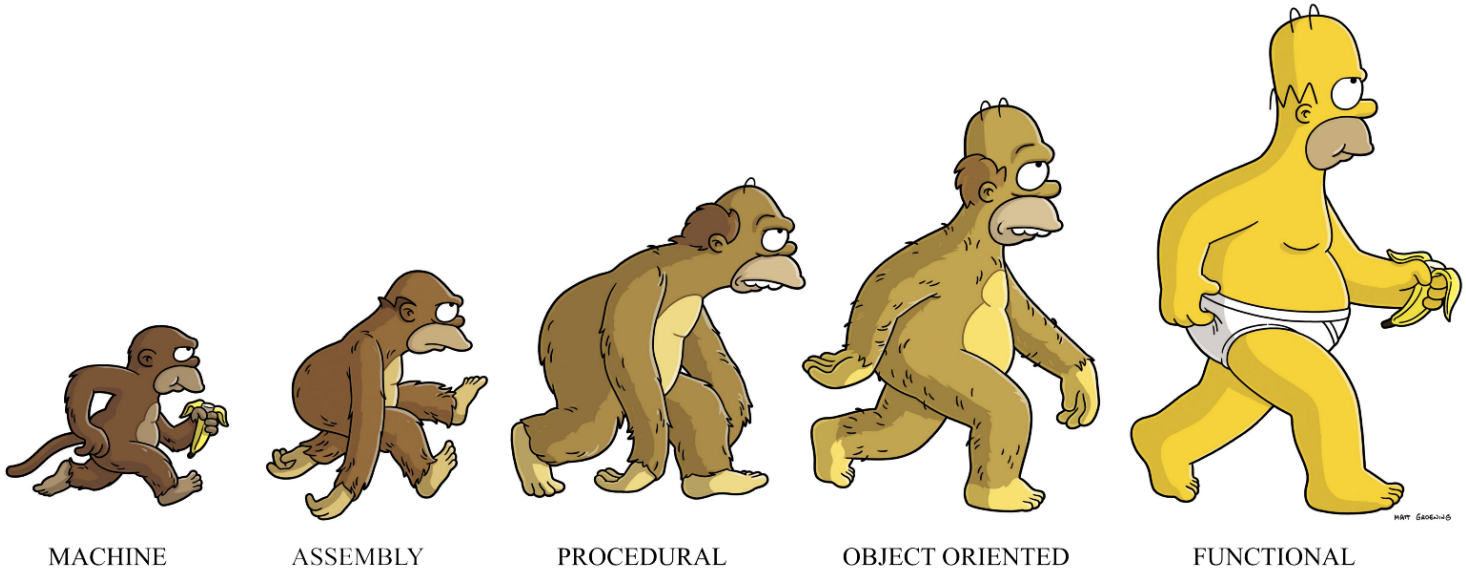
COMP302: Programming Languages and Paradigms

Prof. Brigitte Pientka (Sec 01 / Sec 02)
bpientka@cs.mcgill.ca

School of Computer Science
McGill University

Week 5-1, Fall 2018



UNDERGRADUATE RESEARCH CONFERENCE

14:00	**Poster competition:** start of judging and public viewing	**All are welcome!**
17:00	**Keynote address:** "Having fun with research as a student and a professor" **Prize ceremony** and **networking reception** to follow	**Professor John Stix** Professor, Dept. of Earth & Planetary Sciences Associate Dean of Research, Faculty of Science

Thu., Oct. 4, 2018 | Arts Building Lobby + Moyse Hall | INFORMATION: www.mcgill.ca/science

SCIENCE

McGill

Office for Undergraduate Research in Science

What's on the midterm?

There are different types ...

There are different types ...

Expression `fun x y -> if x * (-1) < 0 then (y,x) else (x,y)`
has the most general type

a) `int * int`

b) `int -> int -> int * int`

c) `int * int -> int * int`

d) `int -> 'a -> int * int`

e) `int -> 'a -> int * 'a`

f) `int * 'a -> int * int`

g) `int * 'a -> int * 'a`

Answer: ?

There are different types ...

What is the most general type of the following expression

```
1 let rec f b l = match l with
2   | [] -> b
3   | x::xs -> f x
```

a) `'a -> 'a list -> 'a`

b) `('a -> 'b) -> 'a -> 'a list -> 'b`

c) ill-typed

d) `int -> int list -> int`

There are different types ...

What is the most general type of the following expression

```
1 type 'a tree = Empty | Node of 'a * 'a tree * 'a tree
2
3 let build_tree x l r = Node (x, l, r)
```

a) `'a -> 'a tree -> 'a tree -> 'a tree`

b) `'a tree`

c) `'a * 'a tree * 'a tree -> 'a tree`

There are different types ...

What is the most general type of the following expression

```
1 let rec zoom x =
2   if zoom x > 3 then zoom (x - 1) else zoom (x + 1)
```

a) infinite loop

b) `int -> int`

c) `int`

There are different types ...

What is the most general type of the following expression

```
1 let double (f, x) = f (f x)
```

a) infinite loop

b) `('a -> 'a) -> 'a -> 'a`

c) `'a -> 'a * 'a -> 'a`

d) `('a -> 'b) -> 'a -> 'b`

There are different types ...

What is the most general type of the following expression

```
1 let apply f x = f x in apply (fun x -> x + 1) 3
```

a) `int`

b) ill typed

c) `(int -> int) -> int -> int`

d) `('a -> 'b) -> 'a -> 'b`

Its' all about values ..

What value does this expression have? – The answer is ...

```
1 let x = 3 in
2 let y = x + 3 in
3 let check x = x = 3 in
4 let x = y in
5   check x
```

Let's do some programming.

Midterm Topics – Summary

1) Reasoning about programs:

- Static type checking: What is the most general type of an expression? When is type checking done?
- Semantics: How is a program evaluated? What value does it produce?

2) Principles of functional programming

- Pattern matching and recursion
- Higher-order functions
  a) Passing functions as arguments: Abstracting over common functionality; proficiency with using built-in higher-order functions
  b) Returning functions as results: examples we have seen: returning a function that swapps the arguments; currying, uncurrying, Church numerals