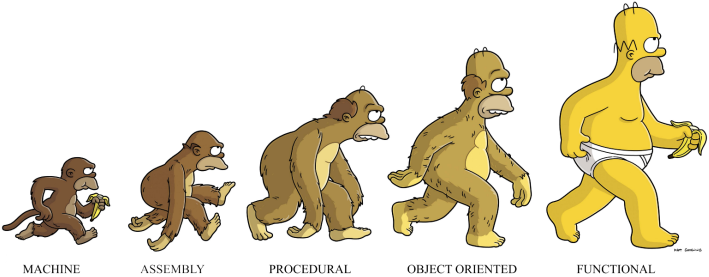
COMP302: Programming Languages and Paradigms

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

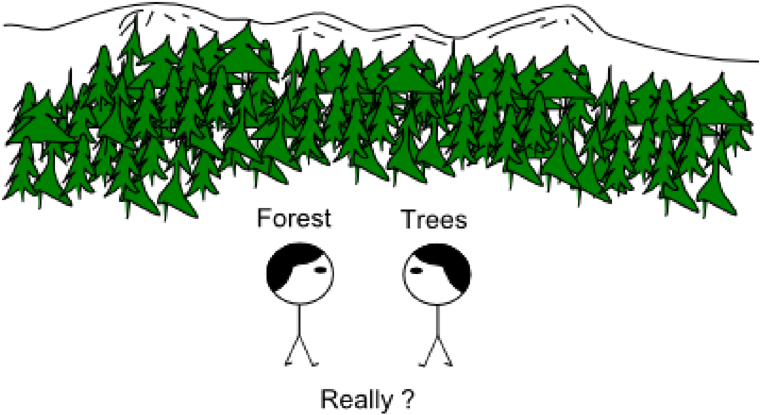
Francisco Ferreira (Sec 02)
fferre8@cs.mcgill.ca

School of Computer Science
McGill University

Week 3-2, Fall 2017



Can't see the forest for the trees



Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `'a` then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `'a` then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

How to define a recursive data type for trees in OCaml?

Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `'a` then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

How to define a recursive data type for trees in OCaml?

```
type 'a tree =
  Empty
| Node of 'a * 'a tree * 'a tree
```

Inductive definition of a binary tree

- The empty binary tree `Empty` is a binary tree
- If `l` and `r` are binary trees and `v` is a value of type `'a` then `Node(v, l, r)` is a binary tree.
- Nothing else is a binary tree.

How to define a recursive data type for trees in OCaml?

```
type 'a tree =
  Empty
| Node of 'a * 'a tree * 'a tree
```

Let's do some programming with trees!

How to prove it?

Step 2: How to reason inductively about trees?

How to prove it?

Step 2: How to reason inductively about trees?

Analyze their structure!

How to prove it?

Step 2: How to reason inductively about trees?

Analyze their structure!

The recipe ...

To prove a property `P(t)` holds about a binary tree `t`

Base Case:	`t = Empty`	
Show	`P(Empty)` holds	
Step Case:	`t = Node(x, l, r)`	
IH	`P(l)`	Assume the property `P` holds
IH	`P(r)`	for trees smaller than `t`.
Show	`P(Node(x, l, r)` holds	Show the property `P` holds for the tree `t`.

Let's prove something!

```
let rec insert ((x,d) as e) t = match t with
  | Empty                  -> Node(e, Empty, Empty)
  | Node ((y,d'), l, r) ->
    if x = y then  Node(e, l, r)
    else  (if x < y then Node((y,d'), insert e l, r)
           else  Node((y,d'), l, insert e r))
```

Theorem: For all trees `t`, keys `x`, and data `dx`,

`lookup x (insert (x, dx) t)` $\Longrightarrow^{*}$ `Some dx`

```
let rec lookup x t = match t with
  | Empty                 -> None
  | Node ((y,d), l, r) ->
  if x = y then  Some(d)
  else (if x < y then lookup x l
        else lookup x r)
```

Remember the slice of cake?

Give an OCaml data type definition for cake!