Recitation 23: Proofs about Programs (1 of 2)

Correctness as defined by <u>spec</u>

(** [fact n] is n factorial,
or [n!] *)

Validation

Informal

PP

TDD

Static analysis

Mathematical Proofs

"Formal" verifications

Formal

Equality of expressions

$$e = e' \qquad e \xrightarrow{*} v \xleftarrow{*} e'$$

$$41 + 1 \xrightarrow{1} 42 \xleftarrow{0} 42$$

Ocaml =

Functions

$$\text{fun } x \to x \overset{?\checkmark}{=} \text{fun } y \to y$$

Define $f = g$ if for all inputs v, $f\ v = g\ v$

$$(\text{fun } x \to x)\ v = (\text{fun } y \to y)\ v$$

$$(\text{fun } x \to x)\ v \xrightarrow{1} v' \xleftarrow{1} (\text{fun } y \to y)\ v$$

Gotchas

to ~~a~~ ask

$e = e'$

e, e' must be

well-typed (evals to value)

pure (no refs, i/o)

total (no ~~∞~~ loops, exceptions)

Example:

let twice f x = f (f x)
let compose f g x = f (g x)

Show: twice f x = compose f f x

twice f x = f (f x).

compose f f x = f (f x)

twice f x
= {eval twice}
f (f x)
= {eval compose}
compose f (f x)

Technically this is Equational Reasoning
- transitivity
- subst of eqs for eqs
- just high school algebra!

Induction (2800 math)

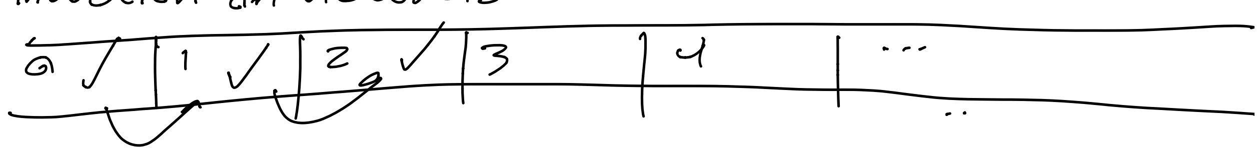
Example:

let rec sumto n =
  if n = 0 then 0 else n + sumto (n-1)

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Show: sumto n = (n * (n+1)) / 2

Induction on naturals



P(0)
if P(k) then P(k+1)

Prove by ind. on n

Base case: n = 0

Show sumto 0 = (0 * (1 + 0)) / 2

sumto 0

= {eval}

0

= {arithmetic}

(0 * (1 + 0)) / 2

Inductive case

IH: sumto k = (k * (k + 1)) / 2

Show: sumto (k + 1) = ((k+1) * ((k+1) + 1)) / 2

sumto (k + 1)

= {eval}

(k + 1) + sumto k

= {IH}

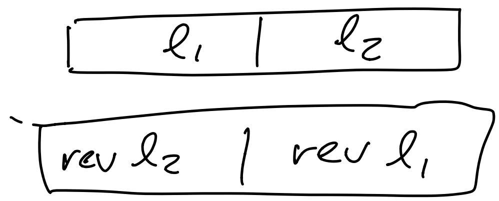
(k + 1) + (k * (k + 1)) / 2

= {algebra}

((k + 1) * ((k + 1) + 1)) / 2

$\square$

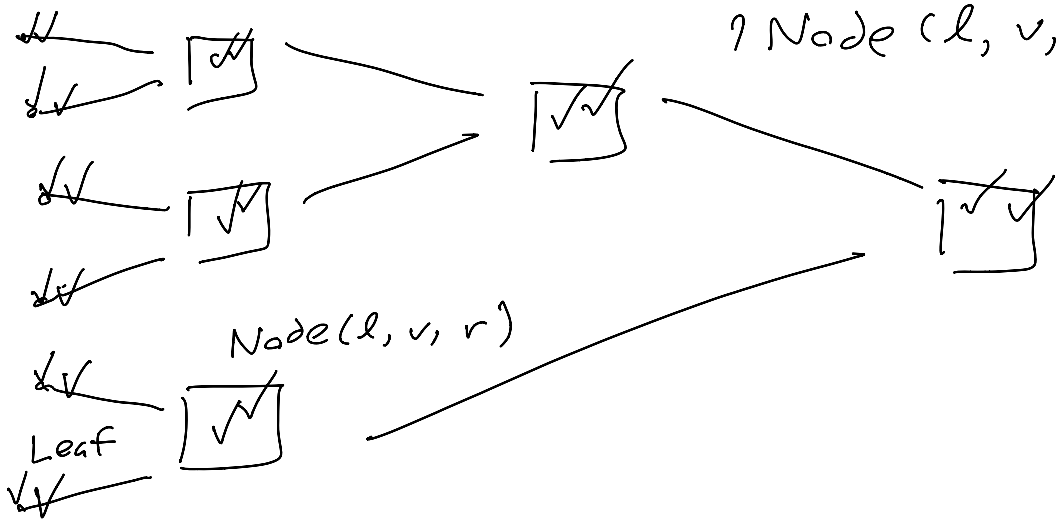
List.rev?



$$\text{rev } l_1 @ l_2 = \text{rev } l_2 @ \text{rev } l_1$$

Structural Induction

```
type 'a tree =
  | Leaf
  | Node (l, v, r)
```



P(Leaf)

if P(l) and P(r) then P(Node(l, v, r))

Example

```
let rec nodes = function
  | Leaf -> 0
  | Node (l, _, r) -> 1 + nodes l + nodes r

let rec leaves = function
  | Leaf -> 1
  | Node (l, _, r) -> leaves l + leaves r
```

Show: leaves t = nodes t + 1

By str. ind. on t

Base Case: Leaf

Show: leaves Leaf = nodes Leaf + 1

1 = 0 + 1

Induction Case: Node (l, _, r)

IH: leaves l = nodes l + 1
leaves r = nodes r + 1

show: leaves (Node (l, _, r)) = nodes (Node (l, _, r)) + 1

leaves (Node (l, _, r))
= {eval leaves}
leaves l + leaves r
= {IH}
(nodes l + 1) + (nodes r + 1)
= {rearrange}
(nodes l + nodes r + 1) + 1
= {eval nodes}
nodes (Node (l, _, r)) + 1

Prove by structural ind. on t

Base case: Leaf
Show leaves Leaf = nodes Leaf + 1

Inductive case:
IH leaves ℓ = 1 + nodes ℓ
leaves r = 1 + nodes r

Show: leaves (Node(ℓ, v, r)) = 1 + nodes(Node(ℓ,v,r))

leaves (Node (ℓ, v, r))
= {eval}
leaves ℓ + leaves r
= {IH}
(1 + nodes ℓ) + (1 + nodes r)

= {algebra}

1 + (1 + nodes ℓ + nodes r)
= {eval}
1 + nodes (Node (ℓ, v, r))

□