

Lecture 5: Lists

int list

A list [ints] is either

- []

"nil"
"empty"

"cons" $x :: xs$, where $x :: \text{int}$

xs is an int list

→ and that's it!

A number is either

- 0, or

- $1 + k$, $k \in \mathbb{N}$

self-referencing

Inductive

→ and that's it!

0 nat

$0+1 = 1$ nat

$0+1+1 = 2$ nat

⋮

$[]$: int list

$(1 :: [])$: int list

~~2~~ $2 :: (1 :: [])$: int list

$2 :: 1 :: []$

$[2, 1]$

$5 :: (2 :: (1 :: []))$: int list

$5 :: 2 :: 1 :: []$

$[5, 2, 1]$

fun f(int list) =

case (int list) of

[] =>

_, _ , ... , 0T

0T



==>



=>

_, _ , ... ,

0T

0T

bound

variables

x:int

xS:int list

x

xS

f(x's)

Recursive

call

case $[]$ of $[] \Rightarrow e_1$
| $x :: xs \Rightarrow e_2$

$\mapsto e_1$

case v :: vs of $[] \Rightarrow e_1$
| $x :: xs \Rightarrow e_2$

\mapsto

e_2 with v for x
 vs for xs

(* Purpose: count the number of numbers in a list *)

(* E.g. length [5, 2, 1] = 3 *)

5 :: ([2 :: 1 :: []])

fun length (l: int list): int =
case l of

[] => 0

| x :: xs => 1 + length(xs)

length (2 3 0 | 3 3 [])

→ case 2 :: 1 :: [] of [] ⇒ 0 | x :: xs ⇒ 1 + length xs

→ 1 + length (1 :: [])

→ 1 + case 1 :: [] of [] ⇒ 0 | x :: xs ⇒ 1 + length xs

→ 1 + (1 + length [])

→ 1 + 1 + case [] of [] ⇒ 0 | ...

→ 1 + (1 + 0)

→ 2 ✓

(* Purpose: add up all the numbers
in a list *)

(* E.g. $\text{sum } [5, 1, 2] = 8$ *)

fun sum (l: int list): int =
case l of

[] =>

0

) x::xs =>

x + sum (xs)

sum [5, 1, 2]

→ case 5 :: [1, 2] of [] => 0 | x :: xs => x + sum (xs)

→ 5 + sum [1, 2]

→ 5 + case 1 :: [2] of [] => 0 | x :: xs => x + sum (xs)

→ 5 + 1 + sum [2]

→ 5 + 1 + case 2 :: [] of [] => 0 | x :: xs => x + sum (xs)

→ 5 + 1 + 2 + sum []

→ 5 + 1 + 2 + case [] of [] => 0 | ...

→ 5 + (1 + (2 + 0))

→ 8

(* Purpose: raise everyone's salaries
by an amount *)

(* E.g. raiseSalaries([11, 14, 16], 4) =
[15, 18, 20] *)

fun raiseSalaries(l: int list, amount: int): int list =

case l of

[] => []

| x :: xs => (x + amount) ::

raiseSalaries(xs, amount)

rs = raiseSalaries

rs([11, 14, 16], 4)

→ rs(11 :: 14 :: 16 :: [])

→ (11+4) :: rs([14, 16], 4)

→ 15 :: rs([14, 16], 4)

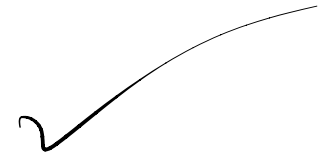
→ 15 :: (14+4) :: rs([16], 4)

→ 15 :: 18 :: rs([16], 4)

→ 15 :: 18 :: (16+4) :: rs([], 4)

→ 15 :: 18 :: 20 :: rs([], 4)

→ 15 :: 18 :: 20 :: []



```
fun length (l: int list): int =  
  case l of  
    [] => 0
```

```
| x :: xs => 1 + length(xs)
```

```
fun sum (l: int list): int =  
  case l of
```

```
    [] =>
```

```
    0
```

```
  | x :: xs =>
```

```
    x + sum(xs)
```

```
fun raiseSalaries (l: int list, amount: int): int list =
```

```
  case l of
```

```
    [] => []
```

```
  | x :: xs => (x + amount) ::
```

```
    raiseSalaries(xs, amount)
```


Induction

"Structural induction
for lists"

To prove something for all
list,

① Prove it for []

② For any x , xs ,
Assume it for a list xs ,

I H

and show it for $x :: xs$

Fusion for all l, a, b

$$rS(rS(l, a), b)$$

$$[11, 14, 16] \quad 4$$



$$[15, 18, 20]$$

$$[17, 20, 22]$$

$$\cong rS(l, a+b)$$

For all l, a, b [values]
 int list int ,

$$\underline{rs(rs(l, a), b)} \cong \underline{rs(l, a+b)}$$

Proof: by structural ind. on l

Case for $[]$:

To show: $rs(rs([], a), b) \cong rs([], a+b)$

$rs(rs([], a), b)$

$\mapsto rs([], b)$

$\mapsto []$

$rs([], a+b)$

$\mapsto []$

$[] \cong []$

✓

Case for $x :: xs$

$$\text{IH: } rS(rS(xs, a), b) \approx rS(xs, a+b)$$

$$\text{TS: } rS(rS(x :: xs, a), b) \approx rS(x :: xs, a+b)$$

$$rS(rS(x :: xs, a), b)$$

$$\mapsto rS(x+a :: rS(xs, a), b)$$

$$\approx (x+a)+b ::$$

$$rS(rS(xs, a), b)$$

$$rS(x :: xs, a+b)$$

$$x+(a+b) ::$$

$$rS(xs, a+b)$$

assoc
+

IH

