Lecture 5: Lists

int list

A list of [ints] is either

- [] "nil" "empty"

- x :: xs, where x : int "cons"

  xs is an int list

→ and that's it!

A ~~number~~ natural number is either

- 0, or
- 1 + k, k nat

→ and that's it!

self-referential

inductive

0 nat

0+1 = 1 nat

0+1+1 = 2 nat

⋮

$[\,]$: int list

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

$2 :: (1 :: [\,])$: int list

$2 :: 1 :: [\,]$

$\underbrace{[2, 1]}$

$\underbrace{5 :: (2 :: (1 :: [\,]))}$: int list

$5 :: 2 :: 1 :: [\,]$

$\underbrace{[5, 2, 1]}$

```
fun f(l: int list) =
  case (l : int list) of
    [] => _ _ _ _ : T
  | x :: xs => _ _ _ _ : T
               _ _ _ _ _ _ _ _
```

: T

x

xs

f(xs) recursive call

bound variables

x : int

xs : int list

case [] of [] => e_1

| x::xs => e_2

$\mapsto e_1$

case $\underline{\underline{v}}$ :: $\underline{\underline{vs}}$ of [] => e_1

| x::xs => $\underline{e_2}$

$\mapsto$

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 *)

$\underline{5} :: \boxed{2 :: 1 :: []}$

```
fun length (l : int list) : int =
    case l of
      [] => 0
    | x :: xs => 1 + length(xs)
```

length (2 :: 1 :: [])

$\mapsto$ case 2 :: 1 :: [] of [] => 0 | x :: xs => 1+length(xs)

$\mapsto$ 1 + length(1 :: [])

$\mapsto$ 1 + case 1 :: [] of [] => 0 | x :: xs => 1 + length(xs)

$\mapsto$ 1 + (1 + length [])

$\mapsto$ 1 + 1 + case [] of [] => 0 | . . .

$\mapsto$ 1 + (1 + 0)

$\mapsto$ 2 ✓

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

(* E.g. sum [5, 1, 2] = 8 *)

```
fun sum (l : int list) : int =
    case l of
        [] => 0
      | x::xs => x + sum(xs)
```

$\mathrm{sum}\ [5, 1, 2]$

$\mapsto \mathrm{case}\ 5 :: [1, 2]\ \mathrm{of}\ [\,] \Rightarrow 0 \mid x :: xs \Rightarrow x + \mathrm{sum}(xs)$

$\mapsto 5 + \mathrm{sum}\ [1, 2]$

$\mapsto 5 + \mathrm{case}\ 1 :: [2]\ \mathrm{of}\ [\,] \Rightarrow 0 \mid x :: xs \Rightarrow x + \mathrm{sum}(xs)$

$\mapsto 5 + 1 + \mathrm{sum}\ [2]$

$\mapsto 5 + 1 + \mathrm{case}\ 2 :: [\,]\ \mathrm{of}\ [\,] \Rightarrow 0 \mid x :: xs \Rightarrow x + \mathrm{sum}(xs)$

$\mapsto 5 + 1 + 2 + \mathrm{sum}\ [\,]$

$\mapsto 5 + 1 + 2 + \mathrm{case}\ [\,]\ \mathrm{of}\ [\,] \Rightarrow 0 \mid \ldots$

$\mapsto 5 + (1 + (2 + 0))$

$\mapsto^{\ldots} 8$

(* 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

$$rS([11, 14, 16], 4)$$

$$\mapsto rS(11 :: 14 :: 16 :: [])$$

$$\mapsto (11+4) :: rS([14, 16], 4)$$

$$\mapsto 15 :: rS([14, 16], 4)$$

$$\mapsto 15 :: (14+4) :: rS([16], 4)$$

$$\mapsto 15 :: 18 :: rS([16], 4)$$

$$\mapsto 15 :: 18 :: (16+4) :: rS([], 4)$$

$$\mapsto 15 :: 18 :: 20 :: rS([], 4)$$

$$\mapsto 15 :: 18 :: 20 :: []$$

✓

```
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)
```

$\boxed{\text{Induction}}$

"structural induction for lists"

To prove something for <u>all</u> list,

1. Prove it for []

2. For any x, xs,
   ~~Assume~~ Assume it for a list xs, and show it for $\underbrace{x :: xs}$

IH

Fusion for all l, a, b

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

$[11, 14, 16]$ 4 2

$[15, 18, 20]$

$[17, 20, 22]$

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

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

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

Proof: by structural ind. on l

Case for [ ]:

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

$$rS(rS([\,], a), b) \qquad\qquad rS([\,], a+b)$$

$$\mapsto rS([\,], b) \qquad\qquad \mapsto [\,]$$

$$\mapsto [\,]$$

$$[\,] \cong [\,] \quad \checkmark$$

Case for $x :: xs$:

IH: $rS(rS(xs, a), b) \stackrel{?}{=} rS(xs, a+b)$

TS: $rS(rS(x :: xs, a), b) \stackrel{?}{=} rS(x :: xs, a+b)$

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

$\mapsto rS(\boxed{\underbrace{x+a}_{x} :: \underbrace{rS(xs, a)}_{xs}}, b)$

$\stackrel{\sim}{=}$

$(x+a)+b ::$

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

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

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

$rS(xs, a+b)$

$=$ assoc $+$

IH