Lec 3 Recursion and Induction

* Higher order function — function that takes function as arg

Example omitted

Computing Exponent

```
(* power : int * int -> int
   REQ : k >= 0
   ENS : power(n, k) = n^k
*)
```

Not using it

(assume $0^0 = 1$)

```
① fun power (_: int, 0: int) : int = 1
    | power (n: int, k: int) : int = n * power (n, k-1)
```

* power is not total — it can't take $k<0$.

power (3,6) $\cong \underbrace{3*3*3*3*3*3}_{O(k)}*1 \cong 729$.

Can we make it faster?

```
② fun power (_ : int, 0: int) : int = 1
    | power (n, k) =
      if k % 2 = 0 then
        square ( power(n, k div 2 ))
      else
        n * power (n, k - 1)
```

power $\underbrace{(3, 7) \cong 3\cdot(3\cdot(3\cdot1)^2)^2}_{O(\log k)} \cong 2187$

Proving this works

Theorem : ① satisfies its specs i.e.
$\forall n, k \in \mathbb{Z}$ with $k \geq 0$, power $(n, k) \hookrightarrow n^k$

<u>Proof</u> by induction on k ← pick the one that decreases
↑ standard induction

↓ reduces to value

Base case when $k = 0$

WTS power $(n, 0) \hookrightarrow n^0$

Showing:

↓ steps to, could be expression

power $(n, 0) \Rightarrow 1$ [1^{st} clause of power]
$\Rightarrow n^0$ [math]

Inductive case

IH: power$(n, k) \hookrightarrow n^k$ for any n
WTS power $(n, k+1) \hookrightarrow n^{k+1}$ for any n

Showing:

power $(n, k+1) \Rightarrow n *$ power (n, k) [2^{nd} clause of power since $k+1 \neq 0$]
$\Rightarrow n * n^k$ [IH]
$\Rightarrow n^{k+1}$ [math]

□

Theorem 2: ③ satisfies its specs i.e.

$\forall n, k \in \mathbb{Z}$ with $k \geq 0$, power $(n, k) \hookrightarrow n^k$

<u>Proof</u> by ** strong ** induction on k

aka second principle (?)

Base case when $k = 0$

WTS power $(n, 0) \to n^0$

Showing:

$$\text{power}(n, 0) \Rightarrow 1 \qquad [1^{st} \text{ clause of power}]$$
$$\Rightarrow n^0 \qquad [\text{math}]$$

← Same stuff

Inductive case

Assume $k > 0$

IH: for all $0 \leq k' < k$, for any n, power $(n, k') \hookrightarrow n^{k'}$

WTS: power$(n, k) \hookrightarrow n^k$ for any n

Showing:

power $(n, k) \Rightarrow$ if $k \% 2 = 0$ then square (power(n, k div 2))
else $n *$ power $(n, k - 1)$

CASE 1 $k \% 2 \cong 0$:

$$\text{power}(n, k) \Rightarrow \text{square}(\text{power}(n, k \text{ div } 2)) \qquad [k \% 2 = 0]$$
$$\Rightarrow \text{square}(n^{k \text{ div } 2}) \qquad [\text{IH and } k > 0]$$
$$\Rightarrow n^{k \text{ div } 2} * n^{k \text{ div } 2} \qquad [\text{def of square}]$$
$$\Rightarrow n^k \qquad [\text{math and k even}]$$

being sloppy

CASE 2 $k \% 2 \neq 0$:

[omitted]

Some ML Syntax (Data Structure)

* Lists

name t list
values $[v_1, \ldots, v_n]$ where $v_1, \ldots, v_n : t$, $n \geq 0$ ← write [ ] for empty or "nil"

Expressions: all values
$e :: es$

$1::[2,3] = [1,2,3] = 1::2::3::[\,]$

Typing:

$[\,] : t$ list
$e :: es : t$ list if $e : t$ and $es : t$ list

Evaluation: left → right

Structural Induction

— dealing with non-number stuff in induction

Consider:

```
(* length : int list -> int
   REQ: true
   ENS: length of list
*)
fun length( [] : int list ) : int = 0
  | length( _ :: xs ) = 1 + length xs
```

But how do we prove this is correct?

Hmm ... but let's prove it's total.

Theorem: length is total for any value L: int list, length L $\hookrightarrow$ v for some v

Proof by structural on L.

Base case
L = [ ]

WTS length [ ] $\hookrightarrow$ v

Showing: length [ ] $\Rightarrow$ 0 [clause 1 of length]

let v = 0

Inductive case

Let L = x::xs with x : int and xs: int list

IH: length xs $\hookrightarrow$ v for some v

WTS: length (x::xs) $\hookrightarrow$ v' for some v'

Showing: length (x::xs) $\Rightarrow$ 1+ length xs [clause 2 of length]

$\Rightarrow$ 1+ v [IH]

$\Rightarrow$ 1+v [math]

let v' = 1+v