binding (name, value)

value ↓ can be functions

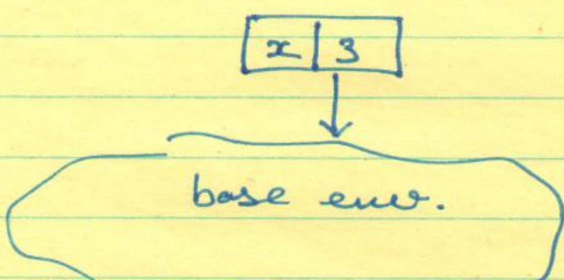
declarations create bindings

$$\underbrace{\underbrace{\boxed{\text{let } x = 3}}_{\text{declaration}} \text{ in } x+2 \text{;;}}_{\text{expression}}$$

in ← shows the scope

Do not confuse this with an assignment.

Bindings are stored in a structure called the environment.



$x+2$ ← to be evaluated

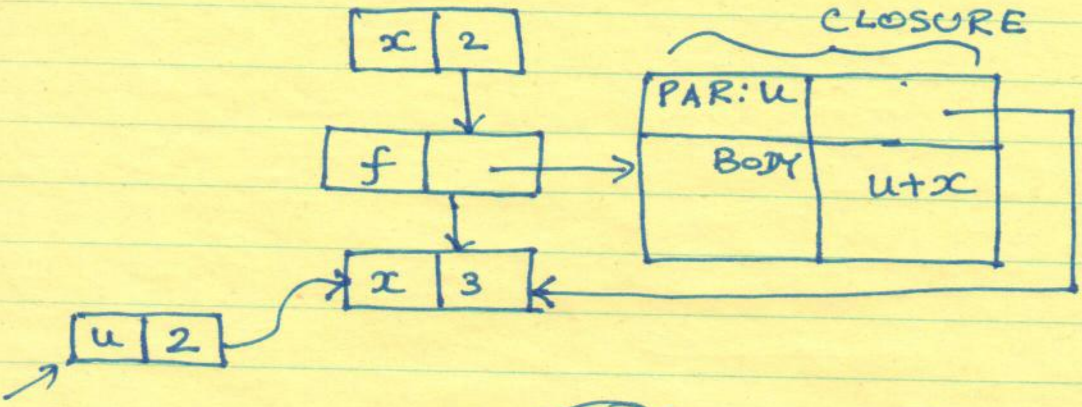
$\rightsquigarrow$

$3+2$

$\rightsquigarrow$

5

When a function is defined it remembers the environment that existed when it was created.



$f\ x \rightsquigarrow f\ 2$

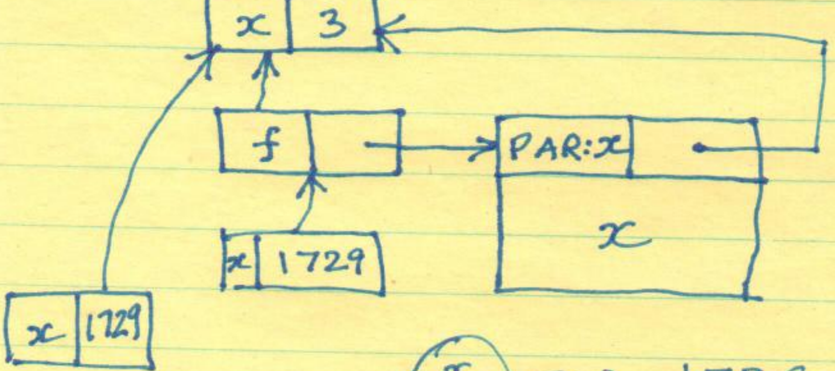
$u+x$

$\rightsquigarrow$

$2+3 \rightsquigarrow 5$

STATIC BINDING
OR LEXICAL SCOPING

```
let x=3 in
 let f = fun x -> x in
  let x=1729 in
   f x;;
```



$f\ 1729$

$x \rightsquigarrow 1729$

How do we reason about recursive programs?

PROVE CORRECTNESS OF PROGRAMS.

RECURSION $\longleftrightarrow$ INDUCTION

```
let rec exp (b, p) =
    if b=0 then 0
    else if p=0 then 1
    else b* exp(b, p-1).
```

$e \Downarrow v$ — e terminates with value v

$e \Rightarrow e'$ — e computes to e' in 1 step

$e \overset{*}{\Rightarrow} e'$ — e computes to e' in several steps

Thm. If $b>0$, $p \geq 0$ then

$$exp(b, p) \Downarrow b^p$$

Proof. By induction on p

BASE $p=0$

From the code : $1 = b^0$

Inductive hypothesis ; $\forall p \leq n \quad exp(b,p)=b^p$

want to see $exp(b, n+1)$: from the code

$$= b * exp(b,n) \underset{\text{from IH}}{=} b * b^n = b^{n+1} \checkmark$$

$e \Downarrow v$
$e \Rightarrow e'$
$e \overset{*}{\Rightarrow} e'$

~~let rec power (base,~~

```
let rec exp(b, p) =
    if b = 0 then 0
    else if p = 0 then 1
    else b * exp (b, p - 1)
```

<u>Thm</u> if $b > 0$ & $p \geq 0$ then
$exp(b,p) \Downarrow b^p$

<u>Proof</u> By induction on p

Base $p = 0$ $exp(b,p) = \underline{1}$

Ind. hyp $p = n$ $exp(b,n) = b^n$

Consider $exp(b,n+1)$

$exp(b,n+1) \overset{*}{\Rightarrow} b * exp(b,n)$
$\overset{*}{\Rightarrow} b * b^n = b^{n+1}$

```
let rec rpe (b, p) =
    if b = 0 then 0
    else if p = 0 then 1
    else if even (p) then
        let t = rpe(b, p/2) in
        t * t
    else b * rpe (b, p-1)
```

$b > 0, p \geq 0$ $rpe(b,p) \Downarrow b^p$

Base as before

IH $\forall k \le n$ tpe$(b, k) \Downarrow b^k$

Case (1) n is odd, $n+1$ is even

$$\text{tpe}(b, n+1) \overset{*}{\Rightarrow} (\text{tpe}(b, n+1/2))^2$$

$$= (b^{n+1/2})^2 = b^{n+1}$$

Case (2) n is even, $n+1$ is odd

$$\text{tpe}(b, n+1) \overset{*}{\Rightarrow} b * \text{tpe}(b, n)$$

$$\overset{*}{\Rightarrow} b * b^n = b^{n+1}$$

```
let rec rev l =
   match l with
   | [] -> []
   | x::t -> rev(t) @ [x]
```

```
let rec trev (l, acc) =
   match l with
   | [] -> acc
   | x::t -> trev(t, x::acc)
```

$\text{rev}(l) \Downarrow l'$ iff $\text{trev}(l, []) \Downarrow l'$

OR $\text{rev}(l) = \text{trev}(l, [])$

Induction on the length of l:

Base $\rightarrow$ ✓

IH $|l| \le n \Rightarrow \text{rev}(l) = \text{trev}(l, [])$

$\text{trev}(x::t, []) \overset{*}{\Rightarrow} \text{trev}(t, [x])$?

We cannot use IH!

If you cannot prove a theorem try to prove a harder theorem.

<u>Thm</u> $rev(l) @ acc = trev(l, acc)$

<u>Proof</u> Base $l = []$

$$\left.\begin{array}{l} rev(l) @ acc = acc \\ trev(l, acc) = acc \end{array}\right\} \checkmark$$

IH $\forall l \; |l| \leq n \Rightarrow \forall acc$

$$rev(l) @ acc = trev(l, acc).$$

let $|t| = n$

$$trev(x::t, acc) \overset{*}{\Rightarrow}$$

$$trev(t, x::acc)$$

$$= rev(t) @ (x::acc)$$

$$\cancel{= trev(t)}$$

$$= (rev(t) @ [x]) @ acc \quad \text{Lemma}$$

$$= rev(x::t) @ acc.$$