Syntax

Semantics → Static
① Name resolution
② Type analysis
checking
inference

→ Dynamic
① Evaluation rules

Idioms

Libraries

Tools

At the top, compilation unit is a sequence of expressions & definitions.

3+4

List.hd [2;8]

let v = 3+8

let f x = x+2

Expr ::= Literals

- Integer | Floating points
- Lists | Strings | Arrays
- Tuples | Records | Bools

| Variables

| Conditionals (if-then-else)

| Boolean connectives (&&, ||)

| Function applications

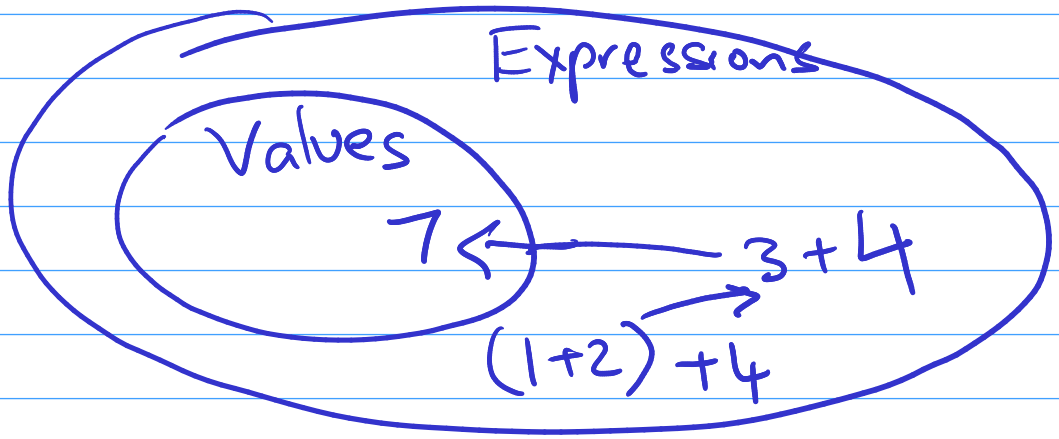
```
List.hd [2;8;9]
```

```
2+3
```

| Let expressions

```
let x = 5 in x+3
```

| Assertions | ...



$(2+3)+5 \Longrightarrow 5+5$

$\Longrightarrow 10$

f e $\Longrightarrow$ f v

$\Longrightarrow \ldots$

Name of some fn.

To evaluate a fn. application

$$f \; e_1 \; e_2 \; \cdots \; e_k$$

① Evaluate the fn. f and each of the arguments

$f \Rightarrow v_f$

$e_1 \Rightarrow v_1$

$e_2 \Rightarrow v_2$

$\vdots$

$e_k \Rightarrow v_k$

② Apply the fn. to the intermediate results

$$v_f \; v_1 \; v_2 \; \cdots \; v_k.$$

To evaluate if e_1 then e_2 else e_3

① Evaluate $e_1 \Rightarrow v_1$

② If $v_1 =$ true

then evaluate $e_2 \Rightarrow v_2$

Output v_2

③ Otherwise $v_1 =$ false !

evaluate $e_3 \Rightarrow v_3$

Output v_3.

let $x = \underline{5+3}$ in $x+8$

e_1

$\Downarrow$

let $x = 8$ in $x+8$

$=$

v_1

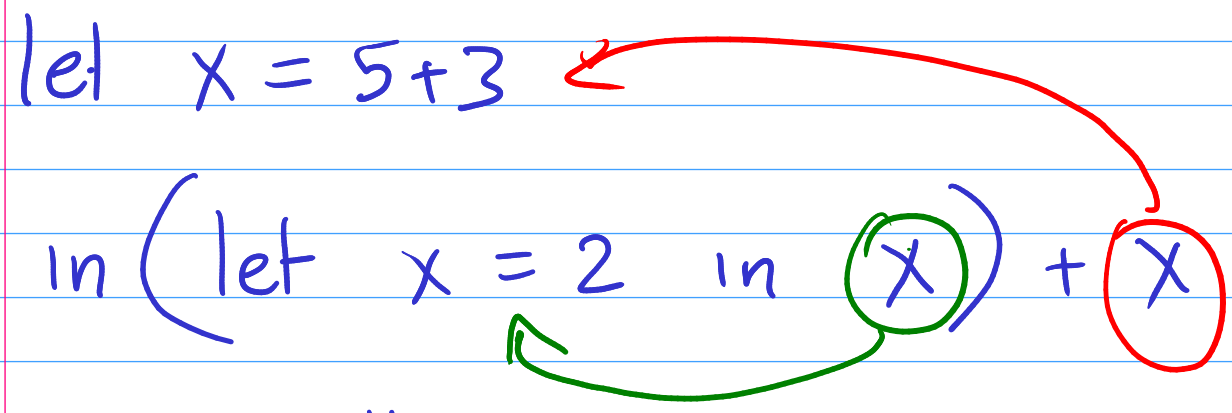
$\Downarrow$

$8+8 \Rightarrow 16$

---

let $x = e_1$ in e_2

1. First evaluate e_1. Let it produce a value v_1
2. Now replace all non-shadowed occurrences of x in e_2 with v_1 $e_2[v_1/x]$
3. Evaluate the new subexpression.



$\Downarrow$

let x=8 in (let x=2 in x)+x

$\Downarrow$

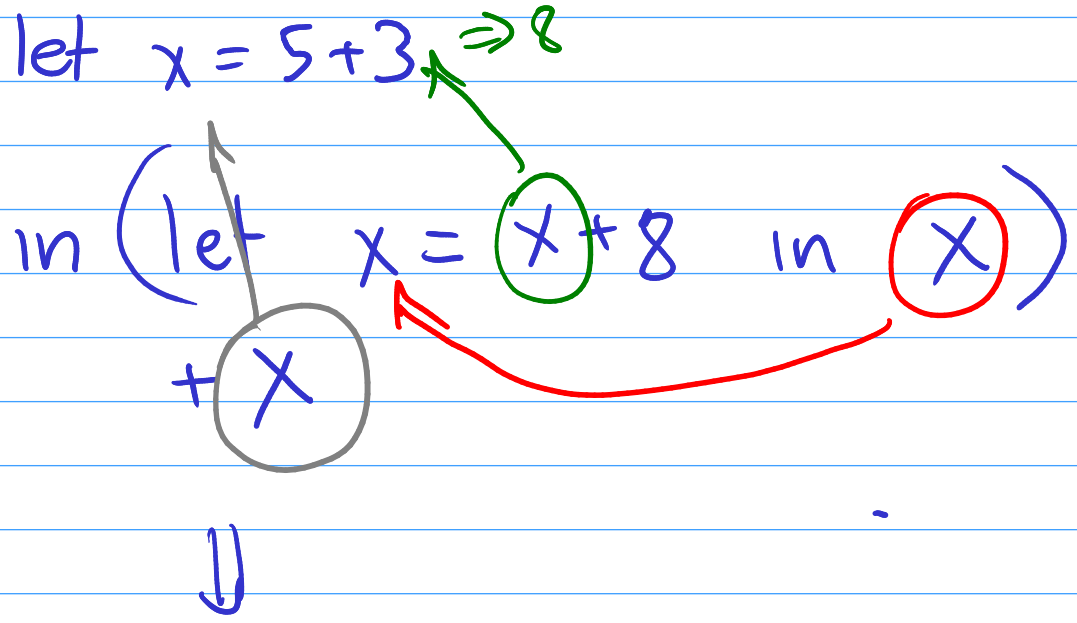
(let x=2 in x) + 8

$\Downarrow$

2 + 8

$\Downarrow$

10



⇓

let x=8 in ⇒ 24
(let x = 16 in x)
+ x

---

let x=e1 ; ⇒ let x = e1 in

let $x = e_1$ in e_2

$\Updownarrow$

$(\text{fun } x \to e_2)\ e_1$

Type checking

$f\ e$

1. Find the type of f.
2. Confirm that it is of the form $T_1 \to T_2$
3. Find the type of e.
   Confirm that $e : T_1$
4. Conclude that $(f\ e) : T_2$.