Macros

```
(:add (:num 7) (:num 8))   => '(+ 7 8)

(define (:add1 x)
  (:add x (:num 1)))

(:add1 8)                  => '(+ 8 1)

'(program (define (add1 x) (+ x 1))
          (add1 8))
```

Functions – arguments are values / no binding

```
(time (fib 888))   => "took 8 secs"
time : int => int          17586672 9
```

```
time' : (->int) -> int
(time' (λ () (fib 888)))
```

```
(match (cons 1 2)          => 3
  [(cons x y) (+ x y)])
```

```
(if (f x)
    42
    (launch-missiles!))
```

"Macros" are ~~the~~ name of a compiler-extension API that used inside the program

```
#define FOO printf("a b %
FOOs is right", "test");
```

```
#define MAX(x,y) (x>y?x:y)
```

```
(define-syntax-rule (time e)
  (time' (λ () e)))
(define (time' f) .... usual ....)
(time (fib 888))
```

⇒

```
(:dsr ...)
(:d ...)
(:call (:ref time) (:call (:ref fib) (:num 888)))
```

```
(define-syntax-rule (let ([i e] ...) b ...)
  ((lambda (i ...) b ...) e ...))
(let ([x 3] [y 5]) (+ x y))
```

⇒

```
((λ (x y) (+ x y)) 3 5)
```

Macro-by-Example 1986

Macro : Syntax AST → AST

21-3

$S = x \mid (S \ \ldots)$ (assume we know all the d-s-r)

$P = () \mid x \mid (P \cdot P) \mid (P \ \ldots)$

B: P · x S → bool

D: P x S → (x → (Num, S)) Env

T: P x Env → S

B (cons x y) (cons 1 2) => T

D => [x ↦ (0, 1) y ↦ (0, 2)]

T (+ x y) " => (+ 1 2)

B () s = (null? s)

B x s = true

B (p1 · p2) s = (and (pair? s) (B p1 (car s)) (B p2 (cdr s)))

B (P ...) s = (and (list? s) (mapand (B P) s))

D () s = ∅

D x s = [x ↦ (1, s)]

D (p1 · p2) s = (D p1 (car s)) ∪ (D p2 (cdr s))

D (P ...) s = Combine ((D P s0) ... (D P sn))

Combine (ξ0, ..., ξn) =

Assume $\xi_i = \{[x_j \mapsto (n, s_{j,i})]$

then ans = $[x_j \mapsto (n+1, (s_{j,0}, \ldots, s_{j,n}))]$

D (x ...) (1 2 3) = [x ↦ (3, (1 2 3))]

Assumes all have same variables

$T\ ()\ \sigma = ()$

$T\ x\ \sigma =$ if $x \in \sigma$, then $(i, s) = \sigma(x)$
if $i = 0$, then return s
o.w. error
o.w. x

(dsr (foo x ...)
x) => (foo 1 2 3) => error
"insufficient ellipse depth"

(dsr (let-values ([(x ...) e ...) ...)
b ...)
???)

$T\ (p_1 \cdot p_2)\ \sigma = cons\ (T\ p_1\ \sigma)\ (T\ p_2\ \sigma)$

$T\ (p \ldots)\ \sigma =$ if ! controllable (p, σ), then error
o.w. let σ' = decompose of p, σ

$p = (x \ldots)$ $p = ((+ x\ 1) \ldots)$
$\sigma = [x \mapsto (2, (a\ b))]$
$= (a\ b\ c)$ $= ((+ a\ 1)\ (+ b\ 1)\ (+ c\ 1))$

$p = ((+ x\ y) \ldots)$
$\sigma = [x \mapsto (2, (a, b, c))$
$y \mapsto (1, 6)]$ (if $y \mapsto (2, (d\ e)) \Rightarrow$ error)
$= ((+ a\ 6)\ (+ b\ 6)\ (+ c\ 6))$

controllable$(p)^{\sigma} = \exists v\ (v \in fv(p) \wedge \sigma(v) > 1)$
"is there a ...var in template?"

decompose = look at σ and $fv(p)$
one-by-one accrue a new envir (always check lengths to be eq)
$(x_i \Rightarrow (n, s))$ o.w. $(x_i \Rightarrow (n-1, s_i))$ for i's
if $n = 1$, then $(x_i \Rightarrow (n, s))$ for all,