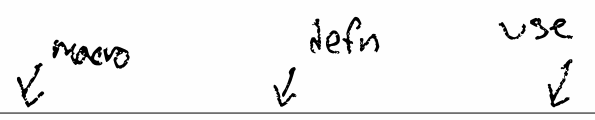


22/



macro-expand:  $id \times [(Pat \times Templ)] \times Stx \Rightarrow Stx$

for pt in defn  
 $(pat, template) := pt$

(rho)  $p = \text{pattern-match } pat \text{ use}$   
 when  $p$   
 return (transcribe  $p$  template)

match:  $pat \times stx \Rightarrow \text{env}(id \rightarrow stx)$

match '()' '()' =  $\emptyset$

match (Atomz id) a =  $\emptyset$   
 ↳ -same → num, string, bool, #:kv

match (Var v) use =  $[v \mapsto use]$

match (cons pa pd) (cons ua ud) =  
 $\oplus$  (match pa ua)  
 (match pd ud)

match \_ \_ = fail

222/

P

transcribe : env (id  $\rightarrow$  stx)  $\times$  template  $\rightarrow$  stx

tr ~~id~~ '()' = '()'

tr p (Atom a) = a

tr p (cons a d) = (cons (tr p a)  
(tr p d))

tr p (var v) = if p(v) = (not false)  
then return p(v)  
o.w. fail

add ... s P : env (id  $\rightarrow$  (nat, stx))

match (var v) use = [v  $\mapsto$  (0, var)]

tr p (var v) = (0, exp) = p(v)  
exp

(dsr (mac x ...) (zero? x))

(mac 1 2 3)  $\Rightarrow$  (zero? (1 2 3))  
(list p '...)

match (DDD p) ~~list~~ (list? u) =

ps = map (match p) u

merge ps ~~cut~~ (match p \_)

=  
( $\lambda$  (x) (match p x))

22-3/

$$\text{tr } p \text{ (DDD } +) = \text{map (tr } - +) \text{ (decompose } p)$$

$$\text{merge} : (\text{list } pat\text{-env}) \Rightarrow \text{pat-env}$$

$$\text{decompose} : \text{pat-env} \Rightarrow (\text{list } pat\text{env})$$

$$\text{merge}^{-1} = \text{decompose}$$

$$\text{merge } ps =$$

$$\text{if } ps \text{ is nil} \Rightarrow \emptyset$$

$$\text{if } ps \text{ contains any fails} \Rightarrow \text{fail}$$

$$p_0 = \text{first } ps$$

$$\text{for } (v, b) \text{ in } p_0 \text{ where } (|v|, -) = b$$

$$\text{add } v \mapsto (|v| + 1,$$

$$\text{map (snd } - [v])$$

$$ps)$$

$$[x \mapsto (0, a)], [x \mapsto (0, b)] \Rightarrow$$

$$[x \mapsto (1, (a b))]$$

22-4/ decompose  $[x \mapsto (1, (a\ b))] =$   
 $[x \mapsto (0, a)] , [x \mapsto (0, b)]$

decompose  $[x \mapsto (0, a) ,$   
 $y \mapsto (1, (b\ c))$   
 $z \mapsto (2, ((d\ e\ f)\ (g\ h\ ~~i~~)))]$

(~~dsr~~ (mac x (y z ...) ...) )  
xxxxxx) (list (x y z) ... ..)

(mac a (b d e f) (c g h ~~i~~))

$\Rightarrow [x \mapsto (0, a), b$   
 $y \mapsto (0, ~~h~~)$   
 $z \mapsto (1, (d\ e\ f))]$

$[x \mapsto (0, a)$   
 $y \mapsto (0, c)$   
 $z \mapsto (1, (g\ h))]$

22-5/ decompose  $P_e =$

find  $v_m$  in  $P$  s.t. level  $v_m$   
is largest

$$\max (|v|, \text{uses}) = P(v_m)$$

len = length uses

build-list len

( $\lambda$  i.

for  $(v, b)$  in  $P$

$$(|v|, \text{use}) = b$$

if  $|v| = 0,$

add  ~~$(v, b)$~~   $(v, b)$  do  $\#$

a.v.

add  $(v, (|v|-1, \text{use}[i]))$ )

---

(dsms  $\cup$  r

(or 1 2 3)

[ (- ) #f ]

$\Rightarrow$  1

[ (- x) x ]

(or #f #f 2)

(let ([v x])

$\Rightarrow$  2

(if v v (or y ...))))]

(let ([v s]) (or #f v))

(L [v s] (L [v #f] (if v v v))))