$$e := \quad \ldots \quad | \quad (\text{lambda} \quad \underset{\text{var}}{\overset{\text{rec} \downarrow}{}} \quad (\underset{}{\overset{\text{arg} \downarrow \; \text{arg type} \downarrow}{[\text{var} : t]}} \ldots) : \underset{}{\overset{\text{return type} \downarrow}{t_y}}$$

$$\underset{\text{body}}{} \Rightarrow e)$$

```
( (λ  fad   ([n: S64]) : S64
     (if   (== n  0)   1
           (+  n   (fad   (- n 1))))))
5)
```

16-2/

```
(let    x = 5   in
 (let    f = (λ _ ([y: s64]): s64
                 (+ y x)) in
  (let    x = 6   in
   (f X))))))      ─→  11   not   12
```

$$\frac{\Delta, \Gamma[a_0 : d_0] \ldots [f : f_{ty}] \vdash e : rng}{\Delta, \Pi \vdash (\lambda \ f \ ([a_0 : d_0] \ldots) : rng \ e) :}$$

$$(\to (d_0 \ldots) \to rng)$$

$$f_{ty}^{\nearrow}$$

16-4/ randp

half of the time rather than generating

(let x = xe   in b)

instead

((λ (x) b) xe)

generalize   to   ≤ rand ∈ [0, 4]

((λ (x₀ x₁ x₂ x₃) b) xe₀ xe₁ xe₂ xe₃)

```
(app fe args) => (let ([x_0 arg_0]
                       [x_1 arg_1]
                       ...)

when fe is "known"            body)

where "known" is
(funref f)
(λ n (x ...) body)
```

16-6/

clo-conversion: R5 → R4

```
(let    x = 5   in
(let    f = (λ   ([y: s64]): s64
                (+ y x))  in
(let    x = 6   in
 (f x))))        ─→  11    not   12
```

⇒

```
(define (f27 [clo: (vector s64)] [y: s64]): s64
   (let   r = f27 in (let   x = (vr clo 0) in
       (+ y x))))
(let x = 5 in
(let f = (vector f27 (vector x)) in
(let x = 6 in  ((vr f 0) (vr f 1) x)))))
```