4-1

LC is simple
we can add pairs, bools, nats by hand

mult = λx. λy. λf. λz.
(y (x f)) z

fac := (λ n.
(((if (is zero n))
~~zero~~ one)
((mult n) (fac n))))

LC infinite loop

P → → → → P

P → → M → → → M

m → m

(X Y) → (X Y)

(λX. M) N →
(λ. M) N)
= M[X ← N]

(λX. M) N ⇒ M[X ← N]

(λX. (L M)) N ⇒ (L M)[X ← N]

(λX. (L X)) N ⇒ (L X) [X ← N]

Ω = (w w) w = λx. x x

((λx. (x x)) [(λx. (x x))]) → ((λx. (x x)) (λx. (x x))) = (w w) = Ω

Ω → Ω

4-2

fac := (λ (n) ... fac ...)

mkfac := (λ (fac) (λ (n) ... fac ...))

(λ (mkfac) (λ (n) ... ((mkfac mkfac) n)))

F = mkfac F

Y = (λ f. ((λx. f (x x)) (λx. f (x x)))) mkfac

((λx. mkfac (x x)) (λx. mkfac (x x))) = mkfac ((λx. mkfa (x x)) (λx. mkfac (x x)))

(Y mkfac) = mkfac (Y mkfac)

Y is the fixed-point combinator

ISWIM E = X | λX. E | E E | b | O_1 E | O_2 E E | On $\vec{E}^n$

b = the constants ∈ B { true, false, 0, 1, 2, ... 77 108, ... }

O_i = primitives operations w/ i arguments

O_1 = { - } O_2 = { +, -, *, / }

b and o_i do not contain variable references

b [X ← N] = b O_2 E_0 E_1 [X ← N] = O_2 E_0 [X ← N] E_1 [X ← N]

operations do not bind variable

Δ-rule : O_i b_0 .. b_i ⟶ a δ describes the real behavior of operation

where $\delta(O_i, b_0 \ldots b_i) = a$

(λY (λX. X + Y) 7) 8 -> (λY (7 + Y)) 8 => 7 + 8 -> 15