Fb Interpreter

$x+1 \not\Rightarrow$

If True Then 0 Else 2 $\not\Rightarrow$

$\{1\}$ $\{2,1\} = \{1,2\}$ $\{1, \text{"apple"}\}$ $\{\Downarrow\}$

$\{\{1\}\}$ $\{1, 2, \{3,4\}\}$ $\{\{\}\}$.

$\{\{\{\ldots\}\}\}$.

Russel's Paradox

R = set of all sets which contain themselves

$R \in R$?

Suppose $R \in R$. Then $R \in R$.
Suppose $R \notin R$. Then $R \in R$.

P = set of all sets which do not contain themselves

$P \in P$?

Suppose $P \in P$. Then $P \notin P$.
Suppose $P \notin P$. Then $P \in P$.

$\{1,2\} \overset{def}{=}$ Function a → a = 1 Or a = 2

$\emptyset \overset{def}{=}$ Function a → False

$\{\{1,2\},\{1\},\{1,2,3\},\ldots\} \overset{def}{=}$ Function a → a 1

all sets containing 1

$P \overset{def}{=}$ Function a → Not (a a)

(Function a → Not (a a)) (Function a → Not (a a))

Not ((Function a → Not (a a)) (Function a → Not (a a)))

Not (Not ((Function a → Not (a a)) (Function a → Not (a a))))

ω-combinator = $R \overset{def}{=}$ Function a → a a

Self - Passing

Let sum' (x) = Function self → Function n → If n=0 Then 0 Else n + (self self (n-1)) (e1)

sum'

In

Let sum = sum' sum' In

sum 5

(e2)

Let sum (x) = sum' sum' (e1) In

sum 5 (e2)

(Function n → If n =0 Then 0 Else n + (sum' sum' (n-1))) 5

If 5 =0 Then 0 Else 5 + (sum' sum' (5-1))

Let sum' = Function self → Function n → If n=0 Then 0 Else n + (self self (n-1)) In (S)

Let mystery' = Function self → Function n → If n =0 Then 0 Else 1 + (sum' self (n-1)) In (M)

mystery' mystery' 5

M M 5 ⇝ (Fun n → If n=0 Then 0 Else 1 + (S M (n-1))) 5

⇝ If 5=0 Then 0 Else 1 + (S M (5-1))

⇝ 1 + (S M 4)

⇝ 1 + (Fun n → If n=0 Then 0 Else n + (M M (n-1)) 4

⇝ 1 + (4 + (M M 3))

⇝ 1 + 4 + 1 + (S M 2)

⇝ 1 + 4 + 1 + 2 + (M M 1)

⇝ 1 + 4 + 1 + 2 + 1 + 0

```
Let sum' = Function recurse -> Function n ->
                If n=0 Then 0 Else n + recurse (n-1)
In
Let recurser =
    Function n ->
        Let me = Function self -> Function k ->
                    sum' (self self) k
        In
        me me n
In
Let sum = sum' recurser In
sum 5
```

```
Let yCombinator =
    Function f ->
        Function n ->
            Let me = Function self -> Function k ->
                        f (self self) k
            In
            me me n
In
Let sum = yCombinator (Fun recurse -> Fun n ->
                If n=0 Then 0 Else n + recurse (n-1))
In
```