Encoding

An "encoding" is an representation of behavior which is not directly supported by a system.

fun a b → a+b

Function a → a+1

Function a → Function b → a+b

Encoding Recursion

Sets:

- ○ {1}
- ○ {1, 3}
- ○ {0, -2, 2, -4, 4, ...}
- ○ {4, True}
- 1 {{1}, {1,3}}
- ○ {1, √2, 🐟}

.....

- × {1, {1}, {{1}}, ...}
- × {1, {1, {1, ...}}}
- × {{{ ....}}} infinitely receeding set

Russel's Paradox

Suppose R is the set of all (sets that contain themselves). $R \stackrel{?}{\in} R$

- $R \in R \longrightarrow R \in R$
- $R \notin R \longrightarrow R \notin R$

Suppose P is the set of all sets that <u>do not</u> contain themselves.

- $P \in P \longrightarrow P \notin P$
- $P \notin P \longrightarrow P \in P$

Encoding Sets in Fb : a set is a function which, when passed a value, returns True/False for in/not in

{1} Function n → n = 1

{1, 3} Function n → n = 1 Or n = 3

{{1}, {1,2}, {1,3}, {1,2,3}, ...} Function n → n 1

P ≡ Function s → Not (s s)

P ≡ Function s → Not (s s)

P P

(Function s → Not (s s)) (Function s → Not (s s))

⇝

Not ((Function s → Not (s s)) (Function s → Not (s s)))

Not (Not ((Function s → Not (s s)) (Function s → Not (s s))))

R R

(Function s → (s s)) (Function s → (s s)) ω-combinator

⇝

(Function s → (s s)) (Function s → (s s))

```
Let summate =
    Function n →
        If n = 0 Then 0 Else summate (n - 1) + n
In
summate 5
```

free

```
Let a = a In a
```

```
Let summate n =
    ....
```

```
Let summate = Function n → ...
```

```
Let summate =
    Function self → Function n →
        If n = 0 Then 0 Else (self self (n - 1)) + n
In
summate summate 5
```

```
Let summate' = Function recurse → Function n →
    If n=0 Then 0 Else recurse (n-1) + n
In
```

Y-combinator

```
Let recurser =
    Function f →
        Let wrapper =
            Function self → Function k → f (self self) k
        In
        wrapper wrapper
In
Let summate = recurser summate' In
summate 5
```