• 1

(Function a -> a+a+a+a+a) 1

(Function f -> Function g -> Function n -> f (g n)) • (Function a->a) 5

⟶ • ((Function a->a) 5)

⟶̃ • 5

Soundness is an ex. of relating three trees: $\Gamma \vdash e : \tau$, $e \Rightarrow v$, $\Gamma \vdash v : \tau$

$e ::= v \mid e \text{ And } e \mid e \text{ Or } e \mid \text{Not } e$

$v ::= \text{True} \mid \text{False}$

Let $e \overset{Fb}{\Rightarrow} v$ denote the operational semantics of Fb. Let $e \overset{BOOL}{\Rightarrow} v$ denote the operational semantics of BOOL. BOOL is a subset of Fb; that is, if $e \overset{BOOL}{\Rightarrow} v$ (given) then $e \overset{Fb}{\Rightarrow} v$ (want to show).

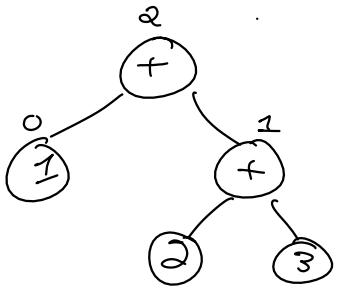
Proof

By induction on the height of the proof of $e \overset{BOOL}{\Rightarrow} v$.
Proceed by case analysis on the rule used in $e \overset{BOOL}{\Rightarrow} v$.

Let $P(k) \equiv$ If $e \overset{BOOL}{\Rightarrow} v$ then $e \overset{Fb}{\Rightarrow} v$ for proof of $e \overset{BOOL}{\Rightarrow} v$ with height k.

Case of Value Rule: If the Value Rule is used to prove $e \overset{BOOL}{\Rightarrow} v$, then $e = v$.
So by the Value Rule of Fb, $v \overset{Fb}{\Rightarrow} v$ and this case is finished.

$P(0), \ldots, P(k)$ implies $P(k+1)$



Normalizing: every expression evaluates to at least one value

$\forall e. \exists v. e \Rightarrow v$

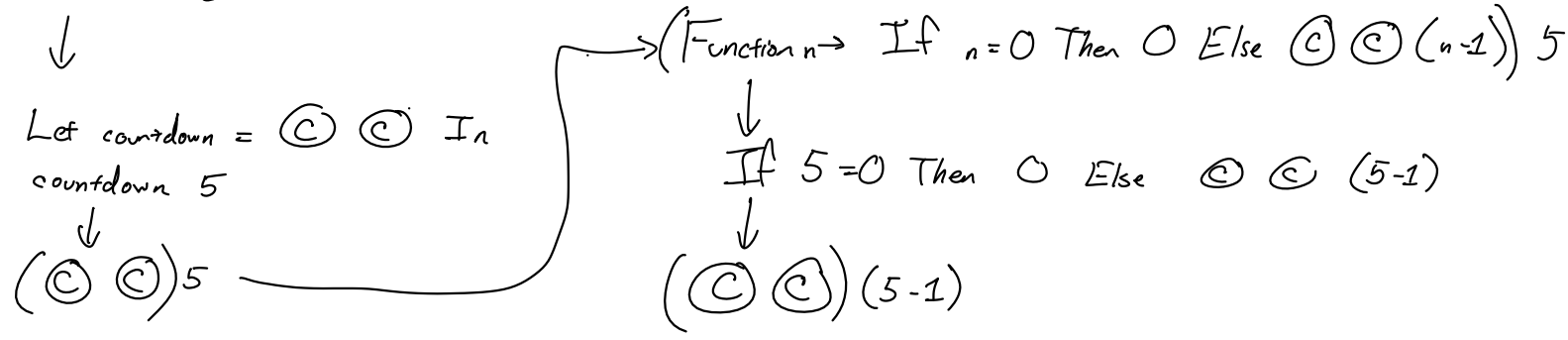
Recursion

```
Let countdown' = Function self -> Function n ->
                    If n=0 Then 0 Else self self (n-1)
In
Let countdown = countdown' countdown' In
countdown 5
```

```
Y (Function recurse -> Function n -> .... recurse (n-1))
```

```
Y2 (Fun f1 -> Fun f2 -> Fun n -> ...)
   (Fun f1 -> Fun f2 -> Fun n -> ...)
```

```
Let countdown' = Function self -> Function n ->
                    If n=0 Then 0 Else self self (n-1)        (C)
In
Let countdown = countdown' countdown' In
countdown 5
```




```
Let somefun = Function recurse -> Function x -> .... In

Let Ycomb = Function f ->
              Let recurser = Function self -> Function arg ->
                               f (self self) arg
              In
              f (recurser recurser)
In
Ycomb somefun 5
```