Encoding recursion, fixpoints

Madhavan Mukund, **S P Suresh**

Programming Language Concepts

Lecture 20, 28 March 2024 & Lecture 21, 04 April 2024

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \longrightarrow \mathbb{N}$?

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \longrightarrow \mathbb{N}$?
- Let **f** be the encoding of f

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let **f** be the encoding of f
- We want $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let **f** be the encoding of f
- We want $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let **f** be the encoding of f
- We want $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$
  - Shown in detail in this lecture

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let **f** be the encoding of f
- We want $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$
  - Shown in detail in this lecture
  - If $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}$ and $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}p\text{»}$, then $m = p$

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let **f** be the encoding of f
- We want $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$
  - Shown in detail in this lecture
  - If $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}$ and $\mathbf{f}\ \text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}p\text{»}$, then $m = p$
  - Follows from more general theorems proved later

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let $\mathbf{f}$ be the encoding of f
- We want $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$
  - Shown in detail in this lecture
  - If $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}$ and $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}p\text{»}$, then $m = p$
  - Follows from more general theorems proved later
  - If $f(n_1, \ldots, n_k)$ is undefined, then $\neg\left(\mathbf{f}\,\text{«}n\text{»}_1 \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}\right)$ for any m

Encoding recursive functions

- Church numerals encode $n \in \mathbb{N}$
- Can we encode recursive functions $f : \mathbb{N}^k \to \mathbb{N}$?
- Let $\mathbf{f}$ be the encoding of f
- We want $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n_k\text{»} \xrightarrow{*}_\beta \text{«}m\text{»}$ iff $f(n_1, \cdots, n_k) = m$
  - If $f(n_1, \cdots, n_k)$ is defined, then $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}f(n_1, \cdots, n_k)\text{»}$
  - Shown in detail in this lecture
  - If $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}$ and $\mathbf{f}\,\text{«}n_1\text{»} \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}p\text{»}$, then $m = p$
  - Follows from more general theorems proved later
  - If $f(n_1, \ldots, n_k)$ is undefined, then $\neg\left(\mathbf{f}\,\text{«}n\text{»}_1 \cdots \text{«}n\text{»}_k \xrightarrow{*}_\beta \text{«}m\text{»}\right)$ for any m
  - Brief discussion at the end of this lecture

Encoding recursive functions

- $\ulcorner n \urcorner = \lambda f x \cdot f^n x$
- **Zero**: $\mathbf{Z} = \lambda x \cdot \ulcorner 0 \urcorner$
- **Successor**: $\mathbf{succ} = \lambda p f x \cdot f(p f x)$
- **Projection**: $\mathbf{proj}_i^k = \lambda x_1 x_2 \cdots x_k \cdot x_i$
- **Composition**: If $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ is defined by $f = g \circ (h_1, \ldots, h_m)$

$$\mathbf{f} = \lambda \vec{x} \cdot \ \mathbf{g} \ \left(\mathbf{h}_1 \vec{x}\right) \cdots \left(\mathbf{h}_m \vec{x}\right)$$

Encoding recursive functions: primitive recursion

- **Primitive recursion**: Suppose f is defined via primitive recursion from $g : \mathbb{N} \to \mathbb{N}$ and $h : \mathbb{N}^3 \to \mathbb{N}$

$$f(0, n) = g(n)$$

$$f(i+1, n) = h(i, f(i, n), n)$$

- We need to eliminate recursion
  - λ-calculus functions are anonymous
  - Cannot directly use name of f inside definition of f
- We convert recursion into iteration

Encoding primitive recursion

- Given m and n, generate a sequence of pairs

$$(0, f(0,n)), (1, f(1,n)), \ldots, (m, f(m,n))$$

- Generate the sequence by the following recursion

$$\begin{array}{lcll} t(0) & = & (0, f(0,n)) & = & (0, g(n)) \\ t(i+1) & = & (i+1, f(i+1,n)) & = & (i+1, h(i, f(i,n), n)) \\ & & & = & (\ succ(\ fst(t(i))), \\ & & & & \quad h(\ fst(t(i)), snd(t(i)), n)\ \ ) \end{array}$$

  - fst and snd return the first and second components of a pair
  - $f(m,n)$ can be retrieved as $snd(t(m))$

Encoding recursive functions

- Generate the sequence by the following recursion

$$\begin{array}{lcll} t(0) & = & (0, f(0,n)) & = & (0, g(n)) \\ t(i+1) & = & (i+1, f(i+1,n)) & = & (i+1, h(i, f(i,n), n)) \\ & & & = & (\ succ(\ fst(t(i))), \\ & & & & \quad h(\ fst(t(i)), snd(t(i)), n)\ ) \end{array}$$

- We generate the $t(i)$'s by iteration
- Define $A = (0, g(n))$ and $S : (t(i)) \mapsto t(i+1)$
- So $t(m) = S^m(A)$ ...
- ...and $f(m,n) = snd(t(m)) = snd(S^m(A))$

Encoding pairs, *fst* and *snd*

- **pair** $= \lambda xyz \cdot zxy$
- **pair** $a\,b \xrightarrow{*}_\beta \lambda z \cdot zab$
- **fst** $= \lambda p.p(\lambda xy \cdot x)$
- **fst** (**pair** $a\,b$) $\xrightarrow{*}_\beta (\lambda p \cdot p(\lambda xy \cdot x))(\lambda z \cdot zab) \longrightarrow_\beta (\lambda z \cdot zab)(\lambda xy \cdot x)$
  $\longrightarrow_\beta (\lambda xy \cdot x)ab \longrightarrow_\beta (\lambda y \cdot a)b \longrightarrow_\beta a$
- **snd** $= \lambda p \cdot (p(\lambda xy \cdot y))$
- **snd** (**pair** $a\,b$) $\xrightarrow{*}_\beta (\lambda p \cdot p(\lambda xy \cdot y))(\lambda z \cdot zab) \longrightarrow_\beta (\lambda z \cdot zab)(\lambda xy \cdot y)$
  $\longrightarrow_\beta (\lambda xy \cdot y)ab \longrightarrow_\beta (\lambda y \cdot y)b \longrightarrow_\beta b$

Encoding primitive recursion

- $A = (0, g(n))$
- $S(t(i)) = (i+1, f(i+1, n)) = (succ(\ fst(t(i))),\quad h(\ fst(t(i)), snd(t(i)), n))$
- $t(m) = S^m(A)$ and $f(m,n) = snd(t(m))$
- n is "free" in the above, but in the lambda encoding we carry an extra argument
- $\mathbf{A} = \lambda x \cdot \mathbf{pair}\ \text{«0»}\ (\mathbf{g}\, x)$
- $\mathbf{S} = \lambda x\, p \cdot \mathbf{pair}\quad (\mathbf{succ}\, (\mathbf{fst}\, p))\quad (\mathbf{h}\ (\mathbf{fst}\, p)\ (\mathbf{snd}\, p)\ x)$
- $\mathbf{f} = \lambda y x \cdot \mathbf{snd}\, (y\, (\mathbf{S}\, x)\, (\mathbf{A}\, x))$

Encoding primitive recursion

- $\mathbf{f} = \lambda yx \cdot \mathbf{snd}\,(y\,(\mathbf{S}\,x)\,(\mathbf{A}\,x))$
- $\mathbf{f}\,\ll m\gg\,\ll n\gg \xrightarrow{*}_\beta \mathbf{snd}\,(\ll m\gg\,(\mathbf{S}\,\ll n\gg)\,(\mathbf{A}\,\ll n\gg)) \xrightarrow{*}_\beta \mathbf{snd}\,((\mathbf{S}\,\ll n\gg)^m\,(\mathbf{A}\,\ll n\gg))$
- Check that $\mathbf{S}\,\ll n\gg\,(\mathbf{pair}\;\ll i\gg\;\ll f(i,n)\gg) \xrightarrow{*}_\beta \mathbf{pair}\;\ll i+1\gg\;\ll f(i+1,n)\gg$

Encoding primitive recursion

- Check that $\mathbf{S}\ \langle\!\langle n \rangle\!\rangle\ (\mathbf{pair}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle) \xrightarrow{*}_{\beta} \mathbf{pair}\ \langle\!\langle i+1 \rangle\!\rangle\ \langle\!\langle f(i+1,n) \rangle\!\rangle$
- $\mathbf{S}\ \langle\!\langle n \rangle\!\rangle\ (\mathbf{pair}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle)$

$$
\begin{aligned}
&\xrightarrow{*}_{\beta} && \mathbf{pair}\ (\mathbf{succ}\ (\mathbf{fst}\ (\mathbf{pair}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle))) \\
& && (\mathbf{h}\ (\mathbf{fst}\ (\mathbf{pair}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle)) \\
& && \quad (\mathbf{snd}\ (\mathbf{pair}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle)) \\
& && \quad \langle\!\langle n \rangle\!\rangle) \\
&\xrightarrow{*}_{\beta} && \mathbf{pair}\ \ (\mathbf{succ}\ \langle\!\langle i \rangle\!\rangle)\ \ (\mathbf{h}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle f(i,n) \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle) \\
&\xrightarrow{*}_{\beta} && \mathbf{pair}\ \langle\!\langle i+1 \rangle\!\rangle\ \langle\!\langle h(i,f(i,n),n) \rangle\!\rangle \\
&= && \mathbf{pair}\ \langle\!\langle i+1 \rangle\!\rangle\ \langle\!\langle f(i+1,n) \rangle\!\rangle
\end{aligned}
$$

Encoding primitive recursion

- Check that $(\mathbf{S}\,\ulcorner n\urcorner)^i\,(\mathbf{A}\,\ulcorner n\urcorner) \xrightarrow{*}_\beta \mathbf{pair}\ \ulcorner i\urcorner\ \ulcorner f(i,n)\urcorner$
- $(\mathbf{S}\,\ulcorner n\urcorner)^0\,(\mathbf{A}\,\ulcorner n\urcorner) \xrightarrow{*}_\beta \mathbf{A}\,\ulcorner n\urcorner = \mathbf{pair}\ \ulcorner 0\urcorner\ (\mathbf{g}\ \ulcorner n\urcorner) = \mathbf{pair}\ \ulcorner 0\urcorner\ \ulcorner f(0,n)\urcorner$
- $$\begin{aligned}(\mathbf{S}\,\ulcorner n\urcorner)^{i+1}\,(\mathbf{A}\,\ulcorner n\urcorner) \quad &= \quad \mathbf{S}\,\ulcorner n\urcorner\,((\mathbf{S}\ulcorner n\urcorner)^i\,(\mathbf{A}\,\ulcorner n\urcorner)) \\ &\xrightarrow{*}_\beta \quad \mathbf{S}\,\ulcorner n\urcorner\,(\mathbf{pair}\ \ulcorner i\urcorner\ \ulcorner f(i,n)\urcorner) \quad \textbf{(ind. hyp.)} \\ &\xrightarrow{*}_\beta \quad \mathbf{pair}\ \ulcorner i+1\urcorner\ \ulcorner f(i+1,n)\urcorner \quad \textbf{(previous slide)}\end{aligned}$$

Encoding primitive recursion

- $\mathbf{f} = \lambda yx \cdot \mathbf{snd}\,(y\,(\mathbf{S}\,x)\,(\mathbf{A}\,x))$
- $\mathbf{f}\,\ulcorner m \urcorner\,\ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{snd}\,(\ulcorner m \urcorner\,(\mathbf{S}\,\ulcorner n \urcorner)\,(\mathbf{A}\,\ulcorner n \urcorner)) \xrightarrow{*}_\beta \mathbf{snd}((\mathbf{S}\,\ulcorner n \urcorner)^m\,(\mathbf{A}\,\ulcorner n \urcorner))$
- $(\mathbf{S}\,\ulcorner n \urcorner)^m\,(\mathbf{A}\,\ulcorner n \urcorner) \xrightarrow{*}_\beta \mathbf{pair}\ \ulcorner m \urcorner\ \ulcorner f(m,n) \urcorner$
- So $\mathbf{f}\,\ulcorner m \urcorner\,\ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{snd}\,(\mathbf{pair}\ \ulcorner m \urcorner\ \ulcorner f(m,n) \urcorner) \xrightarrow{*}_\beta \ulcorner f(m,n) \urcorner$
- The expression **PR** encodes the schema of primitive recursion

$$\begin{aligned}\mathbf{PR} = \lambda hgyx \cdot \mathbf{snd}\,(\,&y\\ &((\lambda xp \cdot \mathbf{pair}\,(\mathbf{succ}\,(\mathbf{fst}\,p))\,(h\,(\mathbf{fst}\,p)\,(\mathbf{snd}\,p)\,x)) \quad x)\\ &((\lambda x \cdot \mathbf{pair}\,\ulcorner 0 \urcorner\,(g\,x)) \quad x)\end{aligned}$$

Encoding μ-recursion

- Suppose $f : \mathbb{N} \longrightarrow \mathbb{N}$ is obtained from $g : \mathbb{N}^2 \longrightarrow \mathbb{N}$ by μ-recursion

$$f(n) = \begin{cases} i & \text{if } g(i,n) = 0 \text{ and } \forall j < i : g(j,n) > 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

- f can be expressed as the following (potentially unbounded) **`while`** loop:

```
i = 0;
while (g(i, n) > 0) {i = i + 1;}
return i;
```

Encoding μ-recursion

- f can be expressed as the following (potentially unbounded) **while** loop:

```
i = 0;
while (g(i, n) > 0) {i = i + 1;}
return i;
```

- Implement the **while** loop using recursion:

```
int search(i, n) {
    if (iszero(g(i, n))) return i;
    else return search(i+1, n);
}
f(n) = search(0, n);
```

Encoding booleans

- Need ways to encode booleans, if-then-else, test for zero and recursive definitions in λ-calculus
- **true** $= \lambda xy \cdot x$
- **false** $= \lambda xy \cdot y$
- **if-then-else** $= \lambda bxy \cdot bxy$
- **Syntactic sugar**: **if-then-else** $b\,f\,g$ is written as **if** b **then** f **else** g

$$
\begin{array}{lcll}
\textbf{if true then } f \textbf{ else } g & = & (\lambda bxy \cdot bxy)(\lambda xy \cdot x)fg & \longrightarrow_\beta \quad (\lambda xy \cdot (\lambda xy \cdot x)xy)fg \\
 & \longrightarrow_\beta & (\lambda y \cdot (\lambda xy \cdot x)fy)g & \longrightarrow_\beta \quad (\lambda xy \cdot x)fg \\
 & \longrightarrow_\beta & (\lambda y \cdot f)g & \longrightarrow_\beta \quad f \\
\textbf{if false then } f \textbf{ else } g & = & (\lambda bxy \cdot bxy)(\lambda xy \cdot y)fg & \xrightarrow{*}_\beta \quad (\lambda xy \cdot y)fg \\
 & \longrightarrow_\beta & (\lambda y \cdot y)g & \longrightarrow_\beta \quad g
\end{array}
$$

Encoding test for zero

- $\textbf{iszero} = \lambda x \cdot x(\lambda z \cdot \textbf{false})\textbf{true}$
- $\textbf{iszero}\ \text{«}n\text{»} \longrightarrow_\beta \text{«}n\text{»}(\lambda z \cdot \textbf{false})\textbf{true} \xrightarrow{*}_\beta (\lambda z \cdot \textbf{false})^n\textbf{true}$
- $(\lambda z \cdot \textbf{false})^0\textbf{true} = \textbf{true}$
- For $n > 0$, $(\lambda z \cdot \textbf{false})^n\textbf{true} = (\lambda z \cdot \textbf{false})((\lambda z \cdot \textbf{false})^{n-1}\textbf{true}) \longrightarrow_\beta \textbf{false}$
- Thus
  - $\textbf{iszero}\ \text{«}0\text{»} \xrightarrow{*}_\beta \textbf{true}$
  - $\textbf{iszero}\ \text{«}n\text{»} \xrightarrow{*}_\beta \textbf{false}$ for $n > 0$

Recursive definitions

- $f(n) = \mu i : g(i, n)$ is expressed as follows:

```
int search(i, n) {
    if (iszero(g(i, n))) return i;
    else return search(i+1, n);
}
f(n) = search(0, n);
```

- The λ-expression **C** encoding `search` satisfies the following property:

$$\mathbf{C}\,\langle\!\langle m \rangle\!\rangle\,\langle\!\langle n \rangle\!\rangle \xrightarrow{*}_{\beta} \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\,\langle\!\langle m \rangle\!\rangle\,\langle\!\langle n \rangle\!\rangle))\ \mathbf{then}\ \langle\!\langle m \rangle\!\rangle\ \mathbf{else}\ (\mathbf{C}\,\langle\!\langle m+1 \rangle\!\rangle\,\langle\!\langle n \rangle\!\rangle)$$

Recursive definitions

- Suppose **C** satisfies the following property:

$$\mathbf{C} \xrightarrow{*}_{\beta} (\lambda cyx \cdot \ \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\, y\, x))\ \mathbf{then}\ y\ \mathbf{else}\ (c\, (\mathbf{succ}\, y)\, x))\ \ \mathbf{C}$$

- Then it satisfies the following:

$$\mathbf{C}\ \text{«}m\text{»}\text{«}n\text{»} \xrightarrow{*}_{\beta} \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\ \text{«}m\text{»}\ \text{«}n\text{»}))\ \mathbf{then}\ \text{«}m\text{»}\ \mathbf{else}\ (\mathbf{C}\ \text{«}m+1\text{»}\ \text{«}n\text{»})$$

- So letting F be

$$\lambda cyx \cdot \ \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\, y\, x))\ \mathbf{then}\ y\ \mathbf{else}\ (c(\mathbf{succ}\, y)\, x)$$

we want $\mathbf{C} \xrightarrow{*}_{\beta} F\, \mathbf{C}$

Recursive definitions and fixed points

- Given a λ-expression F, find an expression $\mathbf{C}$ such that

$$\mathbf{C} \xrightarrow{*}_{\beta} F\,\mathbf{C}$$

- Taking $\mathbf{C} = (\lambda x \cdot F(xx))(\lambda x \cdot F(xx))$, we have

$$\mathbf{C} \xrightarrow{*}_{\beta} F\,\mathbf{C}$$

- Such a $\mathbf{C}$ is a **fixed point** of F
- **Recall**: x is a fixed point of a function f if $f(x) = x$
- Can we abstract the process of finding a fixed point?
- Enter **fixed-point combinators!**

Recursive definitions and the Y combinator

- Define $\mathbf{Y} = \lambda f \cdot (\lambda x \cdot f(xx))(\lambda x \cdot f(xx))$
- $\mathbf{Y}\, F \longrightarrow_\beta (\lambda x \cdot F(xx))(\lambda x \cdot F(xx)) \longrightarrow_\beta F(\lambda x \cdot F(xx))(\lambda x \cdot F(xx))$
- $F(\mathbf{Y}\, F) \longrightarrow_\beta F(\lambda x \cdot F(xx))(\lambda x \cdot F(xx))$
- So there is a G such that $\mathbf{Y}\, F \xrightarrow{*}_\beta G$ and $F(\mathbf{Y}\, F) \xrightarrow{*}_\beta G$
- We say that $\mathbf{Y}\, F =_\beta F\, (\mathbf{Y}\, F)$
- For any F, $\mathbf{Y}\, F$ is a $\mathbf{C}$ such that $\mathbf{C} =_\beta F\, \mathbf{C}$

Recursive definitions and the Θ combinator

- Given a λ-expression F, find an expression $\mathbf{C}$ such that

$$\mathbf{C} \xrightarrow{*}_{\beta} F\,\mathbf{C}$$

- Define $\Theta = (\lambda xy \cdot y(xxy))(\lambda xy \cdot y(xxy))$
- $\Theta\,F = (\lambda xy \cdot y(xxy))(\lambda xy \cdot y(xxy))\,F \longrightarrow_{\beta} (\,\lambda y \cdot y(\,(\lambda xy \cdot y(xxy))\,(\lambda xy \cdot y(xxy))\,y)\,)\,F \longrightarrow_{\beta} F\,(\,(\lambda xy \cdot y(xxy))\,(\lambda xy \cdot y(xxy))\,F) = F\,(\Theta\,F)$
- Thus $\Theta\,F \xrightarrow{*}_{\beta} F\,(\Theta\,F)$
- For any F, $\Theta\,F$ is a $\mathbf{C}$ such that $\mathbf{C} \xrightarrow{*}_{\beta} F\,\mathbf{C}$

Back to μ-recursion

- $f(n) = \mu i : g(i, n)$ is encoded as follows:
  - $F = \lambda cyx \cdot$ **if** (**iszero**(**g** $y\, x$)) **then** y **else** (c(**succ** y)x)
  - **W** $= \Theta F$, and so **W** $\xrightarrow{*}_{\beta} F$ **W**
  - **f** $=$ **W** «0»
- **f** «n» $=$ **W** «0» «n»

Encoding μ-recursion

- Suppose $g(i, n) = 0$
  - Then $\mathbf{g}\ \ll i\gg\ \ll n\gg \xrightarrow{*}_\beta \ll 0\gg$
  - So $\mathbf{iszero}(\mathbf{g}\ \ll i\gg\ \ll n\gg) \xrightarrow{*}_\beta \mathbf{iszero}\ \ll 0\gg \xrightarrow{*}_\beta \mathbf{true}$
  - So

$$
\begin{aligned}
\mathbf{W}\ \ll i\gg\ \ll n\gg &\xrightarrow{*}_\beta F\,\mathbf{W}\ \ll i\gg\ll n\gg \\
&\xrightarrow{*}_\beta \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\ \ll i\gg\ \ll n\gg)) \\
&\quad\ \mathbf{then}\ \ll i\gg \\
&\quad\ \mathbf{else}\ (\mathbf{W}(\mathbf{succ}\ \ll i\gg)\ \ll n\gg) \\
&\xrightarrow{*}_\beta \mathbf{if\ true\ then}\ \ll i\gg\ \mathbf{else}\ (\mathbf{W}\ (\mathbf{succ}\ \ll i\gg)\ \ll n\gg) \\
&\xrightarrow{*}_\beta \ll i\gg
\end{aligned}
$$

Encoding μ-recursion

- Suppose $g(i, n) = k > 0$
  - Then $\mathbf{g}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle \xrightarrow{*}_\beta \langle\!\langle k \rangle\!\rangle$
  - So $\mathbf{iszero}(\mathbf{g}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle) \xrightarrow{*}_\beta \mathbf{iszero}\ \langle\!\langle k \rangle\!\rangle \xrightarrow{*}_\beta \mathbf{false}$
  - So

$$
\begin{aligned}
\mathbf{W}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle &\xrightarrow{*}_\beta F\,\mathbf{W}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle \\
&\xrightarrow{*}_\beta \mathbf{if}\ (\mathbf{iszero}(\mathbf{g}\ \langle\!\langle i \rangle\!\rangle\ \langle\!\langle n \rangle\!\rangle)) \\
&\quad\ \mathbf{then}\ \langle\!\langle i \rangle\!\rangle \\
&\quad\ \mathbf{else}\ (\mathbf{W}(\mathbf{succ}\ \langle\!\langle i \rangle\!\rangle)\ \langle\!\langle n \rangle\!\rangle) \\
&\xrightarrow{*}_\beta (\mathbf{if\ false\ then}\ \langle\!\langle i \rangle\!\rangle\ \mathbf{else}\ (\mathbf{W}\ (\mathbf{succ}\ \langle\!\langle i \rangle\!\rangle)\ \langle\!\langle n \rangle\!\rangle) \\
&\xrightarrow{*}_\beta \mathbf{W}\ (\mathbf{succ}\ \langle\!\langle i \rangle\!\rangle)\ \langle\!\langle n \rangle\!\rangle \\
&\xrightarrow{*}_\beta \mathbf{W}\ \ \langle\!\langle i + 1 \rangle\!\rangle\ \ \langle\!\langle n \rangle\!\rangle
\end{aligned}
$$

Encoding μ-recursion

- If $g(i,n) = 0$ then $\mathbf{W}\ \ulcorner i \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \ulcorner i \urcorner$
- If $g(i,n) > 0$ then $\mathbf{W}\ \ulcorner i \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{W}\ \ulcorner i+1 \urcorner\ \ulcorner n \urcorner$
- Suppose now that $g(b,n) = 0$ and $g(a,n) > 0$ for all $a < b$
- $\mathbf{W}\ \ulcorner 0 \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{W}\ \ulcorner 1 \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{W}\ \ulcorner 2 \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \ldots \xrightarrow{*}_\beta \mathbf{W}\ \ulcorner b \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \ulcorner b \urcorner$
- Thus $\mathbf{f}\ \ulcorner n \urcorner \xrightarrow{*}_\beta \mathbf{W}\ \ulcorner 0 \urcorner\ \ulcorner n \urcorner \xrightarrow{*}_\beta \ulcorner b \urcorner$ where $b = \mu i : g(i, \vec{n})$
- The expression $\mathbf{Mu} = \lambda g \cdot \Theta G \ulcorner 0 \urcorner$ encodes the schema of μ-recursion where

$$G = \lambda cyx \cdot \mathbf{if}\ (\mathbf{iszero}\ (g\,y\,x))\ \mathbf{then}\ y\ \mathbf{else}\ (c(\mathbf{succ}y)x)$$

Encoding μ-recursion

- Suppose $g(a, n)$ is defined and > 0 for all a
- $\mathbf{W}\ \text{«0»}\ \text{«}n\text{»} \xrightarrow{*}_\beta \mathbf{W}\ \text{«1»}\ \text{«}n\text{»} \xrightarrow{*}_\beta \mathbf{W}\ \text{«2»}\ \text{«}n\text{»} \xrightarrow{*}_\beta \mathbf{W}\ \text{«3»}\ \text{«}n\text{»} \ldots$
- Thus no reduction sequence starting from $\mathbf{f}\ \text{«}n\text{»}$ terminates
- Suppose $g(b, n)$ is undefined for some b, and $g(a, n) \neq 0$ for all $a < b$
- Thus $\mathbf{g}\ \text{«}b\text{»}\ \text{«}n\text{»}$ has no terminating reduction sequence, and similarly for $\mathbf{f}\ \text{«}n\text{»}$