Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

We already know that

- Register Machine computable
  - = Turing computable
  - = partial recursive.

Using this, we break the theorem into two parts:

- every partial recursive function is $\lambda$-definable
- $\lambda$-definable functions are RM computable
Recall: Representing primitive recursion

If \( f \in \mathbb{N}^n \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( F \) and \( g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( G \), we want to show \( \lambda \)-definability of the unique \( h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) satisfying \( h = \Phi_{f,g}(h) \)

where \( \Phi_{f,g} \in (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \) is given by

\[
\Phi_{f,g}(h)(\vec{a}, a) \triangleq \text{if } a = 0 \text{ then } f(\vec{a}) \\
\text{else } g(\vec{a}, a - 1, h(\vec{a}, a - 1))
\]
Representing primitive recursion

If \( f \in \mathbb{N}^n \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( F \) and \( g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( G \), we want to show \( \lambda \)-definability of the unique \( h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) satisfying \( h = \Phi_{f,g}(h) \)

where \( \Phi_{f,g} \in (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \) is given by...

**Strategy:**

- show that \( \Phi_{f,g} \) is \( \lambda \)-definable;

\[
\lambda z \, \lambda x . \text{If}(\text{Eq}_0 x)(F \, \overline{x})(G \, \overline{x} \, (\text{pred} \, x)(z \, \overline{x} \, (\text{pred} \, x)))
\]
Representing primitive recursion

If \( f \in \mathbb{N}^n \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( F \) and \( g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N} \) is represented by a \( \lambda \)-term \( G \), we want to show \( \lambda \)-definability of the unique \( h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) satisfying \( h = \Phi_{f,g}(h) \)

where \( \Phi_{f,g} \in (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \) is given by...

Strategy:

- show that \( \Phi_{f,g} \) is \( \lambda \)-definable;
- show that we can solve fixed point equations \( X = MX \) up to \( \beta \)-conversion in the \( \lambda \)-calculus.
Curry’s fixed point combinator $\mathbf{Y}$

\[ \mathbf{Y} \triangleq \lambda f \cdot (\lambda x. f(xx))(\lambda x. f(xx)) \]

So for all $\lambda$-terms $M$ we have

\[ \mathbf{Y}M \equiv_{\beta} M(\mathbf{Y}M) \]
<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>$\lambda$ Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russell set: $R \triangleq { x</td>
<td>\neg(x \in x) }$</td>
</tr>
<tr>
<td>\text{not} $\triangleq \lambda b. \text{If } b \text{ False True}$</td>
<td></td>
</tr>
</tbody>
</table>
### Origins of Y

<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>( \lambda ) Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Russell set:</strong></td>
<td>( R \triangleq \lambda x. \text{not}(x x) )</td>
</tr>
<tr>
<td>( R \triangleq { x</td>
<td>\neg(x \in x) } )</td>
</tr>
<tr>
<td><strong>Russell's Paradox:</strong></td>
<td>( RR =_\beta \text{not}(RR) )</td>
</tr>
<tr>
<td>( R \in R \iff \neg(R \in R) )</td>
<td>( R \in R \iff \neg(R \in R) )</td>
</tr>
</tbody>
</table>
## Origins of Y

<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>λ calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Russell set:</strong></td>
<td></td>
</tr>
<tr>
<td>( R \triangleq { x \mid \neg(x \in x) } )</td>
<td>( R \triangleq \lambda x. \neg \epsilon (x x) )</td>
</tr>
<tr>
<td><strong>Russell's Paradox:</strong></td>
<td></td>
</tr>
<tr>
<td>( R \in R \iff \neg (R \in R) )</td>
<td>( RR =_\beta \neg \epsilon (RR) )</td>
</tr>
</tbody>
</table>

\[
Y \neg \epsilon =_\beta RR = (\lambda x. \neg \epsilon (xx))(\lambda x. \neg \epsilon (xx))
\]
## Origins of Y

<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>λ calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Russell set:</strong></td>
<td></td>
</tr>
<tr>
<td>( R \triangleq { x \mid \neg(x \in x) } )</td>
<td>( R \triangleq \lambda x. \neg x \in x )</td>
</tr>
<tr>
<td><strong>Russell's Paradox:</strong></td>
<td></td>
</tr>
<tr>
<td>( R \in R \iff \neg(R \in R) )</td>
<td>( RR =_{\beta} \neg(RR) )</td>
</tr>
</tbody>
</table>

\[
Y\neg =_{\beta} RR = (\lambda x. \neg x)(\lambda x. \neg x)
\]

\[
Yf = (\lambda x. f(xx))(\lambda x. f(xx))
\]

\[
Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))
\]
Curry’s fixed point combinator $\mathbf{Y}$

$\mathbf{Y} \triangleq \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$

satisfies $\mathbf{Y} M \rightarrow (\lambda x. M(xx))(\lambda x. M(xx))$
Curry’s fixed point combinator \( \mathbf{Y} \)

\[ \mathbf{Y} \triangleq \lambda f. (\lambda x. f(x x))(\lambda x. f(x x)) \]

satisfies \( \mathbf{Y} M \rightarrow (\lambda x. M(x x))(\lambda x. M(x x)) \)
\[ \rightarrow M((\lambda x. M(x x))(\lambda x. M(x x))) \]

hence \( \mathbf{Y} M \rightarrow M((\lambda x. M(x x))(\lambda x. M(x x))) \leftarrow M(\mathbf{Y} M) \).

So for all \( \lambda \)-terms \( M \) we have

\[ \mathbf{Y} M \equiv_\beta M(\mathbf{Y} M) \]
Representing primitive recursion

If \( f \in \mathbb{N}^n \to \mathbb{N} \) is represented by a \( \lambda \)-term \( F \) and \( g \in \mathbb{N}^{n+2} \to \mathbb{N} \) is represented by a \( \lambda \)-term \( G \),

we want to show \( \lambda \)-definability of the unique \( h \in \mathbb{N}^{n+1} \to \mathbb{N} \) satisfying \( h = \Phi_{f,g}(h) \)

where \( \Phi_{f,g} \in (\mathbb{N}^{n+1} \to \mathbb{N}) \to (\mathbb{N}^{n+1} \to \mathbb{N}) \) is given by

\[
\Phi_{f,g}(h)(\vec{a}, a) \triangleq \begin{cases} 
    f(\vec{a}) & \text{if } a = 0 \\
    g(\vec{a}, a - 1, h(\vec{a}, a - 1)) & \text{else} 
\end{cases}
\]

We now know that \( h \) can be represented by

\[
\mathcal{Y}(\lambda z \vec{x}. \text{If}(\text{Eq}_0 x)(F \vec{x})(G \vec{x} (\text{Pred} x)(z \vec{x} (\text{Pred} x))))).
\]
Representing primitive recursion

Recall that the class \textbf{PRIM} of primitive recursive functions is the smallest collection of (total) functions containing the basic functions and closed under the operations of composition and primitive recursion.

Combining the results about \(\lambda\)-definability so far, we have: every \(f \in \text{PRIM}\) is \(\lambda\)-definable.

So for \(\lambda\)-definability of all recursive functions, we just have to consider how to represent minimization. Recall...
Minimization

Given a partial function \( f \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \), define \( \mu^n f \in \mathbb{N}^n \rightarrow \mathbb{N} \) by

\[
\mu^n f(\vec{x}) \triangleq \text{least } x \text{ such that } f(\vec{x}, x) = 0 \text{ and for each } i = 0, \ldots, x - 1, f(\vec{x}, i) \text{ is defined and } > 0
\]

(undefined if there is no such \( x \))

so \( \mu^n f(\vec{x}) = \text{g}(\vec{x}, 0) \) where in general \( \text{g}(\vec{x}, x) \) satisfies

\[
\text{g}(\vec{x}, x) = \begin{cases} 
\text{if } f(\vec{x}, x) = 0 \text{ then } x \\
\text{else } \text{g}(\vec{x}, x+1)
\end{cases}
\]
Minimization

Given a partial function \( f \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \), define \( \mu^n f \in \mathbb{N}^n \rightarrow \mathbb{N} \) by

\[
\mu^n f(\vec{x}) \triangleq \text{least } x \text{ such that } f(\vec{x}, x) = 0 \text{ and for each } i = 0, \ldots, x - 1, f(\vec{x}, i) \text{ is defined and } > 0 \\
\text{(undefined if there is no such } x)\
\]

Can express \( \mu^n f \) in terms of a fixed point equation:

\[ \mu^n f(\vec{x}) \equiv g(\vec{x}, 0) \text{ where } g \text{ satisfies } g = \Psi f(g) \]

with \( \Psi f \in (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{n+1} \rightarrow \mathbb{N}) \) defined by

\[ \Psi f(g)(\vec{x}, x) \equiv \text{if } f(\vec{x}, x) = 0 \text{ then } x \text{ else } g(\vec{x}, x + 1) \]
Representing minimization

Suppose \( f \in \mathbb{N}^{n+1} \to \mathbb{N} \) (totally defined function) satisfies \( \forall \vec{a} \exists a \ (f(\vec{a}, a) = 0) \), so that \( \mu^n f \in \mathbb{N}^n \to \mathbb{N} \) is totally defined.

Thus for all \( \vec{a} \in \mathbb{N}^n \), \( \mu^n f(\vec{a}) = g(\vec{a}, 0) \) with \( g = \Psi_f(g) \) and \( \Psi_f(g)(\vec{a}, a) \) given by

\[
\text{if } (f(\vec{a}, a) = 0) \text{ then } a \text{ else } g(\vec{a}, a + 1).
\]

So if \( f \) is represented by a \( \lambda \)-term \( F \), then \( \mu^n f \) is represented by

\[
\lambda \vec{x}.Y(\lambda z \vec{x} \cdot \text{if} (\text{Eq}_0(F \vec{x} x)) x (z \vec{x} (\text{Succ} x))) \vec{x} 0
\]
Recursive implies $\lambda$-definable

**Fact:** every partial recursive $f \in \mathbb{N}^n \rightarrow \mathbb{N}$ can be expressed in a standard form as $f = g \circ (\mu^n h)$ for some $g, h \in \text{PRIM}$. (Follows from the proof that computable = partial-recursive.)

Hence every (total) recursive function is $\lambda$-definable.

More generally, every partial recursive function is $\lambda$-definable, but matching up $\uparrow$ with $\not\exists\beta - \text{nf}$ makes the representations more complicated than for total functions: see [Hindley, J.R. & Seldin, J.P. (CUP, 2008), chapter 4.]
Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

We already know that computable = partial recursive $\Rightarrow$ $\lambda$-definable. So it just remains to see that $\lambda$-definable functions are RM computable. To show this one can

- code $\lambda$-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) $\beta$-reduction.
Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

We already know that computable = partial recursive $\Rightarrow$ $\lambda$-definable. So it just remains to see that $\lambda$-definable functions are RM computable. To show this one can

- code $\lambda$-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) $\beta$-reduction.
Numerical coding of $\lambda$-terms

Fix an enumeration $x_0, x_1, x_2, \ldots$ of the set of variables. For each $\lambda$-term $M$, define $\overline{M} \in \mathbb{N}$ by

$$\overline{x_i} = \overline{[0, i]}$$

$$\overline{\lambda x_i. M} = \overline{[1, i, \overline{M}]}$$

$$\overline{MN} = \overline{[2, \overline{M}, \overline{N}]}$$

(where $\overline{[n_0, n_1, \ldots, n_k]}$ is the numerical coding of lists of numbers from p.43).
Computable = $\lambda$-definable

**Theorem.** A partial function is computable if and only if it is $\lambda$-definable.

We already know that computable $\equiv$ partial recursive $\Rightarrow$ $\lambda$-definable. So it just remains to see that $\lambda$-definable functions are RM computable. To show this one can

- code $\lambda$-terms as numbers (ensuring that operations for constructing and deconstructing terms are given by RM computable functions on codes)
- write a RM interpreter for (normal order) $\beta$-reduction.

The details are straightforward, if tedious.
Summary

- Formalization of intuitive notion of Algorithm in several equivalent ways (cf. "Church-Turing Thesis")

- Limitative results: undecidable problems, uncomputable functions

  "programs as data" + diagonalization