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

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

\[
\lambda z \overline{x} x. \text{If}(\text{Eq}_0 x)(\overline{F} \overline{x})(\overline{G} \overline{x} (\text{pred} x)(\overline{z} \overline{x} (\text{pred} x)))
\]
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 $\mathbf{Y}$

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

So for all $\lambda$-terms $M$ we have $\mathbf{Y}M \Rightarrow M(\mathbf{Y}M)$
<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>λ calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russell set: ( R \triangleq { x \mid \neg (x \in x) } )</td>
<td>( R \triangleq \lambda x. \not{\not{\not{\not{x}}}} (x \ x) )</td>
</tr>
<tr>
<td></td>
<td>( \not{\not{\not{\not{b}}} \triangleq \lambda b. \text{If } b \text{ False True} )</td>
</tr>
<tr>
<td>Naive set theory</td>
<td>( \lambda ) calculus</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><strong>Russell set</strong>:</td>
<td>( R \triangleq \lambda x. \neg \tau(x x) )</td>
</tr>
<tr>
<td>( R \triangleq { x \mid \neg (x \in x) } )</td>
<td>( R \triangleq \lambda x. \neg \tau(x x) )</td>
</tr>
<tr>
<td><strong>Russell’s Paradox</strong>:</td>
<td>( RR =_\beta \neg \tau(RR) )</td>
</tr>
</tbody>
</table>
| \( R \in R \Leftrightarrow \neg (R \in R) \) | }
# 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></td>
</tr>
<tr>
<td>$R \triangleq { x \mid \neg(x \in x) }$</td>
<td>$R \triangleq \lambda x. \neg(t(x x))$</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>$RR =_\beta \neg(t(RR))$</td>
</tr>
</tbody>
</table>

\[ Y\neg t =_\beta RR = (\lambda x. \neg t(xx))(\lambda x. \neg t(xx)) \]
Origins of $\mathcal{Y}$

<table>
<thead>
<tr>
<th>Naive set theory</th>
<th>\lambda calculus</th>
</tr>
</thead>
</table>
| \textit{Russell set:} | \begin{align*}
R & \triangleq \{ x \mid \neg(x \in x) \} \\
\text{Russell's Paradox:} & \\
R \in R & \iff \neg(R \in R) \\
\end{align*} | \begin{align*}
R & \triangleq \lambda x. \neg \neg \left( x \in x \right) \\
RR & =_{\beta} \neg \neg \left( RR \right) \\
\end{align*} |

\[
\begin{align*}
Y \neg \neg & =_{\beta} RR = (\lambda x. \neg \neg \left( xx \right)) \left( \lambda x. \neg \neg \left( xx \right) \right) \\
Yf & = (\lambda x. f(xx)) \left( \lambda x. f(xx) \right) \\
Y & = \lambda f. (\lambda x. f(xx)) \left( \lambda x. f(xx) \right)
\end{align*}
\]
Curry’s fixed point combinator $\mathsf{Y}$

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

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

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

satisfies $\mathcal{Y} M \rightarrow (\lambda x. M(xx))(\lambda x. M(xx))$

$\rightarrow M((\lambda x. M(xx))(\lambda x. M(xx)))$

hence $\mathcal{Y} M \rightarrow M((\lambda x. M(xx))(\lambda x. M(xx))) \leftrightarrow M(\mathcal{Y} M)$.

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

$\mathcal{Y} M \Rightarrow_M M(\mathcal{Y} M)$
Turing's fixed point combinator

\[ \Theta \triangleq AA \]

where

\[ A \triangleq \lambda xy. y(xxy) \]
Turing's fixed point combinator

\[ \Theta \triangleq A A \]

where \[ A \triangleq \lambda x y . y (x x y) \]

\[ \Theta M = A A M = (\lambda x y . y (x x y)) A M \]
Turing’s fixed point combinator

\[ \Theta \triangleq AA \]

where \[ A \triangleq \lambda xy. y(xx) \]

\[ \Theta M = AAM = (\lambda xy. y(xx))AM \]

\[ \rightarrow M(AAM) \]

\[ = M(\Theta M) \]
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

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

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

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

Recall that the class 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

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

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

\[
g(\vec{x}, x) = \begin{cases} 
0 & \text{if } f(\vec{x}, x) = 0 \\
\text{undefined} & \text{otherwise}
\end{cases}
\]
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

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

(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

\[
\lambda \vec{x}. \text{Y}(\lambda z \vec{x}. \text{if}(\text{Eq}_0(F \vec{x} x)) x (z \vec{x} (\text{Succ} x))) \vec{x} 0
\]
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 $\overline{\beta} - \text{nf}$ 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.
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 \( \llbracket M \rrbracket \in \mathbb{N} \) by

\[
\begin{align*}
\llbracket x_i \rrbracket & = \llbracket [0, i] \rrbracket \\
\llbracket \lambda x_i. M \rrbracket & = \llbracket [1, i, \llbracket M \rrbracket] \rrbracket \\
\llbracket MN \rrbracket & = \llbracket [2, \llbracket M \rrbracket, \llbracket N \rrbracket \rrbracket \rrbracket
\end{align*}
\]

(where \( \llbracket [n_0, n_1, \ldots, n_k] \rrbracket \) is the numerical coding of lists of numbers from p43).
Computable = $\lambda$-definable

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

We already know that computable = partial recursive ⇒ $\lambda$-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.
Recall:

\[
\frac{M_1 \alpha \rightarrow M'_1 \quad M'_1 \rightarrow_{\text{NOR}} M'_2 \quad M'_2 \alpha \rightarrow M_2}{M_1 \rightarrow_{\text{NOR}} M_2}
\]

\[
\frac{M \rightarrow_{\text{NOR}} M'_1}{\lambda x. M \rightarrow_{\text{NOR}} \lambda x. M'_1}
\]

\[
\frac{M_1 \rightarrow_{\text{NOR}} M'_1}{M_1 M_2 \rightarrow_{\text{NOR}} M'_1 M_2}
\]

\[
\frac{(\lambda x. M) M' \rightarrow_{\text{NOR}} M[M'/x]}{(\lambda x. M) M' \rightarrow_{\text{NOR}} M[M'/x]}
\]

\[
\frac{M \rightarrow_{\text{NOR}} M'_1}{U M \rightarrow_{\text{NOR}} U M'_1}
\]

Where

\[
\begin{align*}
U & ::= x \mid U N \\
N & ::= \lambda x. N \mid U
\end{align*}
\]

\(\beta\)-normal forms

\text{"neutral" forms}
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