**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  $f \in \mathbb{N}^n \to \mathbb{N}$  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

$$\Phi_{f,g}(h)(\vec{a},a) \triangleq if a = 0 then f(\vec{a})$$
  
else  $g(\vec{a},a-1,h(\vec{a},a-1))$ 

If  $f \in \mathbb{N}^n \to \mathbb{N}$  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  $\Phi_{f,g} \in (\mathbb{N}^{n+1} \to \mathbb{N}) \to (\mathbb{N}^{n+1} \to \mathbb{N})$  is given by... Strategy:

• show that 
$$\Phi_{f,g}$$
 is  $\lambda$ -definable;  
 $\lambda z \overline{\lambda} x$ . If  $(Eq_{0}x)(F \overline{\lambda})(G \overline{\lambda})(F z \overline{\lambda})(E \overline{\lambda})(F \overline{\lambda})$ 

If  $f \in \mathbb{N}^n \to \mathbb{N}$  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  $h \in \mathbb{N}^{n+1} \to \mathbb{N}$  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... Strategy:

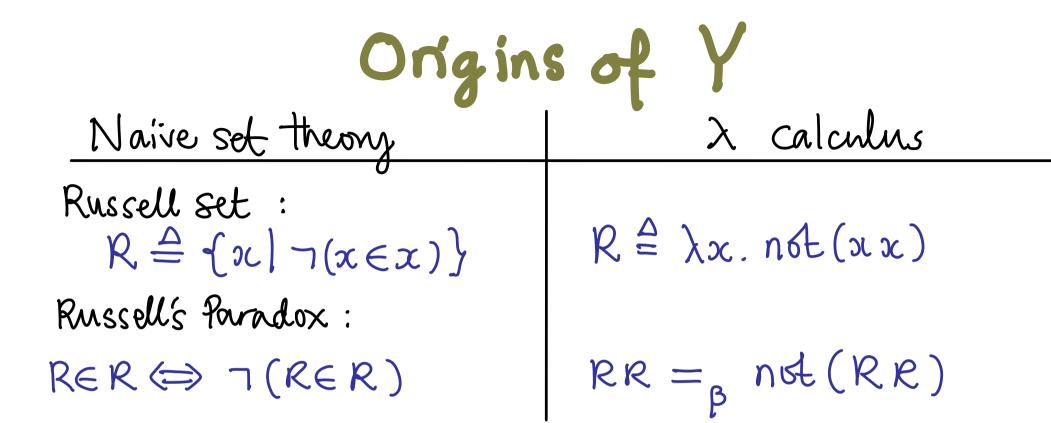
• show that  $\Phi_{f,g}$  is  $\lambda$ -definable;

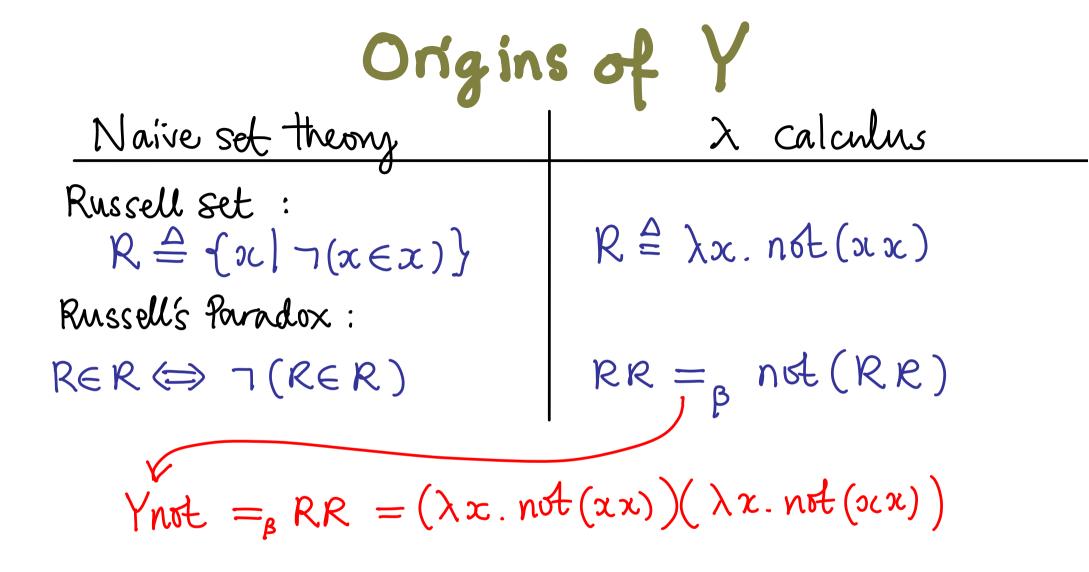
• show that we can solve fixed point equations X = M X up to  $\beta$ -conversion in the  $\lambda$ -calculus.

## Curry's fixed point combinator **Y** $\mathbf{Y} \triangleq \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$

$$\mathbf{Y}M =_{\beta} M(\mathbf{Y}M)$$

Origins of YNaive set theory
$$\lambda$$
 calculusRussell set : $\mathcal{R} \triangleq \{x \mid \forall (x \in x)\}$  $\mathcal{R} \triangleq \{x \mid \forall (x \in x)\}$  $\mathcal{R} \triangleq \lambda x. not(x x)$  $\mathcal{R} \triangleq \lambda b. If b False True$ 





Origins of YNaive set theory
$$\lambda$$
 calculusRussell set : $\chi \in \{x \mid \neg(x \in x)\}$  $R \triangleq \{x \mid \neg(x \in x)\}$  $R \triangleq \lambda x. not(x x)$ Russell's Paradox : $RR = p not(RR)$ 

Ynot = RR = 
$$(\lambda x. not(xx))(\lambda x. not(xx))$$
  
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 **Y**  $\mathbf{Y} \triangleq \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$ 

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

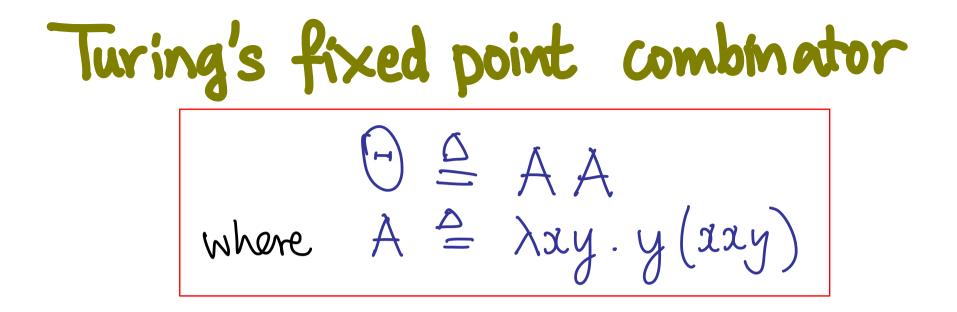
Curry's fixed point combinator **Y**  $\mathbf{Y} \triangleq \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$ 

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

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

 $\mathbf{Y}M =_{\beta} M(\mathbf{Y}M)$ 



 $\Theta M = AAM = (\lambda xy. y(\lambda xy))AM$ 

$$\Theta M = AAM = (\lambda xy. y(\lambda xy))AM$$
  
 $\rightarrow M(AAM)$   
 $= M(\Theta M)$ 

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  $\Phi_{f,g}(h)(\vec{a},a) \triangleq if \ a = 0 \ then \ f(\vec{a})$ else  $g(\vec{a}, a - 1, h(\vec{a}, a - 1))$ 

We now know that h can be represented by

 $Y(\lambda z \vec{x} x. \operatorname{lf}(\operatorname{Eq}_0 x)(F \vec{x})(G \vec{x}(\operatorname{Pred} x)(z \vec{x}(\operatorname{Pred} x)))).$ 

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  $f \in PRIM$  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  $\mu^n f(\vec{x}) \triangleq$  least x such that  $f(\vec{x}, x) = 0$  and for each  $i = 0, \dots, x - 1$ ,  $f(\vec{x}, i)$ is defined and > 0(undefined if there is no such x)

So 
$$\mu^{n}f(\vec{x}) = q(\vec{x},0)$$
 where in  
general  $g(\vec{x},x)$  satisfies  
 $g(\vec{x},x) = iff(\vec{x},x) = 0$  then  $\chi$   
else  $g(\vec{x},x+1)$ 

#### Minimization

Given a partial function  $f \in \mathbb{N}^{n+1} \to \mathbb{N}$ , define  $\mu^n f \in \mathbb{N}^n \to \mathbb{N}$  by  $\mu^n f(\vec{x}) \triangleq$  least x such that  $f(\vec{x}, x) = 0$  and for each  $i = 0, \dots, x - 1$ ,  $f(\vec{x}, i)$ is defined and > 0(undefined if there is no such x)

Can express  $\mu^n f$  in terms of a fixed point equation:  $\mu^n f(\vec{x}) \equiv g(\vec{x}, 0)$  where g 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  $\Psi_f(g)(\vec{x}, x) \equiv if f(\vec{x}, x) = 0$  then x 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 *if*  $(f(\vec{a}, a) = 0)$  *then a else*  $g(\vec{a}, a + 1)$ . So if f is represented by a  $\lambda$ -term F, then  $\mu^n f$  is represented by

 $\lambda \vec{x}.\mathbf{Y}(\lambda z \, \vec{x} \, x. \, \mathbf{lf}(\mathbf{Eq}_0(F \, \vec{x} \, x)) \, x \, (z \, \vec{x} \, (\mathbf{Succ} \, x))) \, \vec{x} \, \underline{0}$ 

## 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  $g, h \in PRIM$ . (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  $\Xi\beta$ -nf makes the representations more complicated than for total functions: see [Hindley, J.R. & Seldin, J.P. (CUP, 2008), chapter 4.]

# **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 λ-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 λ-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 emuration  $x_0, x_1, x_2, ...$  of the set of variables.  
For each  $\lambda$ -term M, define  $\lceil m \rceil \in \mathbb{N}$  by  
 $\lceil x_i^{\ 1} = \lceil [0, \hat{z}]^7$   
 $\lceil \lambda x_i \cdot M^7 = \lceil [1, \hat{z}, \lceil M^7]^7$   
 $\lceil M N^7 = \lceil [2, \lceil M^7, \lceil N^7]^7$   
(where  $\lceil n_0, n_1, ..., n_k \rceil^7$  is the numerical valing of lists  
of numbers from P43).

**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 λ-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.



 Formalization of intuitive notion of ALGORITHM in several equivalent ways
 Cf. "Church-Turing Thesis" 5 • Limitative results: jundecidable problems l'uncomputable functions "programs as data + diagonalization