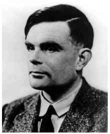
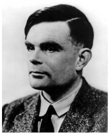
Historische Vorbemerkungen

Bilder: Wikipedia



Allgemeine Theorien von Berechnung (30er Jahre)

Bilder: Wikipedia



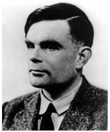
Turing (1912-54)

Gödel (1906-1978)

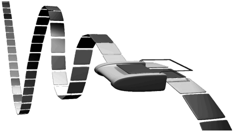
Church (1903-95)

Allgemeine Theorien von Berechnung (30er Jahre)

Bilder: Wikipedia



Turing (1912-54)



Turing Maschine

- x := x + 1
- Prozeduren
- Seiteneffekte

Gödel (1906-1978)

$$\mu f(x_1, \ldots, x_k) = \begin{cases} \min M(f, x_1, \ldots, x_k) & \text{falls } M(f, x_1, \ldots, x_k) \neq \emptyset \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

μ-rekursive Funktionen

- while ... do ...

Church (1903-95)

$$\begin{array}{ll} & (\lambda x.x\,x)\,(\lambda z.z) \\ \longrightarrow_\beta & (\lambda z.z)\,(\lambda z.z) \\ \longrightarrow_\beta & (\lambda z.z) \end{array}$$

λ-Kalkül

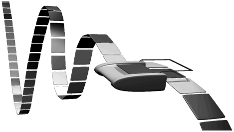
- Funktionen als
  - Objekte
  - Argumente & Resultate
- keine Seiteneffekte

Allgemeine Theorien von Berechnung (30er Jahre)

Bilder: Wikipedia

(Algol, Fortran, Pascal, C) — Imperative Programmierung

Turing (1912-54)



Turing Maschine

- x := x + 1
- Prozeduren
- Seiteneffekte

Gödel (1906-1978)

$$\mu f(x_1, \ldots, x_k) = \begin{cases} \min M(f, x_1, \ldots, x_k) & \text{falls } M(f, x_1, \ldots, x_k) \neq \emptyset \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

μ-rekursive Funktionen

- while . . . do . . .

Church (1903-95)

$$\begin{aligned} & (\lambda x.x\,x)\,(\lambda z.z) \\ \longrightarrow_\beta\ & (\lambda z.z)\,(\lambda z.z) \\ \longrightarrow_\beta\ & (\lambda z.z) \end{aligned}$$

λ-Kalkül

- Funktionen als
  - Objekte
  - Argumente & Resultate
- keine Seiteneffekte

(LISP, ML, OCAML, HASKELL) — Funktionale Programmierung

λ-Calculus: Motivation

λ-Calculus: Motivation

Consider the following arithmetical computations

$$
\begin{aligned}
(-1)^2 - 1 &= 0 \\
(1)^2 - 1 &= 0 \\
(2)^2 - 1 &= 3 \\
&\dots
\end{aligned}
$$

λ-Calculus: Motivation

Consider the following arithmetical computations

$$(-1)^2 - 1 = 0$$
$$(1)^2 - 1 = 0$$
$$(2)^2 - 1 = 3$$
$$\ldots$$

A more general arithmetic expression for the LHS:

$$\mathsf{x}^2 - 1$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \wedge m^2 - 1 = 0 \wedge n \neq m$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \wedge m^2 - 1 = 0 \wedge n \neq m$$

Now we may want to talk about the existence of a function f with two 0's:

$$(1) \quad \exists f. \exists n, m. f(n) = 0 \wedge f(m) = 0 \wedge n \neq m$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \wedge m^2 - 1 = 0 \wedge n \neq m$$

Now we may want to talk about the existence of a function f with two 0's:

$$(1) \quad \exists f. \exists n, m. f(n) = 0 \wedge f(m) = 0 \wedge n \neq m$$

This expression is not a first-order statement; however we want to be able to express such statements. We also want to prove such statements and in a constructive proof we would like to provide witnesses for f and n, m. In first-order logic we can describe f by the following equation

$$f(x) = x^2 - 1$$

λ-Calculus: λ-terms

In λ-calculus the specified function f can be described (without giving it a name) by the witnessing λ-term

$$[\lambda x.x^2 - 1]$$

and the witnesses for n and m are -1 and 1.

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, ..., x, y, z, x1, x2, ...$. The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, ..., x, y, z, x1, x2, ...$ The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>
2. <expr> ::= [λ <identifier> . <expr>] abstraction

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, ..., x, y, z, x1, x2, ...$. The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>
2. <expr> ::= [λ <identifier> . <expr>] — abstraction
3. <expr> ::= [<expr> <expr>] — application

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[F\,A\,B]$ means $[[F\,A]\,B]$. (Application associates to the left.)

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[\mathrm{F\,A\,B}]$ means $[[\mathrm{F\,A}]\,\mathrm{B}]$. (Application associates to the left.)
- $[\lambda \mathrm{x}. \lambda \mathrm{y}.\, \mathrm{B}]$ means $[\lambda \mathrm{x}.[\lambda \mathrm{y}.\, \mathrm{B}]]$.

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[F\ A\ B]$ means $[[F\ A]\ B]$. (Application associates to the left.)
- $[\lambda x.\lambda y.\, B]$ means $[\lambda x.[\lambda y.\, B]]$.
- A dot (except possibly after λ <identifier>) stands for a left bracket whose mate is as far to the right as possible without changing the existing bracketing.

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\exists \mathsf{f}.\exists \mathsf{n}, \mathsf{m}.\mathsf{f}(\mathsf{n}) = 0 \land \mathsf{f}(\mathsf{m}) = 0 \land \mathsf{n} \neq \mathsf{m}$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\begin{aligned} & \exists \mathsf{f}.\exists \mathsf{n}, \mathsf{m}.\mathsf{f}(\mathsf{n}) = 0 \wedge \mathsf{f}(\mathsf{m}) = 0 \wedge \mathsf{n} \neq \mathsf{m} \\ \xrightarrow{\mathsf{f}} \; & \exists \mathsf{n}, \mathsf{m}.[[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{n}] = 0 \wedge [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{m}] = 0 \wedge \mathsf{n} \neq \mathsf{m} \end{aligned}$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\begin{aligned}
& \exists \mathsf{f}.\exists \mathsf{n},\mathsf{m}.\mathsf{f}(\mathsf{n}) = 0 \wedge \mathsf{f}(\mathsf{m}) = 0 \wedge \mathsf{n} \neq \mathsf{m} \\
\stackrel{\mathsf{f}}{\longrightarrow}\ & \exists \mathsf{n},\mathsf{m}.[[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{n}] = 0 \wedge [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{m}] = 0 \wedge \mathsf{n} \neq \mathsf{m} \\
\stackrel{\mathsf{n},\mathsf{m}}{\longrightarrow}\ & [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\ -1] = 0 \wedge [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,1] = 0 \wedge -1 \neq 1
\end{aligned}$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\exists \mathsf{f}.\exists \mathsf{n},\mathsf{m}.\mathsf{f}(\mathsf{n}) = 0 \wedge \mathsf{f}(\mathsf{m}) = 0 \wedge \mathsf{n} \neq \mathsf{m}$$

$$\xrightarrow{\mathsf{f}} \exists \mathsf{n},\mathsf{m}.[[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{n}] = 0 \wedge [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,\mathsf{m}] = 0 \wedge \mathsf{n} \neq \mathsf{m}$$

$$\xrightarrow{\mathsf{n},\mathsf{m}} [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\ -1] = 0 \wedge [[\lambda \mathsf{x}.\mathsf{x}^2 - 1]\,1] = 0 \wedge -1 \neq 1$$

Finally we can ‘evaluate’ function applications by so called β-reduction

$$((-1)^2 - 1) = 0 \wedge (1^2 - 1) = 0 \wedge -1 \neq 1$$

λ-Calculus: β-reduction

The β-reduction rule expresses the idea of function application as motivated on the previous slide. Formally it states that

$$[[\lambda \mathsf{x}.\, \mathsf{A}]\, \mathsf{B}] \longrightarrow_\beta \mathsf{A}[\mathsf{x}/\mathsf{B}]$$

if all free occurrences in B remain free in $\mathsf{A}[\mathsf{x}/\mathsf{B}]$. Here, $\mathsf{A}[\mathsf{x}/\mathsf{B}]$ means the expression E with every free occurrence of x in A replaced with B.

λ-Calculus: Currying

A function of two variables is expressed in lambda calculus as a function of one argument which returns a function of one argument. For instance, the function

$$f(x, y) = x^2 - y$$

is encoded as

$$[\lambda x.\lambda y.x^2 - y]$$

λ-Calculus: α-conversion

The names of the bound variables are unimportant:

$$\lambda x. x^2 - 1 \text{ and } \lambda y. y^2 - 1$$

denote the same function.

λ-Calculus: α-conversion

The names of the bound variables are unimportant:

$$\lambda x.x^2 - 1 \text{ and } \lambda y.y^2 - 1$$

denote the same function.
Formally, the α-conversion rule states that if x and y are variables and A is a λ-expression then

$$[\lambda x.A] \longleftrightarrow_\alpha [\lambda y.A[x/y]]$$

if y does not appear freely in A and y is not bound by a λ in A whenever it replaces a x.

λ-Calculus: η-reduction

η-reduction expresses the idea of (functional) extensionality, which in this context is that two functions are the same iff they give the same result for all arguments:

$$[\lambda \mathsf{x}.\mathsf{F}\mathsf{x}] \longrightarrow_\eta \mathsf{F}$$

whenever x does not appear free in F.

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathsf{M}}$ for $\mathsf{M} \subset \{\alpha, \beta, \eta\}$)

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathsf{M}}$ for M $\subset \{\alpha, \beta, \eta\}$)

- We call two λ-terms E and T $\alpha\beta\eta$-equivalent (or short equivalent) if

$$\mathsf{E} \longleftrightarrow^*_{\alpha\beta\eta} \mathsf{T}$$

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathsf{M}}$ for M $\subset \{\alpha, \beta, \eta\}$)

- We call two λ-terms E and T $\alpha\beta\eta$-equivalent (or short equivalent) if

$$\mathsf{E} \ \longleftrightarrow^*_{\alpha\beta\eta} \ \mathsf{T}$$

(Similarly we may define M-equivalence for M $\subset \{\alpha, \beta, \eta\}$)

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda \mathsf{x} \,.\, \mathsf{A}]\, \mathsf{B}]$$

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda x\ .\ A]\ B]$$

- A λ-expression is called a η-normal form if it does not allow any η-reduction, i.e., has no subexpression of the form (where x does not occur free in E)

$$[\lambda x.E\ x]$$

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda x . A] B]$$

- A λ-expression is called a η-normal form if it does not allow any η-reduction, i.e., has no subexpression of the form (where x does not occur free in E)

$$[\lambda x.E\ x]$$

- A λ-expression is called a $\beta\eta$-normal form if it satisfies both conditions.

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[dr] & & C \arrow[dl] \\ & D & \end{tikzcd}$$

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B \arrow[dr] & & C \arrow[dl] \\
 & D &
\end{tikzcd}$$

- From Church-Rosser it follows that every term has at most one $*$-normal form (up to α-conversion).

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

λ-Calculus: Iteration

Consider twofold iteration of function $\mathsf{f} := [\lambda \mathsf{x}.\mathsf{x}^2 - 1]$

$$\mathsf{f}(\mathsf{f}(\mathsf{x})) = (\mathsf{x}^2 - 1)^2 - 1 = \mathsf{x}^4 - 2\mathsf{x}^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]]$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned} & [[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\ \longrightarrow_\beta\ & [\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y]] \end{aligned}$$

λ-Calculus: Iteration

Consider twofold iteration of function $\mathsf{f} := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned}
&[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\
\longrightarrow_\beta\ &[\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y] \\
\longrightarrow_\beta\ &\lambda y.[\lambda x.x^2 - 1]\ [y^2 - 1]
\end{aligned}$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned}
&[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\
\longrightarrow_\beta\ &[\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y]] \\
\longrightarrow_\beta\ &\lambda y.[\lambda x.x^2 - 1]\ [y^2 - 1] \\
\longrightarrow_\beta\ &[\lambda y.[y^2 - 1]^2 - 1] = \lambda y.y^4 - 2y^2
\end{aligned}$$

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{x}], \qquad \overline{1} = [\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{fx}], \qquad \overline{2} = [\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}(\mathsf{fx})], \qquad \ldots$$

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda f.\lambda x.x], \qquad \overline{1} = [\lambda f.\lambda x.fx], \qquad \overline{2} = [\lambda f.\lambda x.f(fx)], \qquad \ldots$$

Generally a natural number n is encoded as the Church numeral

$$\overline{n} = [\lambda f.\lambda y.f^n\, y]$$

where f^n is an abbreviation for $\underbrace{[f\ [f\ [f \ldots [f}_{n-\text{times}}\ y]]]$.

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda f.\lambda x.x], \qquad \overline{1} = [\lambda f.\lambda x.fx], \qquad \overline{2} = [\lambda f.\lambda x.f(fx)], \qquad \ldots$$

Generally a natural number n is encoded as the Church numeral

$$\overline{n} = [\lambda f.\lambda y.f^n\, y]$$

where f^n is an abbreviation for $\underbrace{[f\ [f\ [f \ldots [f}_{n-\text{times}}\ y]]]]$.

Intuitively, the number n in lambda calculus is a function that takes a function f as argument and returns the n-th iterate of f.

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

Addition is the defined as follows:

$$\overline{\mathsf{PLUS}} = [\lambda \mathsf{m}.\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{mf}[\mathsf{nfx}]]$$

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

Addition is the defined as follows:

$$\overline{\mathsf{PLUS}} = [\lambda \mathsf{m}.\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{mf}[\mathsf{nfx}]]$$

Multiplication can then be defined as

$$\overline{\mathsf{MULT}} = \lambda \mathsf{m}.\lambda \mathsf{n}.\mathsf{m}[\overline{\mathsf{PLUS}}\ \mathsf{n}]\overline{0},$$

the idea being that multiplying m and n is the same as adding n to 0 m times.

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{h}.\mathsf{h}\ [\mathsf{g}\ \mathsf{f}]]\ [\lambda \mathsf{u}.\mathsf{x}]\ [\lambda \mathsf{u}.\mathsf{u}]$$

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{h}.\mathsf{h}\ [\mathsf{g}\ \mathsf{f}]]\ [\lambda \mathsf{u}.\mathsf{x}]\ [\lambda \mathsf{u}.\mathsf{u}]$$

or alternatively

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{k}.[\mathsf{g}\ \overline{1}]\ [\lambda \mathsf{u}.\overline{\mathsf{PLUS}}\ [\mathsf{g}\ \mathsf{k}]\ \overline{1}]\ \mathsf{k}]\ [\lambda \mathsf{l}.\ \overline{0}]\ \overline{0}$$

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{h}.\mathsf{h}\ [\mathsf{g}\ \mathsf{f}]]\ [\lambda \mathsf{u}.\mathsf{x}]\ [\lambda \mathsf{u}.\mathsf{u}]$$

or alternatively

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{k}.[\mathsf{g}\ \overline{1}]\ [\lambda \mathsf{u}.\overline{\mathsf{PLUS}}\ [\mathsf{g}\ \mathsf{k}]\ \overline{1}]\ \mathsf{k}]\ [\lambda \mathsf{l}.\ \overline{0}]\ \overline{0}$$

Note the trick $[\mathsf{g}\overline{1}][\lambda \mathsf{u}.\overline{\mathsf{PLUS}}[\mathsf{g}\ \mathsf{k}]\ \overline{1}]\mathsf{k}$ which evaluates to k if $[\mathsf{g}\ \overline{1}]$ is $\overline{0}$ and to $[\mathsf{g}\ \mathsf{k}] + \overline{1}$ otherwise.

λ-Calculus: Sets

$$\{x | x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

λ-Calculus: Sets

$$\{x | x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A. \exists m, n. m \in A \land n \in A \land m \neq n$$

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A.\exists m, n.m \in A \wedge n \in A \wedge m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

λ-Calculus: Sets

$$\{x | x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A. \exists m, n. m \in A \wedge n \in A \wedge m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

In this expression we talk about 'membership'

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A.\exists m, n.m \in A \land n \in A \land m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

In this expression we talk about 'membership'

Alternatively, we can express the characteristic function of A by the λ-term

$$[\lambda x.[x^2 - 1 = 0]]$$

λ-Calculus: Sets

$$[\lambda x. x^2 - 1 = 0]$$

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$[[\lambda x.x^2 - 1 = 0]\, a]$ evaluates to $a^2 - 1 = 0$

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x.x^2 - 1 = 0]\, a] \text{ evaluates to } a^2 - 1 = 0$$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1.

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x.x^2 - 1 = 0]\, a] \text{ evaluates to } a^2 - 1 = 0$$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1. Otherwise, $a^2 - 1 = 0$ is $\bot$ ($\bot$ denotes Falsehood)

λ-Calculus: Sets

$$[\lambda x. x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x. x^2 - 1 = 0]\ a] \text{ evaluates to } a^2 - 1 = 0$$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1.
Otherwise, $a^2 - 1 = 0$ is $\bot$ ($\bot$ denotes Falsehood)
The characteristic function $[\lambda x. x^2 - 1 = 0]$ provides a witness for

$$\exists P. \exists m, n. [P\ m] \wedge [P\ n] \wedge m \neq n$$

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{\mathsf{n}} = \lambda \mathsf{f}.\lambda \mathsf{y}.[\mathsf{f}^{\mathsf{n}}\, \mathsf{y}]$$

We can also define the *set* $\overline{\mathrm{N}}$ of all Church numerals

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{\mathsf{n}} = \lambda \mathsf{f}.\lambda \mathsf{y}.[\mathsf{f}^{\mathsf{n}}\ \mathsf{y}]$$

We can also define the *set* $\overline{\mathsf{N}}$ of all Church numerals

$\overline{\mathsf{N}}$ must satisfy three properties:

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals

$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^{n}\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals

$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]]$ "$\overline{N}$ is closed under successor"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals
$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]]$ "$\overline{N}$ is closed under successor"
3. $\forall P.[P\,\overline{0}] \wedge [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
   "$\overline{N}$ is the least such set"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals

$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]]$ "$\overline{N}$ is closed under successor"
3. $\forall P.[P\,\overline{0}] \wedge [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
   "$\overline{N}$ is the least such set"

Define $\overline{N}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \;\wedge\; [\forall x.\,[P\,x] \;\supset\; [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{\mathsf{N}}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{N}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$
- $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]$ since if $P\,x$ and P is closed under successor, then $P\,[\overline{SUCC}p]]$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{N}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$
- $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]$ since if $P\,x$ and P is closed under successor, then $P\,[\overline{SUCC}p]]$
- $\forall P.[P\,\overline{0}] \wedge [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
  $\overline{N}$ is the least such set as the intersection of all such sets P

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

We have used quantification over sets (characteristic functions – the variable P) to define $\overline{\mathsf{N}}$.

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

Actually it is so powerful that it is inconsistent!

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

Actually it is so powerful that it is inconsistent!

Russell's paradox:

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.
Actually it is so powerful that it is inconsistent!
Russell's paradox:
Consider the term R:

$$[\lambda x. \neg [x\, x]]$$

As a characteristic function, R represents the set of all sets which do not contain themselves:

$$\{x | x \notin x\}$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x. \neg [x\, x]]$$

Now we evaluate the expression $E := [R\, R]$

$$[[\lambda x. \neg . x\ x]\ R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\ x]\ R] \qquad \text{evaluates to} \qquad \neg[R\,R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\ x]\ R] \qquad \text{evaluates to} \qquad \neg[R\,R]$$

And we evaluate $\neg[R\,R]$

$$\neg[[\lambda x.\neg.x\,x]\,R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to $\neg[R\,R]$

And we evaluate $\neg[R\,R]$

$\neg[[\lambda x.\neg.x\,x]\ R]$ evaluates to

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to $\neg[R\,R]$

And we evaluate $\neg[R\,R]$

$\neg[[\lambda x.\neg.x\,x]\ R]$ evaluates to $\neg\neg[R\,R]$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda \mathsf{x}.\neg[\mathsf{x}\,\mathsf{x}]]$$

Now we evaluate the expression $\mathsf{E} := [\mathsf{R}\,\mathsf{R}]$

$$[[\lambda \mathsf{x}.\neg.\mathsf{x}\ \mathsf{x}]\ \mathsf{R}] \qquad \text{evaluates to} \qquad \neg[\mathsf{R}\,\mathsf{R}]$$

And we evaluate $\neg[\mathsf{R}\,\mathsf{R}]$

$$\neg[[\lambda \mathsf{x}.\neg.\mathsf{x}\,\mathsf{x}]\,\mathsf{R}] \qquad \text{evaluates to} \qquad \neg\neg[\mathsf{R}\,\mathsf{R}]$$

which is equivalent to $[\mathsf{R}\,\mathsf{R}]$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\ x]\ R] \qquad \text{evaluates to} \qquad \neg[R\,R]$$

And we evaluate $\neg[R\,R]$

$$\neg[[\lambda x.\neg.x\,x]\ R] \qquad \text{evaluates to} \qquad \neg\neg[R\,R]$$

which is equivalent to $[R\,R]$
Thus if E holds we can infer $\neg E$ and vice versa. This is Russell's paradox.

λ-Calculus: Nontermination

Note that the term $[\lambda x.\neg.x\,x]$ (just as the standard example $[\lambda x.x\,x]$) does not terminate with respect to β-reduction:

$$[R\,R] \longrightarrow_\beta \neg[R\,R] \longrightarrow_\beta \neg\neg[R\,R] \longrightarrow_\beta \ldots$$