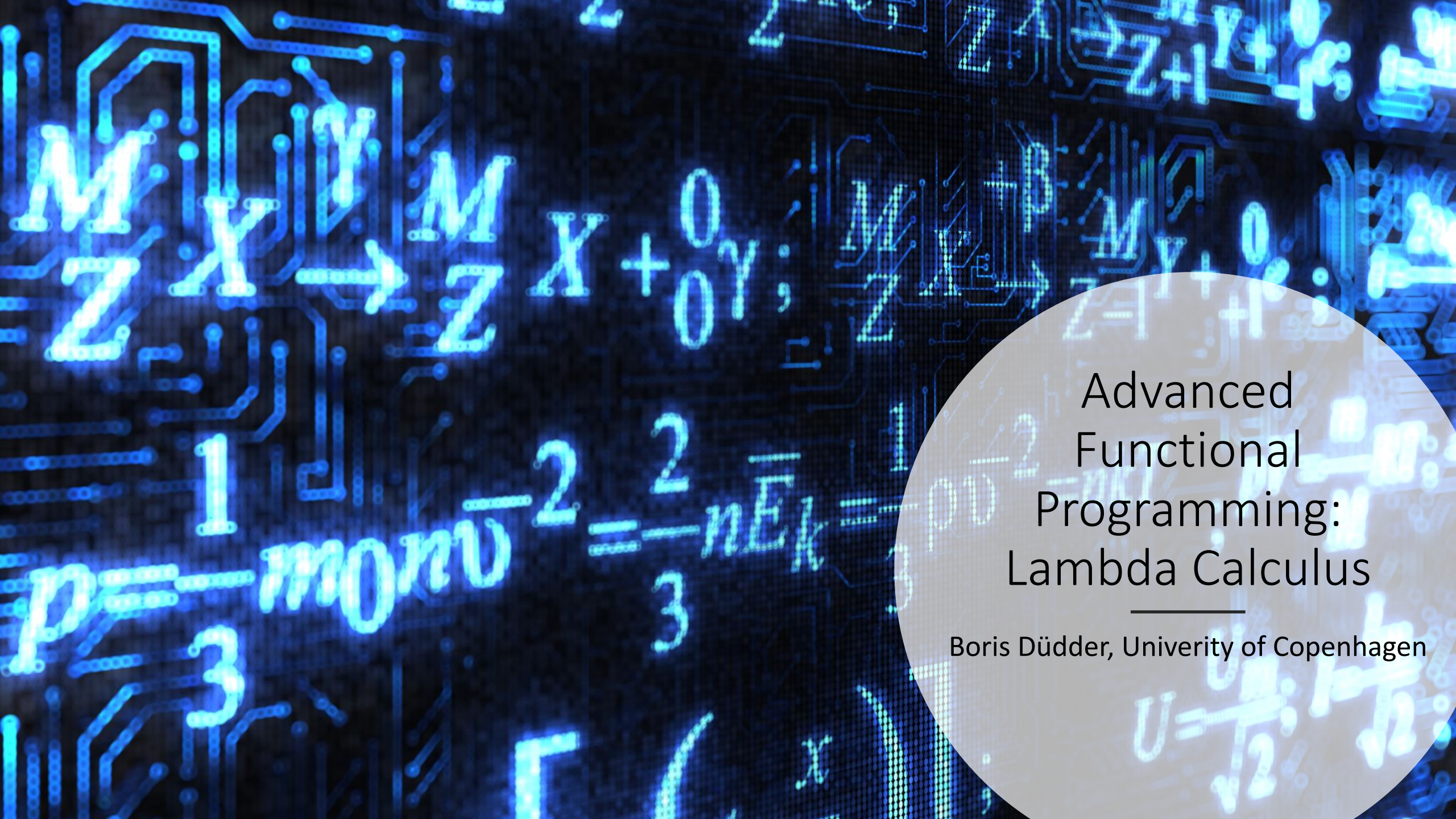


Advanced Functional Programming: Lambda Calculus

Boris Düdder, Univerity of Copenhagen

Literature

- Henk Barendregt, Lambda Calculus with Types, part of Perspectives in Logic. Cambridge University Press. ISBN 9780521766142, 2013

- Henk Barendregt, Erik Barendsen, Introduction to Lambda Calculus, 2000, ftp://ftp.cs.ru.nl/pub/CompMath.Found/lambda.pdf

Learning objectives

- Awareness of functional **calculation**.
- Understanding of the problem of **computation**.
- Different rough **concepts** and methods for the solution.
- Compare selected **approaches**.
- Understanding the **expressiveness** of functional programs.

Alonzo Church

American Logician
June 14, 1903 – August 11, 1995

Work:

- Lambda calculus,
- Church-Turing thesis
- Proof of the undecidability of the decision problem
- Frege-Church ontology
- Church-Rosser's theorem
- Taught at: Princeton, 1929-1967
  University of California, Los Angeles, 1967-1990

Why Lambda calculus

Three fundamental computational models:

- Turing machines (von Neumann architectures, modern computers)
- Primitive Recursive Functions
- Lambda calculus (functional programming languages)

Why lambda calculus?

Mother of all functional programming languages

Theory workshop for programming languages

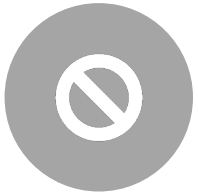
Type theory and the like

Simple semantics → complex calculations

Allows for formal analysis and verification

Lecture content

Definition of the lambda calculus



Examples

Church-Rosser theorem

Encoding of primitive recursive functions

Undecidability of the lambda calculus

Type free (untyped) lambda calculus

1.1.1. Definition. Let

$$V = \{v_0, v_1, \dots\}$$

denote an infinite alphabet. The set Λ^- of *pre-terms* is the set of strings defined by the grammar:

$$\Lambda^- ::= V \mid (\Lambda^- \ \Lambda^-) \mid (\lambda V \ \Lambda^-)$$

Pre-terms

1.1.2. Example. The following are pre-terms.

(i) $((v_0\ v_1)\ v_2) \in \Lambda^-$;

(ii) $(\lambda v_0\ (v_0\ v_1)) \in \Lambda^-$;

(iii) $((\lambda v_0\ v_0)\ v_1) \in \Lambda^-$;

(iv) $((\lambda v_0\ (v_0\ v_0))\ (\lambda v_1\ (v_1\ v_1))) \in \Lambda^-$.

1.1.3. Notation. We use uppercase letters, e.g., K, L, M, N, P, Q, R with or without subscripts to denote arbitrary elements of Λ^- and lowercase letters, e.g., x, y, z with or without subscripts to denote arbitrary elements of V.

1.1.4. Terminology.

(i) A pre-term of form x (i.e., an element of V) is called a *variable*;

(ii) A pre-term of form $(\lambda x\ M)$ is called an *abstraction* (over x);

(iii) A pre-term of form $(M\ N)$ is called an *application* (of M to N).

Notational conventions

1.1.5. NOTATION. We use the shorthands

(i) $(K\ L\ M)$ for $((K\ L)\ M)$;

(ii) $(\lambda x\ \lambda y\ M)$ for $(\lambda x\ (\lambda y\ M))$;

(iii) $(\lambda x\ M\ N)$ for $(\lambda x\ (M\ N))$;

(iv) $(M\ \lambda x\ N)$ for $(M\ (\lambda x\ N))$.

1.1.7. NOTATION. We write $\lambda x_1 \ldots x_n.M$ for $\lambda x_1\ \ldots \lambda x_n\ M$. As a special case, we write $\lambda x.M$ for $\lambda x\ M$.

1.1.9. EXAMPLE. The pre-terms in Example 1.1.2 can be written as follows, respectively:

(i) $v_0\ v_1\ v_2$;

(ii) $\lambda v_0.v_0\ v_1$;

(iii) $(\lambda v_0.v_0)\ v_1$;

(iv) $(\lambda v_0.v_0\ v_0)\ \lambda v_1.v_1\ v_1$.

1.1.10. REMARK. The conventions mentioned above are used in the remainder of these notes. However, we refrain from using them—wholly or partly—when we find this more convenient. For instance, we might prefer to write $(\lambda v_0.v_0\ v_0)\ (\lambda v_1.v_1\ v_1)$ for the last term in the above example.

Example

Free variables, closed term

1.1.11. DEFINITION. For $M \in \Lambda^-$ define the set $\mathrm{FV}(M) \subseteq V$ of *free variables* of M as follows.

$$\begin{aligned} \mathrm{FV}(x) &= \{x\}; \\ \mathrm{FV}(\lambda x.P) &= \mathrm{FV}(P)\backslash\{x\}; \\ \mathrm{FV}(P\,Q) &= \mathrm{FV}(P) \cup \mathrm{FV}(Q). \end{aligned}$$

If $\mathrm{FV}(M) = \{\}$ then M is called *closed.*

1.1.12. EXAMPLE. Let x, y, z denote distinct variables. Then

(i) $\mathrm{FV}(x\ y\ z) = \{x, y, z\}$;

(ii) $\mathrm{FV}(\lambda x.x\ y) = \{y\}$;

(iii) $\mathrm{FV}((\lambda x.x\ x)\ \lambda y.y\ y) = \{\}$.

Example

Substitution

1.1.13. DEFINITION. For $M, N \in \Lambda^-$ and $x \in V$, the *substitution of N for x in M*, written $M[x := N] \in \Lambda^-$, is defined as follows, where $x \neq y$:

$$
\begin{array}{lll}
x[x := N] & = N; & \\
y[x := N] & = y; & \\
(P\ Q)[x := N] & = P[x := N]\ Q[x := N]; & \\
(\lambda x.P)[x := N] & = \lambda x.P; & \\
(\lambda y.P)[x := N] & = \lambda y.P[x := N], & \text{if } y \notin \mathrm{FV}(N) \text{ or } x \notin \mathrm{FV}(P); \\
(\lambda y.P)[x := N] & = \lambda z.P[y := z][x := N], & \text{if } y \in \mathrm{FV}(N) \text{ and } x \in \mathrm{FV}(P).
\end{array}
$$

where z is chosen as the $v_i \in V$ with minimal i such that $v_i \notin \mathrm{FV}(P) \cup \mathrm{FV}(N)$ in the last clause.

1.1.14. EXAMPLE. If x, y, z are distinct variables, then for a certain variable u:

$$((\lambda x.x\ yz)\ (\lambda y.x\ y\ z)\ (\lambda z.x\ y\ z))[x := y] = (\lambda x.x\ yz)\ (\lambda u.y\ u\ z)\ (\lambda z.y\ y\ z)$$

Example

Alpha-equivalence (alpha-conversion)

1.1.15. DEFINITION. Let *α-equivalence*, written $=_\alpha$, be the smallest relation on Λ^-, such that

$$\begin{array}{ll} P =_\alpha P & \text{for all } P; \\ \lambda x.P =_\alpha \lambda y.P[x := y] & \text{if } y \notin \mathrm{FV}(P), \end{array}$$

and closed under the rules:

$$\begin{array}{lcll} P =_\alpha P' & \Rightarrow & \forall x \in V: & \lambda x.P =_\alpha \lambda x.P'; \\ P =_\alpha P' & \Rightarrow & \forall Z \in \Lambda^-: & P\ Z =_\alpha P'\ Z; \\ P =_\alpha P' & \Rightarrow & \forall Z \in \Lambda^-: & Z\ P =_\alpha Z\ P'; \\ P =_\alpha P' & \Rightarrow & & P' =_\alpha P; \\ P =_\alpha P' \ \&\ P' =_\alpha P'' & \Rightarrow & & P =_\alpha P''. \end{array}$$

1.1.16. EXAMPLE. Let x, y, z denote different variables. Then

(i) $\lambda x.x =_\alpha \lambda y.y$;

(ii) $\lambda x.x\ z =_\alpha \lambda y.y\ z$;

(iii) $\lambda x.\lambda y.x\ y =_\alpha \lambda y.\lambda x.y\ x$;

(iv) $\lambda x.x\ y \neq_\alpha \lambda x.x\ z$.

Example

Terms (modulo alpha-equivalence)

1.1.17. DEFINITION. Define for any $M \in \Lambda^-$, the *equivalence class* $[M]_\alpha$ by:

$$[M]_\alpha = \{N \in \Lambda^- \mid M =_\alpha N\}$$

Then define the set Λ of λ-*terms* by:

$$\Lambda \quad = \quad \Lambda^-/=_\alpha \quad = \quad \{[M]_\alpha \mid M \in \Lambda^-\}$$

1.1.19. NOTATION. We write M instead of $[M]_\alpha$ in the remainder. This leads to ambiguity: is M a pre-term or a λ-term? In the remainder of these notes, M should always be construed as $[M]_\alpha \in \Lambda$, *except when explicitly stated otherwise.*

Free variables (modulo alpha)

1.1.20. DEFINITION. For $M \in \Lambda$ define the set $\mathrm{FV}(M) \subseteq V$ of *free variables* of M as follows.

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{x\}; \\ \mathrm{FV}(\lambda x.P) & = & \mathrm{FV}(P)\backslash\{x\}; \\ \mathrm{FV}(P\,Q) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q). \end{array}$$

If $\mathrm{FV}(M) = \{\}$ then M is called *closed.*

1.1.21. REMARK. According to Notation 1.1.19, what we really mean by this is that we define FV as the map from Λ to subsets of V satisfying the rules:

$$\begin{array}{lcl} \mathrm{FV}([x]_\alpha) & = & \{x\}; \\ \mathrm{FV}([\lambda x.P]_\alpha) & = & \mathrm{FV}([P]_\alpha)\backslash\{x\}; \\ \mathrm{FV}([P\,Q]_\alpha) & = & \mathrm{FV}([P]_\alpha) \cup \mathrm{FV}([Q]_\alpha). \end{array}$$

Substitution (modulo alpha)

1.1.22. DEFINITION. For $M, N \in \Lambda$ and $x \in V$, the *substitution of N for x in M*, written $M\{x := N\}$, is defined as follows:

$$
\begin{array}{lcll}
x[x := N] & = & N; & \\
y[x := N] & = & y, & \text{if } x \neq y; \\
(P\ Q)[x := N] & = & P[x := N]\ Q[x := N]; & \\
(\lambda y.P)[x := N] & = & \lambda y.P[x := N], & \text{if } x \neq y, \text{ where } y \notin \mathrm{FV}(N).
\end{array}
$$

1.1.23. EXAMPLE.

(i) $(\lambda x.x\ y)[x := \lambda z.z] = \lambda x.x\ y$;

(ii) $(\lambda x.x\ y)[y := \lambda z.z] = \lambda x.x\ \lambda z.z$.

Beta notion of reduction

1.2.1. DEFINITION. Let $\rightarrow_\beta$ be the smallest relation on Λ such that

$$(\lambda x.P)\ Q \rightarrow_\beta P[x := Q],$$

and closed under the rules:

$$\begin{array}{lcll} P \rightarrow_\beta P' & \Rightarrow & \forall x \in V: & \lambda x.P \rightarrow_\beta \lambda x.P' \\ P \rightarrow_\beta P' & \Rightarrow & \forall Z \in \Lambda: & P\ Z \rightarrow_\beta P'\ Z \\ P \rightarrow_\beta P' & \Rightarrow & \forall Z \in \Lambda: & Z\ P \rightarrow_\beta Z\ P' \end{array}$$

A term of form $(\lambda x.P)\ Q$ is called a *β-redex*, and $P[x := Q]$ is called its *β-contractum*. A term M is a *β-normal form* if there is no term N with $M \rightarrow_\beta N$.

Beta-reduction and conversion

1.2.2. DEFINITION.

(i) The relation $\twoheadrightarrow_\beta$ (*multi-step β-reduction*) is the transitive-reflexive closure of $\rightarrow_\beta$; that is, $\twoheadrightarrow_\beta$ is the smallest relation closed under the rules:

$$\begin{array}{lll} P \rightarrow_\beta P' & \Rightarrow & P \twoheadrightarrow_\beta P'; \\ P \twoheadrightarrow_\beta P' \ \& \ P' \twoheadrightarrow_\beta P'' & \Rightarrow & P \twoheadrightarrow_\beta P''; \\ P \twoheadrightarrow_\beta P. & & \end{array}$$

(ii) The relation $=_\beta$ (*β-equality*) is the transitive-reflexive-symmetric closure of $\rightarrow_\beta$; that is, $=_\beta$ is the smallest relation closed under the rules:

$$\begin{array}{lll} P \rightarrow_\beta P' & \Rightarrow & P =_\beta P'; \\ P =_\beta P' \ \& \ P' =_\beta P'' & \Rightarrow & P =_\beta P''; \\ P =_\beta P; & & \\ P =_\beta P' & \Rightarrow & P' =_\beta P. \end{array}$$

1.2.3. WARNING. In these notes, the symbol $=$ without any qualification is used to express the fact that two objects, e.g., pre-terms or λ-terms are identical. This symbol is very often used in the literature for β-equality.

Warning (many equalities!)

1.2.4. EXAMPLE.

(i) $(\lambda x.x\ x)\ \lambda z.z \to_\beta (x\ x)[x := \lambda z.z] = (\lambda z.z)\ \lambda y.y$;

(ii) $(\lambda z.z)\ \lambda y.y \to_\beta z[z := \lambda y.y] = \lambda y.y$;

(iii) $(\lambda x.x\ x)\ \lambda z.z \twoheadrightarrow_\beta \lambda y.y$;

(iv) $(\lambda x.x)\ y\ z =_\beta y\ ((\lambda x.x)\ z)$.

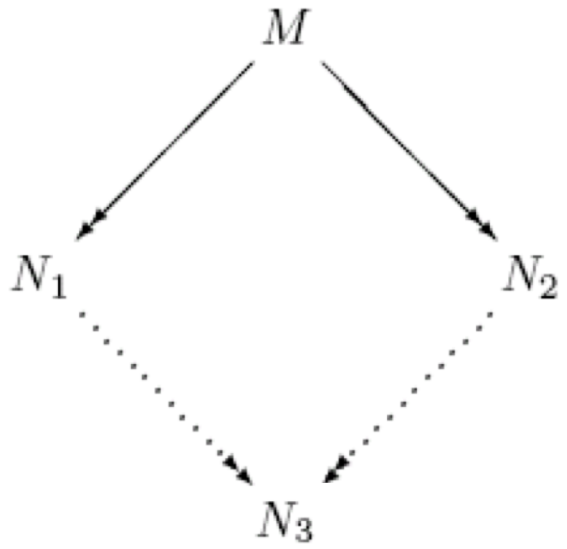
Example

$$\mathbf{K}\ (\mathbf{I}\ \mathbf{I}) \rightarrow_\beta \mathbf{K}\ \mathbf{I} \qquad \mathbf{K}\ (\mathbf{I}\ \mathbf{I}) \rightarrow_\beta \lambda x.(\mathbf{I}\ \mathbf{I})$$

The Church-Rosser theorem (confluence)

Church-Rosser (confluence) property

4.9. Church-Rosser Theorem. *If* $M \twoheadrightarrow_\beta N_1$, $M \twoheadrightarrow_\beta N_2$, *then for some* N_3 *one has* $N_1 \twoheadrightarrow_\beta N_3$ *and* $N_2 \twoheadrightarrow_\beta N_3$; *in diagram*



Question: Why not prove directly by induction?
Question: Why not generalize from local confluence?

Diamond property

1.4.1. DEFINITION. A relation $>$ on Λ satisfies the *diamond property* if, for all $M_1, M_2, M_3 \in \Lambda$, if $M_1 > M_2$ and $M_1 > M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 > M_4$ and $M_3 > M_4$.

Question: Does beta reduction have the diamond property?

Diamond property

1.4.2. LEMMA. *Let $>$ be a relation on Λ and suppose that its transitive closure*[1] *is $\twoheadrightarrow_\beta$. If $>$ satisfies the diamond property, then so does $\twoheadrightarrow_\beta$.*

PROOF. First show by induction on n that $M_1 > N_1$ and $M_1 > \ldots > M_n$ implies that there are $N_2, \ldots, N_n$ such that $N_1 > N_2 > \ldots > N_n$ and $M_n > N_n$.

Using this property, show by induction on m that if $N_1 > \ldots > N_m$ and $N_1 >^* M_1$ then there are $M_2, \ldots, M_m$ such that $M_1 > M_2 > \ldots > M_m$ and $N_m >^* M_m$.

Now assume $M_1 \twoheadrightarrow_\beta M_2$ and $M_1 \twoheadrightarrow_\beta M_3$. Since $\twoheadrightarrow_\beta$ is the transitive closure of $>$ we have $M_1 > \ldots > M_2$ and $M_1 > \ldots > M_3$. By what was shown above, we can find M_4 such that $M_2 > \ldots > M_4$ and $M_3 > \ldots > M_4$. Since $\twoheadrightarrow_\beta$ is the transitive closure of $>$, also $M_2 \twoheadrightarrow_\beta M_4$ and $M_3 \twoheadrightarrow_\beta M_4$. □

Reflexive und transitive Hülle

[1]Let R be a relation on Λ. The *transitive closure* of R is the least relation R^* satisfying:

$$\begin{array}{lcl} PRP' & \Rightarrow & PR^*P' \\ PR^*P' \ \& \ P'R^*P'' & \Rightarrow & PR^*P'' \end{array}$$

The *reflexive closure* of R is the least relation $R^=$ satisfying:

$$\begin{array}{lcl} PRP' & \Rightarrow & PR^=P' \\ PR^=P & & \end{array}$$

Parallel reduction (Tait & Martin-Löf)

1.4.3. DEFINITION. Let $\twoheadrightarrow_l$ be the relation on Λ defined by:

$$
\begin{array}{lcl}
P \twoheadrightarrow_l P & & \\
P \twoheadrightarrow_l P' & \Rightarrow & \lambda x.P \twoheadrightarrow_l \lambda x.P' \\
P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & P\,Q \twoheadrightarrow_l P'\,Q' \\
P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & (\lambda x.P)\,Q \twoheadrightarrow_l P'[x := Q']
\end{array}
$$

1.4.4. LEMMA. $M \twoheadrightarrow_l M' \ \& \ N \twoheadrightarrow_l N' \Rightarrow M[x := N] \twoheadrightarrow_l M'[x := N']$.

PROOF. By induction on the definition of $M \twoheadrightarrow_l M'$. In case M' is M, proceed by induction on M. □

Parallel reduction (Tait & Martin-Löf)

Parallel reduction has diamond property

1.4.5. LEMMA. *$\twoheadrightarrow_l$ satisfies the diamond property, i.e., for all $M_1, M_2, M_3 \in \Lambda$, if $M_1 \twoheadrightarrow_l M_2$ and $M_1 \twoheadrightarrow_l M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 \twoheadrightarrow_l M_4$ and $M_3 \twoheadrightarrow_l M_4$.*

PROOF. By induction on the definition of $M_1 \twoheadrightarrow_l M_2$, using the above lemma. □

Parellel reduction and beta reduction

1.4.6. LEMMA. *$\twoheadrightarrow_\beta$ is the transitive closure of $\twoheadrightarrow_l$.*

PROOF. Clearly[2]

$$(\rightarrow_\beta)^= \subseteq \twoheadrightarrow_l \subseteq \twoheadrightarrow_\beta$$

Then

$$\twoheadrightarrow_\beta = ((\rightarrow_\beta)^=)^* \subseteq \twoheadrightarrow_l^* \subseteq (\twoheadrightarrow_\beta)^* = \twoheadrightarrow_\beta$$

In particular, $\twoheadrightarrow_l^* = \twoheadrightarrow_\beta$. □

Church-Rosser theorem

1.4.7. THEOREM (Church and Rosser, 1936). *For every* $M_1, M_2, M_3 \in \Lambda$, *if* $M_1 \twoheadrightarrow_\beta M_2$ *and* $M_1 \twoheadrightarrow_\beta M_3$, *then there exists an* $M_4 \in \Lambda$ *such that* $M_2 \twoheadrightarrow_\beta M_4$ *and* $M_3 \twoheadrightarrow_\beta M_4$.

PROOF (Tait & Martin-Löf). By the above three lemmas. □

Corollaries of the Church-Rosser theorem

1.4.8. COROLLARY. *For all $M, N \in \Lambda$, if $M =_\beta N$, then there exists an $L \in \Lambda$ such that $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$.*

1.4.9. COROLLARY. *For all $M, N_1, N_2 \in \Lambda$, if $M \twoheadrightarrow_\beta N_1$ and $M \twoheadrightarrow_\beta N_2$ and both N_1 and N_2 are in β-normal form, then $N_1 = N_2$.*

1.4.10. COROLLARY. *For all $M, N \in \Lambda$, if there are β-normal forms L_1 and L_2 such that $M \twoheadrightarrow_\beta L_1$, $N \twoheadrightarrow_\beta L_2$, and $L_1 \neq L_2$, then $M \neq_\beta N$.*

Notions of consistency and completeness for beta-conversion

- Equational consistency (non-triviality)
  - There exists an unprovable equation

- Hilbert-Post completeness
  - Maximal consistency
  - No new equation can be consistently added

Expressibility

- Church numerals (Church, Rosser)
- Booleans, pairs and other data types
- Fixed point operators
- Coding of recursive functions (lambda definability theorem, Kleene)
- Undecidability (Curry, Scott) [Rice‘s theorem for lambda calculus]

Arithmetic

2.14. DEFINITION. (i) $F^n(M)$ with $F \in \Lambda$ and $n \in \mathbb{N}$ is defined inductively as follows.

$$\begin{aligned} F^0(M) &\equiv M; \\ F^{n+1}(M) &\equiv F(F^n(M)). \end{aligned}$$

(ii) The *Church numerals* $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots$ are defined by

$$\boldsymbol{c}_n \equiv \lambda f x.f^n(x).$$

Arithmetic

2.15. Propos J.B. Rosser). *Define*

$$\begin{aligned} \mathbf{A}_{+} &\equiv \lambda xypq.xp(ypq); \\ \mathbf{A}_{*} &\equiv \lambda xyz.x(yz); \\ \mathbf{A}_{\exp} &\equiv \lambda xy.yx. \end{aligned}$$

Then one has for all $n, m \in \mathbb{N}$

(i) $\mathbf{A}_{+}\boldsymbol{c}_n\boldsymbol{c}_m = \boldsymbol{c}_{n+m}$.

(ii) $\mathbf{A}_{*}\boldsymbol{c}_n\boldsymbol{c}_m = \boldsymbol{c}_{n*m}$.

(iii) $\mathbf{A}_{\exp}\boldsymbol{c}_n\boldsymbol{c}_m = \boldsymbol{c}_{(n^m)}$, *except for* $m = 0$ *(Rosser started counting from* 1*).*

LEMMA. (i) $(\boldsymbol{c}_n x)^m(y) = x^{n*m}(y)$.
(ii) $(\boldsymbol{c}_n)^m(x) = \boldsymbol{c}_{(n^m)}(x)$, *for* $m > 0$.

PROOF. (i) Induction on m. If $m = 0$, then LHS $= y =$ RHS. Assume (i) is correct for m (Induction Hypothesis: IH). Then

$$\begin{aligned}
(\boldsymbol{c}_n x)^{m+1}(y) &= \boldsymbol{c}_n x((\boldsymbol{c}_n x)^m(y)) \\
&= \boldsymbol{c}_n x(x^{n*m}(y)) \quad \text{by IH,} \\
&= x^n(x^{n*m}(y)) \\
&\equiv x^{n+n*m}(y) \\
&\equiv x^{n*(m+1)}(y).
\end{aligned}$$

(ii) Induction on $m > 0$. If $m = 1$, then LHS $\equiv \boldsymbol{c}_n x \equiv$ RHS. If (ii) is correct for m, then

$$\begin{aligned}
(\boldsymbol{c}_n)^{m+1}(x) &= \boldsymbol{c}_n((\boldsymbol{c}_n)^m(x)) \\
&= \boldsymbol{c}_n(\boldsymbol{c}_{(n^m)}(x)) \quad \text{by IH,} \\
&= \lambda y.(\boldsymbol{c}_{(n^m)}(x))^n(y) \\
&= \lambda y.x^{n^m*n}(y) \quad \text{by (i),} \\
&= \boldsymbol{c}_{(n^{m+1})}x.
\end{aligned}$$

PROOF OF THE PROPOSITION. (i) Exercise.

(ii) Exercise. Use Lemma 2.16 (i).

(iii) By Lemma 2.16 (ii) we have for $m > 0$

$$\begin{aligned} \mathsf{A}_{\exp}\boldsymbol{c}_n\boldsymbol{c}_m &= \boldsymbol{c}_m\boldsymbol{c}_n \\ &= \lambda x.(\boldsymbol{c}_n)^m(x) \\ &= \lambda x.\boldsymbol{c}_{(n^m)}x \\ &= \boldsymbol{c}_{(n^m)}, \end{aligned}$$

since $\lambda x.Mx = M$ if $M \equiv \lambda y.M'[y]$ and $x \notin \mathrm{FV}(M)$. Indeed,

$$\begin{aligned} \lambda x.Mx &\equiv \lambda x.(\lambda y.M'[y])x \\ &= \lambda x.M'[x] \\ &\equiv \lambda y.M'[y] \\ &\equiv M. \ \square \end{aligned}$$

Booleans and conditionals

1.5.6. PROPOSITION. *Define*

$$\begin{aligned}\mathbf{true} &= \lambda x.\lambda y.x;\\ \mathbf{false} &= \lambda x.\lambda y.y;\\ \mathbf{if}\ B\ \mathbf{then}\ P\ \mathbf{else}\ Q &= B\ P\ Q.\end{aligned}$$

Then

$$\begin{aligned}\mathbf{if\ true\ then}\ P\ \mathbf{else}\ Q &=_\beta P;\\ \mathbf{if\ false\ then}\ P\ \mathbf{else}\ Q &=_\beta Q.\end{aligned}$$

Pairing

1.5.7. PROPOSITION. *Define*

$$\begin{aligned} [P,Q] &= \lambda x.x\ P\ Q; \\ \pi_1 &= \lambda x.\lambda y.x; \\ \pi_2 &= \lambda x.\lambda y.y. \end{aligned}$$

Then

$$\begin{aligned} [P,Q]\ \pi_1 &=_\beta P; \\ [P,Q]\ \pi_2 &=_\beta Q. \end{aligned}$$

1.5.8. REMARK. Note that we do not have $[M\ \pi_1, M\ \pi_2] =_\beta M$ for all $M \in \Lambda$; that is, our pairing operator is not *surjective*.

1.5.9. REMARK. The construction is easily generalized to tuples $[M_1, \ldots, M_n]$ with projections π_i where $i \in \{1, \ldots, n\}$.

Fixed point theorem

1.5.10. THEOREM (Fixed point theorem). *For all F there is an X such that*

$$F\ X =_\beta X$$

In fact, there is a λ-term $\mathbf{Y}$ such that, for all F:

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

PROOF. Put

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

Then

$$\begin{array}{rcl} \mathbf{Y}\ F & = & (\lambda f.(\lambda x.f\ (x\ x))\ \lambda x.f\ (x\ x))\ F \\ & =_\beta & (\lambda x.F\ (x\ x))\ \lambda x.F\ (x\ x) \\ & =_\beta & F\ ((\lambda x.F\ (x\ x))\ \lambda x.F\ (x\ x)) \\ & =_\beta & F\ ((\lambda f.(\lambda x.f\ (x\ x))\ \lambda x.f\ (x\ x))\ F) \\ & = & F\ (\mathbf{Y}\ F) \end{array}$$

… and more

Another common fixed point combinator is
the Turing fixed-point combinator (named after its discoverer, Alan Turing):

Θ = (λx. λy. (y (x x y))) (λx. λy. (y (x x y)))

It also has a simple call-by-value form:
Θv = (λx. λy. (y (λz. x x y z))) (λx. λy. (y (λz. x x y z)))

Some fixed point combinators, such as this one (constructed by J.W.Klop)
are useful chiefly for amusement:

Yk = (L L L L L L L L L L L L L L L L L L L L L L L L L)
where:
L = λabcdefghijklmnopqstuvwxyzr. (r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

Wikipedia: Fixed point combinator

Lambda-definability

1.5.13. Definition.

(i) A *numeric function* is a map

$$f : \mathbb{N}^m \to \mathbb{N}.$$

(ii) A numeric function $f : \mathbb{N}^m \to \mathbb{N}$ is λ-*definable* if there is an $F \in \Lambda$ such that

$$F\ c_{n_1}\ \ldots\ c_{n_m} =_\beta c_{f(n_1,\ldots,n_m)}$$

for all $n_1, \ldots, n_m \in \mathbb{N}$.

Recursive functions

1.5.15. DEFINITION. The class of *recursive functions* is the smallest class of numeric functions containing the *initial functions*

(i) *projections:* $U_i^m(n_1, \ldots, n_m) = n_i$ for all $1 \leq i \leq m$;

(ii) *successor:* $S^+(n) = n + 1$;

(iii) *zero:* $Z(n) = 0$.

and closed under *composition*, *primitive recursion*, and *minimization*:

Recursive functions

(i) *composition:* if $g : \mathbb{N}^k \to \mathbb{N}$ and $h_1, \ldots, h_k : \mathbb{N}^m \to \mathbb{N}$ are recursive, then so is $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = g(h_1(n_1, \ldots, n_m), \ldots, h_k(n_1, \ldots, n_m)).$$

(ii) *primitive recursion:* if $g : \mathbb{N}^m \to \mathbb{N}$ and $h : \mathbb{N}^{m+2} \to \mathbb{N}$ are recursive, then so is $f : \mathbb{N}^{m+1} \to \mathbb{N}$ defined by

$$\begin{aligned} f(0, n_1, \ldots, n_m) &= g(n_1, \ldots, n_m); \\ f(n+1, n_1, \ldots, n_m) &= h(f(n, n_1, \ldots, n_m), n, n_1, \ldots, n_m). \end{aligned}$$

(iii) *minimization:* if $g : \mathbb{N}^{m+1} \to \mathbb{N}$ is recursive and for all $n_1, \ldots, n_m$ there is an n such that $g(n, n_1, \ldots, n_m) = 0$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined as follows is also recursive[3]

$$f(n_1, \ldots, n_m) = \mu n.g(n, n_1, \ldots, n_m) = 0$$

Definability of the initial functions

1.5.16. LEMMA. *The initial functions are λ-definable.*

PROOF. With

$$\begin{aligned} \mathbf{U}_i^m &= \lambda x_1 \ldots . \lambda x_m . x_i \\ \mathbf{S}^+ &= \lambda x . \lambda s . \lambda z . s \ (x \ s \ z) \\ \mathbf{Z} &= \lambda x . c_0 \end{aligned}$$

the necessary properties hold.

Composition

1.5.17. LEMMA. *The λ-definable functions are closed under composition.*

PROOF. If $g : \mathbb{N}^k \to \mathbb{N}$ is λ-definable by $G \in \Lambda$ and $h_1, \ldots, h_k : \mathbb{N}^m \to \mathbb{N}$ are λ-definable by some $H_1, \ldots, H_k \in \Lambda$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = g(h_1(n_1, \ldots, n_m), \ldots, h_k(n_1, \ldots, n_m))$$

is λ-definable by

$$F = \lambda x_1. \ldots \lambda x_m. G\ (H_1\ x_1 \ldots x_m) \ldots (H_k\ x_1 \ldots x_m),$$

as is easy to verify. □

Primitive recursion

1.5.18. LEMMA. *The λ-definable functions are closed under primitive recursion.*

PROOF. If $g : \mathbb{N}^m \to \mathbb{N}$ is λ-definable by some $G \in \Lambda$ and $h : \mathbb{N}^{m+2} \to \mathbb{N}$ is λ-definable by some $H \in \Lambda$, then $f : \mathbb{N}^{m+1} \to \mathbb{N}$ defined by

$$\begin{array}{lcl} f(0, n_1, \ldots, n_m) & = & g(n_1, \ldots, n_m); \\ f(n+1, n_1, \ldots, n_m) & = & h(f(n, n_1, \ldots, n_m), n, n_1, \ldots, n_m), \end{array}$$

is λ-definable by $F \in \Lambda$ where

$$\begin{array}{lcl} F & = & \lambda x.\lambda x_1 \ldots . \lambda x_m . x\, T\, [c_0, G\, x_1 \ldots x_n]\, \pi_2; \\ T & = & \lambda p.[\mathbf{S}^+\, (p\, \pi_1), H\, (p\, \pi_2)\, (p\, \pi_1)\, x_1 \ldots x_m]. \end{array}$$

Minimization

1.5.19. LEMMA. *The λ-definable functions are closed under minimization.*

PROOF. If $g : \mathbb{N}^{m+1} \to \mathbb{N}$ is λ-definable by $G \in \Lambda$ and for all $n_1, \ldots, n_m$ there is an n, such that $g(n, n_1, \ldots, n_m) = 0$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = \mu m.g(n, n_1, \ldots, n_m) = 0$$

is λ-definable by $F \in \Lambda$, where

$$F = \lambda x_1 \ldots . \lambda x_m . H\ c_0$$

and where $H \in \Lambda$ is such that

$$H =_\beta \lambda y.\mathbf{if}\ (\mathbf{zero?}\ (G\ x_1 \ldots x_m\ y))\ \mathbf{then}\ y\ \mathbf{else}\ H\ (\mathbf{S}^+\ y).$$

Here,

$$\mathbf{zero?} = \lambda x.x\ (\lambda y.\mathbf{false})\ \mathbf{true}$$

We leave it as an exercise to verify that the required properties hold. □

Kleene‘s theorem

The following can be seen as a form of *completeness* of the λ-calculus.

1.5.20. Theorem (Kleene). *All recursive functions are λ-definable.*

Proof. By the above lemmas. □

The converse also holds, as one can show by a routine argument. Similar results hold for partial functions as well—see [7].

Gödelization

1.5.21. DEFINITION. Let $\langle \bullet, \bullet \rangle : \mathbb{N}^2 \to \mathbb{N}$ be a bijective, recursive function. The map $\# : \Lambda^- \to \mathbb{N}$ is defined by:

$$\begin{array}{lcl} \#(v_i) & = & \langle 0, i \rangle \\ \#(\lambda x.M) & = & \langle 2, \langle \#(x), \#(M) \rangle \rangle \\ \#(M\ N) & = & \langle 3, \langle \#(M), \#(N) \rangle \rangle \end{array}$$

For $M \in \Lambda$, we take $\#(M)$ to be the least possible number $\#(M')$ where M' is an alpha-representative of M. Also, for $M \in \Lambda$, we define $\lceil M \rceil = c_{\#(M)}$.

Recursive sets of codes

1.5.22. DEFINITION. Let $A \subseteq \Lambda$.

(i) A is *closed under* $=_\beta$ if

$$M \in A \ \& \ M =_\beta N \Rightarrow N \in A$$

(ii) A is *non-trival* if

$$A \neq \emptyset \ \& \ A \neq \Lambda$$

(iii) A is *recursive* if

$$\#A = \{\#(M) \mid M \in A\}$$

is recursive.

Theorem von Curry & Scott

1.5.23. THEOREM (Curry, Scott). *Let A be non-trivial and closed under $=_\beta$. Then A is not recursive.*

PROOF (J. Terlouw). Suppose A is recursive. Define

$$B = \{M \mid M\ \ulcorner M \urcorner \in A\}$$

There exists an $F \in \Lambda$ with

$$\begin{array}{lcl} M \in B & \Leftrightarrow & F\ \ulcorner M \urcorner =_\beta c_0; \\ M \notin B & \Leftrightarrow & F\ \ulcorner M \urcorner =_\beta c_1. \end{array}$$

Let $M_0 \in A$, $M_1 \in \Lambda \backslash A$, and let

$$G = \lambda x.\mathbf{if}\ (\mathbf{zero?}\ (F\ x))\ \mathbf{then}\ M_1\ \mathbf{else}\ M_0$$

Then

$$\begin{array}{lcl} M \in B & \Leftrightarrow & G\ \ulcorner M \urcorner =_\beta M_1 \\ M \notin B & \Leftrightarrow & G\ \ulcorner M \urcorner =_\beta M_0 \end{array}$$

so

$$\begin{array}{lcllcl} G \in B & \Leftrightarrow & G\ \ulcorner G \urcorner =_\beta M_1 & \Rightarrow & G\ \ulcorner G \urcorner \notin A & \Rightarrow & G \notin B \\ G \notin B & \Leftrightarrow & G\ \ulcorner G \urcorner =_\beta M_0 & \Rightarrow & G\ \ulcorner G \urcorner \in A & \Rightarrow & G \in B \end{array}$$

a contradiction. □

Undecidability of lambda calculus

1.5.24. REMARK. The above theorem is analogous to *Rice's theorem* known in recursion theory.

The following is a variant of the halting problem. Informally it states that the formal theory of β-equality mentioned in Remark 1.4.12 is undecidable.

1.5.25. COROLLARY (Church). $\{M \in \Lambda \mid M =_\beta \mathbf{true}\}$ *is not recursive.*

1.5.26. COROLLARY. *The following set is not recursive:*

$$\{M \in \Lambda \mid \exists N \in \Lambda : M \twoheadrightarrow_\beta N \ \& \ N \text{ is a } \beta\text{-normal form } \}.$$

One can also infer from these results the well-known theorem due to Church stating that first-order predicate calculus is undecidable.

Questions?

Questions?

Prof. Dr. Boris Düdder
University of Copenhagen
Copenhagen
Denmark

Email: boris.d@di.ku.dk
Wechat: BorisDuedder
Mobile: +45 93565748

Information:

diku.dk
ebcc.eu
blockchainschool.eu

Co-funded by the Erasmus+ Programme of the European Union

BLOCKNET

Project: BlockChain Network Online Education for interdisciplinary European Competence Transfer
Project No: 2018-1-LT01-KA203-047044