# Proof Theory of The Lambda Calculus

Masahiko Sato
Graduate School of Informatics, Kyoto University
(Joint work with Takafumi Sakurai and Helmut Schwichtenberg)

Workshop on Mathematical Logic and its Applications
Kyoto University
September 17, 2016
(Revised on September 23, 2016)

## Overview

We introduce a free albebra $\mathbb{K}$ of $\mathbb{K}$-expressions, and define an embedding map which injectively embeds the set of closed $\boldsymbol{\lambda}$-terms into $\mathbb{K}$.

Some notable features of the datatype $\mathbb{K}$ are:

1. All the $\mathbb{K}$-*expressions* are constructed without using any variables.
2. Instead of the notion of substitution we have the notion of instantiation and can use this notion to define the $\beta$-reduction step as an algebraic operation on $\mathbb{K}$.

Taking advantage of these features, we can develop a proof theory of $\boldsymbol{\lambda}$-calclulus and can show the Church-Rosser Theorem smoothly within the Minlog proof assistant.

We can also define a category of derivations which admits pushout.

## Frege's view

In §§28 – 31 of *Grundgesetze der Arithmetic, volume 1* (1893), Frege tried to define the syntax and semantics (bedeutung) of the language (Begriffsschrift) he used in the book.

Russell found a technical gap in Frege's definition (Russell Paradox), but it is interesting to note that Frege defined his well-formed expressions (proper names), which include higher-order expressions, without starting from *variables*.

Therefore, I believe that Frege would have rejected the definition of raw lambda-terms given by Church:

$$\Lambda \ni M, N, P ::= x \mid (M\ N) \mid \lambda_x M$$

# Raw $\lambda$-terms

Definition of raw lambda-terms.

$$\Lambda \ni M, N, P ::= x \mid (M\ N) \mid \lambda_x M$$

$(M\ N)$ stands for the application of (function) $M$ to $N$.

We write $[x := N]M$ for the result of substituting $N$ for $x$ in $M$.

# Problems with raw $\lambda$-terms

A problem with raw lambda-terms is that substitution is non-trivial.

Let $M$ be $\boldsymbol{\lambda_y(x\ y)}$. Then, what is $[\boldsymbol{x} := \boldsymbol{y}]M$?

$[\boldsymbol{x} := \boldsymbol{y}]\boldsymbol{\lambda_y(x\ y)} = \boldsymbol{\lambda_y(y\ y)}$ is not correct. $\boldsymbol{y}$ was a free variable before substitution, but it becomes a bound variable after susbstitution.

The problem is solved by renaming $\boldsymbol{y}$ in $M$ to a fresh variable $\boldsymbol{z}$. Then, $[\boldsymbol{x} := \boldsymbol{y}]\boldsymbol{\lambda_z(x\ z)} = \boldsymbol{\lambda_z(y\ z)}$.

We replaced $M = \boldsymbol{\lambda_y(x\ y)}$ by $M' = \boldsymbol{\lambda_z(x\ z)}$ which is obtained by renaming. Such a pair $M$ and $M'$ are called $\alpha$-equivalent.

## Problems with raw $\lambda$-terms (cont.)

A second problem with raw $\boldsymbol{\lambda}$-terms is that the notion of immediate subterm becomes obscure on (raw) $\boldsymbol{\lambda}$-terms.

For example what is (or, are) the immediate subterm(s) of

$$\boldsymbol{\lambda_x \lambda_y (x\ y)?}$$

You may say the answer is $\boldsymbol{\lambda_y (x\ y)}$ (with $\boldsymbol{x}$ free).

But, then what about

$$\boldsymbol{\lambda_y \lambda_x (y\ x)?}$$

Your answer should be $\boldsymbol{\lambda_x (y\ x)}$ (with $\boldsymbol{y}$ free).

Since two given terms are $\boldsymbol{\alpha}$-equivalent, the answers must also be $\boldsymbol{\alpha}$-equivalent. But, this is not the case here.

## Problems with raw $\lambda$-terms (cont.)

All of these difficulties boil down to the following:

1. The raw $\boldsymbol{\lambda}$-terms $\boldsymbol{\lambda}_{x} x$ and $\boldsymbol{\lambda}_{y} y$ are two distinct raw $\boldsymbol{\lambda}$-terms (since they are syntactically different).
2. However, we somehow wish to identify them. And we do this by quotienting $\boldsymbol{\Lambda}$ by the $\boldsymbol{\alpha}$-equivalence relation.

## Raw $\lambda$-terms as an algebra

Raw $\boldsymbol{\lambda}$-terms $\boldsymbol{\Lambda}$ form a free algebra whose generators are the set of variables $\mathbb{X}$. Its signature is:

1. var : $\mathbb{X} \to \boldsymbol{\Lambda}$.
2. apply : $\boldsymbol{\Lambda} \times \boldsymbol{\Lambda} \to \boldsymbol{\Lambda}$.
3. $\boldsymbol{\lambda}$ : $\mathbb{X} \times \boldsymbol{\Lambda} \to \boldsymbol{\Lambda}$.

This is good. However, as we saw, to develop a proof theory of the $\boldsymbol{\lambda}$-caclulus, we must work in the quotient algebra $\boldsymbol{\Lambda}/\equiv_\alpha$.

But, since the quotient algebra is not a free algebra, we cannot use natural inductive argument on the structure of terms. Even worse, since we cannot directly define substitution on $\boldsymbol{\Lambda}$, there is no homomorphism from $\boldsymbol{\Lambda}$ to $\boldsymbol{\Lambda}/\equiv_\alpha$ which commutes with substituion.

# Structure of raw $\lambda$-terms

To see the essence of the $\alpha$-equivalence relation, we make the following observation.
Recall that:

$$\boldsymbol{\Lambda} \ni M, N, P ::= x \mid (M\ N) \mid \lambda_x M$$

By writing $\lambda_{x_1 x_2 \cdots x_n} M$ for $\lambda_{x_1} \lambda_{x_2} \cdots \lambda_{x_n} M$ $(n \geq 0)$, any $\lambda$-term can be uniquely written in one of the following two forms.

1. $\lambda_{x_1 x_2 \cdots x_n} y$.
2. $\lambda_{x_1 x_2 \cdots x_n} (M\ N)$.

## The set $\Lambda^0$ of closed terms

Then, we can define the subset $\boldsymbol{\Lambda^0}$ of $\boldsymbol{\Lambda}$, consiting of closed $\boldsymbol{\lambda}$-terms, as follows.

$$\frac{y \in \bar{x}}{\lambda_{\bar{x}} y \in \Lambda^0} \qquad \frac{\lambda_{\bar{x}} M \in \Lambda^0 \quad \lambda_{\bar{x}} N \in \Lambda^0}{\lambda_{\bar{x}}(M\ N) \in \Lambda^0}$$

Note that the above definition does not rely on the notion of free occurrences of a variable in a term.

This definition suggests that we should be able to develop proof theory of the $\boldsymbol{\lambda}$-calculus with free variables without appealing to the notion of bound variables, and of the $\boldsymbol{\lambda}$-caluculs of closed $\boldsymbol{\lambda}$-terms without using the notion of variables.

# The set $\Lambda^0$ of closed terms

Then, we can define the subset $\mathbf{\Lambda^0}$ of $\mathbf{\Lambda}$, consiting of closed $\boldsymbol{\lambda}$-terms, as follows.

$$\frac{y \in \bar{x}}{\lambda_{\bar{x}} y \in \Lambda^0} \qquad \frac{\lambda_{\bar{x}} M \in \Lambda^0 \quad \lambda_{\bar{x}} N \in \Lambda^0}{\lambda_{\bar{x}} (M\ N) \in \Lambda^0}$$

Note that the above definition does not rely on the notion of free occurrences of a variable in a term.

This definition suggests that we should be able to develop proof theory of the $\boldsymbol{\lambda}$-calculus with free variables without appealing to the notion of bound variables, and of the $\boldsymbol{\lambda}$-caluculs of closed $\boldsymbol{\lambda}$-terms without using the notion of variables.

But, it looks like that we need variables to develop $\boldsymbol{\lambda}$-calculus even on closed $\boldsymbol{\lambda}$-terms.

# $\lambda_\beta$-calculus

$$\frac{}{(\lambda_x M\ N) \to_\beta M[x := N]}\ \beta$$

$$\frac{M \to_\beta M'}{(M\ N) \to_\beta (M'\ N)}\ \mathbf{L} \qquad \frac{N \to_\beta N'}{(M\ N) \to_\beta (M\ N')}\ \mathbf{R}$$

$$\frac{M \to_\beta N}{\lambda_x M \to_\beta \lambda_x N}\ \xi$$

$$\frac{}{M \to_\beta M}\ \mathbf{Rfl} \qquad \frac{M \to_\beta N \quad N \to_\beta P}{M \to_\beta P}\ \mathbf{Trn}$$

The $\beta$-rule captures the informal notion of function application.

# $\mathbb{K}$-expressions

## Definition ($\mathbb{K}$-expressions)

$$\frac{\boldsymbol{i} \in \mathbb{N} \quad \boldsymbol{k} \in \mathbb{N}}{\mathsf{P}^{\boldsymbol{i}}_{\boldsymbol{k}} \in \mathbb{K}} \qquad \frac{\boldsymbol{j} \in \mathbb{N} \quad \boldsymbol{M} \in \mathbb{K} \quad \boldsymbol{N} \in \mathbb{K}}{(\boldsymbol{M}\ \boldsymbol{N})^{\boldsymbol{j}} \in \mathbb{K}}$$

We use $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}, \boldsymbol{N}$ as metavariables ranging over $\mathbb{K}$-expressions $\mathsf{P}^{\boldsymbol{i}}_{\boldsymbol{k}}$ is called a projection. We use $\boldsymbol{I}, \boldsymbol{J}$ as metavariables ranging over projections. $(\boldsymbol{M}\ \boldsymbol{N})^{\boldsymbol{j}}$ is called an application.

## Remark

1. $\mathbb{K}$-expressions are defined without using the notions of variable, $\boldsymbol{\lambda}$-abraction and $\boldsymbol{\alpha}$-equivalence. They are all *closed* terms.
2. $\mathbb{K}$ is a free algebra where projections are free generators and applications are binary operations parameterized by $\boldsymbol{j}$. So, we can study the structure of $\mathbb{K}$-epressions proof-theoretically by inductive arguments.

# Height and Thickness

## Definition (Height)

1. $\mathsf{Ht}(\mathsf{P}^{\boldsymbol{i}}_{\boldsymbol{k}}) := \boldsymbol{i} + \boldsymbol{k} + \mathbf{1}$.
2. $\mathsf{Ht}((\boldsymbol{M}\ \boldsymbol{N})^{\boldsymbol{j}}) := \min\{\boldsymbol{j}, \mathsf{Ht}(\boldsymbol{M}), \mathsf{Ht}(\boldsymbol{N})\}$.

An expression of height $\boldsymbol{h}$ can always be applied to $\boldsymbol{h}$ arguments.

## Definition (Thickness)

1. $\mathsf{Th}(\mathsf{P}^{\boldsymbol{i}}_{\boldsymbol{k}}) := \mathbf{1}$.
2. $\mathsf{Th}((\boldsymbol{M}\ \boldsymbol{N})^{\boldsymbol{j}}) := \mathsf{Th}(\boldsymbol{M}) + \mathsf{Th}(\boldsymbol{N})$.

# Projections

A projection $\mathsf{P}^{i}_{k}$ represents the following $\boldsymbol{\lambda}$-term.

$$\boldsymbol{\lambda}_{\bar{x}y\bar{z}}y,$$

where $\bar{x} = x_1 \cdots x_i$, $\bar{z} = z_1 \cdots z_k$ and $y \notin \bar{z}$.

For example, $\mathsf{P}^{0}_{0} = \boldsymbol{\lambda}_{y}y = \mathsf{I}$ and $\mathsf{P}^{0}_{1} = \boldsymbol{\lambda}_{yz}y = \mathsf{K}$.

## Embedding of $\Lambda^0$ into $\mathbb{L}$

Recall the following definition of $\boldsymbol{\Lambda^0}$.

$$\frac{\boldsymbol{y \in \bar{x}}}{\boldsymbol{\lambda_{\bar{x}} y \in \Lambda^0}} \qquad \frac{\boldsymbol{\lambda_{\bar{x}} M \in \Lambda^0 \quad \lambda_{\bar{x}} N \in \Lambda^0}}{\boldsymbol{\lambda_{\bar{x}}(M\ N) \in \Lambda^0}}$$

We define the embedding $[M]$ of $\boldsymbol{M \in \Lambda^0}$ into $\mathbb{K}$ as follows.

- $[\boldsymbol{\lambda_{x_1 \cdots x_i y z_1 \cdots z_k} y}] := \mathsf{P}^{\boldsymbol{i}}_{\boldsymbol{k}}$.
- $[\boldsymbol{\lambda_{\bar{x}}(M\ N)}] := \boldsymbol{([\lambda_{\bar{x}} M]\ [\lambda_{\bar{x}} N])^k}$, where $\boldsymbol{\bar{x} = x_1 \cdots x_k}$.

### Remark

The definition is well-defined, since $\alpha$-equivalent terms are embedded to the same $\mathbb{K}$-expression.

# Combinators

We can define combinators I, K and S as follows.

1. $\mathsf{I} := \lambda_x x = \mathsf{P}^{\mathbf{0}}_{\mathbf{0}}$.
2. $\mathsf{K} := \lambda_{xy} x = \mathsf{P}^{\mathbf{0}}_{\mathbf{1}}$.
3. $\mathsf{S} := \lambda_{xyz}((x\ z)\ (y\ z)) = (\lambda_{xyz}(x\ z)\ \lambda_{xyz}(y\ z))^{\mathbf{3}}$
   $= ((\lambda_{xyz}x\ \lambda_{xyz}z)^{\mathbf{3}}\ (\lambda_{xyz}y\ \lambda_{xyz}z)^{\mathbf{3}})^{\mathbf{3}}$
   $= ((\mathsf{P}^{\mathbf{0}}_{\mathbf{2}}\ \mathsf{P}^{\mathbf{2}}_{\mathbf{0}})^{\mathbf{3}}\ (\mathsf{P}^{\mathbf{1}}_{\mathbf{1}}\ \mathsf{P}^{\mathbf{2}}_{\mathbf{0}})^{\mathbf{3}})^{\mathbf{3}}$.

# Instantiation

## Definition (**Instantiation**)

Given $K, L \in \mathbb{K}$ such that $\mathrm{Ht}(K) > n$ and $\mathrm{Ht}(L) \geq n$, we define $\langle K\ L\rangle^n \in \mathbb{K}$ as follows.

1. $$\langle \mathsf{P}^i_k\ M\rangle^n := \begin{cases} \mathsf{P}^{i-1}_k & \text{if } n < i, \\ \Uparrow^k_i M & \text{if } n = i, \\ \mathsf{P}^i_{k-1} & \text{if } n > i. \end{cases}$$
2. $\langle (K\ L)^i\ M\rangle^n := (\langle K\ M\rangle^n\ \langle L\ M\rangle^n)^{i-1}$.

## Definition (**Lifting**)

1. $$\Uparrow^k_i \mathsf{P}^j_l := \begin{cases} \mathsf{P}^{j+k}_l & \text{if } i \leq j, \\ \mathsf{P}^j_{l+k} & \text{if } i > j. \end{cases}$$
2. $\Uparrow^k_i (M\ N)^j := (\Uparrow^k_i M\ \Uparrow^k_i N)^{j+k}$.

Note that: $\Uparrow^k_i M = \langle \mathsf{P}^i_k\ M\rangle^i$.

## Instantiation (cont.)

We can combine the previous two definitions and get the following.

**Definition (Instantiation $\langle K\ M\rangle^n$)**

1. $$\langle \mathsf{P}^i_k\ \mathsf{P}^j_l\rangle^n := \begin{cases} \mathsf{P}^{i-1}_k & \text{if } n < i, \\ \mathsf{P}^{j+k}_l & \text{if } n = i \text{ and} i \leq j, \\ \mathsf{P}^{j}_{l+k} & \text{if } n = i \text{ and } i > j, \\ \mathsf{P}^{i}_{k-1} & \text{if } n > i. \end{cases}$$

2. $$\langle \mathsf{P}^i_k\ (M\ N)^j\rangle^n := \begin{cases} \mathsf{P}^{i-1}_k & \text{if } n < i, \\ (\langle \mathsf{P}^i_k\ M\rangle^n\ \langle \mathsf{P}^i_k\ N\rangle^n)^{j+k} & \text{if } n = i, \\ \mathsf{P}^{i}_{k-1} & \text{if } n > i. \end{cases}$$

3. $\langle (K\ L)^i\ M\rangle^n := (\langle K\ M\rangle^n\ \langle L\ M\rangle^n)^{i-1}$.

**Remark**

$n$ is just passed around and does not change. So, for each $n$, instantiation is defined by primitive recursion on $\mathbb{K}$-expressions.

# de Bruijn indices

$$D, E, F ::= i \mid (D\ E) \mid [D]$$

Substitution $\langle D\ F\rangle^i$ (read: substitute $F$ for $i$ in $D$) is defined as follows.

1. $\langle j\ F\rangle^i := \begin{cases} F & \text{if } i = j, \\ j & \text{o.w..} \end{cases}$
2. $\langle (D\ E)\ F\rangle^i := (\langle D\ F\rangle^i\ \langle E\ F\rangle^i)$.
3. $\langle [D]\ F\rangle^i := [\langle D\ F'\rangle^{i+1}]$, where $F'$ is obtained from $F$ by shifting indices of $F$ appropriately.

## Remark

Both $i$ and $F$ are changed in the third item of the definition. So, to define $\langle D\ F\rangle^0$, one has to define $\langle D\ F\rangle^i$ for all $i$.

# Instantiation Lemma

Lemma (Instantiation Lemma)

$$n < m < \mathsf{Ht}(K), m \leq \mathsf{Ht}(L), n \leq \mathsf{Ht}(M) \vdash$$
$$\langle\langle K\ L\rangle^m\ M\rangle^n = \langle\langle K\ M\rangle^n\ \langle L\ M\rangle^n\rangle^{m-1}.$$

Note that we have:

$$\langle(K\ L)^m\ M\rangle^n := (\langle K\ M\rangle^n\ \langle L\ M\rangle^n)^{m-1}, \text{ and}$$

# Substitution and Instantiation

$$x \neq y, x \notin \mathrm{FV}(M) \vdash$$
$$K[x := L][y := M] = K[y := M][x := L[y := M]].$$

$$1 < \mathsf{Ht}(M) \;\vdash\; \langle\langle K\ L\rangle^1\ M\rangle = \langle\langle K\ M\rangle\ \langle L\ M\rangle\rangle.$$

We can see that Instantiation operation naturally represents $\beta$-conversion rule as an algebraic operation.

## $\mathbb{K}_\beta$-calculus

$$\frac{\mathsf{Ht}(M) > n \quad \mathsf{Ht}(N) \geq n}{(M\ N)^n \to_\beta \langle M\ N\rangle^n}\ \beta$$

$$\frac{M \to_\beta M'}{(M\ N)^n \to_\beta (M'\ N)^n}\ \mathsf{L} \qquad \frac{N \to_\beta N'}{(M\ N)^n \to_\beta (M\ N')^n}\ \mathsf{R}$$

$$\frac{}{M \to_\beta M}\ \mathsf{Rfl} \qquad \frac{M \to_\beta N \quad N \to_\beta P}{M \to_\beta P}\ \mathsf{Trn}$$

The $\beta$-rule of $\mathbb{K}_\beta$-calculus subsumes the $\beta$ and $\xi$ rules of $\lambda_\beta$-calculus.

$$\frac{}{(\lambda_x M\ N) \to_\beta M[x := N]}\ \beta \qquad \frac{M \to_\beta N}{\lambda_x M \to_\beta \lambda_x N}\ \xi$$

## Further directions

1. Adding free variables (as constants) to $\mathbb{K}$.

# Further directions

1. Adding free variables (as constants) to $\mathbb{K}$.
   - Then we can compare $\mathbb{K}$-expressions directly with $\boldsymbol{\lambda}$-terms with free variables.

## Further directions

1. Adding free variables (as constants) to $\mathbb{K}$.
   - Then we can compare $\mathbb{K}$-expressions directly with $\boldsymbol{\lambda}$-terms with free variables.
2. First-order theory of $\mathbb{K}_{\boldsymbol{\beta}}$-calculus.

# Further directions

1. Adding free variables (as constants) to $\mathbb{K}$.
   - Then we can compare $\mathbb{K}$-expressions directly with $\boldsymbol{\lambda}$-terms with free variables.
2. First-order theory of $\mathbb{K}_{\boldsymbol{\beta}}$-calculus.
   - Should be straigtforward, just by including instantiation operation as a function symbol. Note that there are no satisfactory first-order theories of $\boldsymbol{\lambda}_{\boldsymbol{\beta}}$-calculus since abstraction cannot be naturally axiomatized.

## Conclusion

- We introduced the datatype $\mathbb{K}$ of $\mathbb{K}$-expressions and showed that it is possible to embed closed $\boldsymbol{\lambda}$-terms into $\mathbb{K}$ faithfully.
- We also showed that it is possible to develop proof theory of the $\boldsymbol{\lambda}$-calculus without ever using the notions of variables, $\boldsymbol{\alpha}$-equivalence or substitution.
- We showed the Church-Rosser Theorem by the residual method, and also showed that it is possible to define a natural category of derivations which admits pushout.
- All the results reported in this talk were formally verified in the Minlog proof assistant.

# Acknowledgement

We thank the Japan Society for the Promotion of Science (JSPS), Core-to-Core Program (A. Advanced Research Networks) for supporting the research.