Lectures on The Lambda Calculus (III)

Masahiko Sato
Graduate School of Informatics, Kyoto University

Autumn school "Proof and Computation"
Fischbachau, Germany
October 7, 2016

Plan of the lectures

I. Background history, philosophy and *main idea*.

II. The free algebra $\mathbb{T}$ of *threads*

III. The free algebra $\mathbb{L}$ of $\mathbb{L}$-*expressions*.

These lectures are based on my work in progress.

What rests on what?

Does knowledge of ‘*what* $\boldsymbol{\lambda}$*-term is*’ rest on knowledge of ‘*what type theory is*’?

What rests on what?

Does knowledge of '*what* *$\boldsymbol{\lambda}$-term* *is*' rest on knowledge of '*what* *type theory* *is*'?

No, it is the other way around!

What rests on what?

Does knowledge of '*what* $\boldsymbol{\lambda}$*-term is*' rest on knowledge of '*what type theory is*'?

No, it is the other way around!

You have to know what $\boldsymbol{\lambda}$-term is *before* you can understand what type theory is.

What rests on what?

Does knowledge of '*what* $\boldsymbol{\lambda}$*-term is*' rest on knowledge of '*what type theory is*'?

No, it is the other way around!

You have to know what $\boldsymbol{\lambda}$-term is *before* you can understand what type theory is.

This is why I am trying to understand $\boldsymbol{\lambda}$-terms as finitary objects *created by finitistic method*.

History

- Frege, in his *Begriffsschrift* (1879), used latin letters for global variables and used german letters for local variables.
- Gentzen (in the 30's) also used different sets of variables for global and local variables. He also introduced *eigen variable*.
- Whitehead-Russell (1910) and, later, Gödel and Church used only one sort of letters for both global and local variables. (I think Church made a *conceptual mistake* here.)
- Quine and Bourbaki (in the 50's) introduced *graphical (two dimensional) notation* for local variable binding.
- McCarthy (1963) introduced *abstract syntax*
- de Bruijn (1972) introduced his *indices* and provided a canonical notation for α-equivalent terms.

Parametrized free algebra $\mathbb{L}$

We generalize the parametrized free algebra $\mathbb{L}$ given in Lecture II as follows.

$$\mathbb{T}_{\boldsymbol{\beta}} = \boldsymbol{\beta} + \boldsymbol{\lambda}\mathbb{T}_{\boldsymbol{\beta}},\ \mathbb{L}_{\boldsymbol{\tau}} = \boldsymbol{\tau} + (\mathbb{L}_{\boldsymbol{\tau}}\ \mathbb{L}_{\boldsymbol{\tau}})^{\mathbb{N}}$$

Here, in the second equation, $\boldsymbol{\tau}$ must be an instance of $\mathbb{T}_{\boldsymbol{\beta}}$ given in the first equation.

Hence, in order to get a concrete instance of $\mathbb{L}_{\boldsymbol{\tau}}$, one only has to specify a concrete algebra $\boldsymbol{\beta}$ (*the base algebra*), and a *height function* $\mathsf{Ht} : \boldsymbol{\beta} \to \mathbb{N}$. Then we put $\boldsymbol{\tau} := \mathbb{T}_{\boldsymbol{\beta}}$ and get $\mathbb{L}_{\boldsymbol{\tau}}$.

Parametrized free algebra $\mathbb{L}$ (cont.)

We saw the following two instances of $\boldsymbol{\beta}$ in Lecture II.

1. $\boldsymbol{\beta_1} = \mathbb{N}$ with Ht $\boldsymbol{n} := \boldsymbol{0}$.
2. $\boldsymbol{\beta_2} = \mathbb{N}$ with Ht $\boldsymbol{n} := \boldsymbol{n} + \boldsymbol{1}$.

In the first case, we interpreted each $\boldsymbol{n}$ as an de Bruin index, and in the second case, we interpreted each $\boldsymbol{n}$ as $\boldsymbol{\lambda}^{\boldsymbol{n}+\boldsymbol{1}}\boldsymbol{n}$ in the algebra $\mathbb{T}_{\boldsymbol{\beta_1}}$ (so that it always becomes a closed thread.)

$\boldsymbol{\beta_1}$ was used to define $\mathbb{L}$ which contains de Bruijn algebra $\mathbb{D}$, and $\boldsymbol{\beta_2}$ was used to define $\mathbb{L}$ containing only closed de Bruijn terms.

In this lecture we choose yet another $\boldsymbol{\beta}$ and use it to define $\mathbb{L}$ to be used throughout the lecture. The choice is based on *Frege-Gentzen's way* of using disjoint sets of letters for free and bound variables.

Definition of $\mathbb{L}$ in this lecture

In this lecture, we choose our $\boldsymbol{\beta}$ as follows and fix it throughout this lecture.

$$\boldsymbol{\beta} := \mathbb{A} + \mathbb{N}',$$

where we assume that we have a fixed bijective correspondence $\mathbb{A} \leftrightarrow \mathbb{N}$, and $\mathbb{N}' = \{\boldsymbol{n}' \mid \boldsymbol{n} \in \mathbb{N}\}$. We call elements of $\mathbb{A}$ *atoms*. Atoms will play the role of *free variables*.

Moreover, we use our knowledge of the above bijective correspondence only to decide the equality of two atoms. This setting automatically endows an *equivariant* structure on each of $\boldsymbol{\beta}, \boldsymbol{\tau} := \mathbb{T}_{\boldsymbol{\beta}}$ and $\mathbb{L}_{\boldsymbol{\tau}}$.

Namely, the group of finite permutations on $\mathbb{A}$ naturally determines equivariant group actions on each of these structures.

Definition of $\mathbb{L}$ in this lecture (cont.)

Now, we have $\mathbb{L}_{\boldsymbol{\tau}}$ determined by $\boldsymbol{\tau} = \mathbb{T}_{\boldsymbol{\beta}}$ and we write $\mathbb{L}$ for $\mathbb{L}_{\boldsymbol{\tau}}$ in this lecture.

$$\mathbb{T} = \mathbb{A} + \mathbb{N}' + \boldsymbol{\lambda}\mathbb{T}, \quad \mathbb{L} = \mathbb{T} + (\mathbb{L}\ \mathbb{L})^{\mathbb{N}}$$

$\mathbb{L}$ has the following abstact syntax.

$$\begin{aligned}
\mathbb{A} &\ni \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots \\
\mathbb{N} &\ni \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}, \boldsymbol{\ell}, \boldsymbol{m}, \boldsymbol{n} ::= \mathbf{0} \mid \boldsymbol{n}' \\
\mathbb{T} &\ni \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t} ::= \boldsymbol{a} \mid \boldsymbol{n}' \mid \boldsymbol{\lambda t} \\
\mathbb{L} &\ni \boldsymbol{M}, \boldsymbol{N}, \boldsymbol{P} ::= \boldsymbol{t} \mid (\boldsymbol{M}\ \ \boldsymbol{N})^{\boldsymbol{n}}
\end{aligned}$$

$\boldsymbol{\lambda}$ as unary operation on $\mathbb{T}$ and $\mathbb{L}$

As can be seen from the abstract syntax of $\mathbb{T}$, $\boldsymbol{\lambda}$ is a *constructor* on $\mathbb{T}$ having arity:

$$\boldsymbol{\lambda} : \mathbb{T} \to \mathbb{T}$$

We can naturally extend $\boldsymbol{\lambda}$ to $\boldsymbol{\lambda}$ so that $\boldsymbol{\lambda}$ will have arity:

$$\boldsymbol{\lambda} : \mathbb{L} \to \mathbb{L}$$

1. $\boldsymbol{\lambda} t := \boldsymbol{\lambda} t$
2. $\boldsymbol{\lambda}(M\ N)^n := (\boldsymbol{\lambda} M\ \boldsymbol{\lambda} N)^{n'}$

Now, writing $\boldsymbol{\lambda}\mathbb{T}$ for $\{\boldsymbol{\lambda} t \in \mathbb{T} \mid t \in \mathbb{T}\}$, we have two *bijections*:

$$\boldsymbol{\lambda} : \mathbb{T} \to \boldsymbol{\lambda}\mathbb{T} \quad \text{and} \quad \overline{\boldsymbol{\lambda}} : \boldsymbol{\lambda}\mathbb{T} \to \mathbb{T},$$

where $\overline{\boldsymbol{\lambda}}$ is the inverse of $\boldsymbol{\lambda}$. We have also two similar bijections for $\mathbb{L}$.

Height of threads and $\mathbb{L}$-terms

We define the height function Ht $: \mathbb{L} \to \mathbb{N}$ as follows. Ht M is called the *height* of M.

1. Ht $a := 0$
2. Ht $n' := n'$
3. Ht $\lambda t :=$ (Ht t)$'$
4. Ht $(M\ N)^n := \min\{n, \text{Ht } M, \text{Ht } N\}$

A term M is called an *abstract* if Ht $M > 0$.

Classification of $\mathbb{L}$ by height

We put

$$\mathbb{L}^n := \{M \in \mathbb{L} \mid \text{Ht } M \geq n\}.$$

We have

$$\mathbb{L} = \mathbb{L}^0 \supsetneq \mathbb{L}^1 \supsetneq \mathbb{L}^2 \cdots$$

We also note that

$$\lambda^n \mathbb{L} := \{\lambda^n M \mid M \in \mathbb{L}\} \subsetneq \mathbb{L}^n \quad (n > 0),$$

since, for example, in case $n = 1$, $1 \in \mathbb{L}^1$ cannot be written as λM.

Closing and opening

The subsets $\mathbb{T}^{\boldsymbol{n}} = \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{T}$ $(\boldsymbol{n} \in \mathbb{N})$ of $\mathbb{T}$ and the subsets $\mathbb{L}^{\boldsymbol{n}} = \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L}$ $(\boldsymbol{n} \in \mathbb{N})$ of $\mathbb{L}$ are bijectively related as follows.

$$\mathbb{T} \xrightarrow{\boldsymbol{\lambda}} \boldsymbol{\lambda}\mathbb{T} \xrightarrow{\boldsymbol{\lambda}} \boldsymbol{\lambda}^{\boldsymbol{2}}\mathbb{T} \xrightarrow{\boldsymbol{\lambda}} \cdots$$

$$\mathbb{T} \xleftarrow{\overline{\boldsymbol{\lambda}}} \boldsymbol{\lambda}\mathbb{T} \xleftarrow{\overline{\boldsymbol{\lambda}}} \boldsymbol{\lambda}^{\boldsymbol{2}}\mathbb{T} \xleftarrow{\overline{\boldsymbol{\lambda}}} \cdots$$

$$\mathbb{L} \xrightarrow{\boldsymbol{\lambda}} \boldsymbol{\lambda}\mathbb{L} \xrightarrow{\boldsymbol{\lambda}} \boldsymbol{\lambda}^{\boldsymbol{2}}\mathbb{L} \xrightarrow{\boldsymbol{\lambda}} \cdots$$

$$\mathbb{L} \xleftarrow{\overline{\boldsymbol{\lambda}}} \boldsymbol{\lambda}\mathbb{L} \xleftarrow{\overline{\boldsymbol{\lambda}}} \boldsymbol{\lambda}^{\boldsymbol{2}}\mathbb{L} \xleftarrow{\overline{\boldsymbol{\lambda}}} \cdots$$

Suggested by these diagrams we will call $\boldsymbol{\lambda}$ a *closing* operator.
Similarly $\overline{\boldsymbol{\lambda}}$ will be called an *opening* operator.

Abstraction and Instantiation

We have seen, in many lectures of this Autumn school, something like the following informal notation:

$$(\forall x.\ A(x)) \to A(t)$$

where t is a term, and $A(x)$ and $A(t)$ are formulas.

Abstraction and Instantiation

We have seen, in many lectures of this Autumn school, something like the following informal notation:

$$(\forall x.\ A(x)) \to A(t)$$

where t is a term, and $A(x)$ and $A(t)$ are formulas.

But, what is A here?

Abstraction and Instantiation

We have seen, in many lectures of this Autumn school, something like the following informal notation:

$$(\forall x.\ A(x)) \to A(t)$$

where t is a term, and $A(x)$ and $A(t)$ are formulas.

But, what is A here?

It is an *abstract*! It is not a formula by itself, but by *instantiating* A by a term, say, t, we get a formula for which we used the notation $A(t)$.

Note that the result of instantiation is completely determined by the abstract A and the term t.

Instantiation

We define the instantiation function

$$\langle - \ - \rangle : \mathbb{L}^{\mathbf{1}} \times \mathbb{L} \to \mathbb{L}$$

$\langle M \ P \rangle$ instantiates abstract M by P.

$$\begin{aligned}
\langle k' \ P \rangle &:= \lambda^{k} P. \\
\langle \lambda t \ P \rangle &:= t. \\
\langle (M \ N)^{n'} \ P \rangle &:= (\langle M \ P \rangle \ \langle N \ P \rangle)^{n} \quad (M, N \in \mathbb{L}^{\mathbf{1}}).
\end{aligned}$$

Note that $\mathbf{1}$ is the identity combinator I and $\mathbf{2}$ is the K combinator.

Instantiation at level n

What we will do here is to generalize the instantiation operation $\langle \boldsymbol{M}\ \boldsymbol{P}\rangle$ (which operates on $\boldsymbol{M} \in \mathbb{L}^{\mathbf{1}}$ and $\boldsymbol{P} \in \mathbb{L}$) to $\langle \boldsymbol{M}\ \boldsymbol{P}\rangle^{\boldsymbol{n}}$ with arity:

$$\langle -\ -\rangle^{\boldsymbol{n}} : \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L}^{\mathbf{1}} \times \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L} \to \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L}$$

We define this operation so that the following diagram commutes:

$$\begin{tikzcd}
\boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L}^{\mathbf{1}} \times \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L} \arrow[r, "\langle -\ -\rangle^{\boldsymbol{n}}"] \arrow[d, "\overline{\boldsymbol{\lambda}}^{\boldsymbol{n}} \times \overline{\boldsymbol{\lambda}}^{\boldsymbol{n}}"'] & \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L} \\
\mathbb{L}^{\mathbf{1}} \times \mathbb{L} \arrow[r, "\langle -\ -\rangle"'] & \mathbb{L} \arrow[u, "\boldsymbol{\lambda}^{\boldsymbol{n}}"']
\end{tikzcd}$$

So, the definition is

$$\langle \boldsymbol{M}\ \boldsymbol{N}\rangle^{\boldsymbol{n}} := \boldsymbol{\lambda}^{\boldsymbol{n}}\langle \overline{\boldsymbol{\lambda}}^{\boldsymbol{n}}\boldsymbol{M}\ \overline{\boldsymbol{\lambda}}^{\boldsymbol{n}}\boldsymbol{N}\rangle \quad (\boldsymbol{M} \in \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L}^{\mathbf{1}} \text{ and } \boldsymbol{N} \in \boldsymbol{\lambda}^{\boldsymbol{n}}\mathbb{L})$$

$\mathbb{L}_{\boldsymbol{\beta}}$-calculus

We define $\mathbb{L}_{\boldsymbol{\beta}}$-calculus as follows.

$$\frac{M \in \lambda^n \mathbb{L}^1 \quad N \in \lambda^n \mathbb{L}}{(M\ N)^n \to_\beta \langle M\ N \rangle^n}\ \beta$$

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

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

The $\boldsymbol{\beta}$-rule of $\mathbb{L}_{\boldsymbol{\beta}}$-calculus subsumes the $\boldsymbol{\beta}$ and $\boldsymbol{\xi}$ rules of $\boldsymbol{\lambda}_{\boldsymbol{\beta}}$-calculus.

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

Freshness

We define the meaning of the judgment 'a is *fresh* for M' (written $a \# M$) for all $a \in \mathbb{A}$ and $M \in \mathbb{L}$ as follows.

$$\frac{a \neq b}{a \# b} \qquad \frac{}{a \# k} \qquad \frac{a \# t}{a \# \lambda t} \qquad \frac{a \# M \quad a \# N}{a \# (M\ N)^n}$$

Note that we can test equality of any two atoms.

The judgment $a \# M$ just says that we can construct M without using a.

Abstraction by an atom

We define $\boldsymbol{\lambda}_{(-)} : \mathbb{A} \times \mathbb{L} \to \mathbb{L}^1$ as follows. $\boldsymbol{\lambda}_a M$ gives an abstract obtained from M by abstacting a in M

1. $\boldsymbol{\lambda}_a b := \begin{cases} 1 & \text{if } a = b \\ \lambda b & \text{if } a \neq b \end{cases}$
2. $\boldsymbol{\lambda}_a k' := \lambda k'$.
3. $\boldsymbol{\lambda}_a \lambda t := \lambda \boldsymbol{\lambda}_a t$.
4. $\boldsymbol{\lambda}_a (M\ N)^{n} := (\boldsymbol{\lambda}_a M\ \boldsymbol{\lambda}_a N)^{n'}$.

Note that $\boldsymbol{\lambda}_a M = \lambda M$ iff $a \# M$.

Translation from Λ to $\mathbb{L}$

Here, we define the set of raw $\boldsymbol{\lambda}$-terms by the following abstract syntax. We assume that $\mathbb{X}$ is a set of variables disjoint from $\mathbb{A}$. We also assume that we have a fixed injection $\mathbb{X} \ni \boldsymbol{x} \mapsto \boldsymbol{a} \in \mathbb{A}$ from $\mathbb{X}$ to $\mathbb{A}$.

$$\begin{aligned} \mathbb{X} &\ni \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \ldots \\ \boldsymbol{\Lambda} &\ni \boldsymbol{M}, \boldsymbol{N}, \boldsymbol{P} ::= \boldsymbol{x} \mid \boldsymbol{\lambda}_{\boldsymbol{x}} \boldsymbol{M} \mid (\boldsymbol{M}\ \boldsymbol{N}) \end{aligned}$$

We define the translation function $[-] : \boldsymbol{\Lambda} \to \mathbb{L}$ as follows.

1. $[\boldsymbol{x}] := \boldsymbol{a}$ where $\boldsymbol{x} \mapsto \boldsymbol{a}$.
2. $[\boldsymbol{\lambda}_{\boldsymbol{x}} \boldsymbol{M}] := \boldsymbol{\lambda}_{[\boldsymbol{x}]} [\boldsymbol{M}]$.
3. $[(\boldsymbol{M}\ \boldsymbol{N})] := ([\boldsymbol{M}]\ [\boldsymbol{N}])$.