Weakening in Simple Type Theory

Masahiko Sato

Graduate School of Informatics, Kyoto University

McGill University, October 18, 2017

Summary

We formulate the Curry-style simple type theory using the map/skeleton representation of untyped lambda terms introduced in [Sato et al. 2013].

We illustarate the usefulness of our approach by showing the admissiblity of the weakening rule.

[Sato et al. 2013] Sato, M., Pollack, R., Schwichtenberg, H. and Sakurai, T., Viewing $\boldsymbol{\lambda}$-terms through maps, *Indag. Math.*, **24**, 1073 – 1104, 2013.

We have formally verified all the technical results in the above paper in the proof assistants Minlog and Isabelle.

Good points of our approach

- The inductive structure of the terms is nicer compared to other approaches.
- Can define closed lambda terms directly without first defining the lambda terms containing free parameters.
- Can use the same technique to define sentences without first defining formulas containing free parameters.
- A special generic constant $\square$ must be included as a term, however.

Part I

Untyped Lambda-terms

Summary of Part I

Two datatypes

We will relate the two datatypes $\mathbf{\Lambda}$ of tradtional raw lambda-terms and $\mathbb{L}$ of our datetype based on map/skeleton.

$$\mathbf{\Lambda} = \text{The datatype of raw } \boldsymbol{\lambda}\text{-terms.}$$

$$\mathbb{L} = \text{The datatype of lambda-expressions.}$$

Two types of abstractions

$$\mathbf{\Lambda} : \text{abstraction by parameters } \boldsymbol{x} \in \mathbb{X}.$$

$$\mathbb{L} : \text{abstraction by maps } \boldsymbol{m} \in \mathbb{M}.$$

Summary of Part I (cont.)

$$
\begin{aligned}
K, L \in \Lambda &::= x \mid \Box \mid \mathsf{app}(K, L) \mid \mathsf{lam}(x, K).\\
M, N \in \mathbb{L} &::= x \mid \Box \mid \mathsf{app}(M, N) \mid \mathsf{mask}(m, M) \quad (m \mid M).\\
x &\in \mathbb{X}.\\
m &\in \mathbb{M}.
\end{aligned}
$$

$\Box$ is a special constant denoting a hole to be filled with lambda expressions.

The notion of map is a generalization of the notion of occurrence of a symbol in syntactic expressions such as formulas or lambda terms.

Plan of the talk

- Part I.1. $\mathbb{L}$.
- Part I.2. $\mathbf{\Lambda}$. Will show $\mathbb{L} \simeq \mathbf{\Lambda}/\equiv_{\boldsymbol{\alpha}}$.
- Part II. Simple type theory

map/skeleton functions will play important roles in all the 3 parts.

Part I.1

$\mathbb{L}$

The Datatype of Lambda-exressions

The Datatype $\mathbb{M}$ of Maps

$$\frac{}{0 \in \mathbb{M}} \qquad \frac{}{1 \in \mathbb{M}}$$

$$\frac{\boldsymbol{m} \in \mathbb{M} \quad \boldsymbol{n} \in \mathbb{M} \quad \boldsymbol{m} \neq 0 \text{ or } \boldsymbol{n} \neq 0}{\mathsf{cons}(\boldsymbol{m}, \boldsymbol{n}) \in \mathbb{M}}$$

Note that

$$\mathsf{cons} : \mathbb{M} \times \mathbb{M} \to \mathbb{M}$$

is a partial function.
We will write $(\boldsymbol{m}\ \boldsymbol{n})$ or $\boldsymbol{mn}$ for $\mathsf{cons}(\boldsymbol{m}, \boldsymbol{n})$.
We will also write 0 for (0 0).

Orthogonality and order relations on maps

$$\frac{}{m \perp 0} \qquad \frac{}{0 \perp n} \qquad \frac{m \perp n \quad m' \perp n'}{mm' \perp nn'}$$

Example: $(1\ 0) \perp (0\ 1)$ but not $(1\ 1) \perp (0\ 1)$.

$$\frac{}{0 \leq n} \qquad \frac{}{1 \leq 1} \qquad \frac{m \leq n \quad m' \leq n'}{mm' \leq nn'}$$

The Datatype $\mathbb{X}$ of Parameters

We assume a countably infinite set $\mathbb{X}$ of parameters.
We will write x, y, z for parameters.
We assume that equality relation on $\mathbb{X}$ is decidable.

The Datatype $\mathbb{L}$ and the Divisibility Relation

$$\frac{}{x \in \mathbb{L}}\ \mathit{par} \qquad \frac{}{\square \in \mathbb{L}}\ \mathit{box}$$

$$\frac{M \in \mathbb{L} \quad N \in \mathbb{L}}{\mathsf{app}(M, N) \in \mathbb{L}}\ \mathsf{app} \qquad \frac{m \in \mathbb{M} \quad M \in \mathbb{L} \quad m \mid M}{\mathsf{mask}(m, M) \in \mathbb{L}}\ \mathsf{mask}$$

$$\frac{}{0 \mid x} \qquad \frac{}{0 \mid \square} \qquad \frac{}{1 \mid \square}$$

$$\frac{m \mid M \quad n \mid N}{\mathsf{mapp}(m, n) \mid \mathsf{app}(M, N)} \qquad \frac{m \mid N \quad n \mid N \quad m \perp n}{m \mid \mathsf{mask}(n, N)}$$

The Datatype $\mathbb{L}$ of lambda-expressions (cont.)

Notational Convention

- We use M, N, P as metavariables ranging over lambda-expressions.
- We write $(M\ N)$ and also MN for $\mathsf{app}(M, N)$.
- We write $m\backslash M$ for $\mathsf{mask}(m, M)$.
- A lambda-expression of the form $m\backslash M$ is called an abstract.
- We use A, B as metavariables ranging over abstarcts, and write $\mathbb{A}$ for the subset of $\mathbb{L}$ consisting of all the abstracts.

map : $\mathbb{X} \times \mathbb{L} \to \mathbb{M}$ and skel : $\mathbb{X} \times \mathbb{L} \to \mathbb{L}$

We write M_x for map(x, M), and M^x for skel(x, M).

$$
\begin{aligned}
y_x &:= \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases} \\
\square_x &:= 0 \\
(M\ N)_x &:= (M_x\ N_x). \\
(m\backslash M)_x &:= M_x.
\end{aligned}
$$

$$
\begin{aligned}
y^x &:= \begin{cases} \square & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases} \\
\square^x &:= \square \\
(M\ N)^x &:= (M^x\ N^x). \\
(m\backslash M)^x &:= m\backslash M^x.
\end{aligned}
$$

Lambda Abstraction in $\mathbb{L}$

We define lam $: \mathbb{X} \times \mathbb{L} \to \mathbb{L}$ by:

$$\mathsf{lam}(x, M) := M_x \backslash M^x.$$

Examples. We assume that x, y and z are distinct parameters.

$$\begin{aligned}
\mathsf{lam}(x, x) &= 1\backslash\Box. \\
\mathsf{lam}(x, y) &= 0\backslash y. \\
\mathsf{lam}(x, \mathsf{lam}(y, x)) &= \mathsf{lam}(x, 0\backslash x) \\
&= 1\backslash 0\backslash\Box. \\
\mathsf{lam}(x, \mathsf{lam}(y, y)) &= \mathsf{lam}(x, \mathsf{lam}(1, \Box)) \\
&= 0\backslash 1\backslash\Box. \\
\mathsf{lam}(x, \mathsf{lam}(y, \mathsf{lam}(z, (xz\ yz)))) &= \\
&(10\ 00)\backslash(00\ 10)\backslash(01\ 01)\backslash(\Box\Box\ \Box\Box)
\end{aligned}$$

Instantiation

We define the instantiation operation $\blacktriangledown : \mathbb{A} \times \mathbb{L} \to \mathbb{L}$ as follows.

$$
\begin{aligned}
0\backslash M \blacktriangledown P &:= M \\
1\backslash \Box \blacktriangledown P &:= P \\
(m\ n)\backslash(M\ \ N) \blacktriangledown P &:= (m\backslash M \blacktriangledown P\ n\backslash N \blacktriangledown P) \\
m\backslash n\backslash N \blacktriangledown P &:= n\backslash(m\backslash N \blacktriangledown P)
\end{aligned}
$$

Remark. We remark that for any fresh x, we have:

$$m\backslash M = \mathsf{lam}(x, m\backslash M \blacktriangledown x).$$

Moreover, putting $N := m\backslash M \blacktriangledown x$, we have $m\backslash M = N_x\backslash N^x$. Namely, $m = N_x$ and $M = N^x$.

Substitution

We can now define substitution operation:
subst : $\mathbb{L} \times \mathbb{X} \times \mathbb{L} \to \mathbb{L}$ as follows.

$$[P/x]M := \mathsf{lam}(x, M) \blacktriangledown P.$$

subst enjoys the following property.

$$[P/x]y = \begin{cases} P & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases}$$

$$[P/x]\Box = \Box.$$

$$[P/x](M\ N) = ([P/x]M\ [P/x]N).$$

$$[P/x](m \backslash M) = (m \backslash [P/x]M).$$

Substitution (cont.)

Example.

$$
\begin{aligned}
[y/x]\mathsf{lam}(y, yx) &= [y/x](10\backslash\Box x) \\
&= 10\backslash[y/x](\Box\ x) \\
&= 10\backslash([y/x]\Box\ [y/x]x) \\
&= 10\backslash\Box y \\
&= \mathsf{lam}(z, zy)
\end{aligned}
$$

Remark. By internalizing the substitution operation, we can easily get an explicit substitution calculus.

Substitution Lemma

If $x \neq y$ and $x \notin \mathsf{FP}(P)$, then

$$[P/y][N/x]M = [[P/y]N/x][P/y]M.$$

Proof. By induction on $M \in \mathbb{L}$. Here, we only treat the case where $M = m\backslash M'$.

$$\begin{aligned}
&[P/y][N/x]M \\
&= [P/y][N/x](m\backslash M') \\
&= m\backslash[P/y][N/x]M' \\
&= m\backslash[[P/y]N/x][P/y]M' && \text{(by IH)} \\
&= [[P/y]N/x][P/y](m\backslash M') \\
&= [[P/y]N/x][P/y]M.
\end{aligned}$$

Substitution (cont.)

We can develop a theory of lambda-calculus on the datatype $\mathbb{L}$ of lambda expressions.

It is remarkable that the theory can be develped without using the notions of variable, lambda abstraction and substitution.

For example, the β-reduction rule is defined by:

$$(m\backslash M\ P) \to_{\beta} m\backslash M \blacktriangledown P$$

without mentioning variables, lambda abstraction and substitution.

Note that in the traditional lambda calculus, the β-rule is:

$$(\lambda_x M\ P) \to_{\beta} [P/x]M$$

Part I.2

$\mathbf{\Lambda}$

The Datatype of Raw Lambda-terms

The Datatype Λ of Raw λ-terms

$$\frac{}{\boldsymbol{x} \in \boldsymbol{\Lambda}}\ \textit{par} \qquad \frac{}{\square \in \boldsymbol{\Lambda}}\ \textit{box}$$

$$\frac{\boldsymbol{K} \in \boldsymbol{\Lambda} \quad \boldsymbol{L} \in \boldsymbol{\Lambda}}{\mathsf{app}(\boldsymbol{K}, \boldsymbol{L}) \in \boldsymbol{\Lambda}}\ \mathsf{app} \qquad \frac{\boldsymbol{x} \in \mathbb{X} \quad \boldsymbol{K} \in \boldsymbol{\Lambda}}{\mathsf{lam}(\boldsymbol{x}, \boldsymbol{K}) \in \boldsymbol{\Lambda}}\ \mathsf{lam}$$

$$\boldsymbol{K}, \boldsymbol{L} \in \boldsymbol{\Lambda} ::= \boldsymbol{x} \mid \square \mid \mathsf{app}(\boldsymbol{K}, \boldsymbol{L}) \mid \mathsf{lam}(\boldsymbol{x}, \boldsymbol{K}).$$

Remark. lam binds parameter $\boldsymbol{x}$ in $\boldsymbol{M}$.

$\mathsf{map} : \mathbb{X} \times \Lambda \to \mathbb{M}$ and $\mathsf{skel} : \mathbb{X} \times \Lambda \to \Lambda$

$$y_x := \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases}$$

$$\square_x := 0.$$

$$(K\ L)_x := (K_x\ L_x).$$

$$\mathsf{lam}(y, K)_x := \begin{cases} 0 & \text{if } x = y, \\ K_x & \text{if } x \neq y. \end{cases}$$

$$y^x := \begin{cases} \square & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases}$$

$$\square^x := \square.$$

$$(K\ L)^x := (K^x\ L^x).$$

$$\mathsf{lam}(y, K)^x := \begin{cases} \mathsf{lam}(y, K) & \text{if } x = y, \\ \mathsf{lam}(y, K^x) & \text{if } x \neq y. \end{cases}$$

Map and Skeleton (cont.)

$$
\begin{aligned}
& \boldsymbol{x} \text{ does not occur free in } \boldsymbol{K} \\
\Longleftrightarrow & \boldsymbol{K}_{\boldsymbol{x}} = 0 \\
\Longleftrightarrow & \boldsymbol{K}^{\boldsymbol{x}} = \boldsymbol{K}
\end{aligned}
$$

Remark. This shows that the notion of map is a generalization of the notion of occurrence.

α-equivalence Relation

We define the α-equivalence relation, $=_\alpha$, using the map/skeleton functions.

$$\frac{}{x =_\alpha x} \qquad \frac{}{\Box =_\alpha \Box}$$

$$\frac{K =_\alpha K' \quad L =_\alpha L'}{KL =_\alpha K'L'} \qquad \frac{K_x = L_y \quad K^x =_\alpha L^y}{\mathsf{lam}(x, K) =_\alpha \mathsf{lam}(y, L)}$$

Remark. No renaming is needed in this definition, and it is easy to verify that this is indeed a decidable equivalence relation.

α-equivalence Relation

We can show that $\mathsf{lam}(x, \mathsf{lam}(y, yx)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(x, xy))$ as follows.

$$\dfrac{01 = 01 \qquad \dfrac{10 = 10 \qquad \dfrac{\dfrac{}{\square =_\alpha \square} \quad \dfrac{}{\square =_\alpha \square}}{\square\square =_\alpha \square\square}}{\mathsf{lam}(y, y\square) =_\alpha \mathsf{lam}(x, x\square)}}{\mathsf{lam}(x, \mathsf{lam}(y, yx)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(x, xy))}$$

Interpretation of Λ in $\mathbb{L}$

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

$$\begin{aligned} [\![x]\!] &:= x. \\ [\![\square]\!] &:= \square. \\ [\![KL]\!] &:= [\![K]\!][\![L]\!]. \\ [\![\mathsf{lam}(x, K)]\!] &:= \mathsf{lam}(x, [\![K]\!]). \end{aligned}$$

Remark. Two raw $\boldsymbol{\lambda}$-terms K and L are α-equivalent iff $[\![M]\!] = [\![N]\!]$.

Part II

Simple Type Theory

We will work in $\mathbb{L}$ and write $\boldsymbol{\lambda}_x M$ for $M_x \backslash M^x$.

Simple types

Simple types are defined by the grammar:

$$\sigma, \rho ::= \iota \mid \rho \to \sigma,$$

where ι ranges over a set of base type symbols.

A typing context is a finite sequence of type assignments $x_i : \sigma_i$:

$$\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n,$$

where $n \geq 0$, and x_i are all distinct from each other.

We put

$$\mathrm{Vars}(\Gamma) := \{x_1, \ldots, x_n\}.$$

Simple type theory in traditional form

The judgments we treat in the traditional calculus are of the form:

$$\Gamma \vdash_\lambda M : \sigma$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_\lambda x : \sigma}\ (\text{Ini}) \qquad \frac{\Gamma \vdash_\lambda M : \rho \to \sigma \quad \Gamma \vdash_\lambda N : \rho}{\Gamma \vdash_\lambda (M\ N) : \sigma}\ (\to\text{E})$$

$$\frac{\Gamma, x : \rho \vdash_\lambda M : \sigma}{\Gamma \vdash_\lambda \lambda_x M : \rho \to \sigma}\ (\to\text{I})$$

Remark.

1. In the (Ini) rule, if x is declared more than once in Γ, then we use the right-most type assignment $x : \sigma$.
2. If $\Gamma \vdash_\lambda M : \sigma$ is derivable then M is $\Box$-free. In particular, $\Gamma \vdash_\lambda \Box : \sigma$ is not derivable.

Simple type theory in traditional form (cont.)

We note that the (→I) rule is an abbreviation of the following rule:

$$\frac{\Gamma, x : \rho \vdash_\lambda M : \sigma}{\Gamma \vdash_\lambda M_x \backslash M^x : \rho \to \sigma} \ (\to\mathsf{I})$$

Putting $A := M_x \backslash M^x$, we have

$$\frac{\Gamma, x : \rho \vdash_\lambda A \blacktriangledown x : \sigma}{\Gamma \vdash_\lambda A : \rho \to \sigma} \ (\to\mathsf{I})$$

So, we can reformulate the type theory as follows.

Simple type theory in traditional form (cont.)

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_\lambda x : \sigma} \text{(Ini)} \qquad \frac{\Gamma \vdash_\lambda M : \rho \to \sigma \quad \Gamma \vdash_\lambda N : \rho}{\Gamma \vdash_\lambda (M\ N) : \sigma} (\to\text{E})$$

$$\frac{\Gamma, x : \rho \vdash_\lambda A \blacktriangledown x : \sigma}{\Gamma \vdash_\lambda A : \rho \to \sigma} (\to\text{I}) \qquad \frac{\Gamma, x : \rho \vdash_\lambda M : \sigma}{\Gamma \vdash_\lambda \lambda_x M : \rho \to \sigma} (\to\text{I})$$

Remark. The conclusion of the original $(\to\text{I})$ rule mentions x, but the new form of the rule does not. Moreover, we have the following theorem.

Theorem

$\Gamma \vdash_\lambda A : \rho \to \sigma$

$\iff \Gamma, x : \rho \vdash_\lambda A \blacktriangledown x : \sigma$ *for some/any x not in* $\mathrm{Vars}(\Gamma)$.

Admissibility of the weakening rule

Proposition

$$\Gamma \vdash_\lambda M : \sigma,\ x \notin \mathrm{Vars}(\Gamma) \Longrightarrow \Gamma, x : \rho \vdash_\lambda M : \sigma.$$

Proof by induction on the derivation (in the traditonal syntax) of $\Gamma \vdash_\lambda M : \sigma$ fails. Consider the case where $M = \lambda_x M'$ and the bottom part of the derivation is:

$$\frac{\Gamma, x : \rho \vdash_\lambda M' : \sigma'}{\Gamma \vdash_\lambda \lambda_x M' : \rho \to \sigma'}.$$

We cannot apply IH in this case. We have to prove by induction on the size of the derivation.

Frege-style formulation of simple type theory

Let M be a hole-free lambda-term such that $\mathrm{Vars}(M)$ is, say, $\{x_1, x_2\}$.

Suppose further that

$$x_1 : \rho_1, x_2 : \rho_2 \vdash_\lambda M : \sigma.$$

Then we have

$$\vdash_\lambda m_1 \backslash m_2 \backslash N \ : \ \rho_1 \to \rho_2 \to \sigma,$$

where $m_i = M_{x_i}$ and $N = (M^{x_2})^{x_1}$.

Conversely, if we have the latter judgment we also have the former judgement. Thus, in general, type-assignments for open terms can be reduced to those for closed terms.

Frege-style simple simple type theory for closed terms

$$\frac{|\vec{u}| = |\vec{\mu}| \quad \vec{u}_i = 1 \quad \vec{\mu}_i = \sigma}{\vdash \vec{u}\backslash\Box : \vec{\mu} \to \sigma} \text{ HOLE}$$

$$\frac{\vdash \vec{m}\backslash M : \vec{\mu} \to \nu \to \sigma \quad \vdash \vec{n}\backslash N : \vec{\mu} \to \nu \quad |\vec{m}| = |\vec{n}| = |\vec{\mu}|}{\vdash (\vec{m}\ \vec{n})\backslash(M\ N) : \vec{\mu} \to \sigma} \text{ APP}$$

In the HOLE rule $\vec{u}$ is a unit sequence. That is, $\vec{u}$ is a sequence of 0 and 1 containing exactly one occurrence of 1.

An example: S combinator

We show an example of type-assignment for the S combinator $\boldsymbol{\lambda}_{xyz}(xz\ yz)$. We write μ_x, μ_y and μ_z for the types of x, y and z. We write σ for the type of $(xz\ yz)$ and write ν for the type introduced by the bottom application of the APP rule. We also write $\vec{\mu} \to \sigma$ for $\mu_x \to \mu_y \to \mu_z \to \sigma$.

$$\dfrac{\dfrac{\dfrac{}{1\backslash 0\backslash 0 : \vec{\mu} \to \mu_x} \quad \dfrac{}{0\backslash 0\backslash 1 : \vec{\mu} \to \mu_z}}{10\backslash 00\backslash 01 : \vec{\mu} \to \nu \to \sigma} \qquad \dfrac{\dfrac{}{0\backslash 1\backslash 0 : \vec{\mu} \to \mu_y} \quad \dfrac{}{0\backslash 0\backslash 1 : \vec{\mu} \to \mu_z}}{00\backslash 10\backslash 01 : \vec{\mu} \to \nu}}{(10\ 00)\backslash(00\ 10)\backslash(01\ 01) : \vec{\mu} \to \sigma}$$

By analyzing the top left application of APP, we have $\mu_x = \mu_z \to \nu \to \sigma$, and from the top right application of APP, we have $\mu_y = \mu_z \to \nu$. Writing μ for μ_z, we have:

$$\mu_x = \mu \to \nu \to \sigma,\ \mu_y = \mu \to \nu,\ \mu_z = \mu.$$

An example: S combinator (cont.)

Using the $\boldsymbol{\lambda}$-notation, the same derivation can be written as follows.

$$
\dfrac{
  \dfrac{
    \dfrac{}{\lambda_{xyz} x : \vec{\mu} \to \mu_x}
    \quad
    \dfrac{}{\lambda_{xyz} z : \vec{\mu} \to \mu_z}
  }{\lambda_{xyz} xz : \vec{\mu} \to \nu \to \sigma}
  \qquad
  \dfrac{
    \dfrac{}{\lambda_{xyz} x : \vec{\mu} \to \mu_y}
    \quad
    \dfrac{}{\lambda_{xyz} z : \vec{\mu} \to \mu_z}
  }{\lambda_{xyz} yz : \vec{\mu} \to \nu}
}{\lambda_{xyz} (xz\ yz) : \vec{\mu} \to \sigma}
$$

A final remark

Hilbert style formulation of the minimal implicational logic can be obtained as follows.

$$\frac{\vec{\mu}_i = \sigma}{\vec{\mu} \to \sigma}\ \text{Axiom} \qquad \frac{\vec{\mu} \to \nu \to \sigma \quad \vec{\mu} \to \nu}{\vec{\mu} \to \sigma}\ \text{MP}$$

For example, we have:

A final remark

Hilbert style formulation of the minimal implicational logic can be obtained as follows.

$$\frac{\vec{\mu}_i = \sigma}{\vec{\mu} \to \sigma}\ \text{Axiom} \qquad \frac{\vec{\mu} \to \nu \to \sigma \quad \vec{\mu} \to \nu}{\vec{\mu} \to \sigma}\ \text{MP}$$

For example, we have:

$$\dfrac{\dfrac{\overline{\vec{\mu} \to \mu \to \nu \to \sigma} \quad \overline{\vec{\mu} \to \mu}}{\vec{\mu} \to \nu \to \sigma} \quad \dfrac{\overline{\vec{\mu} \to \mu \to \nu} \quad \overline{\vec{\mu} \to \mu}}{\vec{\mu} \to \nu}}{\vec{\mu} \to \sigma},$$

where

$$\vec{\mu} \to * = (\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to *$$

for $* = \mu \to \nu \to \sigma, \mu \to \nu$ or μ are axioms.

A final remark (cont.)

Or, equivalently,

1. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to (\mu \to \nu \to \sigma)$ (Axiom)
2. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to (\mu \to \nu)$ (Axiom)
3. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to \mu$ (Axiom)
4. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to \nu \to \sigma$ (MP 1, 3)
5. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to \nu$ (MP 2, 3)
6. $(\mu \to \nu \to \sigma) \to (\mu \to \nu) \to \mu \to \sigma$ (MP 4, 5)