Theorems about typed lambda calculus

Henk Barendregt

Radboud University
Nijmegen, The Netherlands

Let $\mathbb{A}$ be a set of symbols: atomic types

Types over $\mathbb{A}$, notation $\mathbb{T} = \mathbb{T}_{\to}^{\mathbb{A}}$:

$$\boxed{\mathbb{T} \quad = \quad \mathbb{A} \mid \mathbb{T} \to \mathbb{T}}$$

Type assignment $\lambda_{\to}^{\mathbb{A},\mathsf{cu}} = \lambda_{\to}^{\mathsf{cu}} = \lambda_{\to}^{\mathbb{A}} = \lambda_{\to}$ "Simply typed lambda calculus"

Bases: $\Gamma = \{x_1{:}A_1, \cdots, x_n{:}A_n\}$

Statements: $M : A$, with $M \in \Lambda$ and $A \in \mathbb{T}_{\to}^{\mathbb{A}}$

$$\text{(axiom)} \quad \Gamma \vdash x : A, \text{ if } (x{:}A) \in \Gamma$$

$$(\to\mathrm{E}) \quad \frac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B} \qquad (\to\mathrm{I}) \quad \frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)}$$

Write $\vdash M : A$ for $\emptyset \vdash M : A$

Note that $\vdash \mathsf{I} : \alpha \to \alpha$, but also $\vdash \mathsf{I} : A \to A$, $\vdash \mathsf{I} : (A \to B) \to (A \to B)$ for all $A, B \in \mathbb{A}$

We assume that $o \in \mathbb{A}$. We write $A \to B \to C$ for $A \to (B \to C)$ (association to the right).

3.1. Definition. (i) The following types in $\mathbb{T}^{\mathbb{A}}$ are often used.

$$0 \triangleq o,\ 1 \triangleq 0 \to 0,\ 2 \triangleq (0 \to 0) \to 0,\ \ldots.$$

In general

$$0 \triangleq o \text{ and } k+1 \triangleq k \to 0.$$

(ii) Define $A^k \to B$ and n_k by cases on n

$$\begin{array}{rcl} A^0 \to B & \triangleq & B \\ A^{k+1} \to B & \triangleq & A \to A^k \to B \end{array} \qquad \begin{array}{rcl} o_k & \triangleq & o; \\ (n+1)_k & \triangleq & n^k \to o. \end{array}$$

For example

$$\begin{array}{rcl} 1_0 & \equiv & o; \\ 1 \equiv 1_1 & \equiv & o \to o \\ 1_2 & \equiv & o \to o \to o; \\ 2_3 & \equiv & 1 \to 1 \to 1 \to o; \\ 1^2 \to 2 \to o & \equiv & (o \to o) \to (o \to o) \to ((o \to o) \to o) \to o. \end{array}$$

3.2. Definition. Let $M \in \Lambda$

(i) An *M-reduction* is of the form $M \equiv M_0 \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots$

(ii) An M-reduction is *terminating* if it is finite and the last element is in β-nf

(iii) $M \in \mathsf{WN}$ (M is *weakly normalizing*) if there is a terminating M-reduction

(iv) $M \in \mathsf{SN}$ (M is *strongly normalizing*) if all M-reductions are terminating

Note that trivially one has $\mathsf{SN} \subseteq \mathsf{WN}$

Note also that $\emptyset \subsetneq \mathsf{SN} \subsetneq \mathsf{WN} \subsetneq \Lambda$ as $\Omega \in \Lambda - \mathsf{WN}$, $(\mathbf{false}\,\Omega) \in \mathsf{WN} - \mathsf{SN}$, and $(\mathsf{I}\,\mathsf{I}) \in \mathsf{SN}$

The interesting term $\mathbf{false}\,\Omega$ (draw its reduction graph) has a subterm $\notin \mathsf{WN}$

More interesting is $(\lambda x.\mathbf{false}(xx))(\lambda x.\mathbf{false}(xx)) \notin \mathsf{SN}$, with each subterm $\in \mathsf{WN}$

Note that $\mathsf{Y}\mathbf{false} =_\beta (\lambda x.\mathbf{false}(xx))(\lambda x.\mathbf{false}(xx))$

3.3. Theorem (Turing, $\pm$1952). $\Gamma \vdash M : A \Rightarrow M \in \mathsf{WN}$

3.4. Theorem (Tait, 1967). $\Gamma \vdash M : A \Rightarrow M \in \mathsf{SN}$

For $A \in \mathbb{TT}^{\mathbb{A}}$ write A^* for the result of substituting types for the type variables in A

Note that if $\Gamma \vdash M : A$, then $\Gamma^* \vdash M : A^*$

3.5. THEOREM (Hindley; Milner). *There is (after coding) a computable function* pp *such that* $\mathrm{pp}(M) = (\Gamma, A)$ *or* $\mathrm{pp}(M) = \mathtt{fail}$ *such that for all* $M \in \Lambda$

(i) *If* $\mathrm{pp}(M) = (\Gamma, A)$, *then* $\Gamma \vdash M : A$ *and*

$\Gamma' \vdash M : A' \Rightarrow A = \mathrm{pp}(M)^* \ \&\ \Gamma' \supseteq \Gamma^*$ *for some* $^* = [\alpha_1 := A_1, \cdots, \alpha_n := A_n]$

(ii) *If* $\Gamma \vdash M : A$ *for no* Γ, A, *then* $\mathrm{pp}(M) = \mathtt{fail}$

3.6. COROLLARY. *There is a computable function* pt *such that for* $M \in \Lambda^{\emptyset}$

(i) *If* $\mathrm{pt}(M) = A$, *then* $\vdash M : A$ *and* $\vdash M : A' \Rightarrow A' = \mathrm{pt}(M)^*$ *for some* *

(ii) *If* $\vdash M : A$ *for no* A, *then* $\mathrm{pt}(M) = \mathtt{fail}$

3.7. THEOREM (Curry: Subject Reduction). $\vdash M : A \ \&\ M \twoheadrightarrow N \Rightarrow \vdash N : A$
("type checking needed only at compile time")

The pure *functional programming* languages Haskell and Clean are based on $\lambda^{\mathbb{A},\mathsf{cu}}_{\rightarrow}$ with the addition of $\mathsf{Y} : (A \rightarrow A) \rightarrow A$ to represent search. Then WN and SN is no longer valid. A less pure functional programming language is ML, that is based on a similar extension of $\lambda^{\mathbb{A},\mathsf{cu}}_{\rightarrow}$ but also has assignments like $[x := x+1]$. Untyped functional programming language are LISP (with confusion between free and bound variables and and its correction Scheme.

3.8. Definition. (i) For $A\in\mathbb{TT}^{\mathbb{A}}$ we define $\Lambda(A)$ as follows:

$$\begin{array}{rcl} & & x^A\in\Lambda(A) \\ M\in\Lambda(A\to B), N\in\Lambda(A) & \Rightarrow & (MN)\in\Lambda(B) \\ M\in\Lambda(B) & \Rightarrow & (\lambda x^A.M)\in\Lambda(A\to B) \end{array}$$

(ii) As before $=_\beta$ is axiomatized by $(\lambda x^A.M)N\to M[x:=N]$

3.9. Example. $\mathsf{I}_A\equiv(\lambda x^A.x^A)\in\Lambda(A\to A)$ and
$\mathsf{I}_{A\to B}\equiv(\lambda x^{A\to B}.x^{A\to B})\in\Lambda((A\to B)\to(A\to B))$ differ

3.10. Definition. For $M\in\bigcup_A\Lambda(A)$ define $|M|\in\Lambda$:

$$\begin{array}{rcl} |x^A| & = & x \\ |MN| & = & |M\ |N| \\ |\lambda x^A.M| & = & \lambda x.|M| \end{array}$$

3.11. Proposition. (i) $M\in\Lambda(A)\ \&\ M\in\Lambda(B)\Rightarrow A=B$
(ii) $M\in\Lambda(A)\ \&\ M\twoheadrightarrow_\beta N\Rightarrow N\in\Lambda(A)$
(iii) $M\in\Lambda(A)\Rightarrow\Gamma_M\vdash_{\lambda_{\to}^{\mathsf{cu}}}|M|:A$, *where* $\Gamma_M=\{x{:}A\mid x^A\in\mathrm{FV}(M)\}$
(iv) $\Gamma\vdash_{\lambda_{\to}^{\mathsf{cu}}}M:A\Rightarrow\exists M'\in\Lambda(A).|M'|\equiv M\ \&\ \Gamma_{M'}\subseteq\Gamma$

3.12. THEOREM (Tait 1967). $M{\in}\Lambda(A) \Rightarrow M{\in}\mathsf{SN}$

PROOF. We use an induction loading. First we add to $\lambda^{\mathbb{A}}_{\to}$ constants $d_\alpha{\in}\Lambda(\alpha)$ for each atom α, obtaining $\lambda^{\mathsf{ch}\,+}_{\to}$. Then we prove SN for the extended system. It follows *a fortiori* that the system without the constants is SN.

Define for $A{\in}\mathbb{T}^{\mathbb{A}}$ the following class $\mathcal{C}_A$ of *computable* terms of type A.

$$\begin{array}{rcl} \mathcal{C}_\alpha & \triangleq & \{M{\in}\Lambda^\emptyset(\alpha) \mid \mathsf{SN}(M)\};\\ \mathcal{C}_{A\to B} & \triangleq & \{M{\in}\Lambda^\emptyset(A{\to}B) \mid \forall Q{\in}\mathcal{C}_A.MQ{\in}\mathcal{C}_B\};\\ \mathcal{C} & \triangleq & \bigcup\limits_{A\in\mathbf{T}^{\mathbb{A}}} \mathcal{C}_A. \end{array}$$

Then one defines the classes $\mathcal{C}^*_A$ of terms that are *computable under substitution*

$$\mathcal{C}^*_A \triangleq \{M{\in}\Lambda(A) \mid \forall \vec{P}{\in}\mathcal{C}.[M[\vec{x}{:=}\vec{P}]{\in}\Lambda^\emptyset(A) \Rightarrow M[\vec{x}{:=}\vec{P}]{\in}\mathcal{C}_A]\}.$$

Write $\mathcal{C}^* \triangleq \bigcup\{\mathcal{C}^*_A \mid A{\in}\mathbb{T}^{\mathbb{A}}\}$. For $A \equiv A_1{\to}\ldots{\to}A_n{\to}\alpha$ define

$$d_A \triangleq \lambda x_1^{A_1}\ldots\lambda x_n^{A_n}.d_\alpha.$$

Then for A one has

$$M{\in}\mathcal{C}_A \iff \forall \vec{Q}{\in}\mathcal{C}.M\vec{Q}{\in}\mathsf{SN}, \tag{0}$$

$$M{\in}\mathcal{C}^*_A \iff \forall \vec{P},\vec{Q}{\in}\mathcal{C}.M[\vec{x}{:=}\vec{P}]\vec{Q}{\in}\mathsf{SN}, \tag{1}$$

where the $\vec{P}, \vec{Q}$ should have the right types and $M\vec{Q}$ and $M[\vec{x}:=\vec{P}]\vec{Q}$ are of type α, respectively. By an easy simultaneous induction on A one can show

$$M \in \mathcal{C}_A \Rightarrow \mathsf{SN}(M); \tag{2}$$

$$d_A \in \mathcal{C}_A. \tag{3}$$

In particular, since $M[\vec{x}:=\vec{P}]\vec{Q} \in \mathsf{SN} \Rightarrow M \in \mathsf{SN}$, it follows that

$$M \in \mathcal{C}^* \Rightarrow M \in \mathsf{SN}. \tag{4}$$

Now one shows by induction on M, distinguishing cases and using (1), that

$$M \in \Lambda(A) \Rightarrow M \in \mathcal{C}^*_A. \tag{5}$$

Case $M \equiv x$. Then for $P, \vec{Q} \in \mathcal{C}$ one has $M[x:=P]\vec{Q} \equiv P\vec{Q} \in \mathcal{C} \subseteq \mathsf{SN}$, by the definition of $\mathcal{C}$ and (2). Case $M \equiv NL$. Easy. Case $M \equiv \lambda x.N$. Now $\lambda x.N \in \mathcal{C}^*$ iff for all $\vec{P}, Q, \vec{R} \in \mathcal{C}$

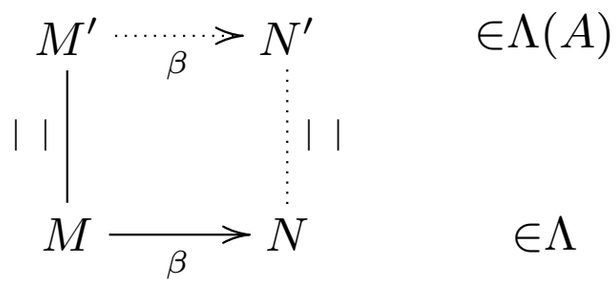
$$(\lambda x.N[\vec{y}:=\vec{P}])Q\vec{R} \in \mathsf{SN}. \tag{6}$$

By the IH one has $N \in \mathcal{C}^* \subseteq \mathsf{SN}$; therefore, if $\vec{P}, Q, \vec{R} \in \mathcal{C} \subseteq \mathsf{SN}$, then

$$N[x:=Q, \vec{y}:=\vec{P}]\vec{R} \in \mathsf{SN}. \tag{7}$$

Now every reduction σ starting from the term in (6) passes through a reduct of the term in (7), as reductions within $N, \vec{P}, Q, \vec{R}$ are finite, hence σ is finite. Therefore we have (6). Finally the result follows by (5) and (4). ■

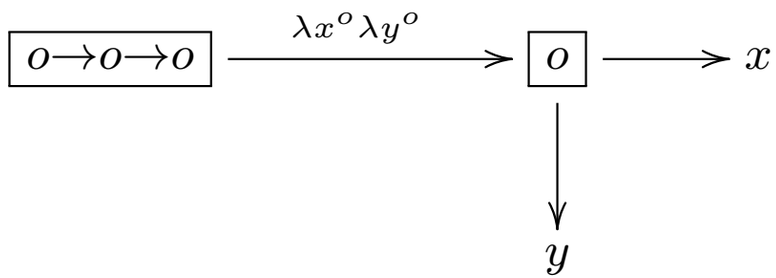
3.13. Lemma. *For $M, N \in \Lambda$, $M' \in \Lambda(A)$ such that $|M'| \equiv M \to_\beta N$ there exists an $N' \in \Lambda(A)$ such that $M' \to_\beta N'$ and $|N'| \equiv N$.*



3.14. Theorem. *Let $\Gamma \vdash M : A$. Then $M \in \mathsf{SN}$.*

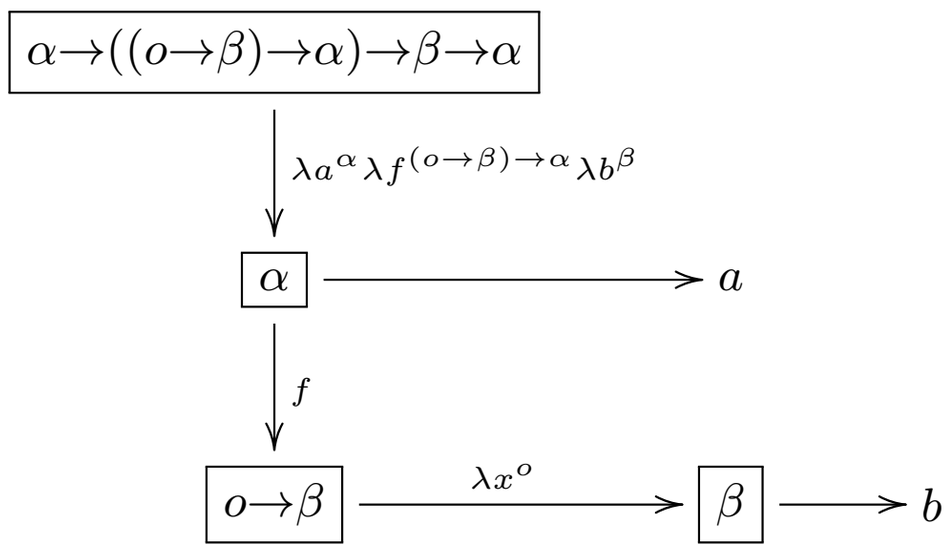
Proof. By Lemma 3.12 an infinite reduction in $\lambda^{\mathsf{cu}}_{\to}$ lifts to $\lambda^{\mathsf{ch}}_{\to}$ which is impossible by Theorem 3.11. ■

3.15. EXAMPLES. (i) $A = o \to o \to o$. Then M_A is



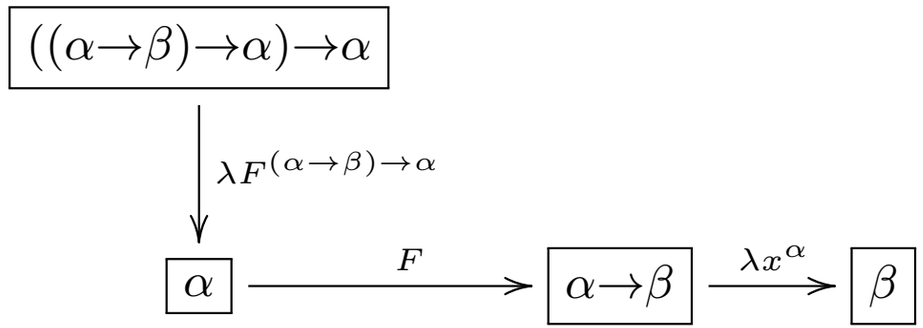
This shows that the type 1_2 has two closed inhabitants: $\lambda xy.x$ and $\lambda xy.y$. We see that the two arrows leaving $\boxed{o}$ represent a choice.

(ii) $A = \alpha \to ((o \to \beta) \to \alpha) \to \beta \to \alpha$. Then M_A is



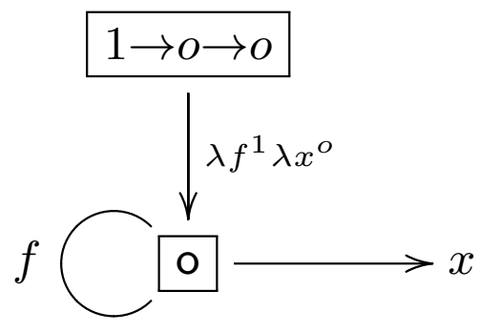
Again there are only two inhabitants: $\lambda afb.a$ and $\lambda afb.f(\lambda x^o.b)$.

(iii) $A = ((\alpha{\to}\beta){\to}\alpha){\to}\alpha$. Then M_A is



This type, corresponding to Peirce's law, does not have any inhabitants.

(iv) $A = 1{\to}o{\to}o$. Then M_A is



This is the type Nat having the Church's numerals $\lambda f^1 x^o . f^n x$ as inhabitants.

See <www.cs.ru.nl/~henk/Inhabitation.pdf> for more on inhabitation machines

On lambda calculus (untyped and typed)

`<http://ftp.cs.ru.nl/CompMath.Found/lambda.pdf>`
Lecture notes Barendregt-Barendsen on lambda calculus

The lambda calculus, its syntax and semantics, H Barendregt, College Publications, 2012. Reprint of the 1984 version. Book on untyped lambda calculus. ± 28E

Lambda Calculus with Types, H Barendregt, W Dekkers, R Statman, CUP, 2013. ± 50E

On functional programming

An Introduction to Functional Programming Through Lambda Calculus, G Michaelson, Dover Books on Mathematics, 2011.

Haskell: The Craft of Functional Programming (3rd Edition) S Thompson, International Computer Science Series, Addison-Wesley, 2011.

On computer mathematics

Interactive Theorem Proving and Program Development, Y Bertot, Springer, 2004.

Handbook of Practical Logic and Automated Reasoning, J Harrison, CUP, 2009