On Unification of Lambda Terms

Andrew Polonsky
jww Giulio Manzonetto

Types 2016, Novi Sad
May 28, 2016

Ω

Bruce Lercher (NDJFL 17-2, 1976) noticed that the term

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

is the only "pure cycle" in the lambda calculus: a term which reduces to itself in one step of beta reduction.

$$(\lambda v.X(v))Y = X(Y)$$

Ω

Bruce Lercher (NDJFL 17-2, 1976) noticed that the term

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

is the only "pure cycle" in the lambda calculus: a term which reduces to itself in one step of beta reduction.

$$\boxed{(\lambda v.X(v))Y = X(Y)}$$

Ω

$$(\lambda v.X(v))Y = X(Y) \tag{1}$$

Ω

$$(\lambda v.X(v))Y = X(Y) \tag{1}$$

- Since $X(Y)$ is an application, either $X(v) = v$, or $X(v)$ is an application.

Ω

$$(\lambda v.X(v))Y = X(Y) \tag{1}$$

- Since $X(Y)$ is an application, either $X(v) = v$, or $X(v)$ is an application.
- Suppose $X(v) = v$. Then $(\lambda v.v)Y = Y$, contradiction.

Ω

$$(\lambda v.X(v))Y = X(Y) \tag{1}$$

- Since $X(Y)$ is an application, either $X(v) = v$, or $X(v)$ is an application.
- Suppose $X(v) = v$. Then $(\lambda v.v)Y = Y$, contradiction.
- So $X(v) = X_1(v)X_2(v)$. Substituting into (1), we obtain

$$(\lambda v.X_1(v)X_2(v))Y = X_1(Y)X_2(Y)$$

Ω

$$(\lambda v.X(v))Y = X(Y) \tag{1}$$

- Since $X(Y)$ is an application, either $X(v) = v$, or $X(v)$ is an application.
- Suppose $X(v) = v$. Then $(\lambda v.v)Y = Y$, contradiction.
- So $X(v) = X_1(v)X_2(v)$. Substituting into (1), we obtain

$$(\lambda v.X_1(v)X_2(v))Y = X_1(Y)X_2(Y)$$

$$\boxed{\begin{aligned}(\lambda v.X_1(v)X_2(v)) &= X_1(Y)\\ Y &= X_2(Y)\end{aligned}}$$

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$

$$Y = X_2(Y) \tag{3}$$

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$

$$Y = X_2(Y) \tag{3}$$

- Regarding (3), either $X_2(v) = v$, or $v \notin X_2(v) = Y$. (Otherwise, $Y = X_2(Y) = X_2(X_2(Y)) = \cdots$ would be infinite.)

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$
$$Y = X_2(Y) \tag{3}$$

- Regarding (3), either $X_2(v) = v$, or $v \notin X_2(v) = Y$. (Otherwise, $Y = X_2(Y) = X_2(X_2(Y)) = \cdots$ would be infinite.)
- The same is true of X_1: If $X_1(y) = \lambda v.X_1(v)X_2'(y, v)$, then certainly X_1 is infinite. Otherwise, y occurs at some position $00{\star}p$, v occurs at position $0000{\star}p$, etc.

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$

$$Y = X_2(Y) \tag{3}$$

- Regarding (3), either $X_2(v) = v$, or $v \notin X_2(v) = Y$. (Otherwise, $Y = X_2(Y) = X_2(X_2(Y)) = \cdots$ would be infinite.)
- The same is true of X_1: If $X_1(y) = \lambda v.X_1(v)X_2'(y, v)$, then certainly X_1 is infinite. Otherwise, y occurs at some position $00{\star}p$, v occurs at position $0000{\star}p$, etc.
- The only possibility remaining is $X_1(v) = v$. Then (2) becomes

$$(\lambda v.vX_2(v)) = Y \tag{4}$$

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$
$$Y = X_2(Y) \tag{3}$$

- Regarding (3), either $X_2(v) = v$, or $v \notin X_2(v) = Y$. (Otherwise, $Y = X_2(Y) = X_2(X_2(Y)) = \cdots$ would be infinite.)
- The same is true of X_1: If $X_1(y) = \lambda v.X_1(v)X_2'(y, v)$, then certainly X_1 is infinite. Otherwise, y occurs at some position $00{\star}p$, v occurs at position $0000{\star}p$, etc.
- The only possibility remaining is $X_1(v) = v$. Then (2) becomes

$$(\lambda v.vX_2(v)) = Y \tag{4}$$

- $X_2(v) = v$ yields the solution $Y = \lambda v.vv$, $X(Y) = \Omega$.

Ω

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y) \tag{2}$$
$$Y = X_2(Y) \tag{3}$$

- Regarding (3), either $X_2(v) = v$, or $v \notin X_2(v) = Y$. (Otherwise, $Y = X_2(Y) = X_2(X_2(Y)) = \cdots$ would be infinite.)
- The same is true of X_1: If $X_1(y) = \lambda v.X_1(v)X_2'(y, v)$, then certainly X_1 is infinite. Otherwise, y occurs at some position $00{\star}p$, v occurs at position $0000{\star}p$, etc.
- The only possibility remaining is $X_1(v) = v$. Then (2) becomes

$$(\lambda v.vX_2(v)) = Y \tag{4}$$

- $X_2(v) = v$ yields the solution $Y = \lambda v.vv$, $X(Y) = \Omega$.
- If $X_2(v) = Y$, then (4) yields $Y = \lambda v.vY$, contradiction.

$M \to N \to M$

That was easy!

$M \to N \to M$

That was easy!
What about "bi-cycles"?

$M \to N \to M$

That was easy!
What about "bi-cycles"?
THEOREM. (Endrullis, Klop, AP; A. Visser Festschrift, to appear)
The pure bicycles in the lambda calculus are of the form

- AAA, where $A = \lambda xy.yxx$;
- AAA, where $A = \lambda xy.yyx$;
- ABA, where $A = \lambda xy.yxy$, B any normal form;
- $AB[A/y]A$, where $A = \lambda xy.yBy$, B a normal form, x does not occur, and y does not occur actively, in B.

$\Lambda[\mathcal{M}]$

- Let $\mathcal{M} = \{X_i(x_1, \ldots, x_{n_i}) \mid i \in \mathbb{N}\}$ be a set of *metavariables with parameters*.

$\Lambda[\mathcal{M}]$

- Let $\mathcal{M} = \{X_i(x_1, \ldots, x_{n_i}) \mid i \in \mathbb{N}\}$ be a set of *metavariables with parameters*. Put

$$\Lambda[\mathcal{M}] ::= x \mid tt \mid \lambda x.t \mid X(t, \ldots, t) \qquad\qquad X \in \mathcal{M}$$

$\Lambda[\mathcal{M}]$

- Let $\mathcal{M} = \{X_i(x_1, \dots, x_{n_i}) \mid i \in \mathbb{N}\}$ be a set of *metavariables with parameters*. Put

$$\Lambda[\mathcal{M}] ::= x \mid tt \mid \lambda x.t \mid X(t, \dots, t) \qquad\qquad X \in \mathcal{M}$$

- The usual notion of substitution is extended to $\Lambda[\mathcal{M}]$ by

$$X(s_1, \dots, s_n)[\vec{t}/\vec{y}] = X(s_1[\vec{t}/\vec{y}], \dots, s_n[\vec{t}/\vec{y}])$$

$\Lambda[\mathcal{M}]$

- Let $\mathcal{M} = \{X_i(x_1, \ldots, x_{n_i}) \mid i \in \mathbb{N}\}$ be a set of *metavariables with parameters*. Put

$$\Lambda[\mathcal{M}] ::= x \mid tt \mid \lambda x.t \mid X(t, \ldots, t) \qquad\qquad X \in \mathcal{M}$$

- The usual notion of substitution is extended to $\Lambda[\mathcal{M}]$ by

$$X(s_1, \ldots, s_n)[\vec{t}/\vec{y}] = X(s_1[\vec{t}/\vec{y}], \ldots, s_n[\vec{t}/\vec{y}])$$

- Given $X(x_1, \ldots, x_{n_X}) \in \mathcal{M}$, $t \in \Lambda[\mathcal{M}](x_1, \ldots, x_{n_X})$, define

$$\begin{array}{rcl} x[X(\vec{x}) := t] & = & x \\ s_1 s_2[X(\vec{x}) := t] & = & s_1[X(\vec{x}) := t]s_2[X(\vec{x}) := t] \\ (\lambda y.s)[X(\vec{x}) := t] & = & \lambda z.s[y/z][X(\vec{x}) := t], \quad z\#t, s \\ X(s_1, \ldots, s_n)[X(\vec{x}) := t] & = & t[\vec{s}[X(\vec{x}) := t]/\vec{x}] \end{array}$$

$\Lambda[\mathcal{M}]$

- A *unification problem* is a finite set of equations between elements of $\Lambda[\mathcal{M}]$.
- A *solution* to a unification problem P is an assignment

$$X_i(x_1, \ldots, x_{n_i}) \longmapsto M_i(x_1, \ldots, x_{n_i})$$

of metavariables occurring in P to pure lambda terms, so that

$$s = t \in E \implies s[\vec{X} := \vec{M}] = t[\vec{X} := \vec{M}]$$

$\Lambda[\mathcal{M}]$

- A *unification problem* is a finite set of equations between elements of $\Lambda[\mathcal{M}]$.
- A *solution* to a unification problem P is an assignment

$$X_i(x_1, \ldots, x_{n_i}) \longmapsto M_i(x_1, \ldots, x_{n_i})$$

of metavariables occurring in P to pure lambda terms, so that

$$s = t \in E \implies s[\vec{X} := \vec{M}] = t[\vec{X} := \vec{M}]$$

- Claim. It is decidable whether a given unification problem has a solution.

Notation

We denote by $P \mid X(\vec{x}) := t$ the result of applying metavariable substitution $[X(\vec{x}) := t]$ to both sides of every equation in P:

$$
\begin{aligned}
\varnothing \mid X(\vec{x}) := t \quad &:= \quad \varnothing \\
P; s = s' \mid X(\vec{x}) := t \quad &:= \quad P \mid X(\vec{x}) := t; s[X(\vec{x}) := t] = s'[X(\vec{x}) := t]
\end{aligned}
$$

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; X(\vec{s}) = y \quad \longmapsto \quad \begin{cases} E; s_1 = y \mid X(\vec{x}) := x_1 \\ \quad \vdots \\ E; s_k = y \mid X(\vec{x}) := x_k \end{cases}$$

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; X(\vec{s}) = t_1 t_2 \quad \longmapsto \quad \begin{cases} E; s_1 = t_1 t_2 \mid X(\vec{x}) := x_1 \\ \quad \vdots \\ E; s_k = t_1 t_2 \mid X(\vec{x}) := x_k \\ E; X_1(\vec{s}) = t_1; X_2(\vec{s}) = t_2 \mid X(\vec{x}) := X_1(\vec{x}) X_2(\vec{x}) \end{cases}$$

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; X(\vec{s}) = \lambda y.t \quad \longmapsto \quad \begin{cases} E; s_1 = \lambda y.t \mid X(\vec{x}) := x_1 \\ \quad \vdots \\ E; s_k = \lambda y.t \mid X(\vec{x}) := x_k \\ E; X_0(z, \vec{s}) = t[z/y] \mid X(\vec{x}) := \lambda z.X_0(z, \vec{x}) \end{cases}$$

where $z \# E, \vec{s}, t$;

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; X(\vec{s}) = Y(\vec{t}) \quad \longmapsto \quad \begin{cases} X(\vec{s}) = Y(\vec{t}); E & \exists (s = t) \in E, \{s, t\} \not\subseteq \mathbb{M} \\ \top & \text{otherwise} \end{cases}$$

where $\mathbb{M} = \{X(\vec{t}) \mid X \in \mathcal{M}, \vec{t} \in \Lambda[\mathcal{M}]\}$.

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; X(\vec{s}) = Y(\vec{t}) \quad \longmapsto \quad \begin{cases} X(\vec{s}) = Y(\vec{t}); E & \exists (s = t) \in E, \{s, t\} \not\subset \mathbb{M} \\ \top & \text{otherwise} \end{cases}$$

where $\mathbb{M} = \{X(\vec{t}) \mid X \in \mathcal{M}, \vec{t} \in \Lambda[\mathcal{M}]\}$.
IOW: *If every equation in the unification problem is an equation between metavariables, then a solution can be obtained by setting all the metavariables simultaneously to* ANY *λ-term. Otherwise, equations of the form $X(\vec{s}) = Y(\vec{t})$ are moved to the back of the equation queue.*

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$E; t = X(\vec{s}) \quad \longmapsto \quad E; X(\vec{s}) = t$$

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$
\begin{aligned}
E; x = x &\quad\longmapsto\quad E \\
E; st = s't' &\quad\longmapsto\quad E; s = s'; t = t' \\
E; \lambda x.s = \lambda y.t &\quad\longmapsto\quad E; s[z/x] = t[z/y], \quad z\#E, s, t
\end{aligned}
$$

Martelli–Montanari for $\Lambda[\mathcal{M}]$

$$\begin{aligned} E; \_\_ &\longmapsto \bot \\ \varnothing &\longmapsto \top \end{aligned}$$

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v)X_2(v)) = X_1(Y)$$

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v, Y)X_2(v)) = X_1(Y, Z)$$

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v, Y)X_2(v)) = X_1(Y, Z)$$

ANSWER: Occurrences *block* decomposition, forcing a variable to be chosen.

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v,Y)X_2(v)) = X_1(Y,Z)$$

ANSWER: Occurrences *block* decomposition, forcing a variable to be chosen. Concretely,

- Every recursive occurrence of a metavariable is marked by a special term constructor, which remembers the metavariable;

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v,Y)X_2(v)) = X_1(Y,Z)$$

ANSWER: Occurrences *block* decomposition, forcing a variable to be chosen. Concretely,

- Every recursive occurrence of a metavariable is marked by a special term constructor, which remembers the metavariable;
- When the metavariable is substituted, the marker is updated; When the metavariable aligns with the marker, only the trivial instances may be chosen;

Cycles again

PROBLEM: *What to do with the occurs-check?*

$$(\lambda v.X_1(v,Y)X_2(v)) = X_1(Y,Z)$$

ANSWER: Occurrences *block* decomposition, forcing a variable to be chosen. Concretely,

- Every recursive occurrence of a metavariable is marked by a special term constructor, which remembers the metavariable;
- When the metavariable is substituted, the marker is updated; When the metavariable aligns with the marker, only the trivial instances may be chosen;
- The markers are calculated and propagated through parameters to metavariables.

Termination

Let P be a unification problem.
For every metavariable $X(\vec{x}) \in P$, the application of decomposition rules to X and all the fresh metavariables generated by it must eventually terminate — either in the variables, the guards, or other metavariables.
The unification problem that results may be much larger than the original one — but it will have one fewer distinct metavariables.
After finitely many (pure) simplifications, some metavariable will become subject to the decomposition rules. At this point, there are again finitely many steps until it will be forced to a variable.

Related work

- *Higher-order unification* concerns unification of simply-typed lambda terms up to beta-eta equality.
- From the perspective of second-order equational logic, it is thus really a form of E-unification.
- *Context unification* is a fragment of higher-order unification where the meta-variables are allowed *unique occurrence* of their argument.
- *Nominal unification* concerns the nominal presentation of higher-order signatures.

Unification in second-order equational logic is decidable!