Barendregt's Theory of the λ-Calculus, Refreshed and Formalized

Adrienne Lancelot[12], Beniamino Accattoli[1], Maxime Vemclefs[3]

[1]Inria & LIX, École Polytechnique
[2]IRIF, Université Paris Cité & CNRS
[3]Independent

September 29th 2025 - ITP

Outline

Barendregt's Theory of the Lambda Calculus

Formalizing in Abella

Partial Recursive Functions

Partial Recursive Functions model which mathematical functions are computable.

There is a natural *extensional preorder* on partial functions

$$f \leq_{\mathrm{PRF}} g \text{ if } \forall n \in \mathbb{N},\ f(n) = \bot \text{ or } f(n) =_{\mathbb{N}} g(n)$$

$f_\bot : n \mapsto \bot$ is the minimum PRF function for $\leq_{\mathrm{PRF}}$

Partial Recursive Functions

Partial Recursive Functions model which mathematical functions are computable.

There is a natural *extensional preorder* on partial functions

$$f \leq_{\mathrm{PRF}} g \text{ if } \forall n \in \mathbb{N},\ f(n) = \bot \text{ or } f(n) =_{\mathbb{N}} g(n)$$

$f_\bot : n \mapsto \bot$ is the minimum PRF function for $\leq_{\mathrm{PRF}}$

Partial Recursive Functions

Partial Recursive Functions model which mathematical functions are computable.

There is a natural *extensional preorder* on partial functions

$$f \leq_{\mathrm{PRF}} g \text{ if } \forall n \in \mathbb{N},\ f(n) = \bot \text{ or } f(n) =_{\mathbb{N}} g(n)$$

$f_\bot : n \mapsto \bot$ is the minimum PRF function for $\leq_{\mathrm{PRF}}$

Lambda Calculus

PRF do not look at how to compute, hence the preorder can only be extensional.

Instead, in the lambda calculus, how to compute is a critical concept.

There are a rich number of possible equivalences (or preorders) of lambda terms, both extensional or intensional.

Lambda Calculus

PRF do not look at how to compute, hence the preorder can only be extensional.

Instead, in the lambda calculus, how to compute is a critical concept.

There are a rich number of possible equivalences (or preorders) of lambda terms, both extensional or intensional.

Lambda Calculus

PRF do not look at how to compute, hence the preorder can only be extensional.

Instead, in the lambda calculus, how to compute is a critical concept.

There are a rich number of possible equivalences (or preorders) of lambda terms, both extensional or intensional.

Computable Functions & Lambda Calculus

Partial recursive functions embed in the lambda calculus.

What is the lambda term that represents **undefined**?

A computation that never ends? Ω!

$\Omega := (\lambda x.xx)(\lambda x.xx) \rightarrow_\beta (\lambda x.xx)(\lambda x.xx) \rightarrow_\beta \cdots$

Now, what is the equivalence class of **undefined**/Ω ?

Computable Functions & Lambda Calculus

Partial recursive functions embed in the lambda calculus.

What is the lambda term that represents **undefined**?

A computation that never ends? Ω!

$\Omega := (\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \cdots$

Now, what is the equivalence class of **undefined**/Ω ?

Computable Functions & Lambda Calculus

Partial recursive functions embed in the lambda calculus.

What is the lambda term that represents **undefined**?

A computation that never ends? Ω!

$\Omega := (\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \cdots$

Now, what is the equivalence class of **undefined**/Ω ?

A first naive attempt

Undefined represents a computation that never ends.

- **undefined** terms = β-diverging terms?

Surprisingly, this would lead to an inconsistency.

>> If all β-diverging terms are equated in an equational theory, then this theory equates all terms.

A first naive attempt

Undefined represents a computation that never ends.

- **undefined** terms $= \beta$-diverging terms?

Surprisingly, this would lead to an inconsistency.

$>>$ If all β-diverging terms are equated in an equational theory, then this theory equates all terms.

A first naive attempt

Undefined represents a computation that never ends.

- **undefined** terms $= \beta$-diverging terms?

Surprisingly, this would lead to an inconsistency.

$>>$ If all β-diverging terms are equated in an equational theory, then this theory equates all terms.

β-diverging terms may be very different

Indeed, let us look at two β-diverging terms

$$\begin{array}{ccc}
\mathit{fix} & \text{and} & \Omega \\
\downarrow_\beta & & \downarrow_\beta \\
\lambda f.f\ (\mathit{fix}\, f) & & \Omega \\
\downarrow_\beta & & \downarrow_\beta \\
\lambda f.f\ (f\ (\mathit{fix}\, f)) & & \Omega \\
\downarrow_\beta & & \downarrow_\beta \\
\lambda f.f\ (f\ (f\ (\mathit{fix}\, f))) & & \Omega \\
\downarrow_\beta & & \downarrow_\beta \\
\lambda f.f\ (f\ (f\ (f\ \ldots))) & & \Omega \\
\downarrow_\beta & & \downarrow_\beta \\
\vdots & & \vdots
\end{array}$$

Recursion does not carry the same meaning as looping on itself.

A second attempt

Instead, one might consider a more restrained reduction

- **undefined** terms $=$ **head**-diverging terms?

The equational theory that identifies **head**-diverging terms is consistent.

$>>$ This theory does not equate all terms.

A second attempt

Instead, one might consider a more restrained reduction

- **undefined** terms = **head**-diverging terms?

The equational theory that identifies **head**-diverging terms is consistent.

>> This theory does not equate all terms.

A second attempt

Instead, one might consider a more restrained reduction

- **undefined** terms = **head**-diverging terms?

The equational theory that identifies **head**-diverging terms is consistent.

$>>$ This theory does not equate all terms.

β-diverging terms may be very different

Fixpoint combinators are **head**-normalizing.

$$\begin{array}{ccc}
\mathit{fix} & \text{and} & \Omega \\
\downarrow_h & & \downarrow_h \\
\lambda f.f\ (\mathit{fix}\, f) & & \Omega \\
\not\downarrow_h & & \downarrow_h \\
 & & \vdots
\end{array}$$

Recursion and looping are nicely separated by **head** reduction.

Consistency

A relation $\mathcal{R} \subseteq \Lambda \times \Lambda$ is consistent if there exists $t, u \in \Lambda$ such that $(t, u) \notin \mathcal{R}$.

An equational theory is an equivalence relation $=_{\mathcal{T}}$ such that:

- *Invariance under Computation*: if $t \to_\beta u$ then $t =_{\mathcal{T}} u$
- *Stability by Contexts*: if $t =_{\mathcal{T}} u$ then $\forall C$, $C\langle t\rangle =_{\mathcal{T}} C\langle u\rangle$.

To validate the choice of undefined terms: Is there a consistent equational theory where undefined terms are collapsed?

Consistency

A relation $\mathcal{R} \subseteq \Lambda \times \Lambda$ is consistent if there exists $t, u \in \Lambda$ such that $(t, u) \notin \mathcal{R}$.

An equational theory is an equivalence relation $=_{\mathcal{T}}$ such that:

- *Invariance under Computation*: if $t \rightarrow_{\beta} u$ then $t =_{\mathcal{T}} u$
- *Stability by Contexts*: if $t =_{\mathcal{T}} u$ then $\forall C$, $C\langle t\rangle =_{\mathcal{T}} C\langle u\rangle$.

To validate the choice of undefined terms: Is there a consistent equational theory where undefined terms are collapsed?

Consistency

A relation $\mathcal{R} \subseteq \Lambda \times \Lambda$ is consistent if there exists $t, u \in \Lambda$ such that $(t, u) \notin \mathcal{R}$.

An equational theory is an equivalence relation $=_{\mathcal{T}}$ such that:

- *Invariance under Computation*: if $t \rightarrow_{\beta} u$ then $t =_{\mathcal{T}} u$
- *Stability by Contexts*: if $t =_{\mathcal{T}} u$ then $\forall C$, $C\langle t\rangle =_{\mathcal{T}} C\langle u\rangle$.

To validate the choice of undefined terms: Is there a consistent equational theory where undefined terms are collapsed?

What is Genericity?

Undefined terms are black holes for the evaluation process.

$$u \longrightarrow u_1 \xrightarrow{n} u_n \xrightarrow{*} \cdots$$

If a program awaits the evaluation of an undefined sub-term

$$\cdots \longrightarrow \boxed{C\; u} \longrightarrow \boxed{C\; u_1}$$

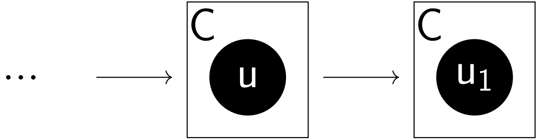
Then it will be unable to produce a result

$$\boxed{C\; u} \longrightarrow \boxed{C\; u_1} \xrightarrow{n} \boxed{C\; u_n} \xrightarrow{*} \cdots$$

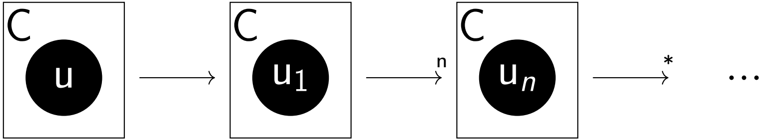
What is Genericity?

Undefined terms are black holes for the evaluation process.

If a program awaits the evaluation of an undefined sub-term



Then it will be unable to produce a result



What is Genericity?

Genericity somehow specifies this fact dually:
If a program terminates while there were **undefined** sub-terms, then it never entered the black hole.

Genericity says: (n is a normal form and s is any term)

Anything can simulate the *generic* **undefined** sub-terms in a terminating term.

What is Genericity?

Genericity somehow specifies this fact dually:
If a program terminates while there were **undefined** sub-terms, then it never entered the black hole.

Genericity says: (n is a normal form and s is any term)

Anything can simulate the *generic* **undefined** sub-terms in a terminating term.

What is Genericity?

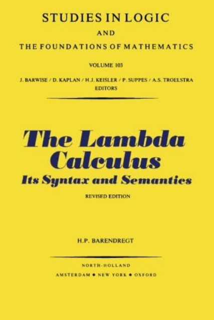
Genericity somehow specifies this fact dually:
If a program terminates while there were **undefined** sub-terms,
then it never entered the black hole.

Genericity says: (n is a normal form and s is any term)

Anything can simulate the *generic* **undefined** sub-terms in a terminating term.

Refreshed and Formalized

We survey some results of Barendregt's theory of the λ-calculus (1984).



Refreshed:

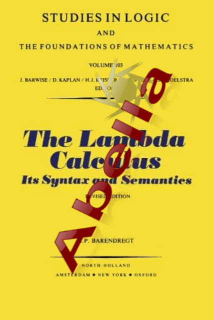
- Takahashi's proof of genericity (1994)
- Accattoli et al. study of normalization (2019)

Formalized with the Abella proof assistant:

- Reasoning with binders close to paper
- Representing contexts (with possible captures)

Refreshed and Formalized

We survey some results of Barendregt's theory of the λ-calculus (1984).



Refreshed:

- Takahashi's proof of genericity (1994)
- Accattoli et al. study of normalization (2019)

Formalized with the Abella proof assistant:

- Reasoning with binders close to paper
- Representing contexts (with possible captures)

Proving/Formalizing Genericity

A Simple Proof of the Genericity Lemma

Masako Takahashi

Department of Information Science
Tokyo Institute of Technology
Ookayama, Meguro, Tokyo 152 Japan
masako@titisha.is.titech.ac.jp

Abstract. A short direct proof is given for the fundamental property of unsolvable λ-terms; if M is an unsolvable λ-term and $C[M]$ is solvable, then $C[N]$ is solvable for any λ-term N. (Here $C[\ ]$ stands for an arbitrary context.)

1. Preliminaries

A term in this note means a λ-term, which is either $x, \lambda x.M$ or MN, (where M, N are terms and x is a variable.) Unless otherwise stated, capital letters $M, N, P, \ldots$ stand for arbitrary terms, $\mathbf{M}, \mathbf{N}, \ldots$ for (possibly null) sequences of terms, $x, y, \ldots$ for variables, and $\mathbf{x}, \mathbf{y}, \ldots$ for (possibly null) sequences of variables. We refer to [1] as the standard text in the field.

A term of the form $\lambda \mathbf{x}.y\mathbf{M}$ (more precisely, $\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.((\ldots((yM_1)M_2)\ldots)M_m))\ldots))$ for some $n, m \geq 0$) is said to be in *head normal form* (*hnf*, for short). If a term M has a hnf (that is, $M =_\beta M'$ for a term M' in hnf), then M is called *solvable*. The following are well-known facts of solvable terms (cf.[1] 8.3.1 - 14).

(1) M is solvable if and only if $\forall P, \exists \mathbf{x}, \exists \mathbf{Q}((\lambda \mathbf{x}.M)\mathbf{Q} =_\beta P)$.
(2) $\lambda x.M$ is solvable if and only if so is M.
(3) if $M[x := N]$ is solvable then so is M.
(4) if MN is solvable then so is M.

A term in *β-normal form* (*β-nf*, for short) is recursively defined as a term of the form $\lambda \mathbf{x}.y\mathbf{M}$ where $\mathbf{M}$ is a (possibly null) sequence of terms in β-nf.

2. Propositions

First we prove a special case of the genericity lemma.

Lemma 1. Let M, N, P be terms with M unsolvable and N in β-nf. Then $P[x := M] =_\beta N$ implies $P[x := M'] =_\beta N$ for any M'.

Proof. We prove the lemma by induction on the structure of N. Suppose $P[x := M] =_\beta N$, and $N \equiv \lambda \mathbf{y}.zN_1N_2\ldots N_n$ where $n \geq 0$ and each N_i is in β-nf. (Here, $\equiv$ denotes syntactic equality of terms.) Then since N is solvable, P is also solvable by (3) above, and hence has a hnf, say $\lambda \mathbf{u}.vP_1P_2\ldots P_p$. Here x and v must be different. (For otherwise $P[x := M] =_\beta \lambda \mathbf{u}.M\mathbf{P}$ for some $\mathbf{P}$, and $P[x := M]$ would by (2) and (4) above be unsolvable, which contradicts our assumption.) Therefore we have $P[x := M] =_\beta \lambda \mathbf{u}.vP'_1P'_2\ldots P'_p$ where $P'_i \equiv P_i[x := M](i = 1, 2, \ldots, p)$. Since $P[x := M] =_\beta N \equiv \lambda \mathbf{y}.zN_1N_2\ldots N_n$, we know from the Church-Rosser theorem that $P'_i =_\beta N_i(i = 1, 2, \ldots, n)$ and $p = n$. Without loss of generality we may also assume $\mathbf{u} \equiv \mathbf{y}$ and $v \equiv z$.

If $n = 0$, then $P =_\beta \lambda \mathbf{u}.v \equiv N$. In this case, we have $P[x := M'] =_\beta (\lambda \mathbf{u}.v)[x := M'] \equiv \lambda \mathbf{u}.v \equiv N$ for any M'. If $n > 0$, then from the fact $P_i[x := M] \equiv P'_i =_\beta N_i$ and the inductive hypothesis, we get $P_i[x := M'] =_\beta N_i(i = 1, 2, \ldots, n)$ for any M'. In this case,

$$\begin{aligned} P[x := M'] &=_\beta (\lambda \mathbf{u}.vP_1P_2\ldots P_n)[x := M'] \\ &\equiv \lambda \mathbf{u}.v(P_1[x := M'])(P_2[x := M'])\ldots(P_n[x := M']) \\ &=_\beta \lambda \mathbf{y}.zN_1N_2\ldots N_n \equiv N. \end{aligned}$$

This proves the lemma. □

Lemma 2. ([1] 14.3.24. Genericity lemma) Let M be an unsolvable term, and $C[\ ]$ be a context such that $C[M]$ has a β-nf. Then $C[M] =_\beta C[M']$ for any M'.

Proof. For given M', let $\mathbf{y}$ be a sequence of all free variables in MM'. Take a new variable x (neither in $C[M]$ nor $C[M']$), and let $P \equiv C[x\mathbf{y}]$. Then since $\lambda \mathbf{y}.M$ and $\lambda \mathbf{y}.M'$ are closed terms, we have

$$P[x := \lambda \mathbf{y}.M] \equiv C[(\lambda \mathbf{y}.M)\mathbf{y}] =_\beta C[M],$$
$$P[x := \lambda \mathbf{y}.M'] \equiv C[(\lambda \mathbf{y}.M')\mathbf{y}] =_\beta C[M'].$$

The term $\lambda \mathbf{y}.M$ therefore satisfies $P[x := \lambda \mathbf{y}.M] =_\beta C[M] =_\beta N$ for some N in β-nf. Here $\lambda \mathbf{y}.M$ is unsolvable because so is M. Hence by applying lemma 1 we get $P[x := \lambda \mathbf{y}.M'] =_\beta N$, which implies $C[M] =_\beta C[M']$. □

Corollary 3. If M is unsolvable and $C[M]$ is solvable, then $C[M']$ is solvable for any M'.

Proof. Since $C[M]$ is solvable, by (1) above there exist $\mathbf{x}$ and $\mathbf{N}$ such that $(\lambda \mathbf{x}.C[M])\mathbf{N}$ has a β-nf. Then by lemma 2 (applied to the context $(\lambda \mathbf{x}.C[\ ])\mathbf{N}$), we know $(\lambda \mathbf{x}.C[M'])\mathbf{N}$ has a β-nf for any M'. This means $(\lambda \mathbf{x}.C[M'])\mathbf{N}$ is solvable, and consequently $C[M']$ is solvable. □

The proof presented provides an alternative to the conventional one which uses a topological argument on Böhm trees (cf.[1] Chapters 10 and 14).

Reference

[1] H. P. Barendregt, *The Lambda Calculus* (North-Holland 1984).

Proving/Formalizing Genericity

A Simple Proof of the Genericity Lemma

Masako Takahashi

Department of Information Science
Tokyo Institute of Technology
Ookayama, Meguro, Tokyo 152 Japan
masako@titisha.is.titech.ac.jp

Abstract. A short direct proof is given for the fundamental property of unsolvable λ-terms; if M is an unsolvable λ-term and $C[M]$ is solvable, then $C[N]$ is solvable for any λ-term N. (Here $C[\ ]$ stands for an arbitrary context.)

1. Preliminaries

A term in this note means a λ-term, which is either $x, \lambda x.M$ or MN, (where M, N are terms and x is a variable.) Unless otherwise stated, capital letters $M, N, P, \ldots$ stand for arbitrary terms, $\mathbf{M}, \mathbf{N}, \ldots$ for (possibly null) sequences of terms, $x, y, \ldots$ for variables, and $\mathbf{x}, \mathbf{y}, \ldots$ for (possibly null) sequences of variables. We refer to [1] as the standard text in the field.

A term of the form $\lambda\mathbf{x}.y\mathbf{M}$ (more precisely, $\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.((\ldots((yM_1)M_2)\ldots)M_m))\ldots))$ for some $n, m \geq 0$) is said to be in *head normal form* (*hnf*, for short). If a term M has a hnf (that is, $M =_\beta M'$ for a term M' in hnf), then M is called *solvable*. The following are well-known facts of solvable terms (cf.[1] 8.3.1 - 14).

(1) M is solvable if and only if $\forall P, \exists\mathbf{x}, \exists\mathbf{Q}((\lambda\mathbf{x}.M)\mathbf{Q} =_\beta P)$.
(2) $\lambda x.M$ is solvable if and only if so is M.
(3) if $M[x := N]$ is solvable then so is M.
(4) if MN is solvable then so is M.

A term in *β-normal form* (*β-nf*, for short) is recursively defined as a term of the form $\lambda\mathbf{x}.y\mathbf{M}$ where $\mathbf{M}$ is a (possibly null) sequence of terms in β-nf.

2. Propositions

First we prove a special case of the genericity lemma.

Lemma 1. Let M, N, P be terms with M unsolvable and N in β-nf. Then $P[x := M] =_\beta N$ implies $P[x := M'] =_\beta N$ for any M'.

Proof. We prove the lemma by induction on the structure of N. Suppose $P[x := M] =_\beta N$, and $N \equiv \lambda\mathbf{y}.zN_1N_2\ldots N_n$ where $n \geq 0$ and each N_i is in β-nf. (Here, $\equiv$ denotes syntactic equality of terms.) Then since N is solvable, P is also solvable by (3) above, and hence has a hnf, say $\lambda\mathbf{u}.vP_1P_2\ldots P_p$. Here x and v must be different. (For otherwise $P[x := M] =_\beta \lambda\mathbf{u}.M\mathbf{P}$ for some $\mathbf{P}$, and $P[x := M]$ would by (2) and (4) above be unsolvable, which contradicts our assumption.) Therefore we have $P[x := M] =_\beta \lambda\mathbf{u}.vP_1'P_2'\ldots P_p'$ where $P_i' \equiv P_i[x := M](i = 1, 2, \ldots, p)$. Since $P[x := M] =_\beta N \equiv \lambda\mathbf{y}.zN_1N_2\ldots N_n$, we know from the Church-Rosser theorem that $P_i' =_\beta N_i(i = 1, 2, \ldots, n)$ and $p = n$. Without loss of generality we may also assume $\mathbf{u} \equiv \mathbf{y}$ and $v \equiv z$.

If $n = 0$, then $P =_\beta \lambda\mathbf{u}.v \equiv N$. In this case, we have $P[x := M'] =_\beta (\lambda\mathbf{u}.v)[x := M'] \equiv \lambda\mathbf{u}.v \equiv N$ for any M'. If $n > 0$, then from the fact $P_i[x := M] \equiv P_i' =_\beta N_i$ and the inductive hypothesis, we get $P_i[x := M'] =_\beta N_i(i = 1, 2, \ldots, n)$ for any M'. In this case,

$$\begin{aligned} P[x := M'] &=_\beta (\lambda\mathbf{u}.vP_1P_2\ldots P_n)[x := M'] \\ &\equiv \lambda\mathbf{u}.v(P_1[x := M'])(P_2[x := M'])\ldots(P_n[x := M']) \\ &=_\beta \lambda\mathbf{y}.zN_1N_2\ldots N_n \equiv N. \end{aligned}$$

This proves the lemma. □

Lemma 2. ([1] 14.3.24. Genericity lemma) Let M be an unsolvable term, and $C[\ ]$ be a context such that $C[M]$ has a β-nf. Then $C[M] =_\beta C[M']$ for any M'.

Proof. For given M', let $\mathbf{y}$ be a sequence of all free variables in MM'. Take a new variable x (neither in $C[M]$ nor $C[M']$), and let $P \equiv C[x\mathbf{y}]$. Then since $\lambda\mathbf{y}.M$ and $\lambda\mathbf{y}.M'$ are closed terms, we have

$$\begin{aligned} P[x := \lambda\mathbf{y}.M] &\equiv C[(\lambda\mathbf{y}.M)\mathbf{y}] =_\beta C[M], \\ P[x := \lambda\mathbf{y}.M'] &\equiv C[(\lambda\mathbf{y}.M')\mathbf{y}] =_\beta C[M']. \end{aligned}$$

The term $\lambda\mathbf{y}.M$ therefore satisfies $P[x := \lambda\mathbf{y}.M] =_\beta C[M] =_\beta N$ for some N in β-nf. Here $\lambda\mathbf{y}.M$ is unsolvable because so is M. Hence by applying lemma 1 we get $P[x := \lambda\mathbf{y}.M'] =_\beta N$, which implies $C[M] =_\beta C[M']$. □

Corollary 3. If M is unsolvable and $C[M]$ is solvable, then $C[M']$ is solvable for any M'.

Proof. Since $C[M]$ is solvable, by (1) above there exist $\mathbf{x}$ and $\mathbf{N}$ such that $(\lambda\mathbf{x}.C[M])\mathbf{N}$ has a β-nf. Then by lemma 2 (applied to the context $(\lambda\mathbf{x}.C[\ ])\mathbf{N}$), we know $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ has a β-nf for any M'. This means $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ is solvable, and consequently $C[M']$ is solvable. □

The proof presented provides an alternative to the conventional one which uses a topological argument on Böhm trees (cf.[1] Chapters 10 and 14).

Reference

[1] H. P. Barendregt, *The Lambda Calculus* (North-Holland 1984).

Main Lemma
~15 lines of text
~90 lines of Abella
~140 tactics

Proving/Formalizing Genericity

A Simple Proof of the Genericity Lemma

Masako Takahashi

Department of Information Science
Tokyo Institute of Technology
Ookayama, Meguro, Tokyo 152 Japan
masako@titisha.is.titech.ac.jp

Abstract. A short direct proof is given for the fundamental property of unsolvable λ-terms; if M is an unsolvable λ-term and $C[M]$ is solvable, then $C[N]$ is solvable for any λ-term N. (Here $C[\ ]$ stands for an arbitrary context.)

1. Preliminaries

A term in this note means a λ-term, which is either $x, \lambda x.M$ or MN, (where M, N are terms and x is a variable.) Unless otherwise stated, capital letters $M, N, P,...$ stand for arbitrary terms, $\mathbf{M}, \mathbf{N},...$ for (possibly null) sequences of terms, $x, y,...$ for variables, and $\mathbf{x}, \mathbf{y},...$ for (possibly null) sequences of variables. We refer to [1] as the standard text in the field.

A term of the form $\lambda\mathbf{x}.y\mathbf{M}$ (more precisely, $\lambda x_1.(\lambda x_2.(...(\lambda x_n.((...((yM_1)M_2)...)M_m))...))$ for some $n, m \geq 0$) is said to be in *head normal form* (*hnf*, for short). If a term M has a hnf (that is, $M =_\beta M'$ for a term M' in hnf), then M is called *solvable*. The following are well-known facts of solvable terms (cf.[1] 8.3.1 - 14).

(1) M is solvable if and only if $\forall P, \exists \mathbf{x}, \exists \mathbf{Q}((\lambda \mathbf{x}.M)\mathbf{Q} =_\beta P)$.
(2) $\lambda x.M$ is solvable if and only if so is M.
(3) if $M[x := N]$ is solvable then so is M.
(4) if MN is solvable then so is M.

A term in *β-normal form* (*β-nf*, for short) is recursively defined as a term of the form $\lambda\mathbf{x}.y\mathbf{M}$ where $\mathbf{M}$ is a (possibly null) sequence of terms in β-nf.

2. Propositions

~~First we prove a special case of the genericity lemma.~~

Lemma 1. Let M, N, P be terms with M unsolvable and N in β-nf. Then $P[x := M] =_\beta N$ implies $P[x := M'] =_\beta N$ for any M'.
Proof. We prove the lemma by induction on the structure of N. Suppose $P[x := M] =_\beta N$, and $N \equiv \lambda\mathbf{y}.zN_1N_2...N_n$ where $n \geq 0$ and each N_i is in β-nf. (Here, $\equiv$ denotes syntactic equality of terms.) Then since N is solvable, P is also solvable by (3) above, and hence has a hnf, say $\lambda\mathbf{u}.vP_1P_2...P_p$. Here x and v must be different. (For otherwise $P[x := M] =_\beta \lambda\mathbf{u}.M\mathbf{P}$ for some $\mathbf{P}$, and $P[x := M]$ would by (2) and (4) above be unsolvable, which contradicts our assumption.) Therefore we have $P[x := M] =_\beta \lambda\mathbf{u}.vP'_1P'_2...P'_p$ where $P'_i \equiv P_i[x := M](i = 1, 2, ..., p)$. Since $P[x := M] =_\beta N \equiv \lambda\mathbf{y}.zN_1N_2...N_n$, we know from the Church-Rosser theorem that $P'_i =_\beta N_i(i = 1, 2, ..., n)$ and $p = n$. Without loss of generality we may also assume $\mathbf{u} \equiv \mathbf{y}$ and $v \equiv z$.

If $n = 0$, then $P =_\beta \lambda\mathbf{u}.v \equiv N$. In this case, we have $P[x := M'] =_\beta (\lambda\mathbf{u}.v)[x := M'] \equiv \lambda\mathbf{u}.v \equiv N$ for any M'. If $n > 0$, then from the fact $P_i[x := M] \equiv P'_i =_\beta N_i$ and the inductive hypothesis, we get $P_i[x := M'] =_\beta N_i(i = 1, 2, ..., n)$ for any M'. In this case,

$$\begin{aligned} P[x := M'] &=_\beta (\lambda\mathbf{u}.vP_1P_2...P_n)[x := M'] \\ &\equiv \lambda\mathbf{u}.v(P_1[x := M'])(P_2[x := M'])...(P_n[x := M']) \\ &=_\beta \lambda\mathbf{y}.zN_1N_2...N_n \equiv N. \end{aligned}$$

This proves the lemma. □

Lemma 2. ([1] 14.3.24. Genericity lemma) Let M be an unsolvable term, and $C[\ ]$ be a context such that $C[M]$ has a β-nf. Then $C[M] =_\beta C[M']$ for any M'.
Proof. For given M', let $\mathbf{y}$ be a sequence of all free variables in MM'. Take a new variable x (neither in $C[M]$ nor $C[M']$), and let $P \equiv C[x\mathbf{y}]$. Then since $\lambda\mathbf{y}.M$ and $\lambda\mathbf{y}.M'$ are closed terms, we have

$$P[x := \lambda\mathbf{y}.M] \equiv C[(\lambda\mathbf{y}.M)\mathbf{y}] =_\beta C[M],$$
$$P[x := \lambda\mathbf{y}.M'] \equiv C[(\lambda\mathbf{y}.M')\mathbf{y}] =_\beta C[M'].$$

The term $\lambda\mathbf{y}.M$ therefore satisfies $P[x := \lambda\mathbf{y}.M] =_\beta C[M] =_\beta N$ for some N in β-nf. Here $\lambda\mathbf{y}.M$ is unsolvable because so is M. Hence by applying lemma 1 we get $P[x := \lambda\mathbf{y}.M'] =_\beta N$, which implies $C[M] =_\beta C[M']$. □

Corollary 3. If M is unsolvable and $C[M]$ is solvable, then $C[M']$ is solvable for any M'.
Proof. Since $C[M]$ is solvable, by (1) above there exist $\mathbf{x}$ and $\mathbf{N}$ such that $(\lambda\mathbf{x}.C[M])\mathbf{N}$ has a β-nf. Then by lemma 2 (applied to the context $(\lambda\mathbf{x}.C[\ ])\mathbf{N}$), we know $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ has a β-nf for any M'. This means $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ is solvable, and consequently $C[M']$ is solvable. □

The proof presented provides an alternative to the conventional one which uses a topological argument on Böhm trees (cf.[1] Chapters 10 and 14).

Reference

[1] H. P. Barendregt, *The Lambda Calculus* (North-Holland 1984).

Preliminaries:
~2000 lines of Abella

Main Lemma
~15 lines of text
~90 lines of Abella
~140 tactics

Proving/Formalizing Genericity

A Simple Proof of the Genericity Lemma

Takahashi's trick
(**Disentangling**)
~60 lines of Abella

Abstract. A short direct p
is an unsolvable λ-term and
stands for an arbitrary cont

1. Preliminaries

A term in this note means a λ-term, which is either $x, \lambda x.M$ or MN, (where M, N are terms and x is a variable.) Unless otherwise stated, capital letters $M, N, P, \ldots$ stand for arbitrary terms, $\mathbf{M}, \mathbf{N}, \ldots$ for (possibly null) sequences of terms, $x, y, \ldots$ for variables, and $\mathbf{x}, \mathbf{y}, \ldots$ for (possibly null) sequences of variables. We refer to [1] as the standard text in the field.

A term of the form $\lambda\mathbf{x}.y\mathbf{M}$ (more precisely, $\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.((\ldots((yM_1)M_2)\ldots)M_m))\ldots))$ for some $n, m \geq 0$) is said to be in *head normal form* (*hnf*, for short). If a term M has a hnf (that is, $M =_\beta M'$ for a term M' in hnf), then M is called *solvable*. The following are well-known facts of solvable terms (cf.[1] 8.3.1 - 14).

(1) M is solvable if and only if $\forall P, \exists \mathbf{x}, \exists \mathbf{Q}((\lambda\mathbf{x}.M)\mathbf{Q} =_\beta P)$.
(2) $\lambda x.M$ is solvable if and only if so is M.
(3) if $M[x := N]$ is solvable then so is M.
(4) if MN is solvable then so is M.

A term in *β-normal form* (*β-nf*, for short) is recursively defined as a term of the form $\lambda\mathbf{x}.y\mathbf{M}$ where $\mathbf{M}$ is a (possibly null) sequence of terms in β-nf.

2. Propositions

First we prove a special case of the genericity lemma.

Lemma 1. Let M, N, P be terms with M unsolvable and N in β-nf. Then $P[x := M] =_\beta N$ implies $P[x := M'] =_\beta N$ for any M'.

Proof. We prove the lemma by induction on the structure of N. Suppose $P[x := M] =_\beta N$, and $N \equiv \lambda\mathbf{y}.zN_1N_2\ldots N_n$ where $n \geq 0$ and each N_i is in β-nf. (Here, $\equiv$ denotes syntactic equality of terms.) Then since N is solvable, P is also solvable by (3) above, and hence has a hnf, say $\lambda\mathbf{u}.vP_1P_2\ldots P_p$. Here x and v must be different. (For otherwise $P[x := M] =_\beta \lambda\mathbf{u}.M\mathbf{P}$ for some $\mathbf{P}$, and $P[x := M]$ would by (2) and (4) above be unsolvable, which contradicts our assumption.) Therefore we have $P[x := M] =_\beta \lambda\mathbf{u}.vP_1'P_2'\ldots P_p'$ where $P_i' \equiv P_i[x := M](i = 1, 2, \ldots, p)$. Since $P[x := M] =_\beta N \equiv \lambda\mathbf{y}.zN_1N_2\ldots N_n$, we know from the Church-Rosser theorem that $P_i' =_\beta N_i(i = 1, 2, \ldots, n)$ and $p = n$. Without loss of generality we may also assume $\mathbf{u} \equiv \mathbf{y}$ and $v \equiv z$.

If $n = 0$, then $P =_\beta \lambda\mathbf{u}.v \equiv N$. In this case, we have $P[x := M'] =_\beta (\lambda\mathbf{u}.v)[x := M'] \equiv \lambda\mathbf{u}.v \equiv N$ for any M'. If $n > 0$, then from the fact $P_i[x := M] \equiv P_i' =_\beta N_i$ and the inductive hypothesis, we get $P_i[x := M'] =_\beta N_i(i = 1, 2, \ldots, n)$ for any M'. In this case,

$$\begin{aligned} P[x := M'] &=_\beta (\lambda\mathbf{u}.vP_1P_2\ldots P_n)[x := M'] \\ &\equiv \lambda\mathbf{u}.v(P_1[x := M'])(P_2[x := M'])\ldots(P_n[x := M']) \\ &=_\beta \lambda\mathbf{y}.zN_1N_2\ldots N_n \equiv N. \end{aligned}$$

This proves the lemma. □

Lemma 2. ([1] 14.3.24. Genericity lemma) Let M be an unsolvable term, and $C[\ ]$ be a context such that $C[M]$ has a β-nf. Then $C[M] =_\beta C[M']$ for any M'.

Proof. For given M', let $\mathbf{y}$ be a sequence of all free variables in MM'. Take a new variable x (neither in $C[M]$ nor $C[M']$), and let $P \equiv C[x\mathbf{y}]$. Then since $\lambda\mathbf{y}.M$ and $\lambda\mathbf{y}.M'$ are closed terms, we have

$$P[x := \lambda\mathbf{y}.M] \equiv C[(\lambda\mathbf{y}.M)\mathbf{y}] =_\beta C[M],$$
$$P[x := \lambda\mathbf{y}.M'] \equiv C[(\lambda\mathbf{y}.M')\mathbf{y}] =_\beta C[M'].$$

The term $\lambda\mathbf{y}.M$ therefore satisfies $P[x := \lambda\mathbf{y}.M] =_\beta C[M] =_\beta N$ for some N in β-nf. Here $\lambda\mathbf{y}.M$ is unsolvable because so is M. Hence by applying lemma 1 we get $P[x := \lambda\mathbf{y}.M'] =_\beta N$, which implies $C[M] =_\beta C[M']$. □

Corollary 3. If M is unsolvable and $C[M]$ is solvable, then $C[M']$ is solvable for any M'.

Proof. Since $C[M]$ is solvable, by (1) above there exist $\mathbf{x}$ and $\mathbf{N}$ such that $(\lambda\mathbf{x}.C[M])\mathbf{N}$ has a β-nf. Then by lemma 2 (applied to the context $(\lambda\mathbf{x}.C[\ ])\mathbf{N}$), we know $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ has a β-nf for any M'. This means $(\lambda\mathbf{x}.C[M'])\mathbf{N}$ is solvable, and consequently $C[M']$ is solvable. □

The proof presented provides an alternative to the conventional one which uses a topological argument on Böhm trees (cf.[1] Chapters 10 and 14).

Reference

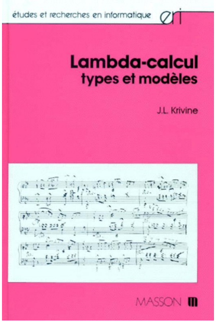
[1] H. P. Barendregt, *The Lambda Calculus* (North-Holland 1984).

Preliminaries:
~2000 lines of Abella

Main Lemma
~15 lines of text
~90 lines of Abella
~140 tactics

Some Related Work

Many other formal developments of the theory of the λ-calculus



$\leftarrow$ **Formalization** of parts of Krivine's book (1990) in Rocq by Larchey-Wendling

- Countless formalized proofs of confluence
- The previous talk!
- ...

Contextual Preorder

The **head (open) contextual preorder** is defined as:

$t \precsim^{\mathtt{h}}_{\mathcal{CO}} u$ if **for all contexts** C, $C\langle t\rangle$ is $\mathtt{h}$-normalizing implies $C\langle u\rangle$ is $\mathtt{h}$-normalizing.

- A natural extensional inequational theory
  The only non-trivial point is the inclusion of β-conversion.
- Strongly Connected with Genericity:
  Genericity says that "head diverging terms are **minimums** for the head contextual preorder".

Contextual Preorder

The **head (open) contextual preorder** is defined as:

$t \precsim^{\mathtt{h}}_{\mathcal{CO}} u$ if **for all contexts** C, $C\langle t\rangle$ is $\mathtt{h}$-normalizing implies $C\langle u\rangle$ is $\mathtt{h}$-normalizing.

- A natural extensional inequational theory

  The only non-trivial point is the inclusion of β-conversion.

- Strongly Connected with Genericity:

  Genericity says that "head diverging terms are **minimums** for the head contextual preorder".

Contextual Preorder

The **head (open) contextual preorder** is defined as:

$t \precsim^{\mathtt{h}}_{\mathcal{CO}} u$ if **for all contexts** C, $C\langle t\rangle$ is $\mathtt{h}$-normalizing implies $C\langle u\rangle$ is $\mathtt{h}$-normalizing.

- A natural extensional inequational theory

  The only non-trivial point is the inclusion of β-conversion.

- Strongly Connected with Genericity:

  Genericity says that "head diverging terms are **minimums** for the head contextual preorder".

Outline

Formalizing λ

$$\textsc{Terms} \quad t, u \quad := \quad x \mid \lambda x.t \mid tu$$

λ-terms and the predicate for inducting on them in Abella:

```
Kind  tm   type.

Type  abs  (tm -> tm) -> tm.
Type  app  tm -> tm -> tm.


Define is_tm : tm -> prop by
  nabla x, is_tm x;
  is_tm (abs T) := nabla x, is_tm (T x);
  is_tm (app T U) := is_tm T /\ is_tm U.
```

Formalizing λ

$$\textsc{Terms} \quad t, u \quad := \quad x \mid \lambda x.t \mid tu$$

λ-terms and the predicate for inducting on them in Abella:

```
Kind  tm   type.

Type  abs  (tm -> tm) -> tm.
Type  app  tm -> tm -> tm.


Define is_tm : tm -> prop by
  nabla x, is_tm x;
  is_tm (abs T) := nabla x, is_tm (T x);
  is_tm (app T U) := is_tm T /\ is_tm U.
```

Formalizing λ and β

$$\frac{}{(\lambda x.t)u \mapsto_\beta t\{x \leftarrow u\}} \quad \frac{t \to_\beta t'}{tu \to_\beta t'u} \quad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \quad \frac{u \to_\beta u'}{tu \to_\beta tu'}$$

```
Define beta : tm -> tm -> prop by
  beta (app (abs T) U) (T U);
  beta (app T U) (app T' U) := beta T T';
  beta (abs T) (abs T') := nabla x, beta (T x) (T' x);
  beta (app T U) (app T U') := beta U U'.
```

$$\frac{}{(\lambda x.t)u \to_{\mathrm{h}} t\{x \leftarrow u\}} \quad \frac{t \to_{\mathrm{h}} u}{ts \to_{\mathrm{h}} us} \quad \frac{t \to_{\mathrm{h}} u}{\lambda x.t \to_{\mathrm{h}} \lambda x.u}$$

Formalizing λ and β

$$\frac{}{(\lambda x.t)u \mapsto_\beta t\{x{\leftarrow}u\}} \qquad \frac{t \to_\beta t'}{tu \to_\beta t'u} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{u \to_\beta u'}{tu \to_\beta tu'}$$

```
Define beta : tm -> tm -> prop by
  beta (app (abs T) U) (T U);
  beta (app T U) (app T' U) := beta T T';
  beta (abs T) (abs T') := nabla x, beta (T x) (T' x);
  beta (app T U) (app T U') := beta U U'.
```

$$\frac{}{(\lambda x.t)u \to_{\mathrm{h}} t\{x{\leftarrow}u\}} \qquad \frac{t \to_{\mathrm{h}} u}{ts \to_{\mathrm{h}} us} \qquad \frac{t \to_{\mathrm{h}} u}{\lambda x.t \to_{\mathrm{h}} \lambda x.u}$$

Formalizing λ and β

$$\frac{}{(\lambda x.t)u \mapsto_\beta t\{x \leftarrow u\}} \quad \frac{t \to_\beta t'}{tu \to_\beta t'u} \quad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \quad \frac{u \to_\beta u'}{tu \to_\beta tu'}$$

```
Define beta : tm -> tm -> prop by
  beta (app (abs T) U) (T U);
  beta (app T U) (app T' U) := beta T T';
  beta (abs T) (abs T') := nabla x, beta (T x) (T' x);
  beta (app T U) (app T U') := beta U U'.
```

$$\frac{}{(\lambda x.t)u \to_{\mathrm{h}} t\{x \leftarrow u\}} \quad \frac{t \to_{\mathrm{h}} u}{ts \to_{\mathrm{h}} us} \quad \frac{t \to_{\mathrm{h}} u}{\lambda x.t \to_{\mathrm{h}} \lambda x.u}$$

Formalizing λ-theories

A λ-theory is stable by contexts...

Contextual equivalence...

We need to formalize contexts

Formalizing λ-theories

A λ-theory is stable by contexts...

Contextual equivalence...

We need to formalize contexts

Formalizing λ-theories

A λ-theory is stable by contexts...

Contextual equivalence...

We need to formalize contexts

Formalizing Contexts

A context is a term with a hole? Not really...

Set $C := \lambda x.\langle\cdot\rangle$, then $C\langle y\rangle = \lambda x.y$ and $C\langle x\rangle = \lambda x.x = \mathtt{I}$.

`ctx T CT` holds iff there exists a context C such that $C\langle \mathtt{T}\rangle = \mathtt{CT}$.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) :=
     nabla x, ctx (T x) (CT x).
```

How to apply a context to two different terms?

Formalizing Contexts

A context is a term with a hole? Not really...
Set $C := \lambda x.\langle\cdot\rangle$, then $C\langle y\rangle = \lambda x.y$ and $C\langle x\rangle = \lambda x.x = \mathtt{I}$.

ctx T CT holds iff there exists a context C such that $C\langle \mathrm{T}\rangle = \mathrm{CT}$.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) :=
     nabla x, ctx (T x) (CT x).
```

How to apply a context to two different terms?

Formalizing Contexts

A context is a term with a hole? Not really...
Set $C := \lambda x.\langle\cdot\rangle$, then $C\langle y\rangle = \lambda x.y$ and $C\langle x\rangle = \lambda x.x = \mathtt{I}$.

`ctx T CT` holds iff there exists a context C such that $C\langle\mathtt{T}\rangle = \mathtt{CT}$.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) :=
     nabla x, ctx (T x) (CT x).
```

How to apply a context to two different terms?

Formalizing Contexts

A context is a term with a hole? Not really...
Set $C := \lambda x.\langle\cdot\rangle$, then $C\langle y\rangle = \lambda x.y$ and $C\langle x\rangle = \lambda x.x = \mathtt{I}$.

`ctx T CT` holds iff there exists a context C such that $C\langle\mathtt{T}\rangle = \mathtt{CT}$.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) :=
     nabla x, ctx (T x) (CT x).
```

How to apply a context to two different terms?

Formalizing Contexts

A context is a term with a hole? Not really...
Set $C := \lambda x.\langle\cdot\rangle$, then $C\langle y\rangle = \lambda x.y$ and $C\langle x\rangle = \lambda x.x = \mathtt{I}$.

`ctx T CT` holds iff there exists a context C such that $C\langle \mathtt{T}\rangle = \mathtt{CT}$.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) :=
     nabla x, ctx (T x) (CT x).
```

How to apply a context to two different terms?

Formalizing Contexts

`ctxs T CT U CU` holds
iff there exists a context C such that $C\langle \texttt{T}\rangle = \texttt{CT}$ and $C\langle \texttt{U}\rangle = \texttt{CU}$.

```
Define ctxs : tm -> tm -> tm -> tm -> prop by
  ctxs T T U U;
  ctxs T (app A B) U (app C D) :=
    (ctxs T A U C /\ B = D /\ tm D)
    \/ (ctxs T B U D /\ A = C /\ tm C);
  nabla x, ctxs (T x) (abs CT) (U x) (abs CU):=
    nabla y, ctxs (T y) (CT y) (U y) (CU y).
```

Formalizing Contexts

`ctxs T CT U CU` holds
iff there exists a context C such that $C\langle \texttt{T}\rangle = \texttt{CT}$ and $C\langle \texttt{U}\rangle = \texttt{CU}$.

```
Define ctxs : tm -> tm -> tm -> tm -> prop by
  ctxs T T U U;
  ctxs T (app A B) U (app C D) :=
    (ctxs T A U C /\ B = D /\ tm D)
    \/ (ctxs T B U D /\ A = C /\ tm C);
  nabla x, ctxs (T x) (abs CT) (U x) (abs CU):=
    nabla y, ctxs (T y) (CT y) (U y) (CU y).
```

Formalizing Contextual Preorder

```
Define ctx_preord : tm -> tm -> prop by
  ctx_preord P Q := forall CP CQ,
    tm P -> tm Q ->
    ctxs P CP Q CQ -> head_terminating CP ->
    head_terminating CQ.
```

- `ctx_preord` is stable by contexts.
- `ctx_preord` is invariant under computation.
- `ctx_preord` has `h`-diverging terms as minimums.

Light Genericity

Light Genericity: head-diverging terms are minimum for the head open contextual preorder.

Unfolded statement:

Light Genericity: let u be head-diverging and C such that $C\langle u\rangle$ is head-normalizing then $C\langle t\rangle$ is head-normalizing for all $t \in \Lambda$.

Main difficulty: reasoning with contexts and reduction.

Light Genericity

Light Genericity: head-diverging terms are minimum for the head open contextual preorder.

Unfolded statement:

Light Genericity: let u be head-diverging and C such that $C\langle u\rangle$ is head-normalizing then $C\langle t\rangle$ is head-normalizing for all $t \in \Lambda$.

Main difficulty: reasoning with contexts and reduction.

Light Genericity

Light Genericity: head-diverging terms are minimum for the head open contextual preorder.

Unfolded statement:

Light Genericity: let u be head-diverging and C such that $C\langle u\rangle$ is head-normalizing then $C\langle t\rangle$ is head-normalizing for all $t \in \Lambda$.

Main difficulty: reasoning with contexts and reduction.

Direct proof of Light Genericity

Takahashi proves Barendregt's heavy genericity with a very short proof [Tak94] and gives as a corollary light genericity.

Key idea/trick: Reason with substitutions instead of contexts!

Light genericity as substitution: let u be $\mathtt{h}$-diverging and t such that $t\{x{\leftarrow}u\}$ is $\mathtt{h}$-normalizing then $t\{x{\leftarrow}s\}$ is $\mathtt{h}$-normalizing for all $s \in \Lambda$.

Direct proof of Light Genericity

Takahashi proves Barendregt's heavy genericity with a very short proof [Tak94] and gives as a corollary light genericity.

Key idea/trick: Reason with substitutions instead of contexts!

Light genericity as substitution: let u be $\mathtt{h}$-diverging and t such that $t\{x{\leftarrow}u\}$ is $\mathtt{h}$-normalizing then $t\{x{\leftarrow}s\}$ is $\mathtt{h}$-normalizing for all $s \in \Lambda$.

Direct proof of Light Genericity

Takahashi proves Barendregt's heavy genericity with a very short proof [Tak94] and gives as a corollary light genericity.

Key idea/trick: Reason with substitutions instead of contexts!

Light genericity as substitution: let u be $\mathtt{h}$-diverging and t such that $t\{x{\leftarrow}u\}$ is $\mathtt{h}$-normalizing then $t\{x{\leftarrow}s\}$ is $\mathtt{h}$-normalizing for all $s \in \Lambda$.

Takahashi's Trick in CbN

$$\begin{array}{rcl} C\langle u\rangle & \leftrightarrow & t_C\{x\leftarrow u_C\} \\ C\langle s\rangle & \leftrightarrow & t_C\{x\leftarrow s_C\} \end{array}$$

Trick:
Let $\mathtt{fv}(u) \cup \mathtt{fv}(s) = \{x_1, \ldots, x_k\}$, and y a fresh variable.

- $u_C := \lambda x_1. \ldots . \lambda x_k.u$ and $s_C := \lambda x_1. \ldots . \lambda x_k.s$ are closed terms.
- Consider $t_C := C\langle yx_1 \ldots x_k\rangle$, and note that:

$$\begin{array}{rcl} t_C\{y\leftarrow u_C\} & = & C\langle u_C x_1 \ldots x_k\rangle \\ & = & C\langle(\lambda x_1. \ldots . \lambda x_k.u)x_1 \ldots x_k\rangle \\ & \to_\beta^k & C\langle u\rangle \end{array}$$

- u is h-diverging implies that u_C is also h-diverging.
- $C\langle u\rangle$ is h-normalizing if and only if $t\{y\leftarrow u_C\}$ is. (also true for s and s_C)

  by the Head Normalization Theorem (and confluence, etc.)

Formalizing Takahashi's Trick

$$C\langle u\rangle \;\leftrightarrow t_C\{x{\leftarrow}u_C\}$$

Disentangling:
For any context C, there exist t_C and a variable $x \notin \mathtt{fv}(C)$ such that:

- for all terms u there exists u_C such that $t_C\{x{\leftarrow}u_C\} \to_\beta^* C\langle u\rangle$. (Moreover, if u is head divergent then u_C is head divergent.)

Some small technicalities in Abella...

Substitution Preorder:

$u \precsim_{\mathcal{S}}^{\mathrm{h}} s$ holds if
for all terms t, variables x, and lists of variables $y_1, \ldots, y_n$ with $n \geq 0$,
$t\{x{\leftarrow}\lambda y_1.\ldots.\lambda y_n.u\}$ $\to_{\mathrm{h}}$-terminating implies that $t\{x{\leftarrow}\lambda y_1.\ldots.\lambda y_n.s\}$ is $\to_{\mathrm{h}}$-terminating

The substitution preorder coincides with the contextual preorder.

Formalizing Takahashi's Trick

$$C\langle u\rangle \ \leftrightarrow t_C\{x{\leftarrow}u_C\}$$

Disentangling:
For any context C, there exist t_C and a variable $x \notin \mathtt{fv}(C)$ such that:

- for all terms u there exists u_C such that $t_C\{x{\leftarrow}u_C\} \rightarrow_\beta^* C\langle u\rangle$. (Moreover, if u is head divergent then u_C is head divergent.)

Some small technicalities in Abella...

Substitution Preorder:

$u \precsim_{\mathcal{S}}^{\mathtt{h}} s$ holds if
for all terms t, variables x, and lists of variables $y_1, \ldots, y_n$ with $n \geq 0$,
$t\{x{\leftarrow}\lambda y_1.\ldots.\lambda y_n.u\}$ $\rightarrow_{\mathtt{h}}$-terminating implies that
$t\{x{\leftarrow}\lambda y_1.\ldots.\lambda y_n.s\}$ is $\rightarrow_{\mathtt{h}}$-terminating

The substitution preorder coincides with the contextual preorder.

Maximality

Another result in Barendregt's book:

Maximality of the Head Contextual Preorder:
if $\precsim^{\mathtt{h}}_{\mathcal{CO}} \subsetneq \leq_{\mathcal{T}}$ then $\leq_{\mathcal{T}}$ is inconsistent.

$\ll$ The head contextual preorder is the largest sensible theory to study. $\gg$

Constructive Contextual Equivalence?

Proofs of maximality always starts by:

If $\mathcal{T} \vdash t \leq u$ and $t \not\lesssim^{\mathtt{h}}_{\mathcal{CO}} u$
Then $\underline{\exists C}$ such that

- $C\langle t\rangle$ is $\mathtt{h}$-normalizing and
- $C\langle u\rangle$ is $\mathtt{h}$-diverging.

...

In general, not valid in intuistionistic logic

$$\neg\forall\phi \not\Rightarrow \exists\neg\phi$$

Constructive Contextual Equivalence?

Proofs of maximality always starts by:

If $\mathcal{T} \vdash t \leq u$ and $t \not\lesssim^{\mathtt{h}}_{\mathcal{CO}} u$
Then $\underline{\exists C}$ such that

- $C\langle t\rangle$ is $\mathtt{h}$-normalizing and
- $C\langle u\rangle$ is $\mathtt{h}$-diverging.

...

In general, not valid in intuistionistic logic

$$\neg\forall\phi \not\Rightarrow \exists\neg\phi$$

Conclusions

- A small subset of Barendregt's book formalized (many rewriting theorems hidden in this presentation)
- Easy proofs that rely mostly on rewriting/operational results
- Faithful formalization of the pen-and-paper proofs

Future work:

- Constructive Contextual (In)Equivalence?
- Many results adapt to the theory of the Call-by-Value calculus (haven't formalized these)
- Other results on program equivalence to be made formal (mechanizing Böhm trees and Böhm 's theorem?
  $\Rightarrow$ intensional presentation of contextual equivalence)

Thank you!

Conclusions

- A small subset of Barendregt's book formalized (many rewriting theorems hidden in this presentation)
- Easy proofs that rely mostly on rewriting/operational results
- Faithful formalization of the pen-and-paper proofs

Future work:

- Constructive Contextual (In)Equivalence?
- Many results adapt to the theory of the Call-by-Value calculus (haven't formalized these)
- Other results on program equivalence to be made formal (mechanizing Böhm trees and Böhm 's theorem?
  $\Rightarrow$ intensional presentation of contextual equivalence)

Thank you!

Masako Takahashi.
A simple proof of the genericity lemma, pages 117–118.
Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.