Self Types for Dependently Typed Lambda Encodings

Peng Fu, Aaron Stump

The University of Iowa
Department of Computer Science

Motivations

- In practice
  - Most dependent type systems include datatype
  - Surprisingly daunting to formalize datatype system

Motivations

- In practice
  - Most dependent type systems include datatype
  - Surprisingly daunting to formalize datatype system
- In lambda calculus
  - Church encoding, Parigot encoding and Scott encoding
  - Church encoding is typable in System $\mathbf{F}$

Motivations

Church-encoded data for dependent type?

- **Inefficient to retrieve subdata**

Motivations

Church-encoded data for dependent type?

- **Inefficient to retrieve subdata**
- **Can not prove** $0 \neq 1$
  A Normalization Proof for an Impredicative Type System with Large Elimination over Integers, B. Werner

Motivations

Church-encoded data for dependent type?

- **Inefficient to retrieve subdata**
- **Can not prove** $0 \neq 1$
  A Normalization Proof for an Impredicative Type System with Large Elimination over Integers, B. Werner
- **Induction principle is not derivable**
  Metamathematical investigations of a calculus of constructions, T. Coquand
  Induction Is Not Derivable in Second Order Dependent Type Theory, H. Geuvers

Church Encoding: Inefficiency

- Church numerals
  $\bar{0} := \lambda s.\lambda z.z, \mathsf{S} := \lambda n.\lambda s.\lambda z.s\ (n\ s\ z)$
  $\bar{3}\ := \lambda s.\lambda z.s\ (s\ (s\ \ z))$
- Linear time predecessor for Church numerals
  $\mathsf{pred}\ n := \mathsf{fst}\ (n\ (\lambda p.(\mathsf{snd}\ p, \mathsf{S}\ (\mathsf{snd}\ p)))\ (0,0))$
- Parigot numerals
  $\bar{0} := \lambda s.\lambda z.z, \mathsf{S} := \lambda n.\lambda s.\lambda z.s\ \mathsf{n}\ (n\ s\ z)$
  $\bar{3}\ := \lambda s.\lambda z.s\ \bar{2}(s\ \bar{1}(s\ \bar{0}\ z))$
- Constant time Parigot predessesor
  $\mathsf{pred}_p\ n := n\ (\lambda x.\lambda y.x)\ 0$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{array}{lll}
x =_A y & := & \forall C : A \to *. C\ x \to C\ y \\
\bot & := & \forall X : *. X \\
0 =_{\mathsf{Nat}} 1 \to \bot & := & (\forall C : \mathsf{Nat} \to *. C\ 0 \to C\ 1) \to \forall X : *. X
\end{array}
$$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{array}{lcl}
x =_A y & := & \forall C : A \to *.C\,x \to C\,y \\
\bot & := & \forall X : *.X \\
0 =_{\mathsf{Nat}} 1 \to \bot & := & (\forall C : \mathsf{Nat} \to *.C\,0 \to C\,1) \to \forall X : *.X
\end{array}
$$

- $0 =_{\mathsf{Nat}} 1 \to \bot$ is underivable
  - $\vdash_{cc} t : 0 \neq_{\mathsf{Nat}} 1$ implies $\vdash_{F_\omega} |t| : |0 \neq_{\mathsf{Nat}} 1|$
  - $|0 =_{\mathsf{Nat}} 1 \to \bot| := \forall C.(C \to C) \to \forall X.X$ in $\mathbf{F}_\omega$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{aligned}
x =_A y &:= \forall C : A \to *. C\ x \to C\ y \\
\bot &:= \forall A : *. \Pi x : A. \Pi y : A. x =_A y \\
0 =_{\mathsf{Nat}} 1 \to \bot &:= (\forall C : \mathsf{Nat} \to *. C\ 0 \to C\ 1) \\
&\quad \to (\forall A : *. \Pi x : A. \Pi y : A. x =_A y)
\end{aligned}
$$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{aligned}
x =_A y &:= \forall C : A \to *. C\ x \to C\ y \\
\bot &:= \forall A : *. \Pi x : A. \Pi y : A. x =_A y \\
0 =_{\mathsf{Nat}} 1 \to \bot &:= (\forall C : \mathsf{Nat} \to *. C\ 0 \to C\ 1) \\
&\quad \to (\forall A : *. \Pi x : A. \Pi y : A. x =_A y)
\end{aligned}
$$

- $\bot$ is uninhabited in $\mathbf{CC}$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{aligned}
x =_A y \quad &:= \quad \forall C : A \to *.C\,x \to C\,y \\
\bot \quad &:= \quad \forall A : *.\Pi x : A.\Pi y : A.x =_A y \\
0 =_{\mathsf{Nat}} 1 \to \bot \quad &:= \quad (\forall C : \mathsf{Nat} \to *.C\,0 \to C\,1) \\
&\quad\;\; \to (\forall A : *.\Pi x : A.\Pi y : A.x =_A y)
\end{aligned}
$$

- $\bot$ is uninhabited in $\mathbf{CC}$
- $0 =_{\mathsf{Nat}} 1 \to \bot$ is derivable in $\mathbf{CC}$

Church Encoding: Underivability of $0 \neq 1$

- Calculus of Construction($\mathbf{CC}$)

$$
\begin{aligned}
x =_A y &:= \forall C : A \to *.C\ x \to C\ y \\
\bot &:= \forall A : *.\Pi x : A.\Pi y : A.x =_A y \\
0 =_{\mathsf{Nat}} 1 \to \bot &:= (\forall C : \mathsf{Nat} \to *.C\ 0 \to C\ 1) \\
&\quad \to (\forall A : *.\Pi x : A.\Pi y : A.x =_A y)
\end{aligned}
$$

- $\bot$ is uninhabited in $\mathbf{CC}$
- $0 =_{\mathsf{Nat}} 1 \to \bot$ is derivable in $\mathbf{CC}$
- $|0 =_{\mathsf{Nat}} 1 \to \bot| := \forall C.(C \to C) \to \forall A.(A \to A \to \forall X.(X \to X))$ in $\mathbf{F}_\omega$

Church Encoding: Induction Principle

- Induction in **CC**
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to \Pi x : \mathsf{Nat}.P\,x$

Church Encoding: Induction Principle

- Induction in $\mathbf{CC}$
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to \Pi x : \mathsf{Nat}.P\,x$
- Underivability
  $P : \mathsf{Nat} \to *, z_1 : \forall y : \mathsf{Nat}.Py \to P(\mathsf{S}y), z_2 : P\,\bar{0}, x : \mathsf{Nat} \vdash \{?\} : P\,x$

Church Encoding: Induction Principle

- Induction in **CC**
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$
- Underivability
  $P : \mathsf{Nat} \to *, z_1 : \forall y : \mathsf{Nat}.Py \to P(\mathsf{S}y), z_2 : P\ \bar{0}, x : \mathsf{Nat} \vdash \{?\} : P\ x$
- Observation
  $... \vdash z_1\ z_2 : P\ (\mathsf{S}\bar{0})$
  $... \vdash z_1(z_1\ z_2) : P\ (\mathsf{SS}\bar{0})$
  $... \vdash z_1(z_1(z_1\ z_2)) : P\ (\mathsf{SSS}\bar{0})$

Church Encoding: Induction Principle

- Induction in **CC**
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$
- Underivability
  $P : \mathsf{Nat} \to *, z_1 : \forall y : \mathsf{Nat}.Py \to P(\mathsf{S}y), z_2 : P\ \bar{0}, x : \mathsf{Nat} \vdash \{?\} : P\ x$
- Observation
  $... \vdash z_1\ z_2 : P\ (\mathsf{S}\bar{0})$
  $... \vdash z_1(z_1\ z_2) : P\ (\mathsf{SS}\bar{0})$
  $... \vdash z_1(z_1(z_1\ z_2)) : P\ (\mathsf{SSS}\bar{0})$
- Self Type: $\iota x.F$
  $\dfrac{\Gamma \vdash t : \iota x.F}{\Gamma \vdash t : [t/x]F}$ *selfInst* $\quad \dfrac{\Gamma \vdash t : [t/x]F}{\Gamma \vdash t : \iota x.F}$ *selfGen*

Church Encoding: Induction Principle

- Induction in **CC**
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$
- Underivability
  $P : \mathsf{Nat} \to *, z_1 : \forall y : \mathsf{Nat}.Py \to P(\mathsf{S}y), z_2 : P\ \bar{0}, x : \mathsf{Nat} \vdash \{?\} : P\ x$
- Observation
  $... \vdash z_1\ z_2 : P\ (\mathsf{S}\bar{0})$
  $... \vdash z_1(z_1\ z_2) : P\ (\mathsf{SS}\bar{0})$
  $... \vdash z_1(z_1(z_1\ z_2)) : P\ (\mathsf{SSS}\bar{0})$
- Self Type: $\iota x.F$
  $\dfrac{\Gamma \vdash t : \iota x.F}{\Gamma \vdash t : [t/x]F}$ *selfInst* $\quad \dfrac{\Gamma \vdash t : [t/x]F}{\Gamma \vdash t : \iota x.F}$ *selfGen*
- Positive recursive type definition and implicit preduct
  $\mathsf{Nat} :=$
  $\iota x.\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to P\ x$

Church Encoding: Induction Principle

- Induction in **CC**
  $\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$
- Underivability
  $P : \mathsf{Nat} \to *, z_1 : \forall y : \mathsf{Nat}.Py \to P(\mathsf{S}y), z_2 : P\ \bar{0}, x : \mathsf{Nat} \vdash \{?\} : P\ x$
- Observation
  $... \vdash z_1\ z_2 : P\ (\mathsf{S}\bar{0})$
  $... \vdash z_1(z_1\ z_2) : P\ (\mathsf{SS}\bar{0})$
  $... \vdash z_1(z_1(z_1\ z_2)) : P\ (\mathsf{SSS}\bar{0})$
- Self Type: $\iota x.F$
  $\dfrac{\Gamma \vdash t : \iota x.F}{\Gamma \vdash t : [t/x]F}$ *selfInst* $\dfrac{\Gamma \vdash t : [t/x]F}{\Gamma \vdash t : \iota x.F}$ *selfGen*
- Positive recursive type definition and implicit preduct
  $\mathsf{Nat} :=$
  $\iota x.\forall P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to P\ x$
- Induction now is derivable
  $ind := \lambda s.\lambda z.\lambda n.n\ s\ z$

System S: Formulation

$$\frac{\Gamma, x : \iota x.T \vdash T : *}{\Gamma \vdash \iota x.T : *} \qquad \frac{\Gamma \vdash t : [t/x]T \quad \Gamma \vdash \iota x.T : *}{\Gamma \vdash t : \iota x.T}$$

$$\frac{\Gamma \vdash t : \iota x.T}{\Gamma \vdash t : [t/x]T} \qquad \frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : *}{\Gamma \vdash \lambda x.t : \Pi x : T_1.T_2}$$

$$\frac{\Gamma \vdash t : \Pi x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash tt' : [t'/x]T_2} \qquad \frac{\Gamma \vdash t : \forall x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash t : [t'/x]T_2}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : * \quad x \notin \mathrm{FV}(t)}{\Gamma \vdash t : \forall x : T_1.T_2}$$

$$\frac{\Gamma \vdash t : T_1 \quad \Gamma \vdash T_1 \cong T_2 \quad \Gamma \vdash T_2 : *}{\Gamma \vdash t : T_2}$$

System S: Parigot Numerals

- Let μ_p be

  $(\mathsf{Nat} : *) \mapsto \iota x.\forall C : \mathsf{Nat} \to *.(\Pi\, n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C\ x$
  $(\mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}) \mapsto \lambda n.\lambda s.\lambda z.s\ n\ (n\ s\ z)$
  $(0 : \mathsf{Nat}) \mapsto \lambda s.\lambda z.z$

System S: Parigot Numerals

- Let μ_p be
  $(\mathsf{Nat} : *) \mapsto \iota x.\forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C\ x$
  $(\mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}) \mapsto \lambda n.\lambda s.\lambda z.s\ n\ (n\ s\ z)$
  $(0 : \mathsf{Nat}) \mapsto \lambda s.\lambda z.z$
- Check $\mu_p \vdash \lambda n.\lambda s.\lambda z.s\ n\ (n\ s\ z) : \mathsf{Nat} \to \mathsf{Nat}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{...\vdash s\ n : C\ n \to C\ (\mathsf{S}\ n) \quad ...\vdash n\ s\ z : C\ n}{n : \mathsf{Nat}, s : \forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)), z : C\ 0 \vdash s\ n\ (n\ s\ z) : C\ (\mathsf{S}n)}}{n : \mathsf{Nat} \vdash \lambda s.\lambda z.s\ n\ (n\ s\ z) : \forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C\ (\mathsf{S}n)}}{\vdash \lambda s.\lambda z.sn(nsz) : \forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C(\lambda s.\lambda z.sn(nsz))}}{n : \mathsf{Nat} \vdash \lambda s.\lambda z.s\ n\ (n\ s\ z) : \iota x.\forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C\ x}}{\mu_p, n : \mathsf{Nat} \vdash \lambda s.\lambda z.s\ n\ (n\ s\ z) : \mathsf{Nat}}$$

Note: $n : \forall C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.C\ n \to C\ (\mathsf{S}\ n)) \to C\ 0 \to C\ n$

System S: Strong Normalization

- Erasure from $\mathbf{S}$ to $\mathbf{F}_\omega$ with positive definitions. $\Gamma \vdash T \triangleright A^\kappa$.

$$\frac{F(\kappa') = \kappa \quad (X : \kappa') \in \Gamma}{\Gamma \vdash X \triangleright X^\kappa} \qquad \frac{\Gamma \vdash T \triangleright T_1^\kappa}{\Gamma \vdash \iota x.T \triangleright T_1^\kappa}$$

$$\frac{\Gamma \vdash T_2 \triangleright T^\kappa}{\Gamma \vdash \forall x : T_1.T_2 \triangleright T^\kappa} \qquad \frac{\Gamma \vdash T \triangleright T_1^\kappa}{\Gamma \vdash \lambda x.T \triangleright T_1^\kappa}$$

System S: Strong Normalization

- Erasure from $\mathbf{S}$ to $\mathbf{F}_\omega$ with positive definitions. $\Gamma \vdash T \triangleright A^\kappa$.

$$\frac{F(\kappa') = \kappa \quad (X : \kappa') \in \Gamma}{\Gamma \vdash X \triangleright X^\kappa} \qquad \frac{\Gamma \vdash T \triangleright T_1^\kappa}{\Gamma \vdash \iota x.T \triangleright T_1^\kappa}$$

$$\frac{\Gamma \vdash T_2 \triangleright T^\kappa}{\Gamma \vdash \forall x : T_1.T_2 \triangleright T^\kappa} \qquad \frac{\Gamma \vdash T \triangleright T_1^\kappa}{\Gamma \vdash \lambda x.T \triangleright T_1^\kappa}$$

- Show SN for $\mathbf{F}_\omega$ with positive type definitions
  - Construct complete lattice $(\rho[\![\kappa]\!], \subseteq_\kappa, \bigcap_\kappa)$ from complete lattice $(\mathfrak{R}_\rho, \subseteq, \cap)$
    where
    $\rho[\![*]\!] := \mathfrak{R}_\rho$
    $\rho[\![\kappa \to \kappa']\!] := \{f \mid \forall a \in \rho[\![\kappa]\!], f(a) \in \rho[\![\kappa']\!]\}$
  - Least fix point exists for $b \mapsto \rho[\![T^\kappa]\!]_{\sigma[b/X^\kappa]}$ with $b \in \rho[\![\kappa]\!]$

System S: Subject Reduction

- View typing as a form of reduction. e.g. $\iota x.T \to_\iota [t/x]T$.
- $\to_\iota$ commutes with $\to_\beta$, thus $\to_{\iota,\beta}$ is confluent.
- Adapt Barendregt's subject reduction proof of $\lambda 2$ to handle implicit product and type level equality.
- If $\Pi x : T_1.T_2 \cong_\Gamma \Pi x : T_1'.T_2'$, then $T_1 \cong_\Gamma T_1'$, $T_2 \cong_\Gamma T_2'$.

Summary

- $0 \neq 1$ is provable with a change of notion of contradiction.
- Introduce Self type to derive induction principle.
- Devised a type system called $\mathbf{S}$.
- We prove $\mathbf{S}$ is convergent(at term level) and type preserving.
- Extended version is available from both author's website.