Church Encoding with Dependent Type and Self Type

Peng Fu, Aaron Stump

Dept. of Computer Science
University of Iowa

September 23, 2013

Motivation

- Church encoding in system $\mathbf{F}$.

Motivation

- Church encoding in system $\mathbf{F}$.
- Dependent type theory.

Motivation

- Church encoding in system $\mathbf{F}$.
- Dependent type theory.
- Primitive inductive data type.

Church Encoding

- Why not use Church encoded data?

Church Encoding

- Why not use Church encoded data?
- Inefficiency to retrieve subdata.

Church Encoding

- Why not use Church encoded data?
- Inefficiency to retrieve subdata.
- Can not prove $0 \neq 1$(e.g. Calculus of Construction $\mathbf{CC}$).

Church Encoding

- Why not use Church encoded data?
- Inefficiency to retrieve subdata.
- Can not prove $0 \neq 1$(e.g. Calculus of Construction $\mathbf{CC}$).
- Induction principle is not derivable(e.g. $\mathbf{CC}$).

Church Encoding: Inefficiency

- Linear time to compute predecessor:
  pred $(\text{Succ } \bar{n}) \underbrace{\rightsquigarrow \ldots \rightsquigarrow}_{\geq n+1} \bar{n}$
- "Unintuitive" predecessor function.
  pred $:= \lambda n.\lambda f.\lambda x.n\ (\lambda g.\lambda h.h\ (g\ f))(\lambda u.x)\ (\lambda u.u)$
- It is inherent to Church encodings.

Underivability of $0 \neq 1$

- Depends on the notion of contradiction.
- Calculus of Construction:

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

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

Underivability of $0 \neq 1$

- A change of notion of contradiction.
- Calculus of Construction:

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

- $\boxed{\bot}$ is uninhabited in $\mathbf{CC}$.
- $0 =_{\mathsf{Nat}} 1 \to \boxed{\bot}$ is derivable in $\mathbf{CC}$.
- $0 \neq 1$ in $\mathbf{CC}$ is mapped to $\Pi C.(C \to C) \to (\Pi A.\Pi C.C \to C)$ in $\mathbf{F}$.

Underivability of Induction Principle

- Depends on the formulation of the logical system.
- Calculus of Construction:
  $\mathit{Ind} := \Pi P : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$.
- Let
  $\Gamma = P : \mathsf{Nat} \to *, s : \Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y)), z : P\ \bar{0}, x : \mathsf{Nat}$
  $\Gamma \vdash ? : P\ x$

Underivability of Induction Principle

$\Gamma = P : \mathsf{Nat} \to *, s : \Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y)), z : P\ \bar{0}, x : \mathsf{Nat}$

- Observe that:
  $\Gamma \vdash z : P\ \bar{0}$
  $\Gamma \vdash s\ \bar{0}\ z : P\ \bar{1}$
  $\Gamma \vdash s\ \bar{1}\ (s\ \bar{0}\ z) : P\ \bar{2}$

Underivability of Induction Principle

$\Gamma = P : \mathsf{Nat} \to *, s : \Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y)), z : P\,\bar{0}, x : \mathsf{Nat}$

- Observe that:
  $\Gamma \vdash z : P\,\bar{0}$
  $\Gamma \vdash s\,\bar{0}\,z : P\,\bar{1}$
  $\Gamma \vdash s\,\bar{1}\,(s\,\bar{0}\,z) : P\,\bar{2}$
- A new notion of Lambda numerals:
  $\bar{0} := \lambda s.\lambda z.z :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\bar{0}$
  $\bar{1} := \lambda s.\lambda z.s\;0\;z :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\bar{1}$
  $\bar{2} := \lambda s.\lambda z.s\;1\;(s\,\bar{0}\,z) :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\bar{2}$
  $\mathsf{S} := \lambda n.\lambda s.\lambda z.s\;n\;(n\;s\;z)$

Underivability of Induction Principle

$\Gamma = P : \mathsf{Nat} \to *, s : \Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y)), z : P\,\bar{0}, x : \mathsf{Nat}$

- Observe that:
  $\Gamma \vdash z : P\,\bar{0}$
  $\Gamma \vdash s\,\bar{0}\,z : P\,\bar{1}$
  $\Gamma \vdash s\,\bar{1}\,(s\,\bar{0}\,z) : P\,\bar{2}$
- A new notion of Lambda numerals:
  $\boxed{\bar{0}} := \lambda s.\lambda z.z :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\boxed{\bar{0}}$
  $\boxed{\bar{1}} := \lambda s.\lambda z.s\,0\,z :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\boxed{\bar{1}}$
  $\boxed{\bar{2}} := \lambda s.\lambda z.s\,1\,(s\,\bar{0}\,z) :$
  $(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\boxed{\bar{2}}$
  $\mathsf{S} := \lambda n.\lambda s.\lambda z.s\,n\,(n\,s\,z)$
- $\mathsf{Nat} := \Pi P : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\boxed{\bar{n}}$ for every $\boxed{\bar{n}}$?

Self Type: Introduction

Nat $:= \Pi P : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,\bar{n}$ for every $\bar{n}$?

- We introduce *self* type.
- Nat $:=$
  $\iota x.\Pi P : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\,\bar{0} \to P\,x$
- Typing rules:
  $\dfrac{\Gamma \vdash t : \iota x.T}{\Gamma \vdash t : [t/x]T}$ *SelfInst* $\quad \dfrac{\Gamma \vdash t : [t/x]T}{\Gamma \vdash t : \iota x.T}$ *SelfGen*
- Self type formation rule:
  $\dfrac{\Gamma, x : \iota x.T \vdash T : *}{\Gamma \vdash \iota x.T : *}$

Self Type: Handling Recursive Definition

Nat $:=$
$\iota x.\Pi P : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to Px$

- The encoding is not quite Church-like yet.
  $0 := \lambda s.\lambda z.z$
  $\mathsf{S} := \lambda n.\lambda s.\lambda z.s\ n\ (n\ s\ z)$
- We need Miquel's implicit product.
  Nat $:=$
  $\iota x.\Pi P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to Px$
- Now we have Church numerals:
  $0 := \lambda s.\lambda z.z$
  $\mathsf{S} := \lambda n.\lambda s.\lambda z.s\ (n\ s\ z)$
- Induction now is derivable:
  $Ind : \Pi P : \mathsf{Nat} \to *.(\forall y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to \Pi x : \mathsf{Nat}.P\ x$
  $Ind := \lambda s.\lambda z.\lambda n.n\ s\ z$.

Summary and Results

- $0 \neq 1$ is provable with a change of notion of contradiction.
- Introduce Self type to derive induction principle.

Some Results

- Self type is incorporated in a type system called $\mathbf{S}$.
- We prove $\mathbf{S}$ can be erased to $\mathbf{F}_\omega$, thus establishing consistency.
- We prove preservation theorem for $\mathbf{S}$.

Thank you for listening!

- Questions?