The Structural Theory of Pure Type Systems

Cody Roux   Floris van Doorn

Carnegie Mellon University

July 15, 2014

Motivation

Consider the induction principle in Peano Arithmetic:

For all formulae $\varphi(x)$ the formula

$$\varphi(0) \to \forall n(\varphi(n) \to \varphi(n+1)) \to \forall n\ \varphi(n).$$

holds.

We can reify the quantification over φ in second order arithmetic:

$$\forall\varphi\big(\varphi(0) \to \forall n(\varphi(n) \to \varphi(n+1)) \to \forall n\ \varphi(n)\big).$$

Motivation

We want to formalize the process of reification of universal quantifiers.

Motivation

Questions:

- How to do this?
  - When reifying quantifiers over formulae in Peano Arithmetic we could obtain both second-order arithmetic and ACA_0.
- Is the reification conservative?
  - ACA_0 is conservative over Peano Arithmetic.
  - Second-order arithmetic is much stronger.

Approach

By the Curry-Howard isomorphism:

Logic	$\Longleftrightarrow$	**Type Theory**
universal quantification ($\forall$)	$\Longleftrightarrow$	(dependent) function type (Π)
$\forall$ proof rule	$\Longleftrightarrow$	λ-abstraction

We will try to answer our questions using Pure Type Systems.

Pure Type Systems

Pure Type Systems

- are a generic framework of type theories.
- only allow universal quantification/dependent function spaces.

Pure Type Systems

A Pure Type Systems consists of

- A set of sorts $\mathcal{S}$
- A set of axioms $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$
- A set of rules $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$

That's it!

Sorts

Informally, sorts $s, *, \Box, \ldots \in \mathcal{S}$ represent a class of objects.

Example

$*$ may represent the class of propositions.

$\Box$ may represent the class of types.

Axioms

Informally, $(s_1, s_2) \in \mathcal{A}$ means that s_1 is a member of the class s_2.

Example

$$(*, \Box) \in \mathcal{A}$$

Rules

Informally, $(s_1, s_2, s_3) \in \mathcal{R}$ means:

You can quantify over an element of s_2 parametrized over an element of s_1, and the result lives in the class s_3.

If $A : s_1$ and $B(x) : s_2$ whenever $x : A$, then $\Pi x\!:\!A.\ B(x) : s_3$.

Example

If we have the rule $(\square, *, *)$ we have

$$\vdash \Pi A\!:\!*.\ A : *$$

Pure Type Systems

Given a PTS, we have the following type system.

Sort formation

$$\textbf{axiom}\ \frac{\Gamma \vdash}{\Gamma \vdash s_1 : s_2}\ (s_1, s_2) \in \mathcal{A}$$

$$\textbf{prod}\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma,\ x : A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.\ B : s_3}\ (s_1, s_2, s_3) \in \mathcal{R}$$

Pure Type Systems

Term formation

$$\textbf{var}\ \frac{\Gamma,\ x : A, \Delta \vdash}{\Gamma,\ x : A, \Delta \vdash x : A}$$

$$\textbf{abs}\ \frac{\Gamma,\ x : A \vdash t : B \qquad \Gamma \vdash \Pi x{:}A.\ B : s}{\Gamma \vdash \lambda x{:}A.\ t : \Pi x{:}A.\ B}\ s \in \mathcal{S}$$

$$\textbf{app}\ \frac{\Gamma \vdash t : \Pi x{:}A.\ B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B\{x \mapsto u\}}$$

Pure Type Systems

Conversion

$$\textbf{conv}\ \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A' : s}{\Gamma \vdash t : A'}\ A \simeq_\beta A',\ s \in \mathcal{S}$$

Here $\simeq_\beta$ is β-conversion, generated by

$$(\lambda x{:}A.\ t)u \rightsquigarrow_\beta t\{x \mapsto u\}.$$

We omit the rules for creating contexts.

Simply Typed Lambda Calculus

The STLC can be encoded as PTS using

$$\mathcal{S} = \{*, \square\}$$
$$\mathcal{A} = \{(*, \square)\}$$
$$\mathcal{R} = \{(*, *, *)\}$$

Example

In STLC

$$A : *, B : * \vdash \lambda a\!:\!A.\lambda b\!:\!B.\ a : A \to B \to A$$

Note: $A \to B$ abbreviates $\Pi x\!:\!A.\ B$.

Examples of PTSs

Name	Sorts $\mathcal{S}$	Axioms $\mathcal{A}$	Rules $\mathcal{R}$
STLC	$*, \Box$	$(*, \Box)$	$(*, *, *)$
$* : *$	$*$	$(*, *)$	$(*, *, *)$
LF/λP	$*, \Box$	$(*, \Box)$	$(*, *, *), (*, \Box, \Box)$
System F	$*, \Box$	$(*, \Box)$	$(*, *, *), (\Box, *, *)$
U^-	$*, \Box, \triangle$	$(*, \Box)$, $(\Box, \triangle)$	$(*, *, *), (\Box, *, *)$, $(\Box, \Box, \Box), (\triangle, \Box, \Box)$
CC	$*, \Box$	$(*, \Box)$	$(*, *, *), (*, \Box, \Box)$, $(\Box, *, *), (\Box, \Box, \Box)$
CC^ω (core of Coq)	$\Box_i$ $(i \in \mathbb{N})$	$(\Box_i, \Box_j)$ $(i < j)$	$(\Box_i, \Box_0, \Box_0)$, $(\Box_i, \Box_j, \Box_k)$ $(k \geq i, j)$

Normalization

A PTS is (weakly) normalizing iff

$$\Gamma \vdash t : T \Rightarrow t \text{ has a } \beta\text{-normal form.}$$

Normalization implies

- the decidability of type-checking.
- the consistency of the system interpreted as a logic.

Normalization

Normalization is hard to predict:

Name	Axioms $\mathcal{A}$	Rules $\mathcal{R}$	Norm.
STLC	$(*, \square)$	$(*, *, *)$	Yes
$* : *$	$(*, *)$	$(*, *, *)$	No
LF/λP	$(*, \square)$	$(*, *, *)$, $(*, \square, \square)$	Yes
System F	$(*, \square)$	$(*, *, *)$, $(\square, *, *)$	Yes
U^-	$(*, \square)$, $(\square, \triangle)$	$(*, *, *)$, $(\square, *, *)$, $(\square, \square, \square)$, $(\triangle, \square, \square)$	No
CC	$(*, \square)$	$(*, *, *)$, $(*, \square, \square)$, $(\square, *, *)$, $(\square, \square, \square)$	Yes
CC^ω (core of Coq)	$(\square_i, \square_j)$ $(i < j)$	$(\square_i, \square_0, \square_0)$, $(\square_i, \square_j, \square_k)$ $(k \geq i, j)$	Yes

Proposal

Our proposal:

- Conder the study of the class of PTSs as a whole rather than individually.
- Examine normalization preserving operations.

We call this the Structual Theory of PTSs.

First observation

Given PTSs $\mathbf{P}$ and $\mathbf{Q}$ we can define the disjoint union $\mathbf{P} + \mathbf{Q}$ by taking the disjoint union of the sorts, axioms and rules.

Theorem

If

$$\Gamma \vdash_{\mathbf{P}+\mathbf{Q}} t : T$$

then

$$\Gamma' \vdash_{\mathbf{P}} t : T \quad \text{or} \quad \Gamma' \vdash_{\mathbf{Q}} t : T$$

for some $\Gamma' \subseteq \Gamma$.

This implies that if $\mathbf{P}$ and $\mathbf{Q}$ are normalizing, then $\mathbf{P} + \mathbf{Q}$ is normalizing.

Second observation

We can add additional rules to $\mathbf{P} + \mathbf{Q}$.

Example

Let Terms be the PTS

$$\begin{aligned}
\mathcal{S}_{\mathsf{Terms}} &= \{\mathsf{Set}, \mathsf{Fun}, \mathsf{Univ}\} \\
\mathcal{A}_{\mathsf{Terms}} &= \{(\mathsf{Set}, \mathsf{Univ})\} \\
\mathcal{R}_{\mathsf{Terms}} &= \{(\mathsf{Set}, \mathsf{Set}, \mathsf{Fun}), (\mathsf{Set}, \mathsf{Fun}, \mathsf{Fun})\}
\end{aligned}$$

This is a term language with only first-order terms.

Example

$$A : \mathsf{Set} \vdash A \to A \to A : \mathsf{Fun}$$

$$A : \mathsf{Set} \vdash \lambda xy\!:\!A.\ x : A \to A \to A$$

Example

Step 1
We can build a new PTS FOL by

- Taking the direct sum Terms $+$ STLC
- Adding the rules $(\mathrm{Set}, *, *)$ and $(\mathrm{Set}, \square, \square)$
  - This allows for parametrized propositions

Example

$$A : \mathrm{Set}, P : A \to * \vdash \Pi a : A.\ P(a) : *$$

This allows us to formulate the induction principle for a single formula:

$$\Gamma = \mathbb{N} : \mathrm{Set},\ 0 : \mathbb{N},\ S : \mathbb{N} \to \mathbb{N},\ \varphi : \mathbb{N} \to *$$

$$\Gamma \vdash \varphi(0) \to (\Pi n{:}\mathbb{N}.\ \varphi(n) \to \varphi(S\ \ n)) \to \Pi m{:}\mathbb{N}.\ \varphi(m) : *$$

Example

Step 2

We can create a new PTS WSOL by

- Taking FOL;
- Adding a new sort $*'$;
- Adding a new rule $(\Box, *, *')$.

Now we can formulate the induction principle for all formulae:

$$\Gamma = \mathbb{N} : \mathsf{Set},\ 0 : \mathbb{N},\ S : \mathbb{N} \to \mathbb{N}$$

$$\Gamma \vdash \Pi\varphi{:}\mathbb{N} \to *.\ \varphi(0) \to (\Pi n{:}\mathbb{N}.\ \varphi(n) \to \varphi(S\ n)) \to \Pi m{:}\mathbb{N}.\ \varphi(m) : *'$$

Reification of quantification!

Example

Fact

The PTSs FOL and WSOL are normalizing

Our result shows this follows from the normalization STLC!

Main result 1

We define $\forall \mathbf{P}.\mathbf{Q}$ to be $\mathbf{P} + \mathbf{Q}$ with added rules

$$(s, k, k), \quad (s \in \mathcal{S}_{\mathbf{P}},\ k \in \mathcal{S}_{\mathbf{Q}})$$

Intuition

$\mathbf{Q}$ is a logic, and $\mathbf{P}$ are terms.
Then $\forall \mathbf{P}.\mathbf{Q}$ is the logic $\mathbf{Q}$ where quantification over the terms in $\mathbf{P}$ is allowed.

Example

$$\mathsf{FOL} \quad \subseteq \quad \forall\, \mathsf{Terms}\,.\,\mathsf{STLC}$$

Main result 1

Theorem

If $\mathbf{P}$ and $\mathbf{Q}$ are normalizing, then $\forall\mathbf{P}.\mathbf{Q}$ is normalizing.

In fact, $\forall\mathbf{P}.\mathbf{Q}$ is a conservative extension of $\mathbf{Q}$.

Main result 2

We define $\mathbf{P}_{\text{poly}}$ to be $\mathbf{P}$ with added sorts

$$k^s, \quad (s, k \in \mathcal{S}_{\mathbf{P}})$$

and added rules

$$(s, k, k^s), (s, k^s, k^s) \quad (s, k \in \mathcal{S}_{\mathbf{P}})$$

Intuition

This allows for quantification over any free variable in $\mathbf{P}$.
k^s is the sort of s-parametrized ks

Example

$$\text{WSOL} \quad \subseteq \quad \text{FOL}_{\text{poly}}$$

Main result 2

Theorem

If $\mathbf{P}$ is normalizing, then $\mathbf{P}_{\mathrm{poly}}$ is normalizing.

Moreover, $\mathbf{P}_{\mathrm{poly}}$ is a conservative extension of $\mathbf{P}$.

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall \mathbf{P}.\mathbf{Q}$ we partition $\to_\beta$ into three reductions:

- $\mathbf{P}$-reductions $\to_{\mathbf{P}}$ from abstractions from $\mathbf{P}$;
- $\mathbf{Q}$-reductions $\to_{\mathbf{Q}}$ from abstractions from $\mathbf{Q}$;
- $\mathbf{I}$-reductions $\to_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall \mathbf{P}.\mathbf{Q}$:

$\forall \mathbf{P}.\mathbf{Q} \qquad t$

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall\mathbf{P}.\mathbf{Q}$ we partition $\rightarrow_\beta$ into three reductions:

- **P**-reductions $\rightarrow_{\mathbf{P}}$ from abstractions from **P**;
- **Q**-reductions $\rightarrow_{\mathbf{Q}}$ from abstractions from **Q**;
- **I**-reductions $\rightarrow_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall\mathbf{P}.\mathbf{Q}$:

$$\begin{array}{cc} \forall\mathbf{P}.\mathbf{Q} & t \\ & \downarrow \\ \mathbf{Q} & \lfloor t \rfloor \end{array}$$

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall\mathbf{P}.\mathbf{Q}$ we partition $\to_\beta$ into three reductions:

- **P**-reductions $\to_{\mathbf{P}}$ from abstractions from **P**;
- **Q**-reductions $\to_{\mathbf{Q}}$ from abstractions from **Q**;
- **I**-reductions $\to_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall\mathbf{P}.\mathbf{Q}$:

$$\begin{array}{llllll} \forall\mathbf{P}.\mathbf{Q} & t & & & & \\ & \downarrow & & & & \\ \mathbf{Q} & \lfloor t \rfloor & \to_{\mathbf{Q}} & \cdots & \to_{\mathbf{Q}} & u \end{array}$$

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall\mathbf{P}.\mathbf{Q}$ we partition $\to_\beta$ into three reductions:

- **P**-reductions $\to_{\mathbf{P}}$ from abstractions from **P**;
- **Q**-reductions $\to_{\mathbf{Q}}$ from abstractions from **Q**;
- **I**-reductions $\to_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall\mathbf{P}.\mathbf{Q}$:

$$\begin{array}{ccccccc}
\forall\mathbf{P}.\mathbf{Q} & t & \to_{\mathbf{I}}^{*}\to_{\mathbf{Q}} & \cdots & \to_{\mathbf{I}}^{*}\to_{\mathbf{Q}} & u' \\
 & \downarrow & & \downarrow & & \downarrow \\
\mathbf{Q} & \lfloor t \rfloor & \to_{\mathbf{Q}} & \cdots & \to_{\mathbf{Q}} & u
\end{array}$$

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall\mathbf{P}.\mathbf{Q}$ we partition $\rightarrow_\beta$ into three reductions:

- $\mathbf{P}$-reductions $\rightarrow_{\mathbf{P}}$ from abstractions from $\mathbf{P}$;
- $\mathbf{Q}$-reductions $\rightarrow_{\mathbf{Q}}$ from abstractions from $\mathbf{Q}$;
- $\mathbf{I}$-reductions $\rightarrow_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall\mathbf{P}.\mathbf{Q}$:

$$\begin{array}{ccccccccc}
\forall\mathbf{P}.\mathbf{Q} & t & \rightarrow_{\mathbf{I}}^{*}\rightarrow_{\mathbf{Q}} & \cdots & \rightarrow_{\mathbf{I}}^{*}\rightarrow_{\mathbf{Q}} & u' & \rightarrow_{\mathbf{I}}^{*} & v \\
 & \downarrow & & \downarrow & & \downarrow & & \\
\mathbf{Q} & \lfloor t \rfloor & \rightarrow_{\mathbf{Q}} & \cdots & \rightarrow_{\mathbf{Q}} & u & &
\end{array}$$

Proof sketch

The proof uses ideas from [Bernardy and Lasson (2011)]

For the normalization of $\forall\mathbf{P}.\mathbf{Q}$ we partition $\to_\beta$ into three reductions:

- $\mathbf{P}$-reductions $\to_{\mathbf{P}}$ from abstractions from $\mathbf{P}$;
- $\mathbf{Q}$-reductions $\to_{\mathbf{Q}}$ from abstractions from $\mathbf{Q}$;
- $\mathbf{I}$-reductions $\to_{\mathbf{I}}$ from the new added rules.

We want to give a β-normal form for a term t with type in $\forall\mathbf{P}.\mathbf{Q}$:

$$
\begin{array}{cccccccccc}
\forall\mathbf{P}.\mathbf{Q} & t & \to_{\mathbf{I}}^{*}\to_{\mathbf{Q}} & \cdots & \to_{\mathbf{I}}^{*}\to_{\mathbf{Q}} & u' & \to_{\mathbf{I}}^{*} & v & \to_{\mathbf{P}}^{*} & w \\
 & \downarrow & & \downarrow & & \downarrow & & & & \\
\mathbf{Q} & \lfloor t \rfloor & \to_{\mathbf{Q}} & \cdots & \to_{\mathbf{Q}} & u & & & &
\end{array}
$$

Conclusions

- Pure Type Systems can be used to answer questions about reification of quantification
- It is interesting to study normalization preserving extensions and combinations of PTSs
- We can build richer type systems with the same logical strength.

Future Work

- Which rules can be added using this method?
- Can we simplify consistency proofs using this approach?
- Extensions to "Impure Type Systems"

Thank you