A general Subject Reduction for Pure Type Systems

COMP527 – Logic and Computation

Charlotte Marchal, Dashiell Rich, Milton Rosenbaum, Zhaoshen Zhai

March 30, 2025

The Cube

$$
\begin{tikzcd}
& \lambda\omega \arrow[rr] & & \lambda C \\
\lambda\mathbf{2} \arrow[ur] \arrow[rr] & & \lambda\mathbf{P2} \arrow[ur] & \\
& \lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \lambda\mathbf{P}\underline{\omega} \arrow[uu] \\
\lambda_{\to} \arrow[uu] \arrow[ur] \arrow[rr] & & \lambda\mathbf{P} \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Lots of Systems

In this diagram, eight systems are defined. The arrow represents $\subseteq$, so the weakest system is the simply typed lambda calculus, at the bottom left. Other named systems are $\lambda 2$, which is equivalent to system **F**, which is the polymorphic lambda calculus. $\lambda\omega$ is equivalent to $F\omega$, which was proposed by Girard, λP is equivalent to the "automath" language, and is sometimes called LF.

Kinds of Dependencies

$(\star, \star)$	Terms can depend on terms
$(\Box, \star)$	Terms can depend on types
$(\star, \Box)$	Types can depend on terms
$(\Box, \Box)$	Types can depend on types

Parameterized & Homogeneous

$\lambda\to$	$(\star,\star)$			
$\lambda\mathbf{2}$	$(\star,\star)$	$(\Box,\star)$		
$\lambda\underline{\omega}$	$(\star,\star)$		$(\Box,\Box)$	
$\lambda\omega$	$(\star,\star)$	$(\Box,\star)$	$(\Box,\Box)$	
$\lambda\mathbf{P}$	$(\star,\star)$			$(\star,\Box)$
$\lambda\mathbf{P2}$	$(\star,\star)$	$(\Box,\star)$		$(\star,\Box)$
$\lambda\mathbf{P}\underline{\omega}$	$(\star,\star)$		$(\Box,\Box)$	$(\star,\Box)$
$\lambda\mathbf{P}\omega$	$(\star,\star)$	$(\Box,\star)$	$(\Box,\Box)$	$(\star,\Box)$

Just change two two rules!

$$\Pi\text{−}rule \qquad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A.B) : s_3}$$

$$\lambda\text{−}rule \qquad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash b : B \quad \Gamma, x : A \vdash B : s_2}{\Gamma(\lambda x : A.b) : (\Pi x : A.B)}$$

What are they

In addition to $\Box$ and $\star$, we can add arbitrary sorts and axiom relations. A *pure type system* $\lambda(S, A, R)$ can be described as

- S : a set of sorts, for example $\star, \Box$

What are they

In addition to $\Box$ and $\star$, we can add arbitrary sorts and axiom relations. A *pure type system* $\lambda(S, A, R)$ can be described as

- S : a set of sorts, for example $\star, \Box$
- A : a set of axioms, $\star : \Box$

What are they

In addition to $\Box$ and $\star$, we can add arbitrary sorts and axiom relations. A *pure type system* $\lambda(S, A, R)$ can be described as

- S : a set of sorts, for example $\star, \Box$
- A : a set of axioms, $\star : \Box$
- R : a set of rules, the valid ways to parameterize the λ and Π rules. λC has all of the pairs of sorts.

What are they

In addition to $\Box$ and $\star$, we can add arbitrary sorts and axiom relations. A *pure type system* $\lambda(S, A, R)$ can be described as

- S : a set of sorts, for example $\star, \Box$
- A : a set of axioms, $\star : \Box$
- R : a set of rules, the valid ways to parameterize the λ and Π rules. λC has all of the pairs of sorts.

We abbreviate 'pure type system' by PTS.

Even more systems

A surprising number of systems can be defined as PTSs:

- $\lambda C' = (\star^t, \star^p, \Box), (\star^t : \Box, \star^p : \Box), (S^2, \text{ that is, all pairs})$

Even more systems

A surprising number of systems can be defined as PTSs:

- $\lambda C' = (\star^t, \star^p, \Box), (\star^t : \Box, \star^p : \Box), (S^2,$ that is, all pairs)
- $\lambda U = [\star, \Box, \Delta], [\star : \Box, \Box : \Delta], [(\star, \star), (\Box, \star), (\Box, \Box), (\Delta, \Box), (\Delta, \star)]$

Mechanize Proof for Each PTS

Since so many systems are easily describable as PTSs, it would be nice to prove some properties about the systems in general.

In our project, we plan to:

- Prove Subject Reduction for generalized PTSs.

Mechanize Proof for Each PTS

Since so many systems are easily describable as PTSs, it would be nice to prove some properties about the systems in general.

In our project, we plan to:

- Prove Subject Reduction for generalized PTSs.
- Mechanize arbitrary PTSs in Beluga.

Tasks

- Milton: Implementing PTSs in Beluga and mechanizing the proof of Subject Reduction.

Tasks

- Milton: Implementing PTSs in Beluga and mechanizing the proof of Subject Reduction.
- Everyone else: Prove Subject Reduction for PTSs.

Tasks

- Milton: Implementing PTSs in Beluga and mechanizing the proof of Subject Reduction.
- Everyone else: Prove Subject Reduction for PTSs.
- Zhaoshen: Write up the extended abstract.

Obstacles

- Programming in Beluga is *hard*.

Obstacles

- Programming in Beluga is *hard*. Like, really hard.

Obstacles

- Programming in Beluga is *hard*. Like, really hard.
- We need to first understand the proofs for System F and $\lambda\mathbf{P}$.