Formalizations of Metatheoretical Properties of Type Theories

Jonas Oberhauser

Programming Systems Lab
Saarland University

Joint work with Chad E. Brown

May 14, 2012

Introduction

Goal:

Formalize proofs of confluence and termination of ECC in Coq

- Confluence of Simply Typed Lambda Calculus (Done)
- Termination of Simply Typed Lambda Calculus (Almost Done)
- Extend those proofs to ECC

Simply Typed Lambda Calculus

base type	Nat
arrow type	$T_1 \to T_2$
lambda	$\lambda \mathrm{x} : T.t$
application	$(t\ u)$
reference	x

De Brujin Indices

Allows for elegant formalization

lambda	$\lambda T.t$
application	$(t\ u)$
reference	$n \in \mathbb{N}$

Shifting and Substitution

Shifting: $t \uparrow_{n'}^{k}$

Shift all references in t, starting with n', by k.

Substitution: t_s^n

Substitute all occurences of n in t by s.

Beta Reduction

$$
\begin{array}{lll}
(\lambda T.t)\, u & \rightarrow_\beta & t_u^0 \\
\lambda T.t & \rightarrow_\beta & \lambda T.t' \\
t\, u & \rightarrow_\beta & t'\, u \\
t\, u & \rightarrow_\beta & t\, u'
\end{array}
$$

We can do any number of steps at once with $\rightarrow_\beta^*$

Diamond Property

$$(\Diamond R) :=$$
$$a\,R\,b_1, a\,R\,b_2 \;\rightarrow\; \exists c.b_1\,R\,c \wedge b_2\,R\,c$$

Confluence

$$\text{Confluence} := \left(\Diamond \to_\beta^*\right)$$

In other words:

$$t \to_\beta^* u_1, t \to_\beta^* u_2 \;\to\; \exists t'.u_1 \to_\beta^* t' \wedge u_2 \to_\beta^* t'$$

How to prove this in a way easy to formalize?

Some Helpful Lemmas

$$\text{DiamondPropagates } R$$
$$(\Diamond R) \ \rightarrow \ (\Diamond R^*)$$

$$\text{ClosureEquivalence } R \ Q$$
$$R \subseteq Q \subseteq R^* \ \rightarrow \ Q^* \equiv R^*$$

$$\text{DiamondClosure } R \ Q$$
$$(\Diamond Q), R \subseteq Q \subseteq R^* \ \rightarrow \ (\Diamond R^*)$$

$$\text{DiamondLemma } \rightsquigarrow$$
$$(\Diamond \rightsquigarrow), \rightarrow_\beta \subseteq \rightsquigarrow \subseteq \rightarrow_\beta^* \ \rightarrow \ \text{Confluence}$$

Some Unhelpful Lemma

But also:

SadnessLemma

$\neg(\Diamond \rightarrow_\beta)$

The Sadness Lemma

$$(\lambda T.t)\, u,\ t = \cdots 0 \cdots 0 \cdots 0 \cdots$$

$$\begin{array}{lll} (\lambda T.t)\, u & \to_\beta & (\lambda T.t)\, u' \\ (\lambda T.t)\, u & \to_\beta & t_u^0 \qquad = \cdots u \cdots u \cdots u \cdots \end{array}$$

join $(\lambda T.t)\, u'$ and $\cdots u \cdots u \cdots u \cdots$?

The Sadness Lemma

$$(\lambda T.t)\, u' \to_\beta t^0_{u'} = \cdots u' \cdots u' \cdots u' \cdots$$

but

$$\cdots u \cdots u \cdots u \cdots \to_\beta \begin{array}{l} \cdots u' \cdots u \cdots u \cdots \\ \cdots u \cdots u' \cdots u \cdots \\ \cdots u \cdots u \cdots u' \cdots \end{array}$$

Parallel Reduction

$$
\begin{array}{ccc}
(\lambda T.t)\, u & \Rightarrow & t'^{0}_{u'} \\
\lambda T.t & \Rightarrow & \lambda T.t' \\
t\, u & \Rightarrow & t'\, u' \\
t & \Rightarrow & t
\end{array}
$$

Parallel Reduction

2 Lambda:
Reflexive case or reduction to another lambda

$$\begin{array}{ccc} \lambda T.t & \Rightarrow & \lambda T.t' \\ t & \Rightarrow & t \end{array}$$

Parallel Reduction

3 Application:
Reflexive case, substitution (actual beta reduction), or reduction to another application

$$\begin{array}{ccc} (\lambda T.t)\, u & \Rightarrow & t'^{0}_{u'} \\ t\, u & \Rightarrow & t'\, u' \\ t & \Rightarrow & t \end{array}$$

Parallel Reduction

Try to prove: $(\Diamond \Rightarrow)$

$$t \Rightarrow u_1, t \Rightarrow u_2 \;\rightarrow\; \exists t'.u_1 \Rightarrow t' \wedge u_2 \Rightarrow t'$$

Induction gives many (redundant) cases

1 reflexive

2x2 lambda

3x3 application

$\rightarrow$ 14 cases

Not formalized in Coq, because we can do better

Parallel Reduction

Eliminate redundant cases with inversion lemmas

Lambda:

$$\lambda T.t \Rightarrow u \;\rightarrow\; u = \lambda T.t'$$

Application:

$$t\ u \Rightarrow v \;\rightarrow\; \begin{cases} t = \lambda T.b \wedge v = b'^{0}_{u'} \\ v = t'\ u' \end{cases}$$

Parallel Reduction

Now prove: $(\Diamond \Rightarrow)$

$$t \Rightarrow u_1, t \Rightarrow u_2 \;\rightarrow\; \exists t'.u_1 \Rightarrow t' \wedge u_2 \Rightarrow t'$$

Induction on $t \Rightarrow u_1$ yields 6 cases:

1 reflexive	just simulate u_2
1x1 lambda	u_2 must be a lambda too
2x2 application	substitution or application?

Fully formalized in Coq

TopElement

$$(\text{TopElement} \rightsquigarrow) := \forall t$$
$$\exists t'.t \rightsquigarrow t' \wedge \forall u.t \rightsquigarrow u \ \rightarrow \ u \rightsquigarrow t'$$

$$(\text{TopElement} \rightsquigarrow) \ \rightarrow \ (\Diamond \rightsquigarrow)$$

In Coq: Sigma!

TopElement

$(\mathrm{TopElement} \Rightarrow)$: contract all the redexes

First attempt at formalization gives 10 cases:
1 reflexive
1 lambda
2x(2x2) application (!)

Patch Reduction

$$
\begin{array}{lccc}
t \triangleright (\lambda T.b) \rightarrow & t\ u & \triangleright & b'^{0}_{u'} \\
 & \lambda T.t & \triangleright & \lambda T.t' \\
 & t\ u & \triangleright & t'\ u' \\
 & t & \triangleright & t
\end{array}
$$

Patch Reduction

Four cases!

Induction on t gives us:

1 reflexive

1x1 lambda

2x1 application

Patch Reduction

For the TopElement of $t\,u$, is $t^\top := \mathrm{TopElement}\ t$ a lambda?
If it is, we need to apply the beta rule.
If it's not, we can give $t^\top u^\top$ and be happy.

$\rightarrow$ no inversion lemmas needed
Fully formalized in Coq

Next Problems to Tackle

Formalize proof of Strong Normalization

Extend Proofs to ECC without Sigma Types

Add Sigma Types

References I


Barendregt.
The Lambda Calculus.
1984.


Luo.
Computation and Reasoning.
1994.


Brown, Smolka.
Semantics Lecture Notes.
2012.


Barras.
Coq in Coq.
1997.