Conservativity of embeddings in the $\lambda\Pi$-calculus modulo rewriting

Ali Assaf

Deducteam, Inria, Paris
Ecole Polytechnique, Palaiseau

TLCA
June 2, 2015

8 years ago...

Embedding Pure Type Systems in the lambda-Pi-calculus modulo

Denis Cousineau and Gilles Dowek

École polytechnique and INRIA
LIX, École polytechnique, 91128 Palaiseau Cedex, France.

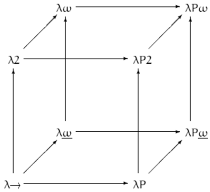
Cousineau@lix.polytechnique.fr, http://www.lix.polytechnique.fr/~cousineau
Gilles.Dowek@polytechnique.edu, http://www.lix.polytechnique.fr/~dowek

Abstract. The lambda-Pi-calculus allows to express proofs of minimal predicate logic. It can be extended, in a very simple way, by adding computation rules. This leads to the lambda-Pi-calculus modulo. We show in this paper that this simple extension is surprisingly expressive and, in particular, that all functional Pure Type Systems, such as the system F, or the Calculus of Constructions, can be embedded in it. And, moreover, that this embedding is conservative under termination hypothesis.

The $\lambda\Pi$*-calculus* is a dependently typed lambda-calculus that allows to express proofs of minimal predicate logic through the Brouwer-Heyting-Kolmogorov interpretation and the Curry-de Bruijn-Howard correspondence. It can be extended in several ways to express proofs of some theory. A first solution is to express the theory in Deduction modulo [7, 9], *i.e.* to orient the axioms as rewrite

Pure type systems

- Large family of typed lambda calculi λS
- Parametrized by a specification S of allowed types
  - dependent types,
  - polymorphism,
  - type operators,
  - ...



The $\lambda\Pi$-calculus modulo rewriting

$$\lambda\Pi R = \text{lambda calculus} + \text{dependent types} + \text{rewriting}$$

- Curry-Howard version of deduction modulo
- Logical framework

Logical framework

Curry-Howard:

$$\Gamma \vdash_{\mathcal{L}} A \iff [\![\Gamma]\!] \vdash_{\lambda_{\mathcal{L}}} M : [\![A]\!]$$

Logical framework

Curry-Howard:

$$\Gamma \vdash_{\mathcal{L}} A \quad \Longleftrightarrow \quad [\![\Gamma]\!] \vdash_{\lambda_{\mathcal{L}}} M : [\![A]\!]$$

Logical framework:

$$\Gamma \vdash_{\mathcal{L}} A \quad \Longleftrightarrow \quad \Sigma_{\mathcal{L}}, [\![\Gamma]\!] \vdash_{LF} M : [\![A]\!]$$

Embedding pure type systems

- Source language = PTS λS
- Target language = $\lambda \Pi R$

$$\Gamma \vdash_{\lambda S} M : A \quad \Longrightarrow \quad \Sigma_S, \llbracket \Gamma \rrbracket \vdash_{\lambda \Pi R} M' : \llbracket A \rrbracket$$

Embedding pure type systems

- Source language = PTS λS
- Target language = $\lambda\Pi R$

$$\Gamma \vdash_{\lambda S} M : A \quad \Longrightarrow \quad \Sigma_S, [\![\Gamma]\!] \vdash_{\lambda\Pi R} M' : [\![A]\!]$$

- This talk:

$$\Gamma \vdash_{\lambda S} M : A \quad \Longleftarrow \quad \Sigma_S, [\![\Gamma]\!] \vdash_{\lambda\Pi R} M' : [\![A]\!] \qquad ?$$

Pure type systems

A PTS *specification* S is a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ where:

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

Pure type systems

A PTS *specification* S is a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ where:

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

Syntax:

$$x \mid s \mid \Pi x : A.\, B \mid \lambda x : A.\, M \mid M\, N$$

$$\boxed{\Gamma \vdash M : A}$$

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

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

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$$

$$\frac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B\,\{x \backslash N\}}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A \equiv_\beta B}{\Gamma \vdash M : B}$$

$$\boxed{\Gamma \text{ well-formed}}$$

$$\frac{}{\varnothing \text{ well-formed}} \qquad \frac{\Gamma \text{ well-formed} \qquad \Gamma \vdash A : s}{\Gamma, x : A \text{ well-formed}}$$

Embedding system F

Signature:

$$
\begin{array}{lcl}
\mathsf{type} & : & \mathsf{Type} \\
\mathsf{arrow} & : & \mathsf{type} \to \mathsf{type} \to \mathsf{type} \\
\mathsf{forall} & : & (\mathsf{type} \to \mathsf{type}) \to \mathsf{type}
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{term} & : & \mathsf{type} \to \mathsf{Type}
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{term}\,(\mathsf{arrow}\; a\; b) & \longmapsto & \mathsf{term}\; a \to \mathsf{term}\; b \\
\mathsf{term}\,(\mathsf{forall}\; f) & \longmapsto & \Pi a : \mathsf{type}.\; \mathsf{term}\,(f\; x)
\end{array}
$$

Embedding system F

Translation:

$$
\begin{array}{lcl}
[a] & = & a \\
[A \to B] & = & \mathsf{arrow}\,[A]\,[B] \\
[\forall a.B] & = & \mathsf{forall}\,(\lambda a.\ [B]) \\
\\
[\![A]\!] & = & \mathsf{term}\,[A] \\
\\
[x] & = & x \\
[\lambda x : A.\, M] & = & \lambda x : [\![A]\!]\,.\ [M] \\
[M\, N] & = & [M]\,[N] \\
[\Lambda a.M] & = & \lambda a : \mathsf{type}.\ [M] \\
[M\,\langle A\rangle] & = & [M]\,[A]
\end{array}
$$

Comparison

	Identity	LF	C&D
Preserves computation	✓	✗	✓
Encodes higher-order	✗	✓	✓

Preservation of reduction/typing

Theorem ([CD07])

If $M \longrightarrow_\beta M'$ *then* $[M] \longrightarrow_\beta^+ [M']$.

Theorem ([CD07])

If $\Gamma \vdash_{\lambda S} M : A$ *then* $\Sigma_S, [\![\Gamma]\!] \vdash_{\lambda \Pi R} [M] : [\![A]\!]$.

Works for any functional pure type system:

- System F
- Calculus of constructions
- Higher-order logic

Preservation of reduction/typing

Theorem ([CD07])

If $M \longrightarrow_\beta M'$ *then* $[M] \longrightarrow_\beta^+ [M']$.

Theorem ([CD07])

If $\Gamma \vdash_{\lambda S} M : A$ *then* $\Sigma_S, [\![\Gamma]\!] \vdash_{\lambda\Pi R} [M] : [\![A]\!]$.

Works for any functional pure type system:

- System F
- Calculus of constructions
- Higher-order logic

What about the converse?

But first...

But first...

... why 8 years?

But first...

... why 8 years?

- Undergrad: 3 years
- Masters: 2 years
- PhD: 3 years

But first...

... why 8 years?

- Undergrad: 3 years
- Masters: 2 years
- PhD: 3 years

- Total: $3 + 2 + 3 =$ **8 years**

In $\lambda\Pi$

In $\lambda\Pi$, conservativity is traditionally proved using strong normalization (SN):

- $\lambda\Pi$ is SN
- Adding declarations in Σ does not affect SN
- Conservativity proved by induction on the normal forms

Fact

Bijection between terms of λS and classes of β-equivalent terms of $\lambda\Pi R_S$.

In $\lambda\Pi R$

- The introduction of rewrite rules could break SN...
  - because R is not SN,
  - or because $\beta \cup R$ is not SN,
  - or because β is not SN anymore.

In $\lambda\Pi R$

- The introduction of rewrite rules could break SN...
  - because R is not SN,
  - or because $\beta \cup R$ is not SN,
  - or because β is not SN anymore.
- Cousineau and Dowek (2007): proof of conservativity assuming $\lambda\Pi R_S$ is SN.

In $\lambda\Pi R$

- The introduction of rewrite rules could break SN...
  - because R is not SN,
  - or because $\beta \cup R$ is not SN,
  - or because β is not SN anymore.
- Cousineau and Dowek (2007): proof of conservativity assuming $\lambda\Pi R_S$ is SN.

Open problem!

Approach 1: absolute normalization

Idea: build a model for $\lambda\Pi R_S$

- in the algebra of reducibility candidates
- or in a general notion of Π-algebra.

The model implies strong normalization of $\lambda\Pi R_S$. Use this to prove conservativity.

Approach 1: absolute normalization

Idea: build a model for $\lambda\Pi R_S$

- in the algebra of reducibility candidates
- or in a general notion of Π-algebra.

The model implies strong normalization of $\lambda\Pi R_S$. Use this to prove conservativity.

Theorem ([Dow14])

There is a model for $\lambda\Pi R_{HOL}$and for $\lambda\Pi R_{CC}$.

Absolute normalization

Problem: Implies strong normalization of λS, so at least as hard proving strong normalization of λS!

$$M \longrightarrow_\beta M' \quad \Longrightarrow \quad [M] \longrightarrow_\beta^+ [M']$$

Absolute normalization

Problem: Implies strong normalization of λS, so at least as hard proving strong normalization of λS!

$$\frac{M \longrightarrow_\beta M' \quad \Longrightarrow \quad [M] \longrightarrow_\beta^+ [M']}{M \notin \mathcal{SN} \quad \Longrightarrow \quad [M] \notin \mathcal{SN}}$$

Absolute normalization

Problem: Implies strong normalization of λS, so at least as hard proving strong normalization of λS!

$$M \longrightarrow_\beta M' \quad \Longrightarrow \quad [M] \longrightarrow_\beta^+ [M']$$

$$M \notin \mathcal{SN} \quad \Longrightarrow \quad [M] \notin \mathcal{SN}$$

$$M \in \mathcal{SN} \quad \Longleftarrow \quad [M] \in \mathcal{SN}$$

Absolute normalization

Problem: Implies strong normalization of λS, so at least as hard proving strong normalization of λS!

$$\frac{\dfrac{M \longrightarrow_\beta M' \quad \Longrightarrow \quad [M] \longrightarrow_\beta^+ [M']}{M \notin \mathcal{SN} \quad \Longrightarrow \quad [M] \notin \mathcal{SN}}}{M \in \mathcal{SN} \quad \Longleftarrow \quad [M] \in \mathcal{SN}}$$

- Tricky to do
  - Anyone wants to try Σ_{CC^ω}? Brrrr...
- Duplicates work
  - Can't we use the fact that λS is SN?

Approach 2: relative normatlization

If $[\![\Gamma]\!] \vdash M : [\![A]\!]$, what can we say about M?

Approach 2: relative normatlization

If $[\![\Gamma]\!] \vdash M : [\![A]\!]$, what can we say about M?

Example

If S is the simply-typed λ-calculus, the polymorphic identity function is not well-typed, so:

$$b : \mathsf{Type} \nvdash (\lambda a : \mathsf{Type}.\, \lambda x : b.\, x)\, b : b \to b$$

Its translation is well-typed in $\lambda\Pi R_S$:

$$b : \mathsf{type} \vdash (\lambda a : \mathsf{type}.\, \lambda x : \mathsf{term}\, a.\, x)\, b : \mathsf{term}\, b \to \mathsf{term}\, b$$

Approach 2: relative normatlization

If $[\![\Gamma]\!] \vdash M : [\![A]\!]$, what can we say about M?

Example

If S is the simply-typed λ-calculus, the polymorphic identity function is not well-typed, so:

$$b : \mathsf{Type} \nvdash (\lambda a : \mathsf{Type}.\, \lambda x : b.\, x)\, b : b \to b$$

Its translation is well-typed in $\lambda\Pi R_S$:

$$b : \mathsf{type} \vdash (\lambda a : \mathsf{type}.\, \lambda x : \mathsf{term}\, a.\, x)\, b : \mathsf{term}\, b \to \mathsf{term}\, b$$

But it reduces to $\lambda x : b.\, x$ of type $b \to b$ in λS.

Relative normalization

What have we learned?

1. $\lambda\Pi R_S$ can type more terms than λS.
2. These terms can be used to construct proofs for the translation of λS types.
3. The $\lambda\Pi R_S$ terms that inhabit the translation of λS types can be reduced to the translation of λS terms.

Relative normalization

What have we learned?

1. $\lambda\Pi R_S$ can type more terms than λS.
2. These terms can be used to construct proofs for the translation of λS types.
3. The $\lambda\Pi R_S$ terms that inhabit the translation of λS types can be reduced to the translation of λS terms.

Idea: reduce only what is necessary to fall back in λS.

Erasure

Define an erasure:

$$\begin{aligned}
\varphi(x) &= x \\
\varphi(\lambda x : A.\, M) &= \lambda x : \psi(A)\,.\,\varphi(M) \\
\varphi(M\, N) &= \varphi(M)\,\varphi(N)
\end{aligned}$$

$$\begin{aligned}
\psi(\text{type}) &= \text{Type} \\
\psi(\text{term}\, A) &= \varphi(A) \\
\psi(A \to B) &= \psi(A) \to \psi(B)
\end{aligned}$$

Erasure is the inverse of the translation:

$$\begin{aligned}
\varphi([M]) &= M \\
\psi([\![A]\!]) &= A
\end{aligned}$$

Erasure

Example

$\varphi\,(((\lambda a : \text{type}.\, \lambda x : \text{term}\, a.\, x)\; b)) = (\lambda a : \text{Type}.\, \lambda x : a.\, x)\; b$

Not well-typed in $\lambda_{\rightarrow}$ because there is no $(\text{Kind}, \text{Type}, -)$ rule.

- Need to allow more types.

Minimal completion

We define the following *completion* ([SP94]).

Definition

The *minimal completion* of S is $S^* = (\mathcal{S}^*, \mathcal{A}^*, \mathcal{R}^*)$ where

- $\mathcal{S}^* = \mathcal{S} \uplus \{\tau\}$
- $\mathcal{A}^* = \mathcal{A} \cup \{(s_1, \tau) \mid \nexists\, s_2 \in \mathcal{S}, (s_1, s_2) \in \mathcal{A}\}$
- $\mathcal{R}^* = \mathcal{R} \cup \{(s_1, s_2, \tau) \mid \nexists\, s_3 \in \mathcal{S}, (s_1, s_2, s_3) \in \mathcal{R}\}$

Example

Example

In $\lambda^*_\rightarrow$,

$$b : \mathsf{Type} \vdash (\lambda a : \mathsf{Type}.\, \lambda x : a.\, x)\, b : b \rightarrow b$$

because

$$\frac{\vdash \mathsf{Type} : \mathsf{Kind} \qquad a : \mathsf{Type} \vdash a \rightarrow a : \mathsf{Type} \qquad (\mathsf{Kind}, \mathsf{Type}, \tau) \in \mathcal{R}^*}{\vdash \Pi a : \mathsf{Type}.\, a \rightarrow a : \tau}$$

Example

Example

In $\lambda^*_\rightarrow$,

$$b : \mathsf{Type} \vdash (\lambda a : \mathsf{Type}.\, \lambda x : a.\, x)\, b : b \rightarrow b$$

because

$$\frac{\vdash \mathsf{Type} : \mathsf{Kind} \qquad a : \mathsf{Type} \vdash a \rightarrow a : \mathsf{Type} \qquad (\mathsf{Kind}, \mathsf{Type}, \tau) \in \mathcal{R}^*}{\vdash \Pi a : \mathsf{Type}.\, a \rightarrow a : \tau}$$

How do we go back to λS?

Reducibility

Let $\Gamma \vdash_{\lambda s}$ well-formed and $\Gamma \vdash_{\lambda s^*} M : A : s$.
Define the predicate $\Gamma \Vdash M : A$ by induction on A:

Reducibility

Let $\Gamma \vdash_{\lambda s}$ well-formed and $\Gamma \vdash_{\lambda s^*} M : A : s$.
Define the predicate $\Gamma \Vdash M : A$ by induction on A:

- If $A : s \neq \tau$ or $A \neq \Pi x : B.\, C$ then $M \longrightarrow^* M'$ and $A \longrightarrow^* A'$ such that $\Gamma \vdash_{\lambda s} M' : A'$.

Reducibility

Let $\Gamma \vdash_{\lambda s}$ well-formed and $\Gamma \vdash_{\lambda s^*} M : A : s$.
Define the predicate $\Gamma \Vdash M : A$ by induction on A:

- If $A : s \neq \tau$ or $A \neq \Pi x : B.\, C$ then $M \longrightarrow^* M'$ and $A \longrightarrow^* A'$ such that $\Gamma \vdash_{\lambda s} M' : A'$.
- If $A : \tau$ and $A = \Pi x : B.\, C$ then for all N such that $\Gamma \Vdash N : B$, $\Gamma \Vdash M\, N : C\, \{x \backslash N\}$.

Reducibility

Let $\Gamma \vdash_{\lambda s}$ well-formed and $\Gamma \vdash_{\lambda s^*} M : A : s$.
Define the predicate $\Gamma \Vdash M : A$ by induction on A:

- If $A : s \neq \tau$ or $A \neq \Pi x : B.\, C$ then $M \longrightarrow^* M'$ and $A \longrightarrow^* A'$ such that $\Gamma \vdash_{\lambda s} M' : A'$.
- If $A : \tau$ and $A = \Pi x : B.\, C$ then for all N such that $\Gamma \Vdash N : B$, $\Gamma \Vdash M\, N : C\,\{x \backslash N\}$.

If σ is a substitution mapping variables to terms:

- $\Gamma \Vdash \sigma : \Delta$ when $\Gamma \Vdash \sigma\,(x) : \sigma\,(A)$ for all $(x : A) \in \Gamma$.

Conservativity

Theorem

If $\Delta \vdash_{\lambda s^*} M : A : s$ *then for any* Γ, σ *such that* $\Gamma \Vdash \sigma : \Delta$,$\Gamma \Vdash \sigma(M) : \sigma(A)$.

Proof.

By induction on the derivation of $\Delta \vdash M : A$. □

Conservativity

Theorem

If $\Delta \vdash_{\lambda S^*} M : A : s$ then for any Γ, σ such that $\Gamma \Vdash \sigma : \Delta$, $\Gamma \Vdash \sigma(M) : \sigma(A)$.

Proof.

By induction on the derivation of $\Delta \vdash M : A$. □

Corollary (Conservativity)

If $\Sigma_S, [\![\Gamma]\!] \vdash_{\lambda \Pi R} M : [\![A]\!]$ then $\varphi(M) \longrightarrow_\beta^* M'$ such that $\Gamma \vdash_{\lambda S} M' : A$.

Proof.

By taking the identity substitution, $\psi(\sigma([\![A]\!])) = \psi([\![A]\!]) = A$. □

Relative normalization

- Avoid complex techniques such as reducibility candidates.
- Works for non-terminating theories! (e.g. system U)

Conclusion

Summary:

- Embedding of PTSs in $\lambda\Pi$-calculus modulo rewriting
- Preserves reductions, preserves typing
- Proof of conservativity by showing relative normalization
- Implies weak normalization of $\lambda\Pi R_S$

Conclusion

Summary:

- Embedding of PTSs in $\lambda\Pi$-calculus modulo rewriting
- Preserves reductions, preserves typing
- Proof of conservativity by showing relative normalization
- Implies weak normalization of $\lambda\Pi R_S$

Future work:

- Adapt proof to show strong normalization of $\lambda\Pi R_S$

Conclusion

Summary:

- Embedding of PTSs in $\lambda\Pi$-calculus modulo rewriting
- Preserves reductions, preserves typing
- Proof of conservativity by showing relative normalization
- Implies weak normalization of $\lambda\Pi R_S$

Future work:

- Adapt proof to show strong normalization of $\lambda\Pi R_S$

Thank you!

References

Denis Cousineau and Gilles Dowek.
Embedding pure type systems in the lambda-Pi-calculus modulo.
In *Typed Lambda Calculi and Applications (TLCA)*, 2007.

Gilles Dowek.
Models and termination of proof-reduction in the $\lambda\Pi$-calculus modulo theory.
arXiv:1501.06522, 2014.

Paula Severi and Erik Poll.
Pure type systems with definitions.
In *Logical Foundations of Computer Science (LFCS)*, 1994.