# Equivalence Between Typed and Untyped Algorithmic Conversion

Meven Lennon-Bertrand

Types Conference – June 22nd 2022

# A Tale of Two (or Four) Conversions

### Typed and Untyped Conversion

- Two traditions: MLTT (typed) vs PTS (untyped)
- Typed: good story for $\eta$ laws
- Untyped: more efficient, thus used in Coq

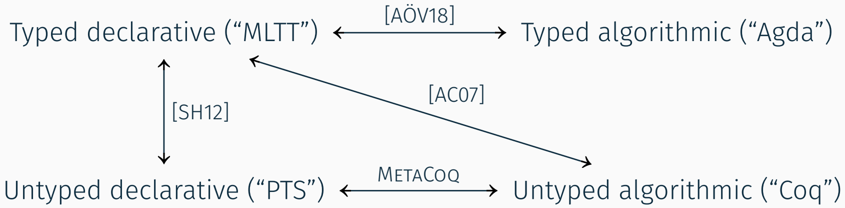
## Typed and Untyped Conversion

- Two traditions: MLTT (typed) vs PTS (untyped)
- Typed: good story for $\eta$ laws
- Untyped: more efficient, thus used in Coq

## Declarative and Algorithmic Conversion

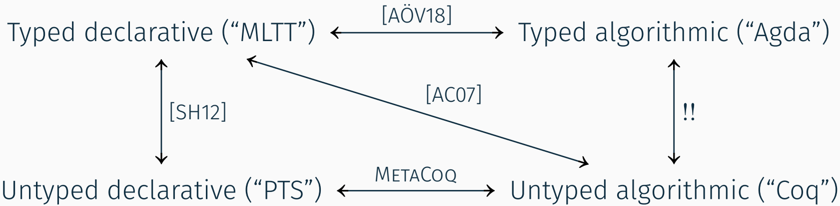
- Declarative: standard presentation, but no direct algorithm
- Algorithmic: easy to relate to an algorithm, but not a good specification

# The Situation so Far




- [AC07], [AÖV18]: stronger logical power than the studied system
- [SH12], MetaCoq: no η laws

# The Situation so Far




- [AC07], [AÖV18]: stronger logical power than the studied system
- [SH12], MetaCoq: no η laws
- !!: Can we do this? With a low logical power?

# How do we do this?

**Typed conversion: put bidirectional lenses on**

- $\Gamma \vdash t \Leftrightarrow t' : T$ with $T$ as *input*, $\Gamma \vdash n \leftrightarrow n' : T$ with $T$ as *output*
- Motto: *Conversion* $\Leftrightarrow$ *checks, neutral comparison* $\leftrightarrow$ *infers*

$$\frac{\Gamma, x\colon A \vdash f\ x \Leftrightarrow g\ x : B}{\Gamma \vdash f \Leftrightarrow g : \Pi\, x\colon A.\ B}$$

## Typed conversion: put bidirectional lenses on

- $\Gamma \vdash t \Leftrightarrow t' : T$ with $T$ as *input*, $\Gamma \vdash n \leftrightarrow n' : T$ with $T$ as *output*
- Motto: *Conversion* $\Leftrightarrow$ *checks, neutral comparison* $\leftrightarrow$ *infers*

$$\frac{\Gamma, x\colon A \vdash f\ x \Leftrightarrow g\ x : B}{\Gamma \vdash f \Leftrightarrow g : \Pi\, x\colon A.\ B}$$

## Untyped conversion

- Same general structure: conversion + neutral comparison
- Main difference: term-directed instead of type-directed

$$\frac{n\ x \Leftrightarrow t \qquad n \text{ neutral}}{n \Leftrightarrow \lambda\, x\colon A.\ t} + \text{symmetric} \qquad\qquad \frac{t \Leftrightarrow t'}{\lambda\, x\colon A.\ t \Leftrightarrow \lambda\, x\colon A'.\ t'}$$

# A Proof in Two Steps

# A Proof in Two Steps

## Step 1: McBride's discipline

- Flow of well-formation information for well-behaved bidirectional rules
- Respected by the relation
- Needs meta-theory of the typed variant

# A Proof in Two Steps

## Step 1: McBride's discipline

- Flow of well-formation information for well-behaved bidirectional rules
- Respected by the relation
- Needs meta-theory of the typed variant

## Step 2: Relate the rules

- Reasoning on weak-head normal forms
- Rather straightforward

# A Proof in Two Steps

## Step 1: McBride's discipline

- Flow of well-formation information for well-behaved bidirectional rules
- Respected by the relation
- Needs meta-theory of the typed variant

## Step 2: Relate the rules

- Reasoning on weak-head normal forms
- Rather straightforward

Work in progress, worked out on a toy system ($\lambda\Pi\Box$) on paper.

# A Proof in Two Steps

## Step 1: McBride's discipline

- Flow of well-formation information for well-behaved bidirectional rules
- Respected by the relation
- Needs meta-theory of the typed variant

## Step 2: Relate the rules

- Reasoning on weak-head normal forms
- Rather straightforward

Work in progress, worked out on a toy system ($\lambda \Pi \square$) on paper.

Does it scale all the way to PCUIC?

# Thank you!

# Bibliography


[AC07] Andreas Abel and Thierry Coquand. "Untyped Algorithmic Equality for Martin-Löf's Logical Framework with Surjective Pairs". In: *Fundamenta Informaticae* 77.4 (2007). TLCA'05 special issue., pp. 345–395. URL: `http://fi.mimuw.edu.pl/abs77.html#15`.

[AÖV18] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. "Decidability of Conversion for Type Theory in Type Theory". In: *Proc. ACM Program. Lang.* (Jan. 2018). DOI: `10.1145/3158111`.

[SH12] Vincent Siles and Hugo Herbelin. "Pure Type System conversion is always typable". In: *J. Funct. Program.* 22.2 (2012), pp. 153–180. DOI: `10.1017/S0956796812000044`.

[Soz+20] Matthieu Sozeau et al. "Coq Coq Correct! Verification of Type Checking and Erasure for Coq, in Coq". In: *Proceedings of the ACM on Programming Languages* (Jan. 2020), pp. 1–28. DOI: `10.1145/3371076`.