Definitional Functoriality for Dependent (Sub)Types

Théo Laurent, **Meven Lennon-Bertrand**, Kenji Maillard

π^3 Seminar — December 11th 2023

Dependent types and subtyping

What users™ want:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\mathrm{sub}} t : A \qquad \Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{sub}} t : A'}$$

What users™ want:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\mathrm{sub}} t : A \qquad \Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{sub}} t : A'}$$

Semanticists hate this: forces $[\![\Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A']\!]$ to be (set) inclusion.

$$\begin{array}{ccc} [\![t]\!] & = & [\![t]\!] \\ \cap & & \cap \\ [\![A]\!] & & [\![A']\!] \end{array}$$

What users™ want:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\mathrm{sub}} t : A \qquad \Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{sub}} t : A'}$$

Semanticists hate this: forces $[\![\Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A']\!]$ to be (set) inclusion.

$$\begin{array}{ccc} [\![t]\!] & = & [\![t]\!] \\ \cap & & \cap \\ [\![A]\!] & & [\![A']\!] \end{array}$$

Too strict: $\dfrac{A' \preccurlyeq A \qquad B' \preccurlyeq B}{A \to B \preccurlyeq A' \to B'}$ is *not* set-theoretic inclusion...

What semanticists want you to do:

$$\textsc{Coe}\ \frac{\Gamma \vdash_{\mathrm{coe}} t : A \qquad \Gamma \vdash_{\mathrm{coe}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{coe}} \mathrm{coe}_{A,A'}\, t : A'}$$

What semanticists want you to do:

$$\textsc{Coe}\ \frac{\Gamma \vdash_{\mathrm{coe}} t : A \qquad \Gamma \vdash_{\mathrm{coe}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{coe}} \mathrm{coe}_{A,A'}\, t : A'}$$

Nicer semantics: $\left[\!\left[\mathrm{coe}_{A,A'}\right]\!\right]$ = "well-chosen functions"

Set-theoretic interpretation for $\left[\!\left[\mathrm{coe}_{A\to B,A'\to B'}\right]\!\right]$ is the one we want

What semanticists want you to do:

$$\textsc{Coe}\ \frac{\Gamma \vdash_{\mathrm{coe}} t : A \qquad \Gamma \vdash_{\mathrm{coe}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{coe}} \mathrm{coe}_{A,A'}\, t : A'}$$

Nicer semantics: $\left[\!\left[\mathrm{coe}_{A,A'}\right]\!\right]$ = "well-chosen functions"

Set-theoretic interpretation for $\left[\!\left[\mathrm{coe}_{A\to B,A'\to B'}\right]\!\right]$ is the one we want

This is the work of a compiler!

Let's just elaborate, then:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\mathrm{sub}} t : A \qquad \Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{sub}} t : A'} \quad \rightsquigarrow \quad \textsc{Coe}\ \frac{\tilde{\Gamma} \vdash_{\mathrm{coe}} \tilde{t} : \tilde{A} \qquad \tilde{\Gamma} \vdash_{\mathrm{coe}} \tilde{A} \preccurlyeq \tilde{A}'}{\tilde{\Gamma} \vdash_{\mathrm{coe}} \mathrm{coe}_{\tilde{A},\tilde{A}'}\, \tilde{t} : \tilde{A}'}$$

Let's just elaborate, then:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\mathrm{sub}} t : A \qquad \Gamma \vdash_{\mathrm{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\mathrm{sub}} t : A'} \quad \rightsquigarrow \quad \textsc{Coe}\ \frac{\tilde{\Gamma} \vdash_{\mathrm{coe}} \tilde{t} : \tilde{A} \qquad \tilde{\Gamma} \vdash_{\mathrm{coe}} \tilde{A} \preccurlyeq \tilde{A}'}{\tilde{\Gamma} \vdash_{\mathrm{coe}} \mathrm{coe}_{\tilde{A},\tilde{A}'}\, \tilde{t} : \tilde{A}'}$$

Coherence: $[\![t]\!] \overset{\mathrm{def}}{=} [\![\tilde{t}]\!]$ should be unambiguous – all elaborations should have the same semantics.

Let's just elaborate, then:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\text{sub}} t : A \qquad \Gamma \vdash_{\text{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\text{sub}} t : A'} \quad \rightsquigarrow \quad \textsc{Coe}\ \frac{\tilde{\Gamma} \vdash_{\text{coe}} \tilde{t} : \tilde{A} \qquad \tilde{\Gamma} \vdash_{\text{coe}} \tilde{A} \preccurlyeq \tilde{A}'}{\tilde{\Gamma} \vdash_{\text{coe}} \ \text{coe}_{\tilde{A},\tilde{A}'}\, \tilde{t} : \tilde{A}'}$$

Coherence: $[\![t]\!] \overset{\text{def}}{=} [\![\tilde{t}]\!]$ should be unambiguous – all elaborations should have the same semantics.

Actually...

```
Cannot unify t and t.
```

Let's just elaborate, then:

$$\textsc{Sub}\ \frac{\Gamma \vdash_{\text{sub}} t : A \qquad \Gamma \vdash_{\text{sub}} A \preccurlyeq A'}{\Gamma \vdash_{\text{sub}} t : A'} \quad \rightsquigarrow \quad \textsc{Coe}\ \frac{\tilde{\Gamma} \vdash_{\text{coe}} \tilde{t} : \tilde{A} \qquad \tilde{\Gamma} \vdash_{\text{coe}} \tilde{A} \preccurlyeq \tilde{A}'}{\tilde{\Gamma} \vdash_{\text{coe}} \ \text{coe}_{\tilde{A},\tilde{A}'}\, \tilde{t} : \tilde{A}'}$$

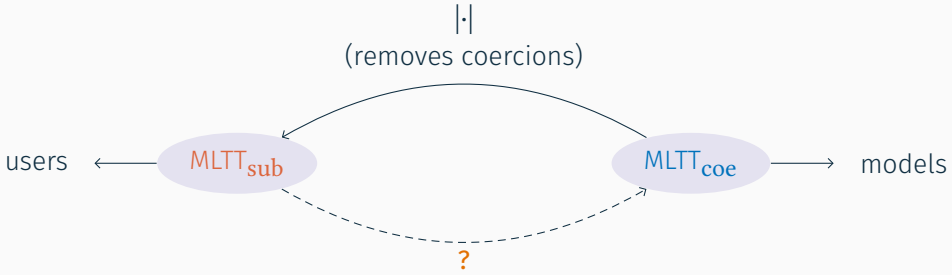
Coherence: $[\![t]\!] \overset{\text{def}}{=} [\![\tilde{t}\,]\!]$ should be unambiguous – all elaborations should have the same semantics.

Actually...

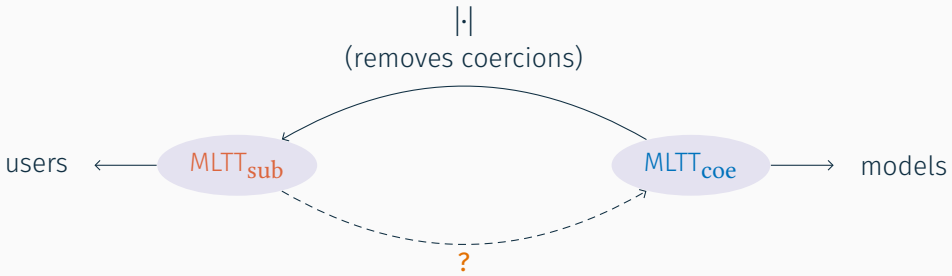
```
Cannot unify t and t.
```

Better coherence: we should always have $\tilde{t} \cong \tilde{t}'$ for two different elaborations of t.

Now we have a goal



Now we have a goal



? = the compilation we want

Only well-defined if a lot of equations hold

In particular, $|t| = |u| \Rightarrow \Gamma \vdash_{\text{coe}} t \cong u : A$ (provided $\Gamma \vdash_{\text{coe}} t, u : A$)

Coercions are maps

What is a reasonable computational behaviour for coercions?

Coercions are maps

What is a reasonable computational behaviour for coercions?

$$
\begin{aligned}
\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}[] &\cong [] \\
\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}(a :: l) &\cong (\mathrm{coe}_{A,A'}\ a) :: (\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}\ l) \\
(\mathrm{coe}_{A\to B,A'\to B'}\ f)\ u &\cong \mathrm{coe}_{B,B'}(f\ (\mathrm{coe}_{A',A}\ u)) \\
&\vdots
\end{aligned}
$$

Coercions are maps

What is a reasonable computational behaviour for coercions?

$$
\begin{aligned}
\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}\,[] &\cong [] \\
\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}\,(a :: l) &\cong (\mathrm{coe}_{A,A'}\ a) :: (\mathrm{coe}_{\mathrm{List}\ A,\mathrm{List}\ A'}\ l) \\
(\mathrm{coe}_{A\to B,A'\to B'}\ f)\ u &\cong \mathrm{coe}_{B,B'}(f\ (\mathrm{coe}_{A',A}\ u)) \\
&\vdots
\end{aligned}
$$

These are map operations! We better understand them before going further.

Definitional functoriality

Definitional functor laws

$\mathsf{MLTT}_{\mathbf{map}}$

Definitional functor laws

$\mathsf{MLTT}_{\mathrm{map}}$ each type former F comes with $\mathrm{dom}(F)$, $\mathrm{hom}(F)$ and map_F such that

Definitional functor laws

MLTT$_{\text{map}}$ each type former F comes with $\mathrm{dom}(F)$, $\mathrm{hom}(F)$ and map_F such that

$$\textsc{Map}\ \frac{\Gamma \vdash_{\text{map}} X, Y : \mathrm{dom}(F) \qquad \Gamma \vdash_{\text{map}} f : \mathrm{hom}_F(X, Y)}{\Gamma \vdash_{\text{map}} \mathrm{map}_F\, f : F\, X \to F\, Y}$$

Definitional functor laws

$\mathrm{MLTT}_{\mathrm{map}}$ each type former F comes with $\mathrm{dom}(F)$, $\mathrm{hom}(F)$ and map_F such that

$$\textsc{Map}\ \frac{\Gamma \vdash_{\mathrm{map}} X, Y : \mathrm{dom}(F) \quad \Gamma \vdash_{\mathrm{map}} f : \mathrm{hom}_F(X, Y)}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, f : F\, X \to F\, Y}$$

$$\textsc{MapId}\ \frac{\Gamma \vdash_{\mathrm{map}} X : \mathrm{dom}(F) \quad \Gamma \vdash_{\mathrm{map}} t : F\, X}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, \mathrm{id}^F_X\ t \cong t : F\, X}$$

Definitional functor laws

$\mathsf{MLTT}_{\mathrm{map}}$ each type former F comes with $\mathrm{dom}(F)$, $\mathrm{hom}(F)$ and map_F such that

$$\textsc{Map}\ \frac{\Gamma \vdash_{\mathrm{map}} X, Y : \mathrm{dom}(F) \quad \Gamma \vdash_{\mathrm{map}} f : \mathrm{hom}_F(X, Y)}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, f : F\,X \to F\,Y} \qquad \textsc{MapId}\ \frac{\Gamma \vdash_{\mathrm{map}} X : \mathrm{dom}(F) \quad \Gamma \vdash_{\mathrm{map}} t : F\,X}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, \mathrm{id}^F_X\ t \cong t : F\,X}$$

$$\textsc{MapComp}\ \frac{\Gamma \vdash_{\mathrm{map}} X, Y, Z : \mathrm{dom}(F) \quad \Gamma \vdash_{\mathrm{map}} g : \mathrm{hom}_F(X, Y) \quad \Gamma \vdash_{\mathrm{map}} f : \mathrm{hom}_F(Y, Z) \quad \Gamma \vdash_{\mathrm{map}} t : F\,X}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\ f\,(\mathrm{map}_F\, g\, t) \cong \mathrm{map}_F(f \circ^F g)\, t : F\,Z}$$

$\mathsf{MLTT}_{\mathrm{map}}$ each type former F comes with $\mathrm{dom}(F)$, $\mathrm{hom}(F)$ and map_F such that

$$\textsc{Map}\ \frac{\Gamma \vdash_{\mathrm{map}} X, Y : \mathrm{dom}(F) \qquad \Gamma \vdash_{\mathrm{map}} f : \mathrm{hom}_F(X, Y)}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, f : F\,X \to F\,Y}$$

$$\textsc{MapId}\ \frac{\Gamma \vdash_{\mathrm{map}} X : \mathrm{dom}(F) \qquad \Gamma \vdash_{\mathrm{map}} t : F\,X}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\, \mathrm{id}^F_X\ t \cong t : F\,X}$$

$$\textsc{MapComp}\ \frac{\Gamma \vdash_{\mathrm{map}} X, Y, Z : \mathrm{dom}(F) \qquad \Gamma \vdash_{\mathrm{map}} g : \mathrm{hom}_F(X, Y) \qquad \Gamma \vdash_{\mathrm{map}} f : \mathrm{hom}_F(Y, Z) \qquad \Gamma \vdash_{\mathrm{map}} t : F\,X}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_F\ f\,(\mathrm{map}_F\, g\, t) \cong \mathrm{map}_F(f \circ^F g)\, t : F\,Z}$$

+ congruences, specific laws for each F

$$\frac{\Gamma \vdash_{\mathrm{map}} (f, g) : \mathrm{hom}_\Pi((A, B), (A', B')) \qquad \Gamma \vdash_{\mathrm{map}} h : \Pi\, x{:}\, A.B \qquad \Gamma \vdash_{\mathrm{map}} a' : A'}{\Gamma \vdash_{\mathrm{map}} \mathrm{map}_\Pi\,(f, g)\, h\, a' \cong g\,(h\,(f\,a')) : B'[a']}$$

New equations for neutrals

This is *not* vanilla MLTT, where

$$\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ x) \not\equiv \mathrm{map}_{\mathrm{List}}(f \circ g)\ x$$

New equations for neutrals

This is *not* vanilla MLTT, where

$$\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ x) \not\equiv \mathrm{map}_{\mathrm{List}}(f \circ g)\ x$$

In general, problems with neutrals at positive types (= without η)

New equations for neutrals

This is *not* vanilla MLTT, where

$$\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ x) \not\equiv \mathrm{map}_{\mathrm{List}}(f \circ g)\ x$$

In general, problems with neutrals at positive types (= without η)

Can we add these equations in?

arXiv:1304.0809v3 [cs.PL] 17 Jun 2013

New Equations for Neutral Terms

A Sound and Complete Decision Procedure, Formalized

Guillaume Allais Conor McBride
University of Strathclyde
{guillaume.allais, conor.mcbride}@strath.ac.uk

Pierre Boutillier
PPS - Paris Diderot
pierre.boutillier@pps.univ-paris-diderot.fr

Abstract

The definitional equality of an intensional type theory is its test of type compatibility. Today's systems rely on ordinary evaluation semantics to compare expressions in types, frustrating users with type errors arising when evaluation fails to identify two 'obviously' equal terms. If only the machine could decide a richer theory! We propose a way to decide theories which supplement evaluation with 'ν-rules', rearranging the neutral parts of normal forms, and report a successful initial experiment.

We study a simple λ-calculus with primitive fold, map and append operations on lists and develop in Agda a sound and complete decision procedure for an equational theory enriched with monoid, functor and fusion laws.

Keywords Normalization by Evaluation, Logical Relations, Simply-Typed Lambda Calculus, Map Fusion

1. Introduction

The programmer working in intensional type theory is no stranger to 'obviously true' equations she wishes held *definitionally* for her program to typecheck without having to chase down ill-typed terms and brutally coerce them. In this article, we present one way to relax definitional equality, thus accommodating some of her longings. We distinguish three types of fundamental relations between terms.

The first denotes computational rules: it is untyped, *oriented* and denoted by $\rightsquigarrow$ in its one step version or $\rightsquigarrow^*$ when the reflexive transitive congruence closure is considered. In Table 1, we introduce a few such rules which correspond to the equations the programmer writes to define functions. They are referred to as δ (for *definitions*) and ι (for pattern-matching on *inductive* data) rules and hold computationally just like the more common β-rule.

The second is the judgmental equality ($\equiv$): it is typed, tractable

```
map : (a → b) → list a → list b
map f []        ⇝ []
map f (x :: xs) ⇝ f x :: map f xs

(++) : list a → list a → list a
[]      ++ ys ⇝ ys
x :: xs ++ ys ⇝ x :: (xs ++ ys)

fold : (a → b → b) → b → list a → b
fold c n []        ⇝ n
fold c n (x :: xs) ⇝ c x (fold c n xs)
```

Table 1. $\delta\iota$-rules - computational

```
Γ ⊢ f ≡ λ x. f x      : a → b
Γ ⊢ p ≡ (π₁ p , π₂ p) : a * b
Γ ⊢ u ≡ ()            : 1
```

Table 2. η-rules - canonicity

fied judgmentally. Table 3 shows a kit for building computationally inert *neutral* terms growing layers of thwarted progress around a variable which we dub the 'nut', together with some equations on neutral terms which hold only propositionally – until now. This paper shows how to extend the judgmental equality with these new 'ν-rules'. We gain, for example, that `map swap . map swap` $\equiv$ `id`, where `swap` swaps the elements of a pair.

```
⊠ □ a   π₁ □   π₂ □   □ ++ ys   map f □   fold n c □
```

DRAFT

New Equations for Neutral Terms

A Sound and Complete Decision Procedure, Formalized

Guillaume Allais Conor McBride
University of Strathclyde
{guillaume.allais, conor.mcbride}@strath.ac.uk

Pierre Boutillier
PPS - Paris Diderot
pierre.boutillier@pps.univ-paris-diderot.fr

arXiv:1304.0809v3 [cs.PL] 17 Jun 2013

Abstract

The definitional equality of an intensional t
of type compatibility. Today's systems rely o
semantics to compare expressions in types, t
type errors arising when evaluation fails to ide
equal terms. If only the machine could decid
propose a way to decide theories which supple
'ψ-rules', rearranging the neutral parts of norm
a successful initial experiment.

We study a simple λ-calculus with primiti
pend operations on lists and develop in Agda a
decision procedure for an equational theory e
functor and fusion laws.

Keywords Normalization by Evaluation, Log
Typed Lambda Calculus, Map Fusion

1. Introduction

The programmer working in intensional type
to 'obviously true' equations she wishes held
program to typecheck without having to chase
and brutally coerce them. In this article, we pr
definitional equality, thus accommodating s
We distinguish three types of fundamental rela

The first denotes computational rules: it is
denoted by $\rightsquigarrow$ in its one step version or $\rightsquigarrow^*$ wh
sitive congruence closure is considered. In Ta
few such rules which correspond to the equat
writes to define functions. They are referred to
and ι (for pattern-matching on *inductive* data)
putationally just like the more common β-rul

The second is the judgmental equality ($\equiv$)

Decidability of Conversion for Type Theory in Type Theory

ANDREAS ABEL, Gothenburg University, Sweden
JOAKIM ÖHMAN, IMDEA Software Institute, Spain
ANDREA VEZZOSI, Chalmers University of Technology, Sweden

Type theory should be able to handle its own meta-theory, both to justify its foundational claims and to obtain a verified implementation. At the core of a type checker for intensional type theory lies an algorithm to check equality of types, or in other words, to check whether two types are convertible. We have formalized in Agda a practical conversion checking algorithm for a dependent type theory with one universe à la Russell, natural numbers, and η-equality for Π types. We prove the algorithm correct via a Kripke logical relation parameterized by a suitable notion of equivalence of terms. We then instantiate the parameterized fundamental lemma twice: once to obtain canonicity and injectivity of type formers, and once again to prove the completeness of the algorithm. Our proof relies on inductive-recursive definitions, but not on the uniqueness of identity proofs. Thus, it is valid in variants of intensional Martin-Löf Type Theory as long as they support induction-recursion, for instance, Extensional, Observational, or Homotopy Type Theory.

CCS Concepts: • **Theory of computation → Type theory**; *Proof theory*;

Martin-Löf *à la* Coq

Jww. A. Adjedj, K. Maillard, P-M. Pédrot and L. Pujet

Martin-Löf logical framework

$$\vdash \Gamma \qquad \Gamma \vdash A \qquad \Gamma \vdash A \cong B \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t \cong u : A$$

Martin-Löf logical framework

$$\vdash \Gamma \qquad \Gamma \vdash A \qquad \Gamma \vdash A \cong B \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t \cong u : A$$

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

Martin-Löf logical framework

$$\vdash \Gamma \qquad \Gamma \vdash A \qquad \Gamma \vdash A \cong B \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t \cong u : A$$

+ type formers ($\mathbf{Type}, \Pi, \Sigma, x =_A y, \mathrm{W} \ldots$)

$$\textsc{AppCong}\ \frac{\Gamma \vdash t \cong t' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[u]}$$

$$\beta\textsc{Fun}\ \frac{\Gamma \vdash A \qquad \Gamma, x{:}\, A \vdash B \qquad \Gamma, x{:}\, A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x{:}\, A.t)\, u \cong t[u] : B[u]} \qquad \eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B}$$

Martin-Löf logical framework

$$\vdash \Gamma \qquad \Gamma \vdash A \qquad \Gamma \vdash A \cong B \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t \cong u : A$$

$$\textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A} \qquad \textsc{Conv}\ \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t : B}$$

+ type formers ($\mathrm{Type}, \Pi, \Sigma, x =_A y, \mathrm{W} \ldots$)

Derivations are not unique!

Declarative typing

Freestanding conversion rule

$$\frac{\Gamma \vdash^{\mathrm{de}} t : A \qquad \Gamma \vdash^{\mathrm{de}} A \cong B}{\Gamma \vdash^{\mathrm{de}} t : B}$$

Algorithmic typing (bidirectional)

Mode-constrained conversion

$$\frac{\Gamma \vdash^{\mathrm{al}} t \triangleright A \qquad \Gamma \vdash^{\mathrm{al}} A \cong B}{\Gamma \vdash^{\mathrm{al}} t \triangleleft B}$$

Declarative typing

Freestanding conversion rule

$$\frac{\Gamma \vdash^{\mathrm{de}} t : A \qquad \Gamma \vdash^{\mathrm{de}} A \cong B}{\Gamma \vdash^{\mathrm{de}} t : B}$$

Conversion mixes arbitrarily:

- Computation steps (β),
- Extensionality steps (η)
- Congruences,
- Transitivity, symmetry and reflexivity.

Algorithmic typing (bidirectional)

Mode-constrained conversion

$$\frac{\Gamma \vdash^{\mathrm{al}} t \triangleright A \qquad \Gamma \vdash^{\mathrm{al}} A \cong B}{\Gamma \vdash^{\mathrm{al}} t \triangleleft B}$$

Term/type-directed conversion alternating:

- Reduction to weak-head normal form,
- type-directed extensionality rules
- congruences

How can we compare the two presentations of MLTT?

How can we compare the two presentations of MLTT?

Algorithmic → Declarative: Admissibility of algorithmic rules

How can we compare the two presentations of MLTT?

Algorithmic → Declarative: Admissibility of algorithmic rules
Declarative → Algorithmic: Need to show that every derivation has a canonical form

How can we compare the two presentations of MLTT?

Algorithmic → Declarative: Admissibility of algorithmic rules
Declarative → Algorithmic: Need to show that every derivation has a canonical form

Gives decidability of conversion/typing... and coherence! (More on this later)

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

For $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

For $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\Gamma \vdash T \rightsquigarrow^{\star} \mathbf{N}}{\Gamma \Vdash T}$$

+ typing side-conditions

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

For $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\Gamma \vdash T \rightsquigarrow^{\star} \mathbf{N}}{\Gamma \Vdash T}$$

+ typing side-conditions

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

For $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\Gamma \vdash t \rightsquigarrow^{\star} 0 : \mathbf{N}}{\Gamma \Vdash_{\mathcal{T}} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} \mathrm{S}(t') : \mathbf{N} \quad \Gamma \Vdash t' : \mathbf{N}}{\Gamma \Vdash_{\mathcal{T}} t : T}$$

+ typing side-conditions

The logical relation (Abel et al.)

A (proof-relevant) predicate $\Gamma \Vdash A$ characterizing types by their weak head normal form.

For $\mathcal{A} :: \Gamma \Vdash A$, 3 predicates:

$$\Gamma \Vdash_{\mathcal{A}} A \cong B \qquad \Gamma \Vdash_{\mathcal{A}} t : A \qquad \Gamma \Vdash_{\mathcal{A}} t \cong u : A$$

Natural numbers:

$$\frac{\Gamma \vdash t \rightsquigarrow^{\star} 0 : \mathbf{N}}{\Gamma \Vdash_{\mathcal{T}} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} \mathrm{S}(t') : \mathbf{N} \quad \Gamma \Vdash t' : \mathbf{N}}{\Gamma \Vdash_{\mathcal{T}} t : T} \qquad \frac{\Gamma \vdash t \rightsquigarrow^{\star} n : \mathbf{N} \quad \Gamma \vdash n \approx n : \mathbf{N}}{\Gamma \Vdash_{\mathcal{T}} t : T}$$

+ typing side-conditions

- mutual definition of $\Gamma \Vdash A$ and $\Gamma \Vdash t : A$
- reducibility at the universe $\Gamma \Vdash A : \mathrm{Type}$ is basically $\Gamma \Vdash A$...

Flirting with logical limits

- mutual definition of $\Gamma \Vdash A$ and $\Gamma \Vdash t : A$
- reducibility at the universe $\Gamma \Vdash A : \mathrm{Type}$ is basically $\Gamma \Vdash A$...

(Small) induction-recursion + stratified definitions

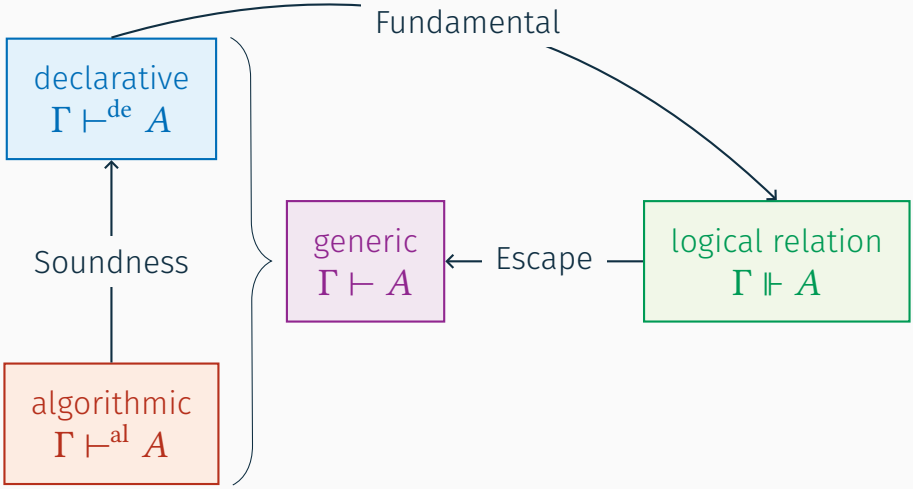
Properties of the logical relation

- Escape: $\Gamma \Vdash A \Rightarrow \Gamma \vdash A$
- Irrelevance
- Equivalence: reflexivity, symmetry, transitivity
- Stability by weakening
- Neutral reflection
- Closure by anti-reduction

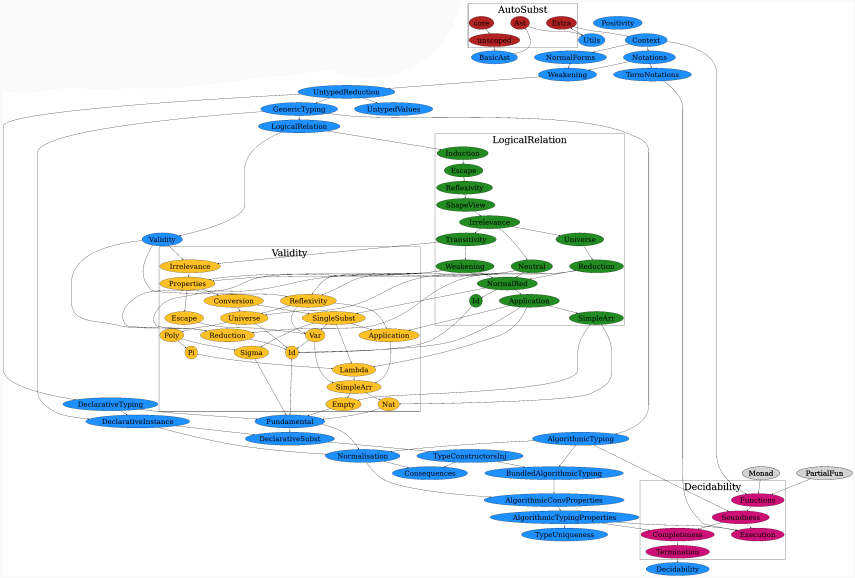
Properties of the logical relation

- Escape: $\Gamma \Vdash A \Rightarrow \Gamma \vdash A$
- Irrelevance
- Equivalence: reflexivity, symmetry, transitivity
- Stability by weakening
- Neutral reflection
- Closure by anti-reduction

Fundamental lemma: if $\Gamma \vdash^{\mathrm{de}} t : A$ then $\mathcal{A} :: \Gamma \Vdash A$ and $\Gamma \Vdash_{\mathcal{A}} t : A$


Fundamental
declarative
$\Gamma \vdash^{\mathrm{de}} A$
Soundness
algorithmic
$\Gamma \vdash^{\mathrm{al}} A$
generic
$\Gamma \vdash A$
Escape
logical relation
$\Gamma \Vdash A$

All formalised!



Back to business

The whole story extends...

The whole story extends...

To handle new equation for neutrals:

1. map fusion in reduction $\dfrac{\mathrm{neutral}\ n}{\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ n) \rightsquigarrow \mathrm{map}_{\mathrm{List}}(f \circ g)\ n}$

The whole story extends...

To handle new equation for neutrals:

1. map fusion in reduction
$$\frac{\mathrm{neutral}\ n}{\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ n) \rightsquigarrow \mathrm{map}_{\mathrm{List}}(f \circ g)\ n}$$

2. identity in comparison
$$\frac{\Gamma \vdash_{\mathrm{map}} n \approx n' : \mathrm{List}\ A \qquad \Gamma, x{:}\ A \Vdash f\ x \cong x : A}{\Gamma \Vdash \mathrm{map}_{\mathrm{List}}\ f\ n \cong n' : \mathrm{List}\ A}$$

The whole story extends...

To handle new equation for neutrals:

1. map fusion in reduction
$$\frac{\mathrm{neutral}\ n}{\mathrm{map}_{\mathrm{List}}\ f\ (\mathrm{map}_{\mathrm{List}}\ g\ n) \rightsquigarrow \mathrm{map}_{\mathrm{List}}(f \circ g)\ n}$$

2. identity in comparison
$$\frac{\Gamma \vdash_{\mathrm{map}} n \approx n' : \mathrm{List}\ A \qquad \Gamma, x{:}\ A \Vdash f\ x \cong x : A}{\Gamma \Vdash \mathrm{map}_{\mathrm{List}}\ f\ n \cong n' : \mathrm{List}\ A}$$

List (⚙), W, Id, + ... (✒)

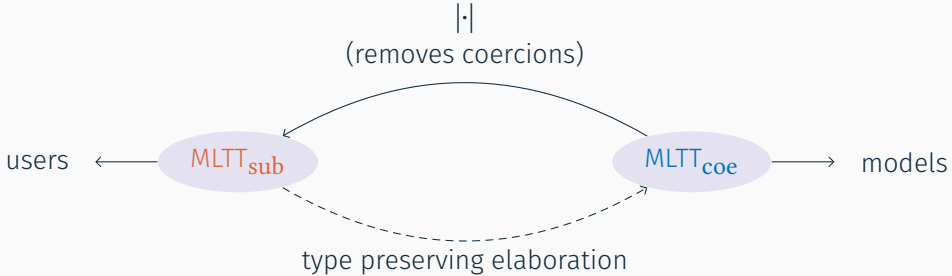
The story still extends...

The story still extends...

- extend conversion to subtyping, reducible conversion to reducible subtyping
- $\mathbf{coe}_{A,B}\, t$ reduces A, B, then applies the relevant $\mathbf{map}$
- $\mathbf{coe}_{A,B}\, \mathbf{coe}_{A',B'}\, t$ is compacted if $A\ B\ A'\ B'$ are all neutral, or are positive types and t is neutral
- identity in comparison as before

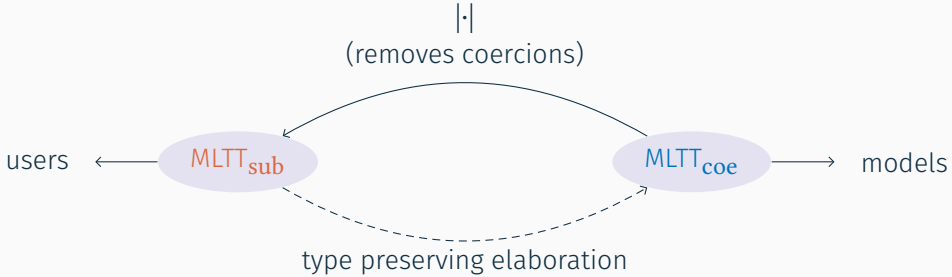
Only the algorithmic system!

Only the algorithmic system!



Much easier to show that elaboration preserves **algorithmic** typing.
Key lemma: coercions never block redexes.

Only the algorithmic system!



Much easier to show that elaboration preserves algorithmic typing.
Key lemma: coercions never block redexes.

For “free”: coherence up to conversion.

Wrapping up

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).
- MLTT_{map}: normalisation, decidability of type-checking… extending the logical relations.

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).
- $\mathrm{MLTT}_{\mathbf{map}}$: normalisation, decidability of type-checking… extending the logical relations.
- $\mathrm{MLTT}_{\mathbf{coe}}$: pen and paper, relatively straightforward extension of $\mathrm{MLTT}_{\mathbf{map}}$ (main difference: reduce types in $\mathbf{coe}_{A,B}$).

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).
- $\mathrm{MLTT}_{\mathrm{map}}$: normalisation, decidability of type-checking… extending the logical relations.
- $\mathrm{MLTT}_{\mathrm{coe}}$: pen and paper, relatively straightforward extension of $\mathrm{MLTT}_{\mathrm{map}}$ (main difference: reduce types in $\mathbf{coe}_{A,B}$).
- $\mathrm{MLTT}_{\mathrm{sub}}$: erasure from $\mathrm{MLTT}_{\mathrm{coe}}$ is type-preserving and invertible; in particular, "syntactic" coherence, up to conversion.

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).
- $\mathrm{MLTT}_{\mathrm{map}}$: normalisation, decidability of type-checking… extending the logical relations.
- $\mathrm{MLTT}_{\mathrm{coe}}$: pen and paper, relatively straightforward extension of $\mathrm{MLTT}_{\mathrm{map}}$ (main difference: reduce types in $\mathbf{coe}_{A,B}$).
- $\mathrm{MLTT}_{\mathrm{sub}}$: erasure from $\mathrm{MLTT}_{\mathrm{coe}}$ is type-preserving and invertible; in particular, "syntactic" coherence, up to conversion.
- Main tool: bidirectional/algorithmic/canonical derivations.

The results

- Martin-Löf *à la* Coq: all meta-theory of MLTT, formalised in Coq (and there's more I have not told you about).
- $\mathrm{MLTT}_{\mathrm{map}}$: normalisation, decidability of type-checking… extending the logical relations.
- $\mathrm{MLTT}_{\mathrm{coe}}$: pen and paper, relatively straightforward extension of $\mathrm{MLTT}_{\mathrm{map}}$ (main difference: reduce types in $\mathbf{coe}_{A,B}$).
- $\mathrm{MLTT}_{\mathrm{sub}}$: erasure from $\mathrm{MLTT}_{\mathrm{coe}}$ is type-preserving and invertible; in particular, "syntactic" coherence, up to conversion.
- Main tool: bidirectional/algorithmic/canonical derivations.

To interpret subsumptive structural subtyping, you (only) need functoriality equations.

What we're still unhappy about

Martin-Löf *à la* Coq

- renamings vs well-typed weakenings
- automation (proofs by reflection?)
- better structure and abstractions (categories?)
- more types!

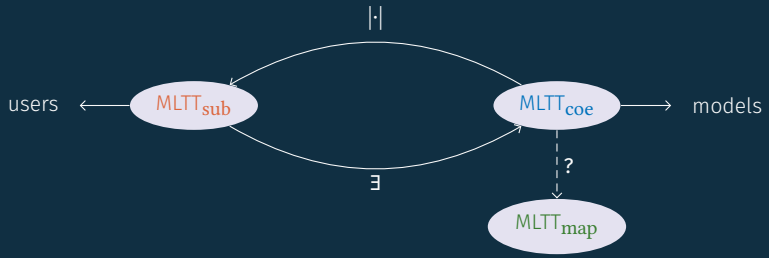
What we're still unhappy about

Martin-Löf *à la* Coq

- renamings vs well-typed weakenings
- automation (proofs by reflection?)
- better structure and abstractions (categories?)
- more types!

Functoriality and subtyping

- No formalisation of $\mathrm{MLTT}_{\mathrm{coe}}$/$\mathrm{MLTT}_{\mathrm{sub}}$
- No good story for elaborating $\mathrm{MLTT}_{\mathrm{coe}}$ to $\mathrm{MLTT}_{\mathrm{map}}$
- No general class of "good inductive types"



Thanks!

$$\Gamma \vdash_{\text{map}} \ \text{map}_F\, f : F\,X \to F\,Y \qquad \Gamma \vdash_{\text{map}} \ \text{map}_F\, \text{id}^F_X\ t \cong t : F\,X$$

$$\Gamma \vdash_{\text{map}} \ \text{map}_F\ f\,(\text{map}_F\, g\, t) \cong \text{map}_F(f \circ^F g)\, t : F\,Z$$