On Deciding Typing in Bidirectional Martin-Löf Type Theory

Making type-checkers complete by construction

Neel Krishnaswami, **Meven Lennon-Bertrand**

Types – June 12th 2023

University of Cambridge
Department of Computer Science and Technology

Why type-checking?

Why type-checking?

Decidable type-checking is crucial practically and philosophically.

Why type-checking?

Decidable type-checking is crucial practically and philosophically.

Elaboration, unification, etc. are nice, but fundamentally incomplete.

Why type-checking?

Decidable type-checking is crucial practically and philosophically.

Elaboration, unification, etc. are nice, but fundamentally incomplete.

Hundreds of papers on conversion… but not so much about decidability of type-checking.

Why type-checking?

Decidable type-checking is crucial practically and philosophically.

Elaboration, unification, etc. are nice, but fundamentally incomplete.

Hundreds of papers on conversion… but not so much about decidability of type-checking.

This talk!

Coping with undecidable type-checking

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).

Coping with undecidable type-checking

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).

Good news: we know how to cope!

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).
Good news: we know how to cope!

Solution 1:
Annotations

$\lambda\, x\colon A.\, t$

Coq, Lean…

All terms infer

Coping with undecidable type-checking

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).
Good news: we know how to cope!

Solution 1: Annotations	Solution 2: Restricted terms
$\lambda\, x\colon A.\, t$	$\lambda\, x.\, t$
Coq, Lean…	Agda…
All terms infer	Neutrals infer Normal forms check

Coping with undecidable type-checking

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).
Good news: we know how to cope!

Solution 1: Annotations	Solution 2: Restricted terms	Solution 3: Free-standing annotations
$\lambda\, x\colon A.\, t$	$\lambda\, x.\, t$	$\lambda\, x.\, t$ and $t :: A$
Coq, Lean…	Agda…	McBride, red* family…
All terms infer	Neutrals infer Normal forms check	Inferring terms Checking terms

Coping with undecidable type-checking

Bad news: type-checking for general dependent types is undecidable (Dowek, 2001).
Good news: we know how to cope!

Solution 1: Annotations	Solution 2: Restricted terms	Solution 3: Free-standing annotations
$\lambda\, x : A.\, t$	$\lambda\, x.\, t$	$\lambda\, x.\, t$ and $t :: A$
Coq, Lean...	Agda...	McBride, RED* family...
All terms infer	Neutrals infer Normal forms check	Inferring terms Checking terms

Our goal: unified setting & completeness proof, formalized.

Inferring and checkable terms, in the syntax

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda\, x{:}\, A.\, i$$

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x : A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\ c \mid \lambda\, x : A.\, i$$

Solution 1

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\, c \mid \lambda\, x{:}\, A.\, i$$

Solution 2

Inferring and checkable terms, in the syntax

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\ c \mid \lambda\, x{:}\, A.\, i$$

Solution 3

Inferring and checkable terms, in the syntax

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda\, x{:}\, A.\, i$$

$(\lambda\, x.\, c)\; u$ is not even valid syntax!

Inferring and checkable terms, in the syntax

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\, c \mid \lambda\, x{:}\, A.\, i$$

$(\lambda\, x.\, c)\, u$ is not even valid syntax!

$$\dfrac{\dfrac{\Gamma \vdash A \triangleleft \qquad \Gamma, x{:}\, A \vdash t \triangleright B}{\Gamma \vdash \lambda\, x{:}\, A.\, t \triangleright \Pi\, x{:}\, A.B} \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash (\lambda\, x{:}\, A.\, t)\, u \triangleright B[u :: A]}$$

INFERRING AND CHECKABLE TERMS, IN THE SYNTAX

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x\colon A.B \mid \lambda\, x.\, c$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda\, x\colon A.\, i$$

$(\lambda\, x.\, c)\; u$ is not even valid syntax!

$$\dfrac{\dfrac{\Gamma \vdash A \triangleleft \qquad \Gamma, x\colon A \vdash t \triangleright B}{\Gamma \vdash \lambda\, x\colon A.\, t \triangleright \Pi\, x\colon A.B} \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash (\lambda\, x\colon A.\, t)\; u \triangleright B[u :: A]}$$

Complete, by construction.

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c \qquad \mid \Sigma\, x{:}\, A.B \mid \langle c, c\rangle \mid \mathbf{W}\, x{:}\, A.B \mid \sup(c, c)$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda\, x{:}\, A.\, i \qquad \mid i._1 \mid i._2 \mid \langle i, c\rangle_{x.B} \mid \mathrm{ind}_{\mathbf{W}}(i; x.A; c) \mid \sup_{x.B}(i, c)$$

$(\lambda\, x.\, c)\; u$ is not even valid syntax!

$$\dfrac{\dfrac{\Gamma \vdash A \triangleleft \qquad \Gamma, x{:}\, A \vdash t \triangleright B}{\Gamma \vdash \lambda\, x{:}\, A.\, t \triangleright \Pi\, x{:}\, A.B} \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash (\lambda\, x{:}\, A.\, t)\; u \triangleright B[u :: A]}$$

Complete, by construction.

Beware of substitutions!

$$(\lambda\, x\colon A.\; t)\; u \to t[u] \qquad ✗$$

Beware of substitutions!

$$(\lambda\, x{:}\, A.\, t)\, u \to t[u :: A] \qquad ✓$$

Beware of substitutions!

$$(\lambda\, x\colon A.\, t)\; u \to t[u :: A] \qquad ✓$$

Annotations reduce, type-directed (see observational equality, gradual cast calculus...):

$$((\lambda\, x.\, t) :: \Pi\, x\colon A.B)\; u \to (\lambda\, x\colon A.\, (t :: B))\; u \to (t[u :: A]) :: B[u :: A]$$

Beware of substitutions!

$$(\lambda\, x : A.\, t)\; u \to t[u :: A] \qquad ✓$$

Annotations reduce, type-directed (see observational equality, gradual cast calculus…):

$$((\lambda\, x.\, t) :: \Pi\, x : A.B)\; u \to (\lambda\, x : A.\, (t :: B))\; u \to (t[u :: A]) :: B[u :: A]$$

Conversion, too, is bidirectional (Abel et al., 2018):

$$\Gamma \vdash A \cong A' \qquad \text{and} \qquad \Gamma \vdash c \cong c' \triangleleft A \qquad \text{but} \qquad \Gamma \vdash n \approx n' \triangleright A$$

Beware of substitutions!

$$(\lambda\, x : A.\, t)\; u \to t[u :: A] \qquad ✓$$

Annotations reduce, type-directed (see observational equality, gradual cast calculus…):

$$((\lambda\, x.\, t) :: \Pi\, x : A.B)\; u \to (\lambda\, x : A.\, (t :: B))\; u \to (t[u :: A]) :: B[u :: A]$$

Conversion, too, is bidirectional (Abel et al., 2018):

$$\Gamma \vdash A \cong A' \qquad \text{and} \qquad \Gamma \vdash c \cong c' \triangleleft A \qquad \text{but} \qquad \Gamma \vdash n \approx n' \triangleright A$$

(Stuck) annotations should be ignored ($\mathrm{TT}^{\mathrm{obs}}$ again):

$$\frac{\Gamma \vdash n \approx n' \triangleright A}{\Gamma \vdash \underline{n} :: A' \approx n' \triangleright A'}$$

Should we compute annotations?

Annotation-free terms are exactly the normal/neutral forms:

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x\colon A.B \mid \lambda\, x.\, c \mid \Sigma\, x\colon A.B \mid \langle c, c\rangle \mid \mathbf{W}\; x\colon A.B \mid \mathrm{sup}(c, c)$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda x\colon A.i \mid i.{}_1 \mid i.{}_2 \mid \langle i, c\rangle_{x.B} \mid \mathrm{ind}_{\mathbf{W}}(i; x.A; c) \mid \mathrm{sup}_{x.B}(i, c)$$

Why?

Should we compute annotations?

Annotation-free terms are exactly the normal/neutral forms:

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c \mid \Sigma\, x{:}\, A.B \mid \langle c, c\rangle \mid \mathbf{W}\; x{:}\, A.B \mid \mathrm{sup}(c, c)$$

$$i ::= c :: A \mid x \mid i\; c \mid \lambda x{:}\, A.i \mid i.{}_1 \mid i.{}_2 \mid \langle i, c\rangle_{x.B} \mid \mathrm{ind}_{\mathbf{W}}(i; x.A; c) \mid \mathrm{sup}_{x.B}(i, c)$$

Why work hard to maintain annotations we know will disappear?

Should we compute annotations?

Annotation-free terms are exactly the normal/neutral forms:

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x\colon A.B \mid \lambda\, x.\, c \mid \Sigma\, x\colon A.B \mid \langle c, c\rangle \mid \mathbf{W}\ x\colon A.B \mid \mathrm{sup}(c, c)$$

$$i ::= c :: A \mid x \mid i\ c \mid \lambda x\colon A.i \mid i.{}_1 \mid i.{}_2 \mid \langle i, c\rangle_{x.B} \mid \mathrm{ind}_{\mathbf{W}}(i; x.A; c) \mid \mathrm{sup}_{x.B}(i, c)$$

Why work hard to maintain annotations we know will disappear?

- Typable intermediate computation steps are nice…
- but if we only care about fast comparison, we should not bother.

The plan is laid out… and the formalization is ongoing.

Where are we at?

The plan is laid out… and the formalization is ongoing.

Luckily, we already have formalized logical relations for ∼Solution 1, in Coq.
Listen to Kenji at 15:00!

Thank you!

$$c, A, B ::= \underline{i} \mid \square_k \mid \Pi x{:}\, A.B \mid \lambda\, x.\, c \mid \Sigma\, x{:}\, A.B \mid \langle c, c\rangle \mid \mathbf{W}\, x{:}\, A.B \mid \mathrm{sup}(c, c)$$

$$i ::= c :: A \mid x \mid i\ c \mid \lambda x{:}\, A.i \mid i._1 \mid i._2 \mid \langle i, c\rangle_{x.B} \mid \mathrm{ind}_{\mathbf{W}}(i; x.A; c) \mid \mathrm{sup}_{x.B}(i, c)$$

Bibliography

[1] Gilles Dowek. “Chapter 16 - Higher-Order Unification and Matching”. In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Handbook of Automated Reasoning. North-Holland, 2001, pp. 1009–1062. DOI: 10.1016/B978-044450813-3/50018-7.

[2] 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.