On Weak Equality Reflection in MLTT with Propositional Truncation

Felix Bradley and Zhaohui Luo

Department of Computer Science
Royal Holloway, University of London

11 June 2025

Presentation Overview

1. Weak Equality Reflection

2. Type Theory for Semantics

3. MLTT with Propositional Truncation

4. Analysing the Metatheory

Motivation: Using MLTT as a foundation/semantics

Introduction

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Problem: MLTT has difficulties for these use-cases

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Problem: MLTT has difficulties for these use-cases

- Does not have both a weak/strong existential

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Problem: MLTT has difficulties for these use-cases

- Does not have both a weak/strong existential
- The counting problem

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Problem: MLTT has difficulties for these use-cases

- Does not have both a weak/strong existential
- The counting problem

Idea: Maybe some small changes might solve these problems?

Motivation: Using MLTT as a foundation/semantics

- Program specification/analysis
- Natural language semantics
- Large body of work using impredicative TTs
- Can it be done with predicative TTs?

Problem: MLTT has difficulties for these use-cases

- Does not have both a weak/strong existential
- The counting problem

Idea: Maybe some small changes might solve these problems?

Clarification: We're using taking MLTT to be Martin-Löf's intensional intuitionistic type theory

Equality Reflection

Two types of equality reflection: strong, and weak

Equality Reflection

Two types of equality reflection: strong, and weak

Strong equality reflection is deriving judgemental equality from propositional equality

Equality Reflection

Two types of equality reflection: strong, and weak

Strong equality reflection is deriving judgemental equality from propositional equality

May involve inclusion of a rule such as

$$\frac{\Gamma \vdash x, y : A \qquad \Gamma \vdash p : \mathrm{Eq}(A, x, y)}{\Gamma \vdash x = y : A}$$

Equality Reflection

Weak equality reflection is where judgemental equality and propositional equality coincide

[1] Identity from indiscernability; $\stackrel{\text{def}}{=} \forall(P : A \to \mathcal{U}), P(x) \to P(y)$

Equality Reflection

Weak equality reflection is where judgemental equality and propositional equality coincide

Admissibility of rules such as

$$\frac{\langle\rangle \vdash \mathsf{Id}(A, x, y)\ \mathsf{true}}{\langle\rangle \vdash x = y : A} \qquad \frac{\langle\rangle \vdash x =_A y\ \mathsf{true}}{\langle\rangle \vdash x = y : A}$$

where $\mathsf{Id}(A, x, y)$ is the identity type

where $x =_A y$ is Leibniz equality[1]

[1] Identity from indiscernability; $\stackrel{\text{def}}{=} \forall(P : A \to \mathcal{U}), P(x) \to P(y)$

Weak equality reflection is where judgemental equality and propositional equality coincide

Admissibility of rules such as

$$\frac{\langle\rangle \vdash \mathsf{Id}(A, x, y)\ \mathsf{true}}{\langle\rangle \vdash x = y : A} \qquad \frac{\langle\rangle \vdash x =_A y\ \mathsf{true}}{\langle\rangle \vdash x = y : A}$$

where $\mathsf{Id}(A, x, y)$ is the identity type

where $x =_A y$ is Leibniz equality[1]

May not have an internal method to go from one to the other

[1] Identity from indiscernability; $\stackrel{\text{def}}{=} \forall (P : A \to \mathcal{U}), P(x) \to P(y)$

Many type theories have weak equality reflection, such as MLTT[2] and UTT[3]

[2]Theorem on p102 of [ML75]

[3]Theorem 5.9 of [Luo94]

Weak Equality Reflection

Many type theories have weak equality reflection, such as MLTT[2] and UTT[3]

Others don't have weak equality reflection, such as traditional homotopy type theory [Uni13]

[2] Theorem on p102 of [ML75]
[3] Theorem 5.9 of [Luo94]

Weak Equality Reflection

Many type theories have weak equality reflection, such as MLTT[2] and UTT[3]

Others don't have weak equality reflection, such as traditional homotopy type theory [Uni13]

Theorem

Weak equality reflection does not hold for homotopy type theory

Proof (Shulman?)

Sketch: Take the higher inductive type S^1 defined by the point $\mathsf{base} : S^1$ and the non-trivial path $\mathsf{loop} : \mathsf{Id}(S^1, \mathsf{base}, \mathsf{base})$. Then the type $\Sigma(x : S^1).\,\mathsf{Id}(S^1, \mathsf{base}, x)$ is a mere proposition, but $(\mathsf{base}, \mathsf{refl}_{\mathsf{base}})$ and $(\mathsf{base}, \mathsf{loop})$ are constructed differently.

[2] Theorem on p102 of [ML75]

[3] Theorem 5.9 of [Luo94]

Type theories can be used for program specification and analysis

MLTT for Program Analysis

Type theories can be used for program specification and analysis

We can define a program specification as a pair (A, p) where A is a type, and p is a predicate $p : A \to \mathcal{U}$

MLTT for Program Analysis

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Type theories can be used for program specification and analysis

We can define a program specification as a pair (A, p) where A is a type, and p is a predicate $p : A \to \mathcal{U}$

Example: Identity functions of a type A

$$(A \to A, \quad \lambda(f : A \to A).\Pi(x : A).\,\mathrm{Id}(A, f(x), x))$$

MLTT for Program Analysis

Type theories can be used for program specification and analysis

We can define a program specification as a pair (A, p) where A is a type, and p is a predicate $p : A \to \mathcal{U}$

Example: Identity functions of a type A

$$(A \to A, \quad \lambda(f : A \to A).\Pi(x : A).\,\mathsf{Id}(A, f(x), x))$$

Example: Sorting functions of $\mathsf{List}(\mathbb{N})$

$$(\mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N}), \quad \lambda(f : \mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N})).\Pi(x : \mathsf{List}(\mathbb{N})). \\ \mathsf{Sorted}(f(x)) \wedge \mathsf{isPermutation}(x, f(x)))$$

MLTT for Program Analysis

We don't necessarily want e.g. function extensionality for program specification/analysis

MLTT for Program Analysis

We don't necessarily want e.g. function extensionality for program specification/analysis

Example: $\mathsf{BubbleSort}, \mathsf{MergeSort} : \mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N})$

MLTT for Program Analysis

We don't necessarily want e.g. function extensionality for program specification/analysis

Example: $\mathsf{BubbleSort}, \mathsf{MergeSort} : \mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N})$

We should expect $\forall \bar{x} : \mathsf{List}(\mathbb{N}), \mathsf{Eq}(\mathsf{List}(\mathbb{N}), \mathsf{BubbleSort}(\bar{x}), \mathsf{MergeSort}(\bar{x}))$

MLTT for Program Analysis

We don't necessarily want e.g. function extensionality for program specification/analysis

Example: $\mathsf{BubbleSort}, \mathsf{MergeSort} : \mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N})$

We should expect $\forall \bar{x} : \mathsf{List}(\mathbb{N}), \mathsf{Eq}(\mathsf{List}(\mathbb{N}), \mathsf{BubbleSort}(\bar{x}), \mathsf{MergeSort}(\bar{x}))$

However BubbleSort and MergeSort are two different algorithms with different computational properties

MLTT for Program Analysis

We don't necessarily want e.g. function extensionality for program specification/analysis

Example: $\mathsf{BubbleSort}, \mathsf{MergeSort} : \mathsf{List}(\mathbb{N}) \to \mathsf{List}(\mathbb{N})$

We should expect $\forall \bar{x} : \mathsf{List}(\mathbb{N}), \mathsf{Eq}(\mathsf{List}(\mathbb{N}), \mathsf{BubbleSort}(\bar{x}), \mathsf{MergeSort}(\bar{x}))$

However BubbleSort and MergeSort are two different algorithms with different computational properties

Here, propositional equality is used to explore expected behaviour of computational (definitional) equality

Type theories can be used for natural language semantics

[4] Also referred to as modern type theories and MTT-semantics

Type theories can be used for natural language semantics

Large body of work primarily based in impredicative type theories

[4] Also referred to as modern type theories and MTT-semantics

MLTT for Natural Language Semantics

Type theories can be used for natural language semantics

Large body of work primarily based in impredicative type theories

Initial work such as Montague semantics based on simple type theory [Mon74]

[4] Also referred to as modern type theories and MTT-semantics

MLTT for Natural Language Semantics

Type theories can be used for natural language semantics

Large body of work primarily based in impredicative type theories

Initial work such as Montague semantics based on simple type theory [Mon74]

Formal semantics further developed using dependent type theories[4] [Ran94, CL20, Luo24]

[4] Also referred to as modern type theories and MTT-semantics

MLTT for Natural Language Semantics

Type theories can be used for natural language semantics

Large body of work primarily based in impredicative type theories

Initial work such as Montague semantics based on simple type theory [Mon74]

Formal semantics further developed using dependent type theories[4] [Ran94, CL20, Luo24]

Example:

"Peter owns a cat."

[4] Also referred to as modern type theories and MTT-semantics

MLTT for Natural Language Semantics

Type theories can be used for natural language semantics

Large body of work primarily based in impredicative type theories

Initial work such as Montague semantics based on simple type theory [Mon74]

Formal semantics further developed using dependent type theories[4] [Ran94, CL20, Luo24]

Example:

"Peter owns a cat."

Correct semantics for MLTT would be:

$$\Sigma(x : \mathrm{Cat}).\mathrm{owns}(\mathrm{Peter}, x)$$

[4] Also referred to as modern type theories and MTT-semantics

Side tangent: why impredicative type theories?

Side tangent: why impredicative type theories?

In MLTT, we only have strong existential:

$$\Sigma(x : \mathrm{Cat}).\mathrm{owns}(\mathrm{Peter}, x)$$

Side tangent: why impredicative type theories?

In MLTT, we only have strong existential:

$$\Sigma(x : \text{Cat}).\text{owns}(\text{Peter}, x)$$

Σ plays two roles in this example: *existential quantifier* and *subset construction*

Side tangent: why impredicative type theories?

In MLTT, we only have strong existential:

$$\Sigma(x : \mathrm{Cat}).\mathrm{owns}(\mathrm{Peter}, x)$$

Σ plays two roles in this example: *existential quantifier* and *subset construction*

This results in some unusual consequences. For example [Esc17], for a function $f : X \to Y$:

$$\mathrm{image}(f) \stackrel{\text{def}}{=} \Sigma(y : Y).\Sigma(x : X).\,\mathrm{Eq}(Y, f(x), y)$$

But then we obtain $\mathrm{image}(f) \cong X$

Semantics for adjectival modification first proposed and studied by Mönnich [Mön85] and Sundholm [Sun86]

Semantics for adjectival modification first proposed and studied by Mönnich [Mön85] and Sundholm [Sun86]

Core concept: Represent how an adjective modifies a noun through a dependent pair type

The Counting Problem

Semantics for adjectival modification first proposed and studied by Mönnich [Mön85] and Sundholm [Sun86]

Core concept: Represent how an adjective modifies a noun through a dependent pair type

Example:

"Abed eats some burnt toast."

The Counting Problem

Semantics for adjectival modification first proposed and studied by Mönnich [Mön85] and Sundholm [Sun86]

Core concept: Represent how an adjective modifies a noun through a dependent pair type

Example:

"Abed eats some burnt toast."

Correct semantics for MLTT would be:

$$\Sigma(x : \text{Toast}).\text{eats}(\text{Abed}, x) \wedge \text{burnt}(x)$$

The Counting Problem

Example:

"There is one black cat in the garden."

The Counting Problem

Example:

"There is one black cat in the garden."

Correct semantics in MLTT would be:

$$\text{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

Example:

"There is one black cat in the garden."

Correct semantics in MLTT would be:

$$\text{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

...right?

$$\mathsf{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \land \text{black}(x)) = 1$$

What if we have more than one proof that a cat is black?

$$\mathsf{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

What if we have more than one proof that a cat is black?

Ideal solution: We want proof irrelevance!

The Counting Problem

$$\text{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

What if we have more than one proof that a cat is black?

Ideal solution: We want proof irrelevance!

Problem: MLTT has a types-as-propositions logic, so adding proof irrelevance everywhere causes other problems

The Counting Problem

$$\text{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

What if we have more than one proof that a cat is black?

Ideal solution: We want proof irrelevance!

Problem: MLTT has a types-as-propositions logic, so adding proof irrelevance everywhere causes other problems

Solution 1: Extend MLTT with an (impredicative) universe of mere propositions [GCST19]

The Counting Problem

$$\mathsf{Cardinality}(\Sigma(x : \text{Cat}).\text{location}(x, \text{Garden}) \wedge \text{black}(x)) = 1$$

What if we have more than one proof that a cat is black?

Ideal solution: We want proof irrelevance!

Problem: MLTT has a types-as-propositions logic, so adding proof irrelevance everywhere causes other problems

Solution 1: Extend MLTT with an (impredicative) universe of mere propositions [GCST19]

Solution 2: Extend MLTT with propositional truncation so we have access to both data types and mere propositions

What is Propositional Truncation?

Propositional truncation forces a type to become a mere proposition

What is Propositional Truncation?

Propositional truncation forces a type to become a mere proposition

Define $\mathsf{isProp}(A) \stackrel{\text{def}}{=} \Pi(x, y : A).\, \mathsf{Id}(A, x, y)$

What is Propositional Truncation?

Propositional truncation forces a type to become a mere proposition

Define $\mathrm{isProp}(A) \stackrel{\text{def}}{=} \Pi(x, y : A).\, \mathrm{Id}(A, x, y)$

$$\frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash \|A\| : \mathcal{U}} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash |a| : \|A\|}$$

$$\frac{\Gamma \vdash \mathrm{isProp}(B)\ \mathrm{true} \qquad \Gamma \vdash f : A \to B}{\Gamma \vdash \kappa_A(f) : \|A\| \to B}$$

$$\frac{\Gamma \vdash \mathrm{isProp}(B)\ \mathrm{true} \qquad \Gamma \vdash f : A \to B \qquad \Gamma \vdash |a| : \|A\|}{\Gamma \vdash \kappa_A(f, |a|) = f(a) : B}$$

What is Propositional Truncation?

Propositional truncation forces a type to become a mere proposition

Define $\text{isProp}(A) \stackrel{\text{def}}{=} \Pi(x, y : A).\,\text{Id}(A, x, y)$

$$\frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash \|A\| : \mathcal{U}} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash |a| : \|A\|}$$

$$\frac{\Gamma \vdash \text{isProp}(B)\ \text{true} \quad \Gamma \vdash f : A \to B}{\Gamma \vdash \kappa_A(f) : \|A\| \to B}$$

$$\frac{\Gamma \vdash \text{isProp}(B)\ \text{true} \quad \Gamma \vdash f : A \to B \quad \Gamma \vdash |a| : \|A\|}{\Gamma \vdash \kappa_A(f, |a|) = f(a) : B}$$

Key point: every $x, y : \|A\|$ are propositionally equal

$MLTT_h$

We define $MLTT_h$ as Martin-Löf's intensional intuitionistic type theory extended with the prior rules for propositional truncation

We define $MLTT_h$ as Martin-Löf's intensional intuitionistic type theory extended with the prior rules for propositional truncation

In $MLTT_h$, we can now given correct semantics to

We define $\mathrm{MLTT_h}$ as Martin-Löf's intensional intuitionistic type theory extended with the prior rules for propositional truncation

In $\mathrm{MLTT_h}$, we can now given correct semantics to

"There is one black cat in the garden."

We define MLTT$_h$ as Martin-Löf's intensional intuitionistic type theory extended with the prior rules for propositional truncation

In MLTT$_h$, we can now given correct semantics to

"There is one black cat in the garden."

Correct semantics would be:

$$\mathsf{Cardinality}(\Sigma(x : \mathrm{Cat}).\ \|\mathrm{location}(x, \mathrm{Garden})\| \wedge \|\mathrm{black}(x)\|) = 1$$

We define $\mathsf{MLTT_h}$ as Martin-Löf's intensional intuitionistic type theory extended with the prior rules for propositional truncation

In $\mathsf{MLTT_h}$, we can now given correct semantics to

"There is one black cat in the garden."

Correct semantics would be:

$$\mathsf{Cardinality}(\Sigma(x : \mathsf{Cat}). \|\mathsf{location}(x, \mathsf{Garden})\| \wedge \|\mathsf{black}(x)\|) = 1$$

Weak Equality Reflection

Propositional truncation of a type behaves like a higher-inductive type

Weak Equality Reflection

Propositional truncation of a type behaves like a higher-inductive type

Theorem

Weak equality reflection does not hold for $\mathsf{MLTT_h}$

Proof

Sketch: Take the mere proposition $\|\mathbf{1}+\mathbf{1}\|$. Then $|\mathsf{inl}\,*|$ and $|\mathsf{inr}\,*|$ are propositionally equal within this type, but are constructed differently and thus judgementally distinct.

While $MLTT_h$ solves the counting problem, weak equality reflection does not hold

While $\mathrm{MLTT_h}$ solves the counting problem, weak equality reflection does not hold

Lack of weak equality reflection may cause other problems for applications

While MLTT_h solves the counting problem, weak equality reflection does not hold

Lack of weak equality reflection may cause other problems for applications

However, MLTT_h is defined as an extension of MLTT, and so contains an MLTT-like subsystem

While $MLTT_h$ solves the counting problem, weak equality reflection does not hold

Lack of weak equality reflection may cause other problems for applications

However, $MLTT_h$ is defined as an extension of MLTT, and so contains an MLTT-like subsystem

Does weak equality reflection hold for this MLTT-like subsystem?

Final Remarks

- Type theories for programme specification/analysis enjoy weak equality reflection
- Prior work for these applications rely on impredicative type theories - we're working towards including MLTT
- Adding propositional truncation to MLTT loses weak equality reflection
- Is there a (useful) subset of $\mathrm{MLTT_h}$ which still has weak equality reflection?

References I

Stergios Chatzikyriakidis and Zhaohui Luo.
Formal Semantics in Modern Type Theories.
Wiley/ISTE, 2020.

Martín Hötzel Escardó.
Logic in univalent type theory, July 2017.
Presented as part of Big Proof.

Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabaeau.
Definitional proof-irrelevance without k.
volume 3. Jan 2019.

Zhaohui Luo.
Computation and Reasoning: A Type Theory for Computer Science.
Oxford University Press, London, March 1994.

Zhaohui Luo.
Modern Type Theories: Their Development and Applications.
Tsinghua University Press, 2024.
(In Chinese).

Per Martin-Löf.
About models for intuitionistic type theories and the notion of definitional equality.
Studies in Logic and the Foundations of Math, 82:81–109, 1975.

Richard Montague.
Formal Philosophy.
Yale University Press, 1974.
Collected papers edited by R. Thomason.

References II

Uwe Mönnich.
Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen.
Habilitation. University of Tübingen, 1985.

Aarne Ranta.
Type-Theoretical Grammar.
Oxford University Press, Oxford, 1994.

Göran Sundholm.
Proof theory and meaning.
In *Handbook of philosophical logic*, pages 471–506. Springer, 1986.

The Univalent Foundations Program.
Homotopy Type Theory: Univalent Foundations of Mathematics.
`https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.