A Multiverse Type Theory

Josselin Poiret

Nantes Université; Gallinette Team, Inria

April 5, 2024

More and more custom type theories are being developed:

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., "Definitional proof-irrelevance without K";

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., “Definitional proof-irrelevance without K”;
- MLTT + Exceptions in Pédrot et al., “A reasonably exceptional type theory”;

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., “Definitional proof-irrelevance without K”;
- MLTT + Exceptions in Pédrot et al., “A reasonably exceptional type theory”;
- Cubical Type Theory in Cohen et al., *Cubical Type Theory: a constructive interpretation of the univalence axiom*;

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., "Definitional proof-irrelevance without K";
- MLTT + Exceptions in Pédrot et al., "A reasonably exceptional type theory";
- Cubical Type Theory in Cohen et al., *Cubical Type Theory: a constructive interpretation of the univalence axiom*;
- MLTT + Internal Language of presheaves;

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., “Definitional proof-irrelevance without K”;
- MLTT + Exceptions in Pédrot et al., “A reasonably exceptional type theory”;
- Cubical Type Theory in Cohen et al., *Cubical Type Theory: a constructive interpretation of the univalence axiom*;
- MLTT + Internal Language of presheaves;
- 2LTT in Annenkov, Capriotti, and Kraus, “Two-Level Type Theory and Applications”;

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., “Definitional proof-irrelevance without K”;
- MLTT + Exceptions in Pédrot et al., “A reasonably exceptional type theory”;
- Cubical Type Theory in Cohen et al., *Cubical Type Theory: a constructive interpretation of the univalence axiom*;
- MLTT + Internal Language of presheaves;
- 2LTT in Annenkov, Capriotti, and Kraus, “Two-Level Type Theory and Applications”;
- Guarded Type Theory in Gratzer et al., “Multimodal Dependent Type Theory”;

More and more custom type theories are being developed:

- MLTT + SProp in Gilbert et al., "Definitional proof-irrelevance without K";
- MLTT + Exceptions in Pédrot et al., "A reasonably exceptional type theory";
- Cubical Type Theory in Cohen et al., *Cubical Type Theory: a constructive interpretation of the univalence axiom*;
- MLTT + Internal Language of presheaves;
- 2LTT in Annenkov, Capriotti, and Kraus, "Two-Level Type Theory and Applications";
- Guarded Type Theory in Gratzer et al., "Multimodal Dependent Type Theory";
- MLTT + Erased Types (Ghost type theory), see next talk.

Introduction to SProp

Two universes in MLTT+SProp: Type and SProp.

Introduction to SProp

Two universes in MLTT+SProp: Type and SProp.

$$\frac{A : \mathrm{SProp} \qquad a : A \qquad b : A}{a \equiv b : A}$$

Introduction to SProp

Two universes in MLTT+SProp: Type and SProp.

$$\frac{A : \mathrm{SProp} \qquad a : A \qquad b : A}{a \equiv b : A}$$

We can squash types:

$$\frac{A : \mathrm{Type}}{\|A\| : \mathrm{SProp}} \qquad \frac{x : A}{\mathrm{sq}\, x : \|A\|}$$

Since $\mathrm{sq}\,\mathrm{true} \equiv \mathrm{sq}\,\mathrm{false} : \|\mathbb{B}\|$, we can't eliminate from $\|\mathbb{B}\|$ into Type the usual way.

Since $\mathrm{sq}\,\mathrm{true} \equiv \mathrm{sq}\,\mathrm{false} : \|\mathbb{B}\|$, we can't eliminate from $\|\mathbb{B}\|$ into Type the usual way.

Only $\bot_{\mathrm{SProp}} : \mathrm{SProp}$ enjoys an elimination principle into Type.

Introduction to (a simplified) ExcTT

Two universes in MLTT+Exc: Type and Exc.

Introduction to (a simplified) ExcTT

Two universes in MLTT+Exc: Type and Exc.

$$\frac{A : \mathrm{Exc}}{\mathrm{raise}_A : A}$$

Introduction to (a simplified) ExcTT

Two universes in MLTT+Exc: Type and Exc.

$$\frac{A : \mathrm{Exc}}{\mathrm{raise}_A : A}$$

$$\frac{A : \mathrm{Type}}{\mathrm{ex}\, A : \mathrm{Exc}} \qquad \frac{a : A}{\mathrm{pure}\, a : \mathrm{ex}\, A}$$

Elimination into Type from Exc needs to handle raise, with catch!

$$\frac{A, B : \text{Type} \qquad a_{\text{ex}} : \text{ex}\, A \qquad a : A \vdash t : B \qquad e : B}{\text{try}\,\text{pure}\, a \leftarrow a_{\text{ex}} \text{ in } t \text{ catch } e : B}$$

These variations on MLTT often combine multiple “universes” (or sorts) of types that behave differently.

These variations on MLTT often combine multiple “universes” (or sorts) of types that behave differently.

The isolation is key to keeping nice properties of the system:

$$\mathrm{raise}_{\bot} : \bot_{\mathrm{Exc}}$$

at the exceptional sort but the “pure” sort is still consistent, because eliminating $\bot_{\mathrm{Exc}}$ requires handling raise.

Might want to use sorts/universes for things more general than just MLTT!

Might want to use sorts/universes for things more general than just MLTT!

In Cubical Agda:

```
I : IUniv
```

Might want to use sorts/universes for things more general than just MLTT!

In Cubical Agda:

```
I : IUniv
```

Isolation of $\mathbb{I}$ from the rest of the system is key in Cubical, because provability of formulas in $\mathbb{I}$ is decidable.

This leads to a problem:

You have to choose your meta-theory and stick with it.

This leads to a problem:

You have to choose your meta-theory and stick with it.

Basically vendor lock-in for type theories!

(a) "Agda"

(b) "Coq"

Figure: Your favorite proof assistants

This talk is more of a survey and rough description of an early work-in-progress to spur discussion.

I don't have any definitive answers to all of these questions.

To support the generality of all these systems: we need a structural framework first.

To support the generality of all these systems: we need a structural framework first.

Some options:

- go PTS-style, an unpublished attempt was made with MuTT in Maillard et al., "The Multiverse: Logical Modularity for Proof Assistants", presented at WG6 2 years ago;
- MTT in Gratzer et al., "Multimodal Dependent Type Theory".

MuTT vs. MTT, structurally

MuTT	MTT

MuTT vs. MTT, structurally

MuTT	MTT
s sort	m mode

MuTT vs. MTT, structurally

MuTT	MTT
s sort	m mode
Γ ctx	Γ ctx @ m

MuTT vs. MTT, structurally

MuTT	MTT
s sort	m mode
Γ ctx	Γ ctx @ m
$\Gamma \vdash A : s$	$\Gamma \vdash A \,@\, m$

MuTT vs. MTT, structurally

MuTT	MTT
s sort	m mode
Γ ctx	Γ ctx @ m
$\Gamma \vdash A : s$	$\Gamma \vdash A @ m$
$\Gamma \vdash a : A$	$\Gamma \vdash a : A @ m$

MuTT vs. MTT, structurally

MuTT	MTT
s sort	m mode
Γ ctx	Γ ctx @ m
$\Gamma \vdash A : s$	$\Gamma \vdash A @ m$
$\Gamma \vdash a : A$	$\Gamma \vdash a : A @ m$
$\dfrac{\Gamma \vdash A : s}{\Gamma, A \text{ ctx}}$	$\dfrac{\mu : m \to n \qquad \Gamma.\text{🔒}_\mu \vdash A @ n}{\Gamma, (\mu \mid A) \text{ ctx} @ m}$

MuTT vs. MTT

MuTT is less verbose than MTT

MuTT vs. MTT

MuTT is less verbose than MTT but seemingly can't handle things like non-idempotent modalities.

MuTT vs. MTT

MuTT is less verbose than MTT but seemingly can't handle things like non-idempotent modalities.

MuTT's framework can accomodate inductives, eg. exceptional booleans in ExcTT, but with a complicated machinery of patterns and rewrite rules.

MuTT vs. MTT

MuTT is less verbose than MTT but seemingly can't handle things like non-idempotent modalities.

MuTT's framework can accomodate inductives, eg. exceptional booleans in ExcTT, but with a complicated machinery of patterns and rewrite rules.

Possible hope of a translation of the structural rules of MuTT to MTT.

Both share some shortcomings:

- No parametrization for `SProp` using them (cannot require *ad-hoc* definitional equalities);

Both share some shortcomings:

- No parametrization for `SProp` using them (cannot require *ad-hoc* definitional equalities);
- Lack *general* treatment of inductive types;

Both share some shortcomings:

- No parametrization for `SProp` using them (cannot require *ad-hoc* definitional equalities);
- Lack *general* treatment of inductive types;
- MLTT at every sort/mode.

Implementations

Implementations

- Simply-typed MTT: Ceulemans, Nuyts, and Devriese, "Sikkel: Multimode Simple Type Theory as an Agda Library";

Implementations

- Simply-typed MTT: Ceulemans, Nuyts, and Devriese, "Sikkel: Multimode Simple Type Theory as an Agda Library";
- Full MTT: Stassen, Gratzer, and Birkedal, "{mitten}: A Flexible Multimodal Proof Assistant";

Implementations

- Simply-typed MTT: Ceulemans, Nuyts, and Devriese, "Sikkel: Multimode Simple Type Theory as an Agda Library";
- Full MTT: Stassen, Gratzer, and Birkedal, "{mitten}: A Flexible Multimodal Proof Assistant";
- MTT with only one mode and predetermined modalities: Agda!

Implementations

- Simply-typed MTT: Ceulemans, Nuyts, and Devriese, “Sikkel: Multimode Simple Type Theory as an Agda Library”;
- Full MTT: Stassen, Gratzer, and Birkedal, “{mitten}: A Flexible Multimodal Proof Assistant”;
- MTT with only one mode and predetermined modalities: Agda!
- Coq’s implementation of Type, Prop and SProp inspired MuTT.

The case of inductive types

Inductives in MTT lead to new questions:

- Do inductives exist at all modes?
- Are there elimination principles through all modalities?

The case of inductive types

Inductives in MTT lead to new questions:

- Do inductives exist at all modes?
- Are there elimination principles through all modalities?

These can only be considered once we factor in each mode's specificities, so not covered by MTT!

The case of inductive types

Inductives in MTT lead to new questions:

- Do inductives exist at all modes?
- Are there elimination principles through all modalities?

These can only be considered once we factor in each mode's specificities, so not covered by MTT!

- Can't match on $\mathbb{B}$ in SProp to produce a value in Type, because $\mathrm{true} \equiv \mathrm{false}$ would definitionally collapse both elimination branches;

The case of inductive types

Inductives in MTT lead to new questions:

- Do inductives exist at all modes?
- Are there elimination principles through all modalities?

These can only be considered once we factor in each mode's specificities, so not covered by MTT!

- Can't match on $\mathbb{B}$ in SProp to produce a value in Type, because $\mathrm{true} \equiv \mathrm{false}$ would definitionally collapse both elimination branches;
- Have to take care of the raise term when eliminating the exceptional $\mathbb{B}$ into Type.

The case of identity types

MTT has identity types at every mode.

The case of identity types

MTT has identity types at every mode.

In ExcTT, are we really interested in $0 =_{\mathrm{ex}} 1$, since raise inhabits it?

The case of identity types

MTT has identity types at every mode.

In ExcTT, are we really interested in $0 =_{\mathrm{ex}} 1$, since raise inhabits it?

What if instead we had equalities all living at one mode, say SProp?

This is part of the approach of Pujet and Tabareau, “Observational Equality: Now for Good”.

The case of universes

MTT has universe types at every mode.

Key for some systems, for example Cubical!

The case of universes

MTT has universe types at every mode.

Key for some systems, for example Cubical!

For others, less so:

- Good luck using $U_{\mathrm{SProp}} \,@\, \mathrm{SProp}$, we want it at Type;

The case of universes

MTT has universe types at every mode.

Key for some systems, for example Cubical!

For others, less so:

- Good luck using $U_{\mathrm{SProp}} @ \mathrm{SProp}$, we want it at Type;
- We might want $U_{\mathrm{ex}} @ \mathrm{ex}$ but also have one at Type.

For all of these, we'd also want *sort-generic* theorems, eg. addition is commutative.

For all of these, we'd also want *sort-generic* theorems, eg. addition is commutative.

This suggests a variation of prenex sort/mode quantification, but would require quantifying over their interactions as well!

For all of these, we'd also want *sort-generic* theorems, eg. addition is commutative.

This suggests a variation of prenex sort/mode quantification, but would require quantifying over their interactions as well!

Note here that Coq already does this with sort polymorphism, see Pierre-Marie Pédrot's talk at TYPES 2023!

On the MTT side, there's also Andreas' talk at TYPES 2023.

Metatheoretical properties become more elusive:

Metatheoretical properties become more elusive:

What is canonicity for types in SProp?

Metatheoretical properties become more elusive:

What is canonicity for types in SProp?

What is consistency when $\mathrm{raise}_{\bot} : \bot_{\mathrm{Exc}}$?

What metrics should we use to evaluate those new type theories?

What metrics should we use to evaluate those new type theories?

- Consistency at Type in our new system;

What metrics should we use to evaluate those new type theories?

- Consistency at Type in our new system;
- More generally, conservativity for usual MLTT terms and types in Type (a form of relative canonicity).

  Note there are difficulties when there is a universe at Type, which will include new type formers;

What metrics should we use to evaluate those new type theories?

- Consistency at Type in our new system;
- More generally, conservativity for usual MLTT terms and types in Type (a form of relative canonicity).

  Note there are difficulties when there is a universe at Type, which will include new type formers;
- Decidability of typing, maybe even only at certain sorts (if that even makes sense).

More pragmatic concerns:

More pragmatic concerns:

How do we ensure the transition to the new system for our proof assistants is painless?

More pragmatic concerns:

How do we ensure the transition to the new system for our proof assistants is painless?

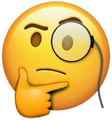
Example: Coq users have had to re-adapt common tactics to work with SProp.

In Agda, I can combine

```
--cohesion --experimental-irrelevance
--cubical --sized-types
```

In Agda, I can combine

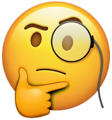
```
--cohesion --experimental-irrelevance
--cubical --sized-types
```



Has anyone in the room studied that exact combination of features?

In Agda, I can combine

```
--cohesion --experimental-irrelevance
--cubical --sized-types
```



Has anyone in the room studied that exact combination of features?

How could we justify this?

Can we glue multimodal/multisorted type theories which "agree" at Type (modulo universes)?

Could we glue their metatheoretical proofs together?

Can we glue multimodal/multisorted type theories which "agree" at Type (modulo universes)?

Could we glue their metatheoretical proofs together?

Roughly what Uemura's recent work attempted to do, presented at WG6 last year.

Will require solving the engineering problem of formalizing high-level proofs in the style of Sterling, Gratzer et al, or Bocquet-Kaposi-Sattler.

Will require solving the engineering problem of formalizing high-level proofs in the style of Sterling, Gratzer et al, or Bocquet-Kaposi-Sattler.

Finally, we want some variation on these systems and combinations thereof to be implemented by Coq/Agda.

Thank you for your attention!

Any questions/ideas?