Multimodal Dependent Type Theory

Daniel Gratzer[0] G.A. Kavvos[0] Andreas Nuyts[1] Lars Birkedal[0]

Wednesday May 27th, 2020

Stockholm University

[0]Aarhus University
[1]imec-DistriNet, KU Leuven

The problem

We'd like extend Martin-Löf Type Theory and apply it to new situations.

- Staged programming [PD01].
- Proof-irrelevance [Pfe01].
- Guarded recursion [Clo+15; BGM17; Gua18].
- Parametric quantification [ND18].
- Exotic models of computation [Bir00].
- (Abstract) topology [Shu18].
- Differential geometry [Wel18].

Where do modalities come in?

Martin-Löf Type Theory satisfies several desirable properties which help make it convenient to use (canonicity, decidability of type-checking, etc.).

Problem If we naively add new features, we will disrupt these properties.

Solution Modalities can manage new features in a controlled way.

Each example uses *modalities* to extended MLTT while preserving crucial properties.

A tangent: what exactly is a modality?

In general, people use *modality* to mean many different things:

1. Any unary type constructor.
2. A unary type constructor which is an *internal functor*.
3. A unary type constructor equipped with a *monad* structure.

For us, a modality is essentially a right adjoint.[1]

This restriction yields a practical syntax and still includes many examples.

[1] More specifically, a modality is essentially a *dependent* right adjoint [Bir+20]

A story of several type theories

Let us consider a representative example of how modal type theories are developed.

1. Work on guarded recursion converges towards the Fitch-style [BGM17; Clo18].
2. Birkedal et al. [Bir+20] isolate this into paradigmatic type theory.
3. Gratzer, Sterling, and Birkedal [GSB19] prove normalization for a similar system.

Each of these type theories build upon each other… but no reuse is possible.

Our Contribution: MTT

We introduce MTT: a type theory parameterized by a collection of modalities.

- MTT features usual connectives of Martin-Löf Type Theory, including a universe.
- The user can instantiate MTT with different collections of modalities.
- Important results such as canonicity are proven irrespective of the modalities.

We have applied MTT to several different situations:

- Axiomatic cohesion
- Degrees of relatedness
- Guarded recursion and warps
- Various classic modal type theories

Revisiting our story

Let us reconsider the previous example with [Bir+20] and [GSB19].

With MTT we would not design two separate type theories!

- Instantiate MTT twice to yield type theories similar to the originals.
- Prove normalization for MTT *once*, and transfer the result to both instantiations.

MTT makes the superficial similarity into a formal relationship.

Modal type theories, generally

Before diving into MTT, let's take the time to review modal type theories generally.

Main questions:

1. Why is it challenging to add a modality to MLTT?
2. What are the main lines of prior work?
3. What needs to be done to adapt any previous type theories handle multiple interacting modalities?

The functorial introduction rule

1. Let's say we're adding a single modality: $\langle \mu \mid - \rangle$.
2. We'd like to assume that $\langle \mu \mid - \rangle$ is in some sense functorial or left-exact, but not that it's fibered.

The functorial introduction rule

1. Let's say we're adding a single modality: $\langle \mu \mid - \rangle$.
2. We'd like to assume that $\langle \mu \mid - \rangle$ is in some sense functorial or left-exact, but not that it's fibered.

As a functor on (some variant of) the syntactic category

The functorial introduction rule

1. Let's say we're adding a single modality: $\langle \mu \mid - \rangle$.
2. We'd like to assume that $\langle \mu \mid - \rangle$ is in some sense functorial or left-exact, but not that it's fibered.

We don't assume $\langle \mu \mid A \rangle[\gamma] = \langle \mu \mid A[\gamma] \rangle$.

The functorial introduction rule

1. Let's say we're adding a single modality: $\langle \mu \mid - \rangle$.
2. We'd like to assume that $\langle \mu \mid - \rangle$ is in some sense functorial or left-exact, but not that it's fibered.

There is a tantalizing (but incorrect!) choice for an introduction rule, based on the functorial action:

$$\frac{\Gamma \vdash M : A}{\langle \mu \mid \Gamma \rangle \vdash \mathsf{mod}_{\mu}(M) : \langle \mu \mid A \rangle}$$

(NB: $\langle \mu \mid \Gamma \rangle$ can be thought of as traversing Γ and applying $\langle \mu \mid - \rangle$ to each type.)

The problem with the naïve functor rule

How do we commute substitutions past $\mathsf{mod}_\mu(M)$?

1. Suppose Γ, Δ are contexts, $\Gamma \vdash M : A$, and $\Delta \vdash \gamma : \langle \mu \mid \Gamma \rangle$ is a substitution.
2. What should $\Delta \vdash \mathsf{mod}_\mu(M)[\gamma] : \langle \mu \mid A \rangle[\gamma]$ be equal to?

There's no reason to expect that $\gamma = \mathsf{mod}_\mu(\gamma')$, so there's no way to push this substitution under $\mathsf{mod}_\mu(-)$ or $\langle \mu \mid - \rangle$.

Without additional structure this type theory will not enjoy a substitution principle.

A first solution: split contexts

- The issue is that a substitution into a modal context might not be modal itself.
- We can fix this by changing both contexts and substitutions to explicitly separate out a modal component.

Dual-context modal type theory

This is the *dual-context* or split context approach:

- We can split the context into pairs $\Delta; \Gamma$ (representing $\langle \mu \mid \Delta \rangle \times \Gamma$).
- The variable rule allows us to access Γ normally, but not Δ.
- The introduction rule is functoriality combined with weakening:

$$\frac{\cdot; \Delta \vdash M : A}{\Delta; \Gamma \vdash \mathsf{mod}_\mu(M) : \langle \mu \mid A \rangle}$$

- A substitution is a pair of substitutions $\Delta'; \Gamma' \vdash [\delta; \gamma] : \Delta; \Gamma$.
- To commute $[\delta; \gamma]$ past modal introduction, we push in $[\cdot; \delta]$.

This is one of the original approaches to modal type theory [PD01; Bd00; Shu18]

What about the elimination rule?

The elimination rule in the dual-context style smooths out the difference between

1. $x : \langle \mu \mid A \rangle$ in the normal zone
2. $x : A$ in the modal zone

$$\frac{x : \langle \mu \mid A \rangle \in \Gamma \qquad \Delta; \Gamma, b : \langle \mu \mid A \rangle \vdash B \text{ type} \qquad \Delta, y : A; \Gamma \vdash M : B[\text{mod}_\mu(y)/b]}{\Delta; \Gamma \vdash \text{let mod}_\mu(y) \leftarrow x \text{ in } M : B[x/b]}$$

What about the elimination rule?

The elimination rule in the dual-context style smooths out the difference between

1. $x : \langle \mu \mid A \rangle$ in the normal zone
2. $x : A$ in the modal zone

$$\frac{x : \langle \mu \mid A \rangle \in \Gamma \qquad \Delta; \Gamma, b : \langle \mu \mid A \rangle \vdash B \text{ type} \qquad \Delta, y : A; \Gamma \vdash M : B[\text{mod}_\mu(y)/b]}{\Delta; \Gamma \vdash \text{let mod}_\mu(y) \leftarrow x \text{ in } M : B[x/b]}$$

$$\frac{\begin{array}{c}\Delta; \Gamma, b : \langle \mu \mid A \rangle \vdash B \text{ type} \\ \Delta; \Gamma \vdash M_0 : \langle \mu \mid A \rangle \qquad \Delta, y : A; \Gamma \vdash M_1 : B[\text{mod}_\mu(y)/b]\end{array}}{\Delta; \Gamma \vdash \text{let mod}_\mu(y) \leftarrow M_0 \text{ in } M_1 : B[M_0/b]}$$

(For intuition, compare to J)

Limitations of the dual-context style

The dual-context approach works for a reasonable modality [Kav17].

However, it scales poorly to multiple modalities:

1. There's a explosion in contexts (one for every *combination* of modalities).
2. There is no clear introduction rules if these modalities are allowed to interact.
3. Even with one modality, the split-context style is awkward for dependence.

The problem with the introduction rule

There is no canonical choice for what variables to keep when introducing a modality.

Example:

1. Suppose $\langle \mu \mid - \rangle, \langle \nu \mid - \rangle, \langle \xi_0 \mid - \rangle, \langle \xi_1 \mid - \rangle$ are modalities.
2. Suppose further that $\langle \nu \mid - \rangle \cong \langle \nu \mid \langle \xi_i \mid - \rangle \rangle$.
3. When introducing ν, we should retain access to some form of μ zone.
4. But we could pick either to move them to the ξ_0 or ξ_1 zone.
5. There's no right answer.

It gets worse! We could have coercions, not isomorphisms, more modalities, etc..

Solving the easy problems: dependence and combinatorics

We can solve two of our three problems by switching away from split contexts.

$$\textit{(Ann. context)} \quad \Gamma, \Delta \quad ::= \quad \cdot \mid \Gamma, x : (\mu \mid A)$$

We need a bit more structure on our modalities: they form a category.

1. Instead of a μ-zone, we just tag μ-modal variables as such.
2. The identity modality tags normal variables.
3. Dependency is now more easily managed: a type depends on the prior context.

The elimination and variable rules

The elimination rule from the split-context style scales in an almost direct way:

$$\frac{\Gamma \vdash M_0 : \langle \mu \mid A \rangle \qquad \Gamma, x : (\mu \mid A) \vdash M_1 : A[\mathsf{mod}_\mu(x)/y]}{\Gamma \vdash \mathsf{let}\ \mathsf{mod}_\mu(x) \leftarrow M_0\ \mathsf{in}\ M_1 : A[M_0/y]}$$

In fact, the variable rule is also straightforward to fix for this new setting.

What about the problematic introduction rule?

The problem we didn't solve

How do we write an introduction rule?

The same problem persists. Suppose $\mu = \nu \circ \xi_0 = \nu \circ \xi_1$

$$\frac{x_0 : (\chi \mid A_0), \cdots, x_n : (\xi_i \mid A_n) \vdash M : A}{x_0 : (\nu \circ \chi \mid A_0), \cdots, x_n : (\mu \mid A_n) \vdash \mathsf{mod}_\nu(M) : \langle \nu \mid A \rangle}$$

What should we pick for i? We haven't really introduced any new ways to resolve this.

We have made the problem easier to write down however! (Small victories...)

Four potential ways to resolve this problem

I am aware of 4 distinct solutions to this problem:

1. We can introduce *delayed substitutions*.
2. Or require a *left-division* structure on modalities.
3. Or switch to the *Fitch-style*.
4. Or, finally, we can use MTT!

Four potential ways to resolve this problem

I am aware of 4 distinct solutions to this problem:

1. We can introduce *delayed substitutions*.
2. Or require a *left-division* structure on modalities.
3. Or switch to the *Fitch-style*.
4. Or, finally, we can use MTT!

We'll discuss these now, and spend the second half of the talk on MTT.

Delayed substitutions

A first solution to this problem is simply to give up on finding the right introduction.

Make the user tell us!

$$\frac{\Gamma \vdash \gamma : \mu \cdot \Gamma' \qquad \Gamma' \vdash M : A}{\Gamma \vdash \mathsf{mod}_{\mu}(M)^{\gamma} : \langle \mu \mid A \rangle^{\gamma}}$$

The advantage is uniformity

Delayed substitutions

A first solution to this problem is simply to give up on finding the right introduction.

Make the user tell us!

$$\frac{\Gamma \vdash \gamma : \mu \cdot \Gamma' \qquad \Gamma' \vdash M : A}{\Gamma \vdash \mathsf{mod}_\mu(M)^\gamma : \langle \mu \mid A \rangle^\gamma}$$

$\mu \cdot \Gamma'$ = precompose every annotation with μ

The advantage is uniformity

Delayed substitutions

A first solution to this problem is simply to give up on finding the right introduction.

Make the user tell us!

$$\frac{\Gamma \vdash \gamma : \mu \cdot \Gamma' \qquad \Gamma' \vdash M : A}{\Gamma \vdash \mathsf{mod}_\mu(M)^\gamma : \langle \mu \mid A \rangle^\gamma}$$

The advantage is uniformity .... but we need some way to resolve these substitutions:

$$\frac{\Gamma \vdash \delta_0 : \mu \cdot \Delta_0 \qquad \Delta_0 \vdash \delta_1 : \Delta_1 \qquad \Delta_1 \vdash M : A}{\Gamma \vdash \mathsf{mod}_\mu(M)^{(\mu \cdot \delta_1) \circ \delta_0} = \mathsf{mod}_\mu(M[\delta_1])^{\delta_0} : \langle \mu \mid A \rangle^{(\mu \cdot \delta_1) \circ \delta_0}}$$

Delayed substitutions

A first solution to this problem is simply to give up on finding the right introduction.

Make the user tell us!

$$\frac{\Gamma \vdash \gamma : \mu \cdot \Gamma' \qquad \Gamma' \vdash M : A}{\Gamma \vdash \mathsf{mod}_{\mu}(M)^{\gamma} : \langle \mu \mid A \rangle^{\gamma}}$$

The advantage is uniformity .... but we need some way to resolve these substitutions:

$$\frac{\Gamma \vdash \delta_0 : \mu \cdot \Delta_0 \qquad \Delta_0 \vdash \delta_1 : \Delta_1 \qquad \Delta_1 \vdash M : A}{\Gamma \vdash \mathsf{mod}_{\mu}(M)^{(\mu \cdot \delta_1) \circ \delta_0} = \mathsf{mod}_{\mu}(M[\delta_1])^{\delta_0} : \langle \mu \mid A \rangle^{(\mu \cdot \delta_1) \circ \delta_0}}$$

No more unique normal forms! This prevents us from proving decidability of type-checking.

Delayed substitutions: pros and cons

The main advantage of delayed substitutions is that it is uniform.

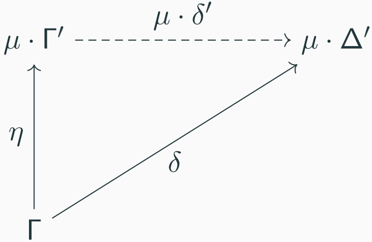
- It applies to substructural settings as well [LSR17].
- It scales in a straightforward way to dependent types [Biz+16].

The downside is that it almost surely destroys decidability of typechecking.

Avoiding delayed substitutions

The remaining approaches (left-division, Fitch-style, MTT) ensure a *universal delayed substitution*.

Suppose for every substitution $\Gamma \to \mu \cdot \Delta$, we have the following factorization:

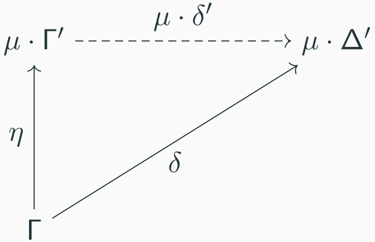


Then we can pick always pick η as our delayed substitution!

Avoiding delayed substitutions

The remaining approaches (left-division, Fitch-style, MTT) ensure a *universal delayed substitution*.

Suppose for every substitution $\Gamma \to \mu \cdot \Delta$, we have the following factorization:



This makes $\mu \cdot -$ into a right adjoint!

Then we can pick always pick η as our delayed substitution!

Left-division

We can require a division operation [Pfe01; Abe08; NVD17; ND18] on modalities in order to make this adjoint exist:

$$\mu \leq \nu \circ \xi \iff \mu/\nu \leq \xi$$

The right adjoint to $\nu \cdot -$ is pointwise application of $-/\nu$.

Left-division

We can require a division operation [Pfe01; Abe08; NVD17; ND18] on modalities in order to make this adjoint exist:

$$\mu \leq \nu \circ \xi \iff \mu/\nu \leq \xi$$

The right adjoint to $\nu \cdot -$ is pointwise application of $-/\nu$.

The new introduction rule uses this universal solution:

$$\frac{\Gamma/\mu \vdash M : A}{\Gamma \vdash \mathsf{mod}_{\mu}(M) : \langle \mu \mid A \rangle}$$

Left-division: pros and cons

When left-division exists, this is completely solid and implementable! [Nuy19].

The issue is that it often doesn't exist.
The only example I know is parametricity/degrees of relatedness.

Left-division: pros and cons

When left-division exists, this is completely solid and implementable! [Nuy19].

The issue is that it often doesn't exist.
The only example I know is parametricity/degrees of relatedness.

The Fitch style

- What the left adjoint doesn't need to respect context extension or the terminal?
- We can formally *add* application of the left adjoint to our grammar for contexts.

$$\textit{(Fitch-style contexts)} \quad \Gamma, \Delta \quad ::= \quad \cdot \mid \Gamma, x : A \mid \Gamma, \text{🔒}_{\mu}$$

The Fitch style

- What the left adjoint doesn't need to respect context extension or the terminal?
- We can formally *add* application of the left adjoint to our grammar for contexts.

$$\textit{(Fitch-style contexts)} \quad \Gamma, \Delta \quad ::= \quad \cdot \mid \Gamma, x : A \mid \Gamma, \text{🔒}_\mu$$

Now we can use this left adjoint in the introduction rule:

$$\frac{\Gamma, \text{🔒}_\mu \vdash M : A}{\Gamma \vdash \mathsf{mod}_\mu(M) : \langle \mu \mid A \rangle}$$

The variable rules

There's no way to extract $x : A$ from under a lock, so we adapt the variable rule:

$$\frac{🔒 \notin \Gamma_0}{\Gamma_1, x : A, \Gamma_0 \vdash x : A}$$

After all, $\Gamma, 🔒_\mu$ represents the application of some functor to Γ.

But now we need some way to *remove locks*, otherwise those variables are gone forever.

The rub: the elimination rule for modalities

A natural candidate for the elimination rule is to transpose in the other direction:

$$\frac{\Gamma \vdash M : \langle \mu \mid A \rangle}{\Gamma, \text{🔒}_{\mu} \vdash \mathsf{open}(M) : A}$$

The rub: the elimination rule for modalities

A natural candidate for the elimination rule is to transpose in the other direction:

$$\frac{\Gamma \vdash M : \langle \mu \mid A \rangle}{\Gamma, 🔒_\mu \vdash \mathsf{open}(M) : A}$$

Once again, however, we have a problem with substitutions:

- Suppose we have a substitution $\Delta \to \Gamma, 🔒_\mu$.
- Can we always factor it through unique substitution $\Delta', 🔒_\mu \to \Gamma, 🔒_\mu$?
- In specific cases, substitution is at least *admissible*.
- With multiple modalities or certain modal operations however, it seems impossible.

An aside: an open-scope eliminator

Fitch-style type theories *do not* have a pattern matching elimination principle.

We can given $\langle \mu \mid A \rangle$ an η-principle, the only time we can do so today!

$$\frac{\Gamma \vdash M : \langle \mu \mid A \rangle}{\Gamma \vdash \mathsf{mod}_{\mu}(\mathsf{open}(M)) = M : \langle \mu \mid A \rangle}$$

I used to think this was very important [GSB19]. No longer so sure [Gra+20].

The Fitch-style: pros and cons

The Fitch style works when substitution can be proven admissible [DP01; BGM17; Clo18; Bir+20; GSB19].

Furthermore, there are multiple implementations of Fitch-style type theories.

On the other hand...

- Restricts the possible modalities (right adjoints), but workable in practice.
- There's a certain asymmetry here: the left adjoints cannot act as types!
- The admissibility of substitution is too weak to work as an internal language.
- The admissibility approach seems completely intractable for multiple modalities.

Our "long march through modal type theory"

After this long tour of modal type theories, where do we stand?

- If we have only one modality, then we can use a split-context.
- If we require no structure on a collection of modalities, we lose normalization
- We can require a division, which works nicely *if* it applies.
- If we ask only that a modality is a right adjoint, multiple modalities are still problematic.

So, the challenge for MTT is to have a system which is (1.) well-behaved (2.) less restrictive than left-division.

Break time

Let's get some coffee...

Our Contribution: MTT

We introduce MTT: a type theory parameterized by a collection of modalities.

- MTT features usual connectives of Martin-Löf Type Theory, including a universe.
- The user can instantiate MTT with different collections of modalities.
- Important results such as canonicity are proven irrespective of the modalities.

We have applied MTT to several different situations:

- Axiomatic cohesion
- Degrees of relatedness
- Guarded recursion and warps
- Various classic modal type theories

Modes and Mode theories

MTT is a multi*mode* type theory, not just multi*modal*.

- Each mode is its own separate type theory, with modalities bridging between them.
- As an example, spatial type theory has two modes: sets and spaces.

Modes and Mode theories

MTT is a multi*mode* type theory, not just multi*modal*.

- Each mode is its own separate type theory, with modalities bridging between them.
- As an example, spatial type theory has two modes: sets and spaces.

We follow [LS16; LSR17] and specify our modalities as a mode theory, a 2-category:

$$\begin{aligned} \text{object} &\sim \text{mode} \\ \text{morphism} &\sim \text{modality} \\ \text{2-cell} &\sim \text{natural map between modalities} \end{aligned}$$

An example: an idempotent comonad

The mode theory for an idempotent comonad is generated from the following data:

objects: $\{m\}$ morphisms: $\{\mu : m \to m\}$ 2-cells: $\{\epsilon : \mu \Rightarrow 1\}$

Furthermore, $\mu \circ \mu = \mu$ and that $\alpha = \beta$ for any pair of 2-cells.

An example: an idempotent comonad

The mode theory for an idempotent comonad is generated from the following data:

objects: $\{m\}$ morphisms: $\{\mu : m \to m\}$ 2-cells: $\{\epsilon : \mu \Rightarrow 1\}$

Furthermore, $\mu \circ \mu = \mu$ and that $\alpha = \beta$ for any pair of 2-cells.

This induces a single modality $\langle \mu \mid - \rangle$ with the following operations:

$$\frac{\Gamma \vdash M : \langle \mu \mid A \rangle \mathbin{@} m}{\Gamma \vdash \mathsf{extract}(M) : A \mathbin{@} m} \qquad \frac{\Gamma \vdash M : \langle \mu \mid A \rangle \mathbin{@} m}{\Gamma \vdash \mathsf{duplicate}(M) : \langle \mu \mid \langle \mu \mid A \rangle \rangle \mathbin{@} m}$$

An example: an idempotent comonad

The mode theory for an idempotent comonad is generated from the following data:

objects: $\{m\}$ morphisms: $\{\mu : m \to m\}$ 2-cells: $\{\epsilon : \mu \Rightarrow 1\}$

Furthermore, $\mu \circ \mu = \mu$ and that $\alpha = \beta$ for any pair of 2-cells.

This induces a single modality $\langle \mu \mid - \rangle$ with the following operations:

A *mode annotation*.

$$\frac{\Gamma \vdash M : \langle \mu \mid A \rangle \mathbin{@} m}{\Gamma \vdash \mathsf{extract}(M) : A \mathbin{@} m} \qquad \frac{\Gamma \vdash M : \langle \mu \mid A \rangle \mathbin{@} m}{\Gamma \vdash \mathsf{duplicate}(M) : \langle \mu \mid \langle \mu \mid A \rangle \rangle \mathbin{@} m}$$

MTT, more formally

We will now introduce MTT a bit more carefully. Let us fix a mode theory $\mathcal{M}$.

MTT is stratified into the following judgments:

$$\Gamma \ \mathsf{ctx} @ m \qquad \Gamma \vdash A \ \mathsf{type} @ m \qquad \Gamma \vdash M : A @ m \qquad \Gamma \vdash \delta : \Delta @ m$$

Each judgment is localized to a mode and each mode contains a copy of MLTT.

MTT, more formally

We will now introduce MTT a bit more carefully. Let us fix a mode theory $\mathcal{M}$.

MTT is stratified into the following judgments: We'll mostly ignore these today.

$$\Gamma \text{ ctx} @ m \qquad \Gamma \vdash A \text{ type} @ m \qquad \Gamma \vdash M : A @ m \qquad \Gamma \vdash \delta : \Delta @ m$$

Each judgment is localized to a mode and each mode contains a copy of MLTT.

Modal types

Slogan: modalities act like functors between modes.

Given a closed type $A \mathbin{@} n$ and $\mu : n \to m$, there is a closed type $\langle \mu \mid A \rangle \mathbin{@} m$.

This doesn't easily scale to open types:

$$\frac{\Gamma \vdash A \text{ type} \mathbin{@} n \qquad \mu : n \to m}{\Gamma \vdash \langle \mu \mid A \rangle \text{ type} \mathbin{@} m}$$

Modal types

Slogan: modalities act like functors between modes.

Given a closed type $A \,@\, n$ and $\mu : n \to m$, there is a closed type $\langle \mu \mid A \rangle \,@\, m$.

This doesn't easily scale to open types:

$$\frac{\Gamma \vdash A \text{ type} \,@\, n \qquad \mu : n \to m}{\Gamma \vdash \langle \mu \mid A \rangle \text{ type} \,@\, m}$$

One of these must live in the wrong mode.

Modal types

Slogan: modalities act like functors between modes.

Given a closed type $A \mathbin{@} n$ and $\mu : n \to m$, there is a closed type $\langle \mu \mid A \rangle \mathbin{@} m$.

This doesn't easily scale to open types:

One of these must live in the wrong mode.

$$\frac{\Gamma \vdash A \text{ type} \mathbin{@} n \qquad \mu : n \to m}{\Gamma \vdash \langle \mu \mid A \rangle \text{ type} \mathbin{@} m}$$

We require additional *judgmental* structure to make sense of modal types.

Fitch-style contexts

MTT uses a Fitch-style context so modalities to have an *adjoint* action on contexts:

$$\frac{\mu : n \to m \qquad \Gamma\ \mathsf{ctx} \mathbin{@} m}{\Gamma, 🔒_\mu\ \mathsf{ctx} \mathbin{@} n}$$

While it is not entirely accurate, it is helpful to imagine $-, 🔒_\mu \dashv \langle \mu \mid - \rangle$.

Fitch-style contexts

MTT uses a Fitch-style context so modalities to have an *adjoint* action on contexts:

$$\frac{\mu : n \to m \qquad \Gamma \ \mathsf{ctx} \mathbin{@} m}{\Gamma, \text{🔒}_\mu \ \mathsf{ctx} \mathbin{@} n}$$

While it is not entirely accurate, it is helpful to imagine $-, \text{🔒}_\mu \dashv \langle \mu \mid - \rangle$.

Accordingly, the introduction and formation rules are transposition:

$$\frac{\mu : n \to m \qquad \Gamma, \text{🔒}_\mu \vdash A \ \mathsf{type} \mathbin{@} n}{\Gamma \vdash \langle \mu \mid A \rangle \ \mathsf{type} \mathbin{@} m} \qquad \frac{\mu : n \to m \qquad \Gamma, \text{🔒}_\mu \vdash M : A \mathbin{@} n}{\Gamma \vdash \mathsf{mod}_\mu(M) : \langle \mu \mid A \rangle \mathbin{@} m}$$

These rules follow other Fitch-style type theories [BGM17; Clo18; Bir+20; GSB19].

Fitch-style contexts with multiple modalities

Prior work had one modality, hence one lock. How do we scale to many modalities?

$$\frac{\Gamma\ \mathsf{ctx} \mathbin{@} m}{\Gamma = \Gamma, \text{🔒}_1\ \mathsf{ctx} \mathbin{@} m} \qquad \frac{\nu : o \to n \qquad \mu : n \to m \qquad \Gamma\ \mathsf{ctx} \mathbin{@} m}{\Gamma, \text{🔒}_\mu, \text{🔒}_\nu = \Gamma, \text{🔒}_{\mu \circ \nu}\ \mathsf{ctx} \mathbin{@} o}$$

Fitch-style contexts with multiple modalities

Prior work had one modality, hence one lock. How do we scale to many modalities?

$$\frac{\Gamma \ \mathsf{ctx} \mathbin{@} m}{\Gamma = \Gamma, \text{🔒}_1 \ \mathsf{ctx} \mathbin{@} m} \qquad \frac{\nu : o \to n \qquad \mu : n \to m \qquad \Gamma \ \mathsf{ctx} \mathbin{@} m}{\Gamma, \text{🔒}_\mu, \text{🔒}_\nu = \Gamma, \text{🔒}_{\mu \circ \nu} \ \mathsf{ctx} \mathbin{@} o}$$

In fact, 🔒 is part of a 2-functor from $\mathcal{M}^{\mathsf{coop}}$ to contexts and substitutions.

Definition
Given a 2-cell $\alpha : \mu \Rightarrow \nu$ and a term $\Gamma, \text{🔒}_\nu \vdash M : A \mathbin{@} m$, there is a derived operation $(-)^\alpha$ such that $\Gamma, \text{🔒}_\mu \vdash M^\alpha : A^\alpha \mathbin{@} m$.

Secretly, this is built from a modal substitution behaving like a natural transformation.

What about variables?

Locks allow us to state the formation rule for modalities, but what about variables?

With the standard variable rule, we again have a mode error!

$$\frac{\mu : n \to m}{x : A, 🔒_\mu \vdash x : A \mathbin{@} n}$$

What about variables?

Locks allow us to state the formation rule for modalities, but what about variables?

With the standard variable rule, we again have a mode error!

$$\frac{\mu : n \to m}{x : A, 🔒_\mu \vdash x : A \mathbin{@} n}$$

A must live in mode *m*

A must live in mode *n*

What about variables?

Locks allow us to state the formation rule for modalities, but what about variables?

With the standard variable rule, we again have a mode error!

$$\frac{\mu : n \to m}{x : A, \text{🔒}_{\mu} \vdash x : A \mathbin{@} n}$$

- Previous Fitch-style type theories handled this through an elimination rule.
- In MTT, we will introduce a final piece of judgmental structure.

Variable annotations

In addition to locks, each variable in the context will be annotated with a modality.

$$\frac{\mu : n \to m \qquad \Gamma \text{ ctx} @ m \qquad \Gamma, \text{🔒}_\mu \vdash A \text{ type} @ n}{\Gamma, x : (\mu \mid A) \text{ ctx} @ m}$$

Another rough intuition: $\Gamma, x : (\mu \mid A) \cong \Gamma, x : \langle \mu \mid A \rangle$.

Variable annotations

In addition to locks, each variable in the context will be annotated with a modality.

$$\frac{\mu : n \to m \qquad \Gamma \text{ ctx} @ m \qquad \Gamma, 🔒_\mu \vdash A \text{ type} @ n}{\Gamma, x : (\mu \mid A) \text{ ctx} @ m}$$

Another rough intuition: $\Gamma, x : (\mu \mid A) \cong \Gamma, x : \langle \mu \mid A \rangle$.

We can use these annotations to give a more sensible variable rule:

$$\frac{\mu : n \to m}{\Gamma, x : (\mu \mid A), 🔒_\mu \vdash x : A @ n}$$

Intuition: this is the counit of the adjunction $-, 🔒_\mu \dashv \langle \mu \mid - \rangle$.

What about 2-cells?

This variable rule is a bit restrictive; it requires the annotation and lock to match.

We can relax a bit and require only a *2-cell* between the annotation and the lock.

$$\frac{\mu, \nu : n \to m \qquad \alpha : \mu \Rightarrow \nu}{\Gamma, x : (\mu \mid A), 🔒_{\nu} \vdash x^{\alpha} : A^{\alpha} \mathbin{@} n}$$

In fact, this rule is a combination of the 'counit' rule and the action of 🔒 on 2-cells.

Modal induction

The final piece of the puzzle is the elimination rule for $\langle \mu \mid A \rangle$.

- At a high-level this rule let's us replace $x : \langle \mu \mid A \rangle$ with $y : (\mu \mid A)$.
- We must generalize to replacing $x : (\nu \mid \langle \mu \mid A \rangle)$ with $y : (\nu \circ \mu \mid A)$.
- We rephrase this as a pattern-matching or cut-style rule.

Modal induction

The final piece of the puzzle is the elimination rule for $\langle \mu \mid A \rangle$.

- At a high-level this rule let's us replace $x : \langle \mu \mid A \rangle$ with $y : (\mu \mid A)$.
- We must generalize to replacing $x : (\nu \mid \langle \mu \mid A \rangle)$ with $y : (\nu \circ \mu \mid A)$.
- We rephrase this as a pattern-matching or cut-style rule.

Fix $\nu : m \to o$, $\mu : n \to m$. The elimination rule for μ is as follows:

$$\frac{\begin{array}{c}\Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 \ @ \ o \\ \Gamma, y : (\nu \circ \mu \mid A) \vdash M_1 : B[\mathsf{mod}_\mu(y)/x] \ @ \ o \\ \Gamma, 🔒_\nu \vdash M_0 : \langle \mu \mid A \rangle \ @ \ m\end{array}}{\Gamma \vdash \mathsf{let}_\nu \ \mathsf{mod}_\mu(y) \leftarrow M_0 \ \mathsf{in} \ M_1 : B[M_0/x] \ @ \ o}$$

Modal induction

The final piece of the puzzle is the elimination rule for $\langle \mu \mid A \rangle$.

- At a high-level this rule let's us replace $x : \langle \mu \mid A \rangle$ with $y : (\mu \mid A)$.
- We must generalize to replacing $x : (\nu \mid \langle \mu \mid A \rangle)$ with $y : (\nu \circ \mu \mid A)$.
- We rephrase this as a pattern-matching or cut-style rule.

Fix $\nu : m \to o$, $\mu : n \to m$. The elimination rule for μ is as follows:

The motive → $\Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 \ @ \ o$

$$\frac{\Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 \ @ \ o \qquad \Gamma, y : (\nu \circ \mu \mid A) \vdash M_1 : B[\mathsf{mod}_\mu(y)/x] \ @ \ o \qquad \Gamma, 🔒_\nu \vdash M_0 : \langle \mu \mid A \rangle \ @ \ m}{\Gamma \vdash \mathsf{let}_\nu \ \mathsf{mod}_\mu(y) \leftarrow M_0 \ \mathsf{in} \ M_1 : B[M_0/x] \ @ \ o}$$

Modal induction

The final piece of the puzzle is the elimination rule for $\langle \mu \mid A \rangle$.

- At a high-level this rule let's us replace $x : \langle \mu \mid A \rangle$ with $y : (\mu \mid A)$.
- We must generalize to replacing $x : (\nu \mid \langle \mu \mid A \rangle)$ with $y : (\nu \circ \mu \mid A)$.
- We rephrase this as a pattern-matching or cut-style rule.

Fix $\nu : m \to o$, $\mu : n \to m$. The elimination rule for μ is as follows:

The base case

$$\frac{\Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 \ @\ o \qquad \Gamma, y : (\nu \circ \mu \mid A) \vdash M_1 : B[\mathsf{mod}_\mu(y)/x] \ @\ o \qquad \Gamma, \text{🔒}_\nu \vdash M_0 : \langle \mu \mid A \rangle \ @\ m}{\Gamma \vdash \mathsf{let}_\nu\ \mathsf{mod}_\mu(y) \leftarrow M_0 \ \mathsf{in}\ M_1 : B[M_0/x] \ @\ o}$$

Modal induction

The final piece of the puzzle is the elimination rule for $\langle \mu \mid A \rangle$.

- At a high-level this rule let's us replace $x : \langle \mu \mid A \rangle$ with $y : (\mu \mid A)$.
- We must generalize to replacing $x : (\nu \mid \langle \mu \mid A \rangle)$ with $y : (\nu \circ \mu \mid A)$.
- We rephrase this as a pattern-matching or cut-style rule.

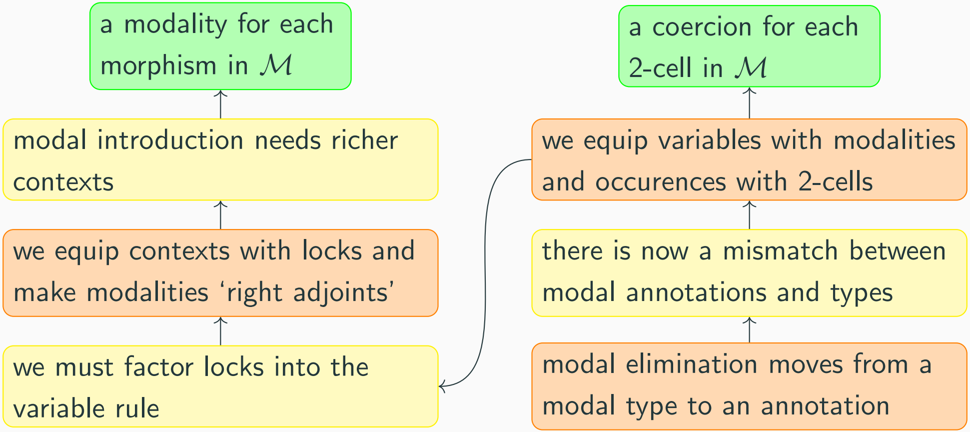
Fix $\nu : m \to o$, $\mu : n \to m$. The elimination rule for μ is as follows:

$$\frac{\Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 \ @\ o \qquad \Gamma, y : (\nu \circ \mu \mid A) \vdash M_1 : B[\mathsf{mod}_{\mu}(y)/x] \ @\ o \qquad \Gamma, \text{🔒}_{\nu} \vdash M_0 : \langle \mu \mid A \rangle \ @\ m}{\Gamma \vdash \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow M_0 \ \mathsf{in}\ M_1 : B[M_0/x] \ @\ o}$$

The term we pattern match on.

Taking stock of MTT

It's easy to feel this is just "one damn rule after another", but at a high-level:



Summary of crucial modal rules

$$\frac{\mu : n \to m \qquad \Gamma \ \mathsf{ctx} @ m}{\Gamma, 🔒_\mu \ \mathsf{ctx} @ n}
\qquad
\frac{\mu : n \to m \qquad \Gamma \ \mathsf{ctx} @ m \qquad \Gamma, 🔒_\mu \vdash A \ \mathsf{type}_1 @ n}{\Gamma, x : (\mu \mid A) \ \mathsf{ctx} @ m}$$

$$\frac{\mu : n \to m \qquad \Gamma, 🔒_\mu \vdash A \ \mathsf{type}_\ell @ n}{\Gamma \vdash \langle \mu \mid A \rangle \ \mathsf{type}_\ell @ m}
\qquad
\frac{\nu : m \to n \qquad \alpha : \nu \Rightarrow \mathsf{locks}(\Gamma_1)}{\Gamma_0, x : (\nu \mid A), \Gamma_1 \vdash x^\alpha : A^\alpha @ m}$$

$$\frac{\mu : n \to m \qquad \Gamma, 🔒_\mu \vdash M : A @ n}{\Gamma \vdash \mathsf{mod}_\mu(M) : \langle \mu \mid A \rangle @ m}$$

$$\frac{\begin{array}{c}\mu : n \to m \qquad \nu : m \to o \qquad \Gamma, x : (\nu \mid \langle \mu \mid A \rangle) \vdash B \ \mathsf{type}_1 @ o \\ \Gamma, 🔒_\nu \vdash M_0 : \langle \mu \mid A \rangle @ m \qquad \Gamma, x : (\nu \circ \mu \mid A) \vdash M_1 : B[\mathsf{mod}_\mu(x)/x] @ o\end{array}}{\Gamma \vdash \mathsf{let}_\nu \ \mathsf{mod}_\mu(x) \leftarrow M_0 \ \mathsf{in} \ M_1 : B[M_0/x] @ o}$$

Modal combinators

The mode theory is reflected into MTT as a series of modal combinators:

$$\langle 1 \mid A \rangle \simeq A$$

$$\langle \mu \mid \langle \nu \mid A \rangle \rangle \simeq \langle \mu \circ \nu \mid A \rangle$$

$$\langle \mu \mid A \rangle \to \langle \nu \mid A \rangle \qquad \text{(For each } \alpha : \mu \Rightarrow \nu\text{)}$$

$$\langle \mu \mid A \to B \rangle \to (\langle \mu \mid A \rangle \to \langle \mu \mid B \rangle)$$

All of these follow because 🔒 is a 2-functor out of $\mathcal{M}^{\mathsf{coop}}$:

$$\Gamma, 🔒_1 = \Gamma \ \mathsf{ctx} \mathbin{@} m \qquad \Gamma, 🔒_\mu, 🔒_\nu = \Gamma, 🔒_{\mu \circ \nu} \ \mathsf{ctx} \mathbin{@} m \qquad \Gamma, 🔒_\nu \vdash 🔑^\alpha_\Gamma : \Gamma, 🔒_\mu \mathbin{@} m$$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq ?$

Holes

$x : (1 \mid \langle \mu \mid A \rangle) \vdash ? : \langle \nu \mid A \rangle$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ ?$

Holes

$x : (1 \mid \langle \mu \mid A \rangle), y : (\mu \mid A) \vdash ? : \langle \nu \mid A \rangle$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\nu}(?)$

Holes

$x : (1 \mid \langle \mu \mid A \rangle), y : (\mu \mid A), 🔒_{\nu} \vdash\ ?\ : A$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\nu}(y^{\alpha})$

Holes

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$$

$$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\nu}(y^{\alpha})$$

$$\mathsf{comp}_{\mu,\nu}(-) : \langle \mu \circ \nu \mid A \rangle \to \langle \mu \mid \langle \nu \mid A \rangle \rangle$$

$$\mathsf{comp}(x) \triangleq ?$$

Holes

$$x : (1 \mid \langle \mu \circ \nu \mid A \rangle) \vdash ? : \langle \mu \mid \langle \nu \mid A \rangle \rangle$$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x \text{ in } \mathsf{mod}_{\nu}(y^{\alpha})$

$\mathsf{comp}_{\mu,\nu}(-) : \langle \mu \circ \nu \mid A \rangle \to \langle \mu \mid \langle \nu \mid A \rangle \rangle$

$\mathsf{comp}(x) \triangleq \mathsf{let}\ \mathsf{mod}_{\mu \circ \nu}(y) \leftarrow x \text{ in } ?$

Holes

$x : (\cdots), y : (\mu \circ \nu \mid A) \vdash ? : \langle \mu \mid \langle \nu \mid A \rangle \rangle$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\nu}(y^{\alpha})$

$\mathsf{comp}_{\mu,\nu}(-) : \langle \mu \circ \nu \mid A \rangle \to \langle \mu \mid \langle \nu \mid A \rangle \rangle$

$\mathsf{comp}(x) \triangleq \mathsf{let}\ \mathsf{mod}_{\mu \circ \nu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\mu}(?)$

Holes

$x : (\cdots), y : (\mu \circ \nu \mid A), \text{🔒}_{\mu} \vdash ? : \langle \nu \mid A \rangle$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$

$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x \mathsf{\ in\ } \mathsf{mod}_{\nu}(y^{\alpha})$

$\mathsf{comp}_{\mu,\nu}(-) : \langle \mu \circ \nu \mid A \rangle \to \langle \mu \mid \langle \nu \mid A \rangle \rangle$

$\mathsf{comp}(x) \triangleq \mathsf{let\ mod}_{\mu \circ \nu}(y) \leftarrow x \mathsf{\ in\ } \mathsf{mod}_{\mu}(\mathsf{mod}_{\nu}(?))$

Holes

$x : (\cdots), y : (\mu \circ \nu \mid A), 🔒_{\mu \circ \nu} \vdash ? : A$

Example definitions of modal combinators

To get a feel for MTT, let us define some of these combinators.

Programs

Holes

$$\mathsf{coe}[\alpha : \mu \Rightarrow \nu](-) : \langle \mu \mid A \rangle \to \langle \nu \mid A^{\alpha} \rangle$$

$$\mathsf{coe}[\alpha](x) \triangleq \mathsf{let}_{\nu}\ \mathsf{mod}_{\mu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\nu}(y^{\alpha})$$

$$\mathsf{comp}_{\mu,\nu}(-) : \langle \mu \circ \nu \mid A \rangle \to \langle \mu \mid \langle \nu \mid A \rangle \rangle$$

$$\mathsf{comp}(x) \triangleq \mathsf{let}\ \mathsf{mod}_{\mu \circ \nu}(y) \leftarrow x\ \mathsf{in}\ \mathsf{mod}_{\mu}(\mathsf{mod}_{\nu}(y))$$

Results about MTT

A major strength of MTT is that we can prove theorems *irrespective* of $\mathcal{M}$.

Theorem (Consistency)
There is no term $\cdot \vdash M : \mathsf{Id}_{\mathbb{B}}(\mathsf{tt}, \mathsf{ff}) \mathbin{@} m$.

Theorem (Canonicity)
Subject to a technical restriction, if $\cdot \vdash M : A \mathbin{@} m$ *is a closed term, then the following conditions hold:*

- *If* $A = \mathbb{B}$, *then* $\cdot \vdash M = \mathsf{tt} : \mathbb{B} \mathbin{@} m$ *or* $\cdot \vdash M = \mathsf{ff} : \mathbb{B} \mathbin{@} m$.
- *If* $A = \mathsf{Id}_{A_0}(N_0, N_1)$ *then* $\cdot \vdash M = \mathsf{refl}(N_0) : \mathsf{Id}_{A_0}(N_0, N_1) \mathbin{@} m$.
- *If* $A = \langle \mu \mid A_0 \rangle$ *then* $\cdot \vdash M = \mathsf{mod}_\mu(N) : \langle \mu \mid A_0 \rangle \mathbin{@} m$ *for some* N.

As time permits we'll return to canonicity, but for now just take it on faith.

Example: guarded recursion

The other major strength of MTT is that we can use it to model interesting examples!

- We'll be interested in using MTT to model *guarded recursion*.
- Guarded recursion is naturally multimode.
- This situation crucially requires the *interaction* of modalities.

Guarded recursion: a brief introduction

Guarded recursion uses two modalities to isolate productive and coinductive programs.

Guarded recursion: a brief introduction

Guarded recursion uses two modalities to isolate productive and coinductive programs.

1. The later modality $\blacktriangleright$ tags computation which are available at the next step:

$$\text{next} : A \to \blacktriangleright A \qquad\qquad \text{löb} : (\blacktriangleright A \to A) \to A$$

2. The always modality $\Box$ tags computation which do not depend on the time step:

$$\text{extract} : \Box A \to A \qquad\qquad \text{dup} : \Box A \simeq \Box\Box A$$

These two modalities interact in a crucial way to give *coinductive* programs:

$$\text{now} : \Box \blacktriangleright A \to \Box A$$

Splitting guarded recursion into 2 modes

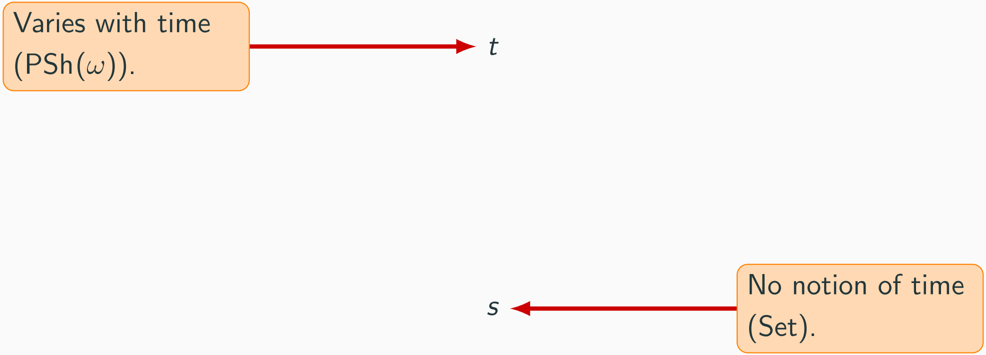
Previous guarded type theories had a single mode, we opt for 2 *modes*.

t

s

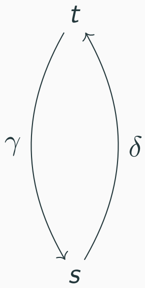
Splitting guarded recursion into 2 modes

Previous guarded type theories had a single mode, we opt for 2 *modes*.



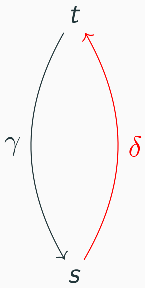
Splitting guarded recursion into 2 modes

Previous guarded type theories had a single mode, we opt for 2 *modes*.



Splitting guarded recursion into 2 modes

Previous guarded type theories had a single mode, we opt for 2 *modes*.

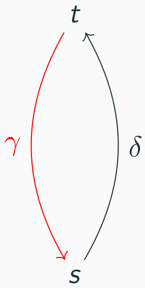


Allow a constant type to trivially vary over time.

Splitting guarded recursion into 2 modes

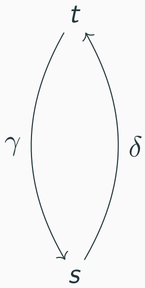
Previous guarded type theories had a single mode, we opt for 2 *modes*.

Restrict a varying type to global elements.



Splitting guarded recursion into 2 modes

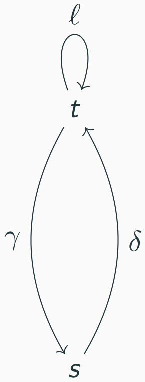
Previous guarded type theories had a single mode, we opt for 2 *modes*.



$$\Box A \triangleq \langle \delta \circ \gamma \mid A \rangle$$

Splitting guarded recursion into 2 modes

Previous guarded type theories had a single mode, we opt for 2 *modes*.



$$\Box A \triangleq \langle \delta \circ \gamma \mid A \rangle$$

$$\blacktriangleright A \triangleq \langle \ell \mid A \rangle$$

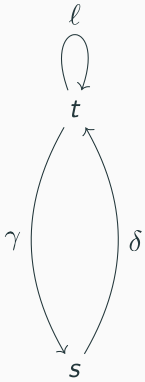
$$\Gamma A \triangleq \langle \gamma \mid A \rangle$$

$$\Delta A \triangleq \langle \delta \mid A \rangle$$

Splitting guarded recursion into 2 modes

Previous guarded type theories had a single mode, we opt for 2 *modes*.

$$\delta \circ \gamma \leq 1 \quad 1 = \gamma \circ \delta$$
$$1 \leq \ell \quad \gamma = \gamma \circ \ell$$



$$\Box A \triangleq \langle \delta \circ \gamma \mid A \rangle$$
$$\blacktriangleright A \triangleq \langle \ell \mid A \rangle$$
$$\Gamma A \triangleq \langle \gamma \mid A \rangle$$
$$\Delta A \triangleq \langle \delta \mid A \rangle$$

Modal operations

The terms for the following operations are all induced by generic combinators:

$$\text{next} : A \to \blacktriangleright A \qquad - \circledast - : \blacktriangleright(A \to B) \to \blacktriangleright A \to \blacktriangleright B \qquad \text{extract} : \Box A \to A$$

$$\text{dup} : \Box A \simeq \Box\Box A \qquad \text{now} : \Box \blacktriangleright A \to \Box A$$

In particular, next and extract are instances of coercions, while dup and now follow from associativity.

What about Löb?

- We can use the standard operations on MTT to derive all operations except Löb.
- Löb is actually a bit of problem; it's a modality-specific operation!

What about Löb?

- We can use the standard operations on MTT to derive all operations except Löb.
- Löb is actually a bit of problem; it's a modality-specific operation!

Instead, we have to simply axiomitize Löb:

$$\frac{\Gamma, x : (\ell \mid A^{1 \leq \ell}) \vdash M : A \mathbin{@} t}{\Gamma \vdash \text{löb}(x.\ M) : A \mathbin{@} t}$$

$$\frac{\Gamma, x : (\ell \mid A^{1 \leq \ell}) \vdash M : A \mathbin{@} t}{\Gamma \vdash \text{löb}(x.\ M) = \text{let mod}_\ell(x) \leftarrow \text{next}(\text{löb}(x.\ M)) \text{ in } M : A \mathbin{@} t}$$

(NB: $A^{1 \leq \ell}$ moves A from the Γ to $\Gamma, \text{🔒}_\ell$)

What about Löb?

- We can use the standard operations on MTT to derive all operations except Löb.
- Löb is actually a bit of problem; it's a modality-specific operation!

Instead, we have to simply axiomitize Löb:

$$\frac{\Gamma, x : (\ell \mid A^{1 \leq \ell}) \vdash M : A \mathbin{@} t}{\Gamma \vdash \text{löb}(x.\ M) : A \mathbin{@} t}$$

$$\frac{\Gamma, x : (\ell \mid A^{1 \leq \ell}) \vdash M : A \mathbin{@} t}{\Gamma \vdash \text{löb}(x.\ M) = \text{let } \text{mod}_\ell(x) \leftarrow \text{next}(\text{löb}(x.\ M)) \text{ in } M : A \mathbin{@} t}$$

(NB: $A^{1 \leq \ell}$ moves A from the Γ to $\Gamma, \text{🔒}_\ell$)

(NB: We follow Bizjak et al. [Biz+16] and just work in an extensional type theory.)

An aside: modality-specific operations

It's not clear yet what the proper formulation of Löb would be.

It belongs to a large class of operations which entangle a modality and another connective (in this case, $\rightarrow$ and $\blacktriangleright$) which we term *modality-specific*.

Question
Is there a reasonable class of modality-specific operations that can be handled uniformly?

Programming in Guarded MTT

Now that we have these combinators established, we can use them to write guarded programs.

$$\mathsf{Str}'_A \triangleq \mathsf{l\ddot{o}b}(S.\ \Delta(A) \times \blacktriangleright S)$$

$$\mathsf{Str} : \mathsf{U} \to \mathsf{U} \mathbin{@} s$$
$$\mathsf{Str}(A) \triangleq \Gamma(\mathsf{Str}'_A)$$

Programming in Guarded MTT

Now that we have these combinators established, we can use them to write guarded programs.

$$\mathsf{Str}'_A \triangleq \mathsf{löb}(S.\ \Delta(A) \times \blacktriangleright S)$$

$$\mathsf{Str} : \mathsf{U} \to \mathsf{U} \mathbin{@} s$$
$$\mathsf{Str}(A) \triangleq \Gamma(\mathsf{Str}'_A)$$

We only have one clock so coinductive streams only work on constant data.

Programming in Guarded MTT: zipWith

Now that streams are defined, we can write down an operator on them:

$$\begin{aligned}&\mathsf{go} : \Delta(A \to B \to C) \to \mathsf{Str}'_A \to \mathsf{Str}'_B \to \mathsf{Str}'_C \\ &\mathsf{go}(f) \triangleq \mathsf{l\ddot{o}b}(r.\ \lambda x, y.\ (f \circledast_\delta x_h \circledast_\delta y_h, \mathsf{mod}_\ell(r) \circledast_\ell x_t \circledast_\ell y_t))\end{aligned}$$

Now that streams are defined, we can write down an operator on them:

$$\begin{aligned}
&\mathsf{go} : \Delta(A \to B \to C) \to \mathsf{Str}'_A \to \mathsf{Str}'_B \to \mathsf{Str}'_C \\
&\mathsf{go}(f) \triangleq \mathsf{l\ddot{o}b}(r.\ \lambda x, y.\ (f \circledast_\delta x_h \circledast_\delta y_h, \mathsf{mod}_\ell(r) \circledast_\ell x_t \circledast_\ell y_t))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{zipWith} : (A \to B \to C) \to \mathsf{Str}(A) \to \mathsf{Str}(B) \to \mathsf{Str}(C) \\
&\mathsf{zipWith}(f) \quad \triangleq \quad \lambda x, y.\ \mathsf{mod}_\gamma(\mathsf{go}(\mathsf{mod}_\delta(f))) \circledast_\gamma x \circledast_\gamma y
\end{aligned}$$

Programming in Guarded MTT: zipWith

Now that streams are defined, we can write down an operator on them:

$$\mathsf{go} : \Delta(A \to B \to C) \to \ \mathsf{Str}'_A \to \mathsf{Str}'_B \to \mathsf{Str}'_C$$
$$\mathsf{go}(f) \triangleq \mathsf{l\ddot{o}b}(r.\ \lambda x, y.\ (f \circledast_\delta x_h \circledast_\delta y_h, \mathsf{mod}_\ell(r) \circledast_\ell x_t \circledast_\ell y_t))$$

$$\mathsf{zipWith} : (A \to B \to C) \to \ \mathsf{Str}(A) \to \mathsf{Str}(B) \to \mathsf{Str}(C)$$
$$\mathsf{zipWith}(f) \ \triangleq\ \lambda x, y.\ \mathsf{mod}_\gamma(\mathsf{go}(\mathsf{mod}_\delta(f))) \circledast_\gamma x \circledast_\gamma y$$

We can use the ambient dependent type theory to show that zipWith preserves e.g. commutativity.

Guarded recursion conclusions

Experimentally, this calculus for guarded recursion is reasonably pleasant for pen-and-paper calculations!

There are a few missing things:

1. Using extensional type theory makes a standard implementation impossible.
2. Using only $\Box$ and $\blacktriangleright$ makes a few things simple, but lacks the expressivity of clocks.

Going forward, we'd like to address these limitations, especially the first!

Conclusions

We introduce MTT: a type theory parameterized by a collection of modalities.

- MTT features usual connectives of Martin-Löf Type Theory, including a universe.
- The user can instantiate MTT with different collections of modalities.
- Important results such as canonicity are proven irrespective of the modalities.

We have applied MTT to several different situations:

- Axiomatic cohesion
- Degrees of relatedness
- Guarded recursion and warps
- Various classic modal type theories

`https://jozefg.github.io/papers/multimodal-dependent-type-theory.pdf`

`https://jozefg.github.io/papers/type-theory-a-la-mode.pdf`

Bonus slides!

Bonus slides is code for “very technical slides I liked too much to delete entirely”.

The modal substitution calculus

Before we talk about the canonicity proof, we need to quickly show the explicit substitution calculus.

$$\frac{\mu : n \to m \qquad \Gamma \vdash \delta : \Delta \mathbin{@} m}{\Gamma, \text{🔒}_\mu \vdash \delta.\text{🔒}_\mu : \Delta, \text{🔒}_\mu \mathbin{@} n} \qquad \frac{\alpha : \mu \Rightarrow \nu}{\Gamma, \text{🔒}_\nu \vdash \text{🔑}^\alpha_\Gamma : \Gamma, \text{🔒}_\mu \mathbin{@} n} \qquad \frac{\Gamma \vdash \delta : \Delta \mathbin{@} m}{\Gamma \vdash \delta.\text{🔒}_1 = \delta : \Delta \mathbin{@} m}$$

$$\frac{\mu : n \to m \qquad \nu : o \to n \qquad \Gamma \vdash \delta : \Delta \mathbin{@} m}{\Gamma, \text{🔒}_{\mu \circ \nu} \vdash \delta.\text{🔒}_{\mu \circ \nu} = \delta.\text{🔒}_\mu.\text{🔒}_\nu : \Delta, \text{🔒}_{\mu \circ \nu} \mathbin{@} m}$$

$$\frac{\mu, \nu : n \to m \qquad \alpha : \nu \Rightarrow \mu \qquad \Gamma \vdash \delta : \Delta \mathbin{@} m}{\Gamma, \text{🔒}_\mu \vdash \text{🔑}^\alpha_\Delta \circ (\delta.\text{🔒}_\mu) = (\delta.\text{🔒}_\nu) \circ \text{🔑}^\alpha_\Gamma : \Delta, \text{🔒}_\nu \mathbin{@} n}$$

Proving canonicity via gluing proof

How does gluing work?

1. Define a category of models, syntax is (by definition) the initial model.
2. Define $\mathcal{G}$ which equips elements of some model $\mathcal{M}$ with a proof of e.g. canonicity.
3. Define a projection from the $\mathcal{M}$ to the $\mathcal{G}$ which forgets the proof.
4. Use the initiality of syntax to obtain a section to projection.
5. Conclude that every element of the initial model enjoys e.g. canonicity.

Best thought of as a categorification of logical relations, where we also allow proof relevance.

Advantage of the gluing approach

- We can already attempt to prove e.g. canonicity via standard syntactic logical relations.
- The goal is to make these proofs simpler by excavating the categorical structure.
- Another big advantage is the switch to proof-relevance: necessary to handle universes well!

Where do I learn more?

Gluing is not a new idea, but two recent preprints cover some of my own perspectives on it:

1. Sterling and Spitters [SS18], an arxiv preprint about the simply-typed case.
2. Sterling, Angiuli, and Gratzer [SAG20], another preprint covering the dependently-typed case.

The key line of development at the moment is how to make gluing more *mathematical*. Frustratingly, the ideas proposed in the latter are complex to scale to modalities.

The models of MTT

In order to apply gluing, we construct a category of models for MTT [Car78].

- A model of MTT is built around a 2-functor $\mathcal{M} : \mathcal{M}^{\mathsf{coop}} \to \mathsf{Cat}$ which sends each mode to a category of contexts.
- We require a CWF for each $\mathcal{M}[m]$ (including $\sum$,$\prod$,Id,etc.).
- Each 1-cell in $\mathcal{M}$ must induce a modality relating the two different modes.

NB: If we instead worked with dependent right adjoints, this becomes even simpler because a modality has such a nice semantic characterization!

The technical restriction

The difficulty in the gluing proof for MTT is handling modalities in the glued model.

1. I know how to do this in a very clean way for simply-typed languages or when the modalities are dependent right adjoints, but it's hard in MTT.
2. In order to simplify, we insist $\cdot, \text{🔒}_{\mu} = \cdot$ (the left adjoint preserves terminals).
3. We are presently working to remove this restriction.

Why does this restriction help? It allows us to work exclusively with closed terms. Otherwise we need to also work with terms in the context $\cdot, \text{🔒}_{\mu}$.

Further details for our proof are given in the accompanying technical report.

References

Andreas Abel. "Polarised subtyping for sized types". In: *Mathematical Structures in Computer Science* 18.5 (2008), pp. 797–822. DOI: 10.1017/S0960129508006853. URL: https://doi.org/10.1017/S0960129508006853 (cit. on pp. 32, 33).

G. M. Bierman and V. C. V. de Paiva. "On an Intuitionistic Modal Logic". In: *Studia Logica* 65.3 (2000). DOI: 10.1023/A:1005291931660 (cit. on p. 15).

Patrick Bahr, Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. "The clocks are ticking: No more delays!" In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2017. DOI: 10.1109/LICS.2017.8005097. URL: http://www.itu.dk/people/mogel/papers/lics2017.pdf (cit. on pp. 2, 5, 42, 56, 57).

Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. "Modal dependent type theory and dependent right adjoints". In: *Mathematical Structures in Computer Science* 30.2 (2020), pp. 118–138. DOI: 10.1017/S0960129519000197. eprint: 1804.05236 (cit. on pp. 4, 5, 7, 42, 56, 57).

Lars Birkedal. “Developing Theories of Types and Computability via Realizability”. In: *Electronic Notes in Theoretical Computer Science* 34 (2000) (cit. on p. 2).

Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus E. Møgelberg, and Lars Birkedal. “Guarded Dependent Type Theory with Coinductive Types”. In: *Foundations of Software Science and Computation Structures*. Ed. by Bart Jacobs and Christof Löding. Springer Berlin Heidelberg, 2016, pp. 20–35 (cit. on pp. 29, 96–98).

John Cartmell. “Generalised Algebraic Theories and Contextual Categories”. PhD thesis. University of Oxford, 1978 (cit. on p. 112).

Ranald Clouston, Aleš Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. "Programming and Reasoning with Guarded Recursion for Coinductive Types". In: *Foundations of Software Science and Computation Structures*. Ed. by Andrew Pitts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 407–421. ISBN: 978-3-662-46678-0 (cit. on p. 2).

Ranald Clouston. "Fitch-Style Modal Lambda Calculi". In: *Foundations of Software Science and Computation Structures*. Ed. by Christel Baier and Ugo Dal Lago. Springer International Publishing, 2018, pp. 258–275 (cit. on pp. 5, 42, 56, 57).

Rowan Davies and Frank Pfenning. "A Modal Analysis of Staged Computation". In: *Journal of the ACM* 48.3 (May 2001), pp. 555–604 (cit. on p. 42).

Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. "Multimodal Dependent Type Theory". To Appear at LICS 2020. Availabe at https://jozefg.github.io/papers/multimodal-dependent-type-theory.pdf. 2020 (cit. on p. 41).

Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. "Implementing a Modal Dependent Type Theory". In: *Proc. ACM Program. Lang.* 3 (ICFP 2019). DOI: 10.1145/3341711 (cit. on pp. 5, 7, 41, 42, 56, 57).

Adrien Guatto. "A Generalized Modality for Recursion". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. ACM, 2018. DOI: 10.1145/3209108.3209148 (cit. on p. 2).

G. A. Kavvos. "Dual-context calculi for modal logic". In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005089. arXiv: 1602.04860 (cit. on p. 18).

Daniel R. Licata and Michael Shulman. "Adjoint Logic with a 2-Category of Modes". In: *Logical Foundations of Computer Science*. Ed. by Sergei Artemov and Anil Nerode. Springer International Publishing, 2016, pp. 219–235. DOI: 10.1007/978-3-319-27683-0_16 (cit. on pp. 46, 47).

Daniel R. Licata, Michael Shulman, and Mitchell Riley. "A Fibrational Framework for Substructural and Modal Logics". In: *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*. Ed. by Dale Miller. Vol. 84. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 25:1–25:22. DOI: 10.4230/LIPIcs.FSCD.2017.25 (cit. on pp. 29, 46, 47).

Andreas Nuyts and Dominique Devriese. "Degrees of Relatedness: A Unified Framework for Parametricity, Irrelevance, Ad Hoc Polymorphism, Intersections, Unions and Algebra in Dependent Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. ACM, 2018. DOI: 10.1145/3209108.3209119 (cit. on pp. 2, 32, 33).

Andreas Nuyts. *Menkar*. https://github.com/anuyts/menkar. 2019 (cit. on pp. 34, 35).

Andreas Nuyts, Andrea Vezzosi, and Dominique Devriese. "Parametric Quantifiers for Dependent Type Theory". In: *Proc. ACM Program. Lang.* 1.ICFP (2017). DOI: 10.1145/3110276 (cit. on pp. 32, 33).

Frank Pfenning and Rowan Davies. "A Judgmental Reconstruction of Modal Logic". In: *Mathematical Structures in Computer Science* 11.4 (2001), pp. 511–540. DOI: 10.1017/S0960129501003322. URL: http://www.cs.cmu.edu/~fp/papers/mscs00.pdf (cit. on pp. 2, 15).

F. Pfenning. “Intensionality, extensionality, and proof irrelevance in modal type theory”. In: *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 2001, pp. 221–230. DOI: 10.1109/LICS.2001.932499. URL: https://www.cs.cmu.edu/~fp/papers/lics01.pdf (cit. on pp. 2, 32, 33).

Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. *A Cubical Language for Bishop Sets*. 2020. arXiv: 2003.01491 [cs.LO] (cit. on p. 111).

Michael Shulman. “Brouwer’s fixed-point theorem in real-cohesive homotopy type theory”. In: *Mathematical Structures in Computer Science* 28.6 (2018), pp. 856–941. DOI: 10.1017/S0960129517000147. URL: https://doi.org/10.1017/S0960129517000147 (cit. on pp. 2, 15).

Jonathan Sterling and Bas Spitters. *Normalization by gluing for free λ-theories*. 2018. arXiv: 1809.08646 [cs.LO] (cit. on p. 111).

Felix Wellen. *Cartan Geometry in Modal Homotopy Type Theory*. 2018. arXiv: 1806.05966 [math.DG] (cit. on p. 2).