Implementing a Modal Dependent Type Theory

Daniel Gratzer[0] Jonathan Sterling[1] Lars Birkedal[0]

August 21, 2019

ICFP '19

[0]Aarhus University

[1]Carnegie Mellon University

Modalities

We want to add a single modality MLTT, $\Box$.

$$\Gamma \vdash M : \Box A$$

$$\leftrightsquigarrow$$

$$M \text{ ":" } A \text{ and } M \text{ only mentions variables of the shape } \Box B$$

- In staged programming, $\Box A$ represents precomputed values.
- In modal FRP, $\Box A$ represents stable types.
- In distributed programming, $\Box A$ represents globally available values.

Our Contribution: $\mathsf{MLTT}_{🔒}$

We contribute $\mathsf{MLTT}_{🔒}$, a dependent type theory with...

$$\left.\begin{array}{l} \bullet\ \text{the box modality, } \Box A \\ \bullet\ \text{dependent sums, } \Sigma(A, B) \\ \bullet\ \text{dependent products, } \Pi(A, B) \end{array}\right\} \text{With both } \beta \text{ and } \eta$$

- natural numbers, nat
- intensional identity types, $\mathsf{Id}(A, M, N)$
- a cumulative hierarchy of universes, $\mathsf{U}_0, \mathsf{U}_1 \ldots$

We have constructed a precise syntactic account of $\mathsf{MLTT}_{🔒}$, and proved the decidability of type-checking for it.

Typical Problems with Modalities

We could imagine just dropping all local variables when constructing $\Box A$:

$$\frac{\Box\Gamma \vdash M : A}{\Box\Gamma, \Delta \vdash \mathsf{box}(M) : \Box A}\ \textsc{tm/lock?!}$$

In this case $\mathsf{box}(M)$ cannot commute with substitution:

$$\mathsf{box}(M)[N/x] \text{ could be well-typed while } \mathsf{box}(M[N/x]) \text{ is ill-typed!}$$

We can try versions of this rule,[1] but we'll opt for another approach.

[1] Prawitz 1967

Adding Judgmental Structure

We'll incorporate Fitch-style judgmental structure[2] to handle $\Box A$:

$$(\textit{Contexts}) \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x : A \mid \Gamma.🔒$$

Instead of dropping part of the context we can lock it away:

$$\frac{\Gamma.🔒 \vdash M : A}{\Gamma \vdash [M]_{🔒} : \Box A}\ \textsc{tm/lock} \qquad \frac{\Gamma = \Gamma_0, x : A, \Gamma_1 \qquad 🔒 \notin \Gamma_1}{\Gamma \vdash x : A}\ \textsc{tm/var}$$

[2] Clouston 2018

We'll incorporate Fitch-style judgmental structure[2] to handle $\Box A$:

$$\textit{(Contexts)} \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x : A \mid \Gamma.\text{🔒}$$

Instead of dropping part of the context we can lock it away:

$$\textsc{tm/lock}\ \frac{\Gamma.\text{🔒} \vdash M : A}{\Gamma \vdash [M]_{\text{🔒}} : \Box A} \qquad \textsc{tm/var}\ \frac{\Gamma = \Gamma_0, x : A, \Gamma_1 \qquad \text{🔒} \notin \Gamma_1}{\Gamma \vdash x : A}$$

Crucially, later on we are able to *unlock* the context:

$$\textsc{tm/unlock}\ \frac{\Gamma^{\text{🔓}} \vdash M : \Box A}{\Gamma \vdash [M]_{\text{🔓}} : A}$$

[2] Clouston 2018

Adding Judgmental Structure

We'll incorporate Fitch-style judgmental structure[2] to handle $\Box A$:

$$(\textit{Contexts}) \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x : A \mid \Gamma.\text{🔒}$$

Instead of dropping part of the context we can lock it away:

$$\frac{\Gamma.\text{🔒} \vdash M : A}{\Gamma \vdash [M]_{\text{🔒}} : \Box A}\ \textsc{tm/lock} \qquad \frac{\Gamma = \Gamma_0, x : A, \Gamma_1 \qquad \text{🔒} \notin \Gamma_1}{\Gamma \vdash x : A}\ \textsc{tm/var}$$

Crucially, later on we are able to *unlock* the context:

$$\frac{\Gamma^{\text{🔓}} \vdash M : \Box A}{\Gamma \vdash [M]_{\text{🔓}} : A}\ \textsc{tm/unlock}$$

Not obvious, but these rules respect substitution!

[2] Clouston 2018

A Small Programming Break

How does our intuition for $\Box A$ square with $[-]_{\text{unlock}}$ and $[-]_{\text{lock}}$?

Programs

$$\mathrm{extract}_A : \Box A \longrightarrow A$$
$$\mathrm{extract}_A(x) \triangleq [x]_{\text{unlock}}$$

How does our intuition for $\Box A$ square with $[-]_{🔓}$ and $[-]_{🔒}$?

Programs

$$\mathsf{extract}_A : \Box A \longrightarrow A$$

$$\mathsf{extract}_A(x) \triangleq [x]_{🔓}$$

$$\mathsf{dup}_A : \Box A \longrightarrow \Box\Box A$$

$$\mathsf{dup}_A(x) \triangleq ?$$

Holes

$$x : \Box A \vdash ? : \Box\Box A$$

A Small Programming Break

How does our intuition for $\Box A$ square with $[-]_{\faUnlock}$ and $[-]_{\faLock}$?

Programs

$$\mathrm{extract}_A : \Box A \longrightarrow A$$

$$\mathrm{extract}_A(x) \triangleq [x]_{\faUnlock}$$

$$\mathrm{dup}_A : \Box A \longrightarrow \Box\Box A$$

$$\mathrm{dup}_A(x) \triangleq [?]_{\faLock}$$

Holes

$$x : \Box A, \faLock \vdash ? : \Box A$$

A Small Programming Break

How does our intuition for $\Box A$ square with $[-]_{🔓}$ and $[-]_{🔒}$?

Programs	Holes
$\mathsf{extract}_A : \Box A \longrightarrow A$	
$\mathsf{extract}_A(x) \triangleq [x]_{🔓}$	
$\mathsf{dup}_A : \Box A \longrightarrow \Box\Box A$	
$\mathsf{dup}_A(x) \triangleq [[?]_{🔒}]_{🔒}$	$x : \Box A, 🔒, 🔒 \vdash ? : A$

A Small Programming Break

How does our intuition for $\Box A$ square with $[-]_{🔓}$ and $[-]_{🔒}$?

Programs

$$\mathsf{extract}_A : \Box A \longrightarrow A$$

$$\mathsf{extract}_A(x) \triangleq [x]_{🔓}$$

$$\mathsf{dup}_A : \Box A \longrightarrow \Box\Box A$$

$$\mathsf{dup}_A(x) \triangleq [[[?]_{🔓}]_{🔒}]_{🔒}$$

Holes

$$x : \Box A \vdash \, ? : \Box A$$

A Small Programming Break

How does our intuition for $\square A$ square with $[-]_{🔓}$ and $[-]_{🔒}$?

Programs

Holes

$$\mathsf{extract}_A : \square A \longrightarrow A$$

$$\mathsf{extract}_A(x) \triangleq [x]_{🔓}$$

$$\mathsf{dup}_A : \square A \longrightarrow \square\square A$$

$$\mathsf{dup}_A(x) \triangleq [[[x]_{🔓}]_{🔒}]_{🔒}$$

Making Hard Choices: Definitional Equalities for $\mathrm{MLTT}_{🔒}$

We are able to equip $\Box A$ with both a β and η rule in $\mathrm{MLTT}_{🔒}$:

$$\textsc{tm/unlock-lock}\quad \frac{\Gamma^{🔓}.🔒 \vdash M : A}{\Gamma \vdash [[M]_{🔒}]_{🔓} = M : A} \qquad\qquad \textsc{tm/lock-unlock}\quad \frac{\Gamma \vdash M : \Box A}{\Gamma \vdash M = [[M]_{🔓}]_{🔒} : \Box A}$$

Notice, no commutating conversions, this is a win from the Fitch style.[3]

[3] Clouston 2018 and Birkedal, Clouston, Mannaa, Møgelberg, Pitts, Spitters 2019

Making Hard Choices: Definitional Equalities for $\mathrm{MLTT}_{\text{🔒}}$

We are able to equip $\Box A$ with both a β and η rule in $\mathrm{MLTT}_{\text{🔒}}$:

$$\frac{\Gamma^{\text{🔓}}.\text{🔒} \vdash M : A}{\Gamma \vdash [[M]_{\text{🔒}}]_{\text{🔓}} = M : A}\ \textsc{tm/unlock-lock} \qquad \frac{\Gamma \vdash M : \Box A}{\Gamma \vdash M = [[M]_{\text{🔓}}]_{\text{🔒}} : \Box A}\ \textsc{tm/lock-unlock}$$

Notice, no commutating conversions, this is a win from the Fitch style.[3]

The premises of these rules are subtle and important!

$$\Gamma^{\text{🔓}}.\text{🔒} \vdash M : A \implies \Gamma \vdash M : A$$

$$\Gamma \vdash [[M]_{\text{🔓}}]_{\text{🔒}} : \Box A \not\Longrightarrow \Gamma \vdash M : \Box A$$

[3] Clouston 2018 and Birkedal, Clouston, Mannaa, Møgelberg, Pitts, Spitters 2019

Taking Stock

What do we have at this point?

- $\mathrm{MLTT}_{\text{🔒}}$: a declarative modal dependent type theory.
- We can prove the expected admissibilities: substitution, presupposition, ...
- As well as modal admissibilities: lock contraction, strengthening...

These are important checks to ensure that $\mathrm{MLTT}_{\text{🔒}}$ behaves well.

Taking Stock

Complication: *non-local* and sensitive to extensions.

What do we have at this point?

- $\mathrm{MLTT}_{\text{🔒}}$: a declarative modal dependent type theory.
- We can prove the expected admissibilities: substitution, presupposition, ...
- As well as modal admissibilities: lock contraction, strengthening...

These are important checks to ensure that $\mathrm{MLTT}_{\text{🔒}}$ behaves well.

Taking Stock

What do we have at this point?

- $\mathrm{MLTT}_{🔒}$: a declarative modal dependent type theory.
- We can prove the expected admissibilities: substitution, presupposition, …
- As well as modal admissibilities: lock contraction, strengthening…

These are important checks to ensure that $\mathrm{MLTT}_{🔒}$ behaves well.

Big remaining question: can we implement this?

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. **Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.**
2. Prove that $\mathbb{T}$ admits a *normalization* theorem.
3. Conclude that $\mathbb{T}$ enjoys decidable conversion.
4. Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.
5. Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.
6. Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.

Many of these proofs rely on the admissiblities we established!

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.
2. **Prove that $\mathbb{T}$ admits a *normalization* theorem.**
3. Conclude that $\mathbb{T}$ enjoys decidable conversion.
4. Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.
5. Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.
6. Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.

Many of these proofs rely on the admissiblities we established!

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.
2. Prove that $\mathbb{T}$ admits a *normalization* theorem.
3. **Conclude that $\mathbb{T}$ enjoys decidable conversion.**
4. Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.
5. Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.
6. Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.

Many of these proofs rely on the admissiblities we established!

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.
2. Prove that $\mathbb{T}$ admits a *normalization* theorem.
3. Conclude that $\mathbb{T}$ enjoys decidable conversion.
4. **Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.**
5. Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.
6. Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.

Many of these proofs rely on the admissiblities we established!

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.
2. Prove that $\mathbb{T}$ admits a *normalization* theorem.
3. Conclude that $\mathbb{T}$ enjoys decidable conversion.
4. Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.
5. **Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.**
6. Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.

Many of these proofs rely on the admissiblities we established!

Implementing a Type Theory: A General Recipe

The process of implementing some type theory $\mathbb{T}$ might follow these steps:

1. Construct a bidirectional syntax for $\mathbb{T}$: $\mathbb{T}^{\leftrightarrows}$.
2. Prove that $\mathbb{T}$ admits a *normalization* theorem.
3. Conclude that $\mathbb{T}$ enjoys decidable conversion.
4. Prove that $\mathbb{T}^{\leftrightarrows}$ enjoys decidable type-checking.
5. Prove that every term of $\mathbb{T}$ is convertible with a term from $\mathbb{T}^{\leftrightarrows}$.
6. **Conclude that $\mathbb{T}^{\leftrightarrows}$ presents $\mathbb{T}$ and is implementable.**

Many of these proofs rely on the admissiblities we established!

Implementing $\mathrm{MLTT}_{\text{🔒}}$: Bidirectional Syntax

$\mathrm{MLTT}_{\text{🔒}}$ is a simple enough that we can extend a bidirectional presentation of MLTT:

- Terms are split into two categories:

$$
\begin{array}{llcl}
\textit{(Checkable)} & N, M & ::= & R \mid \lambda x.\, M \mid \ \ldots \\
\textit{(Synthesizable)} & R, S & ::= & (M : A) \mid x \mid R(M) \mid \ \ldots
\end{array}
$$

- We split the judgments along these lines as well:

$$
\begin{array}{cc}
\text{CHECK} & \text{SYNTH} \\
\Gamma \vdash M \Leftarrow A & \Gamma \vdash S \Rightarrow A
\end{array}
$$

- We can extend the standard rules with the new rules for $\Box A$:

$$
\frac{\Gamma.\text{🔒} \vdash M \Leftarrow A}{\Gamma \vdash [M]_{\text{🔒}} \Leftarrow \Box A}
\qquad\qquad
\frac{\Gamma^{\text{🔓}} \vdash M \Rightarrow \Box A}{\Gamma \vdash [M]_{\text{🔓}} \Rightarrow A}
$$

By restricting $\mathrm{MLTT}_{🔒}$ to $\mathrm{MLTT}_{🔒}^{\leftrightarrows}$ we can obtain the following result:

Theorem
If we can $\Gamma \vdash A = B\ \mathit{type}$ *is decidable*[*] *then so are* $\Gamma \vdash M \Leftarrow A$ *and* $\Gamma \vdash M \Rightarrow A$.

We've restricted $\mathrm{MLTT}_{🔒}^{\leftrightarrows}$ so that most one rule applies in each case.

---

[*] We also need whnfs, but this will follow from how we prove the decidability of conversion.

Normalization is a common way to decide equality.

Definition
The normalization function has the following type:

$$\underline{\mathbf{norm}}^{A}_{\Gamma} : \mathbf{Term}_{\Gamma,A} \to \mathbf{Term}_{\Gamma,A},$$

Completeness:

If $\Gamma \vdash M_1 = M_2 : A$ then $\underline{\mathbf{norm}}^{A}_{\Gamma}(M_1) = \underline{\mathbf{norm}}^{A}_{\Gamma}(M_2)$.

Soundness:

If $\Gamma \vdash M : A$ then $\Gamma \vdash M = \underline{\mathbf{norm}}^{A}_{\Gamma}(M) : A$

Normalization is a common way to decide equality.

Definition
The normalization function has the following type:

$$\underline{\mathbf{norm}}_{\Gamma}^{A} : \mathbf{Term}_{\Gamma,A} \to \mathbf{Term}_{\Gamma,A},$$

Completeness:

If $\Gamma \vdash M_1 = M_2 : A$ then $\underline{\mathbf{norm}}_{\Gamma}^{A}(M_1) = \underline{\mathbf{norm}}_{\Gamma}^{A}(M_2)$.

Soundness:

If $\Gamma \vdash M : A$ then $\Gamma \vdash M = \underline{\mathbf{norm}}_{\Gamma}^{A}(M) : A$

Corollary
$\Gamma \vdash M = N : A \iff \underline{\mathbf{norm}}_{\Gamma}^{A}(M)$ *and* $\underline{\mathbf{norm}}_{\Gamma}^{A}(N)$ *are* *identical*.

In order to actually define $\underline{\mathbf{norm}}_{\Gamma}^{A}$ we use *normalization-by-evaluation*.[4]

Slogan: evaluate syntax to a computational domain, quote it to a normal form.

- Evaluation performs β-reduction.
- Quotation is *type-directed* and handles η-expansion.
- The algorithm scales to support $\square A$, even with η.

[4] Martin-Löf 1975, see Abel 2013 for an overview.

A Sketch of a Proof Sketch

Lots of details to balance here, since we also support a full dependent type theory!

Completeness:

Construct a PER model on the computational domain.

Soundness:

Construct a Kripke cross-language logical relation between the computational domain and syntax.

The main sources of complexity are the modality and universes.

A Sketch of a Proof Sketch

Lots of details to balance here, since we also support a full dependent type theory!

Completeness:

Construct a Kripke PER model on the computational domain.

Soundness:

Construct a double Kripke cross-language logical relation between the computational domain and syntax.

The main sources of complexity are the modality and universes.

After normalization we end up with normal forms, but what do these look like?

$$\frac{\Gamma \vdash^{\mathsf{ne}} R : \Pi(A, x.B) \qquad \Gamma \vdash^{\mathsf{nf}} M : A}{\Gamma \vdash^{\mathsf{ne}} R(M) : B[M/x]} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} S : \mathsf{U}_i}{\Gamma \vdash^{\mathsf{nf}} S : \mathsf{U}_i}$$

$$\frac{\Gamma.\text{🔒} \vdash^{\mathsf{nf}} M : A}{\Gamma \vdash^{\mathsf{nf}} [M]_{\text{🔒}} : \Box A} \qquad \frac{\Gamma^{\text{🔓}} \vdash^{\mathsf{ne}} M : \Box A}{\Gamma \vdash^{\mathsf{ne}} [M]_{\text{🔓}} : A}$$

Corollary
There is no term $\cdot \vdash \mathrm{bad} : \Pi(A, \Box A)$

Observe that neutral terms are synthesizable, normal forms are checkable.[5]

[5] Coquand 1996

The Full Theorem

Theorem (Decidability of Type-Checking)

- *Both* $\Gamma \vdash M \Rightarrow A$ *and* $\Gamma \vdash M \Leftarrow A$ *are decidable.*
- *If* $\Gamma \vdash M \Rightarrow A$ *or* $\Gamma \vdash M \Leftarrow A$ *then* $\Gamma \vdash M : A$.
- *If* $\Gamma \vdash M : A$, *there exists*[*] *some* N *such that* $\Gamma \vdash M = N : A$ *and* $\Gamma \vdash N \Leftarrow A$.

This theorem provides the foundation for our implementation of $\text{MLTT}_{🔒}$.

To our knowledge, this is the first such result for MLTT with $\square A$.

[*] In particular, $N \triangleq \underline{\mathbf{norm}}^{A}_{\Gamma}(M)$.

Conclusions

We contribute $\mathrm{MLTT}_{🔒}$, a dependent type theory with...

$$\left.\begin{array}{l}\bullet\ \text{the box modality, } \square A \\ \bullet\ \text{dependent sums, } \Sigma(A, B) \\ \bullet\ \text{dependent products, } \Pi(A, B)\end{array}\right\}\ \text{With both } \beta \text{ and } \eta$$

- natural numbers, nat
- intensional identity types, $\mathsf{Id}(A, M, N)$
- a cumulative hierarchy of universes, $\mathsf{U}_0, \mathsf{U}_1 \dots$

We have proved the decidability of typechecking for $\mathrm{MLTT}_{🔒}$, and implemented it.

`http://github.com/jozefg/blott`