Notes on elaborating MTT

Zoe Stafford

December 5, 2025

Contents

1 Introduction

In this report, we document our work on implementing elaboration for Multimodal type theory[3] (MTT) based on an elaboration algorithm presented in András Kovács' Elaboration Zoo[4]. We start by describing normalisation-by-evaluation (NbE), bidirectional type checking and unification for non-modal dependent type theory, then we introduce MTT and show how to extend those algorithms to MTT and explore the problems we encounter. Stafford [6] extends `mitten`[7] with metavariables and elaboration.

Notation. We denote $\theta_1 \circ \theta_2$ by θ_{12}. De Bruijn index n is written v_n and de Bruijn level n is written l_n. We denote definitional equality by $=$ and structural equality by $\equiv$.

2 Elaboration Zoo

This section describes a variant of the bidirectional type checking algorithm with metavariables presented by Kovács [4]. Unlike Kovács [4], we allow metavariable to be open by associating them with a potentially non-empty context. This is important when for elaborating MTT, since contexts in MTT can contain locks, and are not, in general, equivalent to some iterated Π-type.

Metacontexts:	$\Theta\ \mathsf{mctx}$	$\Theta = \Theta'\ \mathsf{mctx}$
Metasubstitutions:	$\Theta \vdash \theta : \Theta'$	$\Theta \vdash \theta = \theta' : \Theta'$
Contexts:	$\Theta \vdash \Gamma\ \mathsf{ctx}$	$\Theta \vdash \Gamma = \Delta$
Substitutions:	$\Theta \vert \Gamma \vdash \delta : \Delta$	$\Theta \vert \Gamma \vdash \gamma = \delta : \Delta$
Types:	$\Theta \vert \Gamma \vdash A\ \mathsf{type}$	$\Theta \vert \Gamma \vdash A = B\ \mathsf{type}$
Terms:	$\Theta \vert \Gamma \vdash M : A$	$\Theta \vert \Gamma \vdash M = N : A$

$$\frac{}{\cdot\ \mathsf{mctx}} \qquad \frac{\Theta\ \mathsf{mctx} \qquad \Theta \vdash \Gamma\ \mathsf{ctx} \qquad \Theta|\Gamma \vdash A\ \mathsf{type} \qquad \alpha \notin \mathsf{Subjects}(\Theta)}{\Theta.(\Gamma \vdash \alpha : A)\ \mathsf{mctx}}$$

$$\frac{\Theta_1 = \Theta_2\ \mathsf{mctx} \qquad \Theta_1 \vdash \Gamma_1 = \Gamma_2\ \mathsf{ctx} \qquad \Theta_1|\Gamma_1 \vdash A = B\ \mathsf{type}}{\Theta.(\Gamma_1 \vdash \alpha : A) = \Theta.(\Gamma_2 \vdash \alpha : B)\ \mathsf{mctx}} \qquad \frac{\Theta\ \mathsf{mctx}}{\Theta \vdash \cdot : \cdot}$$

$$\frac{\Theta \vdash \theta : \Theta' \qquad \Theta|\Gamma[\theta] \vdash M : A[\theta]}{\Theta \vdash \theta.(\alpha \coloneqq M) : \Theta'.(\Gamma \vdash \alpha : A)} \qquad \frac{(\Gamma \vdash \alpha : A) \in \Theta}{\Theta|\Gamma \vdash \alpha : A} \qquad \frac{}{\Theta \vdash \cdot = \theta : \cdot}$$

$$\frac{\Theta|\Gamma \vdash M = N : A}{\Theta \vdash \theta_1.(\alpha \coloneqq M) = \theta_2.(\alpha \coloneqq N) : \Delta.(\Gamma \vdash \alpha : A)}$$

Figure 1: Judgements and rules for MLTT with metavariables

2.1 Core Syntax

Our core syntax is based on the core syntax presented by Gratzer et al. [3] without modalities, modal types, locks and keys. Since we care about representing terms on a computer, we do not consider definitionally equal terms identical.

There are 2 kinds of types: small types and large types. Small types are written $\Gamma \vdash A\ \mathsf{type}_0$ and large types $\Gamma \vdash A\ \mathsf{type}_1$, however we omit the size of the type when it is not relevant. We have a single universe of small types and an operation $\Uparrow$ to include small types in large types. We omit $\Uparrow$ when it is clear from context.

$$\frac{}{\Gamma \vdash \mathbf{U}\ \mathsf{type}_1} \qquad \frac{\Gamma \vdash A\ \mathsf{type}_0}{\Gamma \vdash \mathbf{Code}(A) : \mathbf{U}} \qquad \frac{\Gamma \vdash A : \mathbf{U}}{\Gamma \vdash \mathbf{El}(A)\ \mathsf{type}_0} \qquad \frac{\Gamma \vdash A\ \mathsf{type}_0}{\Gamma \vdash \Uparrow A\ \mathsf{type}_1}$$

We add metacontexts and metasubstitutions to track metavariables, by parameterising existing judgements by a metacontext and adding rules governing metacontexts, metasubstitutions and metavariables, as shown in Figure 1. Metavariables can't be bound, so we use named syntax for them. Note that substitutions get stuck on metavariables; substitutions stuck on metavariables are called *delayed substitutions.* We write $\mathsf{Subjects}(\Theta)$ for the set of metavariables bound by Θ.

We define some operations to work with metavariables:

- $\mathsf{freshMeta}_\Theta(\Gamma, A) = (\Theta.(\Gamma \vdash \alpha : A), \mathbf{p}, \alpha)$ where $\alpha \notin \mathsf{Subjects}(\Theta)$
- $\mathsf{solveMeta}_\Theta(\alpha, M)$

$\mathsf{solveMeta}$ is defined such that if $(\Gamma \vdash \alpha : A) \in \Theta$ is a metavariable and $\Theta|\Gamma \vdash M : A$ is a term such that α doesn't occur in M then $\mathsf{solveMeta}_\Theta(\alpha, M)$ satsfies:

- $\mathsf{solveMeta}_{\Theta}(\alpha, M) = (\Theta', \theta)$
- $\Theta' \vdash \theta : \Theta$
- $\mathsf{Subjects}(\Theta') = \mathsf{Subjects}(\Theta) - \{\alpha\}$
- $\alpha[\mathsf{solveMeta}_{\Theta}(\alpha, M)] = M$
- For any metavariable $(\Delta \vdash \beta : B) \in \Theta$ where $\beta \neq \alpha$, $(\Gamma'[M/\alpha] \vdash \beta : B[M/\alpha]) \in \Theta'$

2.2 Pre-syntax

The core syntax of the dependent type theory we use in this report requires a lot of type annotations, which we, as users, don't want to write. Additionally, variable names are often more convenient than the de Bruijn indices used by the core syntax. To allow convenience features for users while still having a formally defined core syntax, many type checkers and elaborators use an informal pre-syntax that allow the user to use variable names, omit type annotations and use holes to omit entire terms. We then translate this to core syntax in a process known as elaboration. To formalise elaboration we must first define the pre-syntax we will use:

$$\begin{array}{lll}
\text{Variables:} & x, y, z & \\
\text{Preterms:} & A, B, M, N, O & ::= \Pi(x : A)B \mid \lambda x.M \mid M(N) \\
 & & \mid \Sigma(x : A)B \mid (M, N) \mid \textsf{\textbf{fst}}(M) \mid \textsf{\textbf{snd}}(m) \\
 & & \mid \textsf{\textbf{Nat}} \mid \textsf{\textbf{zero}} \mid \textsf{\textbf{suc}}(M) \mid \textsf{\textbf{rec}}(A; M; N; O) \\
 & & \mid \textsf{\textbf{U}} \mid \_
\end{array}$$

The pre-term _ is a *hole*, which is a term that the elaborator will fill in. We shall see that holes are elaborated to metavariables, which can later be solved.

2.3 Normalisation by Evaluation

There are 2 stages to Normalisation by Evaluation (NbE): *evaluation* which computes β-normal forms, and type directed *quoting* which η-expands terms. Formally we will define 2 operations:

- $\mathsf{eval}^{\sigma}(M)$ which takes a type or term M and computes its β-normal form using the environment σ to record the value of variables in M.
- $\mathsf{quote}^{s}(M : A)$ which takes a term M of type A (both in β-normal form), and returns the η-long form of M. We also defined a variant of this for quoting types: $\mathsf{quoteType}^{s}(A)$.

This allows us to define an operation $\mathsf{norm}^{\Gamma}(A)$, which takes a type A in context Γ and returns its β-normal, η-long form:

$$\mathsf{norm}^{\Gamma}(A) \coloneqq \mathsf{quoteType}^{|\Gamma|}(\mathsf{eval}^{\sigma}(A))$$

for some environment σ which we define later. We use this in Section 2.4 to reduce definitional equality checking to structural equality checking.

We need to weaken the environment every time we pass under a binder, so we represent terms in β-normal form as *values* using de Bruijn levels. This requires us to track the size of the context we are quoting into, in order to resolve de Bruijn levels to de Bruijn indices.

Variables:	x	$\in \mathbb{N}$ (de Bruijn level)
Metavariables:	α	
Terms:	t	Unevaluated terms
Normal Sub:	σ	$::= \cdot \mid \sigma.N$
Values:	A, u, v	$::= n : A \mid \Pi(A, c)\lambda c \mid \Sigma(A, c) \mid (u, v) \mid \mathbf{zero} \mid \mathbf{suc}(u) \mid \mathbf{Nat} \mid \mathbf{U}$
Neutrals:	n	$::= x \mid \alpha[\sigma] \mid n(N) \mid \mathbf{fst}(n) \mid \mathbf{snd}(n) \mid \mathbf{rec}(c_1; c_2; u; n)$
Normals:	N	$::= \mathsf{Normal}(u, A)$
Closure:	c	$::= t \ @\ \sigma$

We define an evaluation function, which computes the β-normal form of a term using an environment to avoid applying substitutions to terms.

Definition 1. An *environment* σ for a context Γ maps variables of Γ to values. Formally, it is a list of values satisfying:

- If $\Gamma = \cdot$ then $\sigma = \cdot$
- If $\Gamma = \Delta.A$ then $\sigma = \delta.M$ where δ is an environment for Δ and M is a value

For term $\Theta|\Gamma \vdash M : A$ and environment σ for Γ, we define $\mathsf{eval}^{\sigma}(M)$ by induction on M:

$$
\begin{aligned}
&\mathsf{eval}^{\sigma.u}(\mathbf{v}_0) := u\\
&\mathsf{eval}^{\sigma.u}(\mathbf{v}_{n+1}) := \mathsf{eval}^{\sigma}(\mathbf{v}_n)\\
&\mathsf{eval}^{\sigma}(M(N)) := \mathbf{do\_ap}(\mathsf{eval}^{\sigma}(M), \mathsf{eval}^{\sigma}(N))\\
&\mathbf{do\_ap}(\lambda(t \ @\ \sigma), u) := \mathsf{eval}^{\sigma.u}(c)\\
&\mathbf{do\_ap}(\mathsf{Normal}(n, \Pi(A, (B \ @\ \sigma))), u) := \mathsf{Normal}(n(\mathsf{Normal}(u, A)), \mathsf{eval}^{\sigma.u}(B))\\
&\qquad\vdots
\end{aligned}
$$

We define a type-directed quoting procedure to read back β-normal values into β-normal, η-long term. To convert from de Bruijn levels in values to de Bruijn indices in terms, we must track the size s of the context we are quoting into.

$$
\begin{aligned}
&\mathsf{quote}^{s}(u, \Pi(A, B)) := \lambda_{\mathsf{quoteType}^{s}(A)}(\mathsf{quote}^{s+1}(\mathbf{do\_ap}(u, \mathbf{l}_s), B))\\
&\mathsf{quote}^{s}(\mathsf{Normal}(n, A), \mathsf{Normal}(A, \mathbf{U})) := \mathsf{quoteNeutral}^{s}(n)\\
&\mathsf{quoteNeutral}^{s}(\mathbf{l}_n) := \mathbf{v}_{\mathsf{lvl\_to\_ix}^{s}(n)}\\
&\mathsf{quoteNeutral}^{s}(n(N)) := \mathsf{quoteNeutral}^{s}(n)(\mathsf{quote}^{s}(N))\\
&\mathsf{quoteType}^{s}(\Pi(A, B)) := \Pi(\mathsf{quoteType}(A), \mathsf{quoteType}^{s+1}(B))\\
&\mathsf{quoteType}^{s}(\mathbf{Id}_A(u, v)) := \mathbf{Id}_{\mathsf{quoteType}^{s}(A)}(\mathsf{quote}^{s}(u, A), \mathsf{quote}^{s}(v, A))\\
&\mathsf{quoteType}^{s}(\mathbf{El}(A)) := \mathbf{El}(\mathsf{quote}^{s}(A, \mathbf{U}))\\
&\qquad\vdots
\end{aligned}
$$

We now complete the definition of $\mathsf{norm}^{\Gamma}(A)$:

$$\mathsf{norm}^{\Gamma}(A) := \mathsf{quoteType}^{|\Gamma|}(\mathsf{eval}^{\sigma}(A))$$

where $\sigma = \cdot . \mathbf{l}_0 . \mathbf{l}_1 \ldots \mathbf{l}_n$ for $\Gamma = \cdot . A_0 . A_1 \ldots A_n$.

2.4 Unification

In this section we describe a unification algorithm, which is an algorithm that computes a metasubstitution that unifies two types or terms. Except for metavariables, this algorithm procedes by checking the two terms for structural equality, which for β-normal, η-long terms coincides with definitional equality. When one of the terms is stuck on a metavariable, the algorithm checks the unification problem satisfies certain conditions in order to solve the metavariable.

Definition 2. A *unifier* of terms $\Theta_1|\Gamma \vdash M : A$ and $\Theta_1|\Gamma \vdash N : A$ is a metacontext Θ_2 and metasubstitution $\Theta_2 \vdash \theta : \Theta_1$ such that $\Theta_2|\Gamma[\theta] \vdash M[\theta] = N[\theta] : A[\theta]$. A unifier of types is defined similarly.

Definition 3. A *most general unifier* (MGU) of terms $\Theta_1|\Gamma \vdash M : A$ and $\Theta_1|\Gamma \vdash N : A$ is a unifier (Θ_2, θ_1) such that for any other unifier (Θ_3, θ_2) of M and N, there exists a unique metasubstitution $\Theta_3 \vdash \theta_3 : \Theta_2$ such that $\Theta_3 \vdash \theta_2 = \theta_1 \circ \theta_3 : \Theta_1$. A most general unifier of types is defined similarly.

We define a procedure to unify β-normal, η-long terms (or types): $\Theta|\Gamma \vdash M \overset{?}{=}_{\beta\eta} N : A \rightsquigarrow (\Theta', \theta)$ (or $\Theta|\Gamma \vdash A \overset{?}{=}_{\beta\eta} B\,\mathsf{type} \rightsquigarrow (\Theta', \theta)$ respectively) that produces a most general unifier (Θ', θ) of terms M and N (or types A and B respectively). Except for solving metavariables, this just checks for structural equality, as demonstrated by the rule for Π types.

Unify/Pi

$$\frac{\Theta_1|\Gamma \vdash A_1 \overset{?}{=}_{\beta\eta} A_2\,\mathsf{type} \rightsquigarrow (\Theta_2, \theta_1) \qquad \Theta_2|\Gamma[\theta_1].A_1[\theta_1] \vdash B_1[\theta_1] \overset{?}{=}_{\beta\eta} B_2[\theta_1]\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_2)}{\Theta_1|\Gamma \vdash \Pi(A_1, B_1) \overset{?}{=}_{\beta\eta} \Pi(A_2, B_2)\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_{12})}$$

Lemma 1. *The rule* Unify/Pi *is correct (ie it produces an MGU).*

Proof.

- By IH, (Θ_2, θ_1) is an MGU of A_1 and A_2, hence $A_1[\theta_1] = A_2[\theta_1]$
- So $B_1[\theta_1]$ and $B_2[\theta_1]$ are types in context $\Theta_2|\Gamma[\theta_1].A_1[\theta_1]$
- By IH, (Θ_3, θ_2) is an MGU of $B_1[\theta_1]$ and $B_2[\theta_2]$ so $B_1[\theta_1 \circ \theta_2] = B_2[\theta_1 \circ \theta_2]$
- Hence $\Pi(A_1, B_1)[\theta_{12}] = \Pi(A_1[\theta_{12}], B_2[\theta_{12}]) = \Pi(A_2[\theta_{12}], B_2[\theta_{12}]) = \Pi(A_2, B_2)[\theta_{12}]$ so (Θ_3, θ_{12}) is a unifier of $\Pi(A_1, B_1)$ and $\Pi(A_2, B_2)$
- For any unifier (Δ, δ) of $\Pi(A_1, B_1)$ and $\Pi(A_2, B_2)$, $\Pi(A_1[\delta], B_1[\delta]) = \Pi(A_2[\delta], B_2[\delta])$
- Hence $A_1[\delta] = A_2[\delta]$ so δ is a unifier of A_1 and A_2 so $\delta = \theta_1 \circ \delta_1$
- Also $B_1[\theta_1][\delta_1] = B_1[\delta] = B_2[\delta] = B_2[\theta_1][\delta_1]$ so δ_1 is a unifier of $B_1[\theta_1]$ and $B_2[\theta_1]$ so $\delta_1 = \theta_2 \circ \delta_2$
- Hence $\delta = \theta_{12} \circ \delta_2$
- We now show the factorisation of δ is unique.

$$\begin{tikzcd}
\Theta_3 \arrow[r, "\theta_2"] & \Theta_2 \arrow[r, "\theta_1"] & \Theta_1 \\
& \Delta \arrow[ul, dashed, "\delta_2"] \arrow[u, dashed, "\delta_1"] \arrow[ur, "\delta"'] &
\end{tikzcd}$$

- Let $\Delta \vdash \delta_2' : \Theta_1$ such that $\delta = \theta_{12} \circ \delta_2'$
- TODO
- So (Θ_3, θ_{12}) is an MGU of $\Pi(A_1, B_1)$ and $\Pi(A_2, B_2)$

$\square$

We can extend this procedure to unify arbitrary types, by first normalising the types:

$$\frac{\Theta|\Gamma \vdash \mathsf{norm}^\Gamma(M) \overset{?}{=}_{\beta\eta} \mathsf{norm}^\Gamma(N)\,\mathsf{type} \rightsquigarrow (\Theta', \theta)}{\Theta|\Gamma \vdash M \overset{?}{=} N\,\mathsf{type} \rightsquigarrow (\Theta', \theta)}$$

This leaves unification problems where one side is stuck on a metavariable. We only solve the metavariable when the problem is in the *pattern fragment*[5], ie the problem is of the form $\Theta|\Gamma \vdash \alpha[\cdot.x_1 \dots x_n] \overset{?}{=}_{\beta\eta} t$ and satisfies the pattern conditions:

1. Each $x_1, \dots, x_m$ is a distinct bound variable
2. The free variables of t all occur in $x_1, \dots, x_m$
3. α doesn't occur in t

The rest of this section is spent defining $\mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t)$ and proving that the following unification rule is correct:

$$\frac{(\Delta \vdash \alpha : B) \in \Theta \qquad \Theta|\Gamma \vdash \alpha[\cdot.x_1 \dots x_n] \overset{?}{=}_{\beta\eta} t : B \text{ satisfies pattern conditions}}{\Theta|\Gamma \vdash \alpha[\cdot.x_1 \dots x_n] \overset{?}{=}_{\beta\eta} t : B \rightsquigarrow \mathsf{solveMeta}_\Theta(\alpha, \mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t))}$$

We may extend the pattern fragment to problems of the form $\alpha[\cdot.x_1 \dots x_n](x_{n+1}) \dots (x_m) \overset{?}{=}_{\beta\eta} t$, by adding λs to the solution, but for the sake of simplicity, we don't allow that in this report.

Lemma 2. *Any substitution γ can be expressed as a* normal form*: $\gamma = \cdot.M_1 \dots M_n$, and this normal form is unique up to definitional equality.*

Definition 4. A *renaming* is a substitution where the substitution extension rule is restricted to a variable.

Lemma 3. *Let $\Delta \vdash \gamma : \Gamma$ be a renaming and $\Gamma \vdash \mathbf{v}_k : A$. Then $\mathbf{v}_k[\gamma]$ is a variable.*

Lemma 4. *Let $\Delta \vdash \gamma : \Gamma$ be a renaming and $\Gamma \vdash t : A$ be a term in β-normal, η-long form. Then $t[\gamma]$ is in β-normal, η-long form, and $t[\gamma]$ has the same head constructor as t unless t is a variable.*

Lemma 5. *Let $\mathsf{norm}(t)$ denote the β-normal, η-long form of t.*

Let $\Delta \vdash \gamma : \Gamma$ be a renaming and $\Gamma \vdash t : A$ be a term. Then $\mathsf{norm}(t)[\delta] = \mathsf{norm}(t[\delta])$.

Proof. TODO $\square$

Definition 5. A *projection* is a renaming generated from the following rules:

$$\textsc{Empty}\ \frac{}{\Theta|\Gamma \vdash \cdot : \cdot} \qquad \textsc{Keep}\ \frac{\Theta|\Gamma \vdash \delta : \Delta \qquad \Theta|\Delta \vdash A\,\mathsf{type}}{\Theta|\Gamma.A[\delta] \vdash \mathsf{keep}(\delta) := (\delta \circ \mathbf{p}).\mathbf{v}_0 : \Delta.A} \qquad \textsc{Drop}\ \frac{\Theta|\Gamma \vdash \delta : \Delta \qquad \Theta|\Gamma \vdash A\,\mathsf{type}}{\Theta|\Gamma.A \vdash \mathsf{drop}(\delta) := \delta \circ \mathbf{p} : \Delta}$$

Definition 6. For projection $\Theta|\Gamma \vdash \delta : \Delta$, we define the set $\mathsf{vars}(\delta)$ of variables used by δ by induction on δ:

- $\mathsf{vars}(\cdot) = \emptyset$
- $\mathsf{vars}(\mathsf{keep}(\delta)) = \{\mathbf{v}_0\} \cup \{\mathbf{v}_{k+1} \mid \mathbf{v}_k \in \mathsf{vars}(\mathsf{keep}(\delta))\}$
- $\mathsf{vars}(\mathsf{keep}(\delta)) = \{\mathbf{v}_{k+1} \mid \mathbf{v}_k \in \mathsf{vars}(\mathsf{keep}(\delta))\}$

We define the relation $\leq_{\mathsf{var}}$ on terms by: $x \leq_{\mathsf{var}} y$ when $x = \mathbf{v}_m$ and $y = \mathbf{v}_n$ and $m \leq n$.

Lemma 6. *The normal form of a projection $\Theta|\Gamma \vdash \delta : \Delta$ is $\delta = \cdot.x_n.x_{n-1} \ldots x_1$ where $\mathsf{vars}(\delta) = \{x_n, x_{n-1}, \ldots, x_n\}$ with $x_1 \leq_{\mathsf{var}} \cdots \leq_{\mathsf{var}} x_n$.*

We define vars on general renamings using the normal form of the renaming: $\mathsf{vars}(\cdot.x_1 \ldots x_n) \coloneqq \{x_1, \ldots, x_n\}$ where $x_1, \ldots, x_n$ are variables.

Lemma 7. *Let $\Theta|\Gamma \vdash \delta : \Delta$ be a projection. Then for any substitutions $\Theta|\Delta \vdash \gamma_1 : \Delta'$ and $\Theta|\Delta \vdash \gamma_2 : \Delta'$, types $\Theta|\Delta \vdash A, B\,\mathsf{type}$ or terms $\Theta|\Delta \vdash M, N : A$:*

$$\gamma_1 \circ \delta = \gamma_2 \circ \delta \iff \gamma_1 = \gamma_2 \tag{1}$$
$$A[\delta] = B[\delta] \iff A = B \tag{2}$$
$$M[\delta] = N[\delta] \iff M = N \tag{3}$$

Proof. For (1), consider the normal forms γ_1 and γ_2:

$$\gamma_1 = \cdot.M_1 \ldots M_n$$
$$\gamma_2 = \cdot.N_1 \ldots N_n$$

Then

$$\begin{aligned} \gamma_1 \circ \delta = \gamma_2 \circ \delta &\iff \cdot.M_1[\delta] \ldots M_n[\delta] = \cdot.N_1[\delta] \ldots N_n[\delta] \\ &\iff \cdot.M_1 \ldots M_n = \cdot.N_1 \ldots N_n && \text{(By (3))} \\ &\iff \gamma_1 = \gamma_2 \end{aligned}$$

For (2) and (3), it suffices to show this for β-normal, η-long forms, since

$$\begin{aligned} A[\delta] = B[\delta] &\iff \mathsf{norm}(A[\delta]) = \mathsf{norm}(B[\delta]) \\ &\iff \mathsf{norm}(A)[\delta] = \mathsf{norm}(B)[\delta] && \text{(Lemma 5)} \end{aligned}$$

and $A = B \iff \mathsf{norm}(A) = \mathsf{norm}(B)$.

We procede by induction on β-normal, η-long forms A, B, x and y:

- $x \equiv \mathbf{v}_m$: By Lemma 3, $x[\delta] = y[\delta]$ is a variable, so $y = \mathbf{v}_n$. We show $m = n$ by induction on m, n and δ:
  - $m = 0$ and $\delta = \mathsf{keep}(\delta')$: $x[\delta] = \mathbf{v}_0[\mathsf{keep}(\delta')] = \mathbf{v}_0$. Assume for sake of contradiction $n > 0$. Then $y[\delta] = \mathbf{v}_n[\mathsf{keep}(\delta')] = \mathbf{v}_{n-1}[\delta'][\mathbf{p}]$, but δ' is a renaming, so $\mathbf{v}_{n-1}[\delta'][\mathbf{p}] = \mathbf{v}_k[\mathbf{p}] = \mathbf{v}_{k+1}$ for some k and $k + 1 > 0$ so $\sharp$. Hence $n = 0 = m$.
  - $m > 0$ and $\delta = \mathsf{keep}(\delta')$: $x[\delta] = \mathbf{v}_m[\mathsf{keep}(\delta')] = \mathbf{v}_{m-1}[\delta'][\mathbf{p}]$ and $y[\delta] = \mathbf{v}_{n-1}[\delta'][\mathbf{p}]$. By IH, $m - 1 = n - 1$ so $m = n$.

  - $\delta = \mathsf{drop}(\delta')$: $x[\delta] = \mathbf{v}_n[\mathsf{drop}(\delta')] = \mathbf{v}_n[\delta'][\mathbf{p}]$ and $y[\delta] = \mathbf{v}_m[\delta'][\mathbf{p}]$. Since δ' is a renaming, $\mathbf{v}_n[\delta'] = \mathbf{v}_r$ and $\mathbf{v}_m[\delta'] = \mathbf{v}_k$, so $\mathbf{v}_{r+1} = \mathbf{v}_r[\mathbf{p}] = x[\delta] = y[\delta] = \mathbf{v}_k[\mathbf{p}] = \mathbf{v}_{k+1}$ so $r = k$ so $\mathbf{v}_n[\delta'] = \mathbf{v}_m[\delta']$. By IH, $n = m$.
  - $\delta = \cdot$, this case is impossible.
- $A \equiv \Pi(A_1, A_2)$:
  - $A[\delta] = \Pi(A_1[\delta], A_2[\mathsf{keep}(\delta)])$.
  - By Lemma 4, $B = \Pi(B_1, B_2)$, since δ is a renaming.
  - Hence $A_1[\delta] = B_1[\delta]$ and $A_2[\mathsf{keep}(\delta)] = B_2[\mathsf{keep}(\delta)]$.
  - By IH, $A_1 = B_1$ and $A_2 = B_2$ so $\Pi(A_1, A_2) = \Pi(B_1, B_2)$.
- $x \equiv \alpha[\cdot.x_1 \dots x_n]$:
  - $x[\delta] = \alpha[\cdot.x_1[\delta] \dots x_n[\delta]]$.
  - So $y = \alpha[\cdot.y_1 \dots y_n]$ with $y_i[\delta] = x_i[\delta]$ for $1 \le i \le n$, since δ is a renaming.
  - By IH, $x_i = y_i$ for $1 \le i \le n$, so $x = y$.
- ...

□

Lemma 8. *Let $\Theta|\Gamma \vdash \delta : \Delta$ be a projection and $\Theta|\Gamma \vdash t : A$ with $\mathsf{FV}(t) \subseteq \mathsf{vars}(\delta)$. Then there is a unique (up to definitional equality) term $\Theta|\Delta \vdash t' : A'$ such that $t = t'[\delta]$.*

Proof. We first prove such a t' exists by induction on t:

- $t = \mathbf{v}_k$: then $\Gamma = \Gamma_1.A.\Gamma_2$ with $|\Gamma_2| = k$. $\mathsf{FV}(t) = \{\mathbf{v}_k\} \subseteq \mathsf{vars}(\delta)$. By Lemma 6, $\delta = \cdot.x_1 \dots x_n$ and there is an i such that $x_i = \mathbf{v}_k$, so let $t' = \mathbf{v}_i$, then $t'[\delta] = \mathbf{v}_k$.
- $t = \Pi(A, B)$: By IH, $\exists A'.A = A'[\delta]$. By IH with $\mathsf{keep}(\delta), \exists B'.B = B'[\mathsf{keep}(\delta)]$. Then:

$$\Pi(A', B')[\delta] = \Pi(A'[\delta], B'[\mathsf{keep}(\delta)]) = \Pi(A, B)$$

- ...

We now show t' is unique up to definitional equality. Let x and y be terms of type A such that $\Theta|\Gamma \vdash x[\delta] = y[\delta] : A[\delta]$. By Lemma 7, $\Theta|\Delta \vdash x = y : A$. □

Definition 7. Let Γ be a context annotated with names in metacontext Θ and $S \subseteq \mathsf{Subjects}(\Gamma)$, such that for any variable $x \in S$ with $(x : A) \in \Gamma$, $\mathsf{FV}(A) \subseteq S$. The restriction of Γ to S is the context $\Gamma \restriction S$, defined by:

$$\cdot \restriction S := \cdot$$
$$(\Gamma.(x : A)) \restriction S := \begin{cases} (\Gamma \restriction S).(x : \mathsf{rename}_{\Gamma,S}(A)) & \text{if } x \in S \\ \Gamma \restriction S & \text{if } x \notin S \end{cases}$$

Where $\mathsf{rename}_{\Gamma,S}(A)$ is the unique type A' such that $A'[\mathsf{restrict}_{\Gamma,S}] = A$, defined by Lemma 8.

And the projection $\Gamma \vdash \mathsf{restrict}_{\Gamma,S} : \Gamma \restriction S$ such that $\mathsf{vars}(\mathsf{restrict}_{\Gamma,S}) = S$, is defined by induction on Γ:

$$\mathsf{restrict}_{\cdot,S} \coloneqq \cdot$$
$$\mathsf{restrict}_{\Gamma.(x:A),S} \coloneqq \begin{cases} \mathsf{keep}(\mathsf{restrict}_{\Gamma,S-\{x\}}) & \text{if } x \in S \\ \mathsf{drop}(\mathsf{restrict}_{\Gamma,S-\{x\}}) & \text{if } x \notin S \end{cases}$$

For the $x \in S$ clause to be well-typed, we need a type A' in context $\Gamma \restriction S - \{x\}$ with $A'[\mathsf{restrict}_{\Gamma,S}] = A$. Since $(x : A) \in \Gamma, \mathsf{FV}(A) \subseteq S = \mathsf{vars}(\mathsf{restrict}_{\Gamma,S})$ so by Lemma 8, there is a unique such A'.

Lemma 9. *Let $\Gamma_1 \vdash \theta : \Gamma_2$ be a renaming in normal form. Let S be the set of variables used by θ. Then θ can be uniquely factored as $\theta = \iota \circ \theta^p$ where $\Gamma_1 \restriction S \vdash \iota : \Gamma_2$ is a renaming and $\Gamma_1 \vdash \theta^p : \Gamma_1 \restriction S$ is a projection.*

Proof. Let $\cdot.x_1 \ldots x_n$ be the normal form of θ. Let $\theta^p = \mathsf{restrict}_{\Gamma,S}$. For $1 \le i \le n$, let x'_i be the unique term such that $x'_i[\theta^p] = x_i$, defined by Lemma 8. Let $\iota = \cdot.x'_1 \ldots x'_n$. Then:

$$\iota \circ \theta^p = \cdot.(x'_1[\theta^p]) \ldots (x'_n[\theta^p]) = \cdot.x_1 \ldots x_n$$

TODO: uniqueness of θ^p

Suppose $\theta = \iota_1 \circ \theta^p = \iota_2 \circ \theta^p$. Then by Lemma 7, $\iota_1 = \iota_2$. □

Lemma 10. *Metasubstitutions and renamings commute. Let $\Theta_1 \vdash \theta : \Theta_2$ be a metasubstitution and $\Theta_2|\Gamma_1 \vdash \gamma : \Gamma_2$ be a renaming. We note that γ doesn't contain any metavariables, so γ is well-formed in metacontext Θ_1 and $\gamma[\theta] = \gamma$. Then for any type $\Theta_2|\Gamma_2 \vdash A\,\mathsf{type}$ (resp. term $\Theta_2|\Gamma_2 \vdash M : A$), $\Theta_1|\Gamma_1[\theta] \vdash A[\theta][\gamma] = A[\gamma][\theta]\,\mathsf{type}$ (resp. $\Theta_1|\Gamma_1[\theta] \vdash M[\theta][\gamma] = M[\gamma][\theta] : A[\gamma][\theta]$).*

Proof. We proceeded by induction on the judgement $\Theta_2|\Gamma_2 \vdash A\,\mathsf{type}$ or $\Theta_2|\Gamma_2 \vdash M : A$:

- $A = \Pi(B,C)$: $\Pi(B,C)[\theta][\gamma] = \Pi(B[\theta][\gamma], C[\theta][\gamma]) \overset{\text{IH}}{=} \Pi(B[\gamma][\theta], C[\gamma,\theta]) = \Pi(B,C)[\gamma][\theta]$.
- $M = \mathbf{v}_k$: $\mathbf{v}_k[\theta][\gamma] = \mathbf{v}_k[\gamma]$. Since γ is a renaming, by Lemma 3, $\mathbf{v}_k[\gamma] = \mathbf{v}_k[\gamma][\theta]$.
- $M = \alpha[\delta]$: Let $\cdot.M_1 \ldots M_n$ be the normal form of δ. Then:

$$\begin{aligned}\alpha[\delta][\theta][\gamma] &= \alpha[\cdot.M_1 \ldots M_n][\theta][\gamma] \\ &= (\alpha[\theta])[\cdot.M_1[\theta] \ldots M_n[\theta]][\gamma] \\ &= (\alpha[\theta])[\cdot.M_1[\theta][\gamma] \ldots M_n[\theta][\gamma]] \\ &\overset{\text{IH}}{=} (\alpha[\theta])[\cdot.M_1[\gamma][\theta] \ldots M_n[\gamma][\theta]] \\ &= (\alpha[\gamma][\theta])[\cdot.M_1[\gamma][\theta] \ldots M_n[\gamma][\theta]] \\ &= \alpha[\cdot.M_1 \ldots M_n][\gamma][\theta]\end{aligned}$$

□

Lemma 11. *Consider unification problem that satisfies the pattern conditions:*

$$\Theta|\Gamma \vdash \alpha[\cdot.x_1 \ldots x_n] \overset{?}{=}_{\beta\eta} t : A[\cdot.x_1 \ldots x_n]$$

for metavariable $(\Delta \vdash \alpha : A) \in \Theta$. Let $\gamma = \cdot.x_1 \ldots x_n$. By Lemma 9, γ can be uniquely factored as $\gamma = \iota \circ \gamma^p$ for some renaming $\Gamma \restriction \{x_1, \ldots, x_n\} \vdash \iota : \Delta$ and projection $\Gamma \vdash \gamma^p : \Gamma \restriction \{x_1, \ldots, x_n\}$. By Lemma 8, $\exists! t'$ such that $t = t'[\gamma^p]$. Then ι is invertible and $\mathsf{solveMeta}_\Theta(\alpha, t'[\iota^{-1}])$ is an MGU of $\alpha[\cdot.x_1 \ldots x_n]$ and t. We define $\mathsf{solve}(\alpha[\cdot.x_1 \ldots x_n], t) \coloneqq t'[\iota^{-1}]$.

Proof. First, we show ι is invertible: We define a function $f : |\Delta| \to |\Gamma \restriction \{x_1, \ldots, x_n\}|$:

$$f(k) = \textbf{let}\,\mathbf{v}_r = \mathbf{v}_k[\iota]\,\textbf{in}\,r$$

Let i, j such that $f(i) = f(j)$. Then $\mathbf{v}_i[\iota] = \mathbf{v}_j[\iota]$, so $\mathbf{v}_i[\gamma] = \mathbf{v}_j[\gamma]$ so $x_i = x_j$, but by the pattern conditions, $x_1, \ldots, x_n$ are distinct bound variables, so $i = j$. Hence f is injective. Since $|\Delta| = |\Gamma \restriction \{x_1, \ldots, x_n\}|$, f is an iso. Let $\iota^{-1} = \cdot.\mathbf{v}_{f^{-1}(n-1)}.\mathbf{v}_{f^{-1}(n-2)} \ldots \mathbf{v}_{f^{-1}(0)}$. We note the normal form of ι:

$$\iota = \textbf{id} \circ \iota = \cdot.\mathbf{v}_{n-1}[\iota] \ldots \mathbf{v}_0[\iota] = \cdot.\mathbf{v}_{f(n-1)} \ldots \mathbf{v}_{f(0)}$$

Claim: $\Delta \vdash \iota^{-1} : \Gamma \restriction \{x_1, \ldots, x_n\}$

Let $\Gamma' \coloneqq \Gamma \restriction \{x_1, \ldots, x_n\} = \cdot.A_{n-1}.A_{n-2} \ldots A_0$ and $\Delta = \cdot.B_{n-1}.B_{n-2} \ldots B_0$. $\Gamma' \vdash \iota : \Delta$ so:

$$\Gamma' \vdash \mathbf{v}_k : A_k[\mathbf{p}^k]$$
$$\Gamma' \vdash \mathbf{v}_{f(i)} : B_i[\cdot.\mathbf{v}_{f(n-1)} \ldots \mathbf{v}_{f(i+1)}]$$

Hence $\Gamma' \vdash A_{f(i)}[\mathbf{p}^{f(i)}] = B_i[\cdot.\mathbf{v}_{f(n-1)} \ldots \mathbf{v}_{f(i+1)}]\ \mathsf{type}$.

We procede by induction on m with IH:

$$\forall m \in \{0, \ldots, n\}.\Delta \vdash \cdot.\mathbf{v}_{f^{-1}(n-1)} \ldots \mathbf{v}_{f^{-1}(n-m)} : \cdot.A_{n-1} \ldots A_{n-m}$$

- $m = 0$: $\Delta \vdash \cdot : \cdot$
- $m = m' + 1$: By IH, $\Delta \vdash \gamma \coloneqq \cdot.\mathbf{v}_{f^{-1}(n-1)} \ldots \mathbf{v}_{f^{-1}(n-m')} : \cdot.A_{n-1} \ldots A_{n-m'}$. It remains to show: $\Delta \vdash \mathbf{v}_{f^{-1}(n-m)} : A_{n-m}[\gamma]$.

  We have:

$$\Delta \vdash \mathbf{v}_{f^{-1}(n-m)} : B_{f^{-1}(n-m)}[\mathbf{p}^{f^{-1}(n-m)}]$$
$$\Gamma' \vdash B_{f^{-1}(n-m)}[\cdot.\mathbf{v}_{f(n-1)} \ldots \mathbf{v}_{f(f^{-1}(n-m)+1)}] = A_{n-m}[\mathbf{p}^{n-m}]$$

$$\Delta \vdash A_{n-m}[\cdot.\mathbf{v}_{f^{-1}(n-1)} \ldots \mathbf{v}_{f^{-1}(n-m')}]\ \mathsf{type}$$

  TODO

Claim: ι^{-1} is the inverse to ι.

$$\begin{aligned}
\iota &= \textbf{id} \circ \iota \\
&= \cdot.\mathbf{v}_{n-1}[\iota] \ldots \mathbf{v}_0[\iota] \\
&= \cdot.\mathbf{v}_{f(n-1)} \ldots \mathbf{v}_{f(0)} \\
\iota \circ \iota^{-1} &= \cdot.\mathbf{v}_{f(n-1)}[\iota^{-1}] \ldots \mathbf{v}_{f(0)}[\iota^{-1}] \\
&= \cdot.\mathbf{v}_{f^{-1}(f(n-1))} \ldots \mathbf{v}_{f^{-1}(f(0))} \\
&= \textbf{id} \\
\iota^{-1} \circ \iota &= \cdot.\mathbf{v}_{f^{-1}(n-1)}[\iota] \ldots \mathbf{v}_{f^{-1}(0)}[\iota] \\
&= \cdot.\mathbf{v}_{f(f^{-1}(n-1))} \ldots \mathbf{v}_{f(f^{-1}(0))} \\
&= \textbf{id}
\end{aligned}$$

By the pattern conditions, α doesn't occur in t. Since ι is a renaming, α occurs in t' iff α occurs in t, so α doesn't occur in t, and since ι^{-1} is a renaming, α doesn't occur in $t'[\iota^{-1}]$. Hence $\mathsf{solveMeta}_{\Theta}(\alpha, t'[\iota^{-1}])$ is well-defined.

Second, we show $(\Theta', \theta) := \mathsf{solveMeta}_{\Theta}(\alpha, t'[\iota^{-1}])$ is a unifier of $\alpha[\gamma]$ and t:

$$\begin{aligned}
\alpha[\gamma][\theta] &= \alpha[\theta][\gamma] && \text{(Lemma 10)}\\
&= t'[\iota^{-1}][\gamma]\\
&= t'[\iota^{-1} \circ \iota \circ \gamma^p]\\
&= t'[\gamma^p]\\
&= t
\end{aligned}$$

By pattern condition 3, α doesn't occur in t, so $t[\theta] = t$, so $\alpha[\gamma][\theta] = t[\theta]$.

Finally, we show (Θ', θ) is a most general unifier of $\alpha[\gamma]$ and t. Suppose (Φ, ϕ) is a unifier of α and t.

$$\begin{aligned}
\alpha[\gamma][\phi] = t[\phi] &\iff \alpha[\iota \circ \gamma^p][\phi] = t'[\gamma^p][\phi]\\
&\iff \alpha[\phi][\iota \circ \gamma^p] = t'[\phi][\gamma^p] && \text{(Lemma 10)}\\
&\iff \alpha[\phi][\iota] = t'[\phi] && \text{(Lemma 7)}\\
&\iff \alpha[\phi] = t'[\phi][\iota^{-1}]\\
&\iff \alpha[\phi] = t'[\iota^{-1}][\phi] && \text{(Lemma 10)}
\end{aligned}$$

To show θ is an MGU we must show $\phi = \theta \circ \hat{\phi}$ for some unique $\Phi \vdash \hat{\phi} : \Theta'$, or equivalently, $\beta[\phi] = \beta[\theta][\hat{\phi}]$ for any metavariable β in Θ.

Suppose β is a metavariable in Θ', then β is a metavariable in Θ and $\beta \neq \alpha$. We have $\beta[\mathsf{solveMeta}_{\Theta}(\alpha, \mathsf{solve}(\alpha, t'[\iota^{-1}]))] = \beta$, so define $\beta[\hat{\phi}] := \beta[\phi]$. Then $\beta[\phi] = \beta[\hat{\phi}] = \beta[\theta][\hat{\phi}]$.

Since α doesn't occur in $t'[\iota^{-1}]$, $\Theta'|\Delta[\theta] \vdash t'[\iota^{-1}] : A[\theta]$. For metavariables $\beta \neq \alpha$, we have $\beta[\hat{\phi}] = \beta[\phi]$, so $t'[\iota^{-1}][\phi] = t'[\iota^{-1}][\hat{\phi}]$, so $\alpha[\phi] = t'[\iota^{-1}][\hat{\phi}] = \alpha[\theta][\hat{\phi}]$.

Let $\Phi \vdash \tilde{\phi} : \Theta'$ such that $\phi = \theta \circ \tilde{\phi}$. Then for metavariable β of Θ' we have $\beta \neq \alpha$, so $\beta[\theta \circ \tilde{\phi}] = \beta[\tilde{\phi}]$, so $\beta[\tilde{\phi}] = \beta[\phi] = \beta[\hat{\phi}]$ so $\tilde{\phi} = \hat{\phi}$. Hence the factorisation $\phi = \theta \circ \hat{\phi}$ is unique. $\square$

2.5 Bidirectional type checking

Bidirectional type checking can work in 2 modes: checking a preterm against a type, and infering the type of a preterm. When the type checker tries to check a term against type A if only knows how to infer, then it first infers type B for that term, then checks types A and B are equal. However, A and B may only be equal after solving some metavariables, so the type checker instead has to unify A and B using the algorithm described in Section 2.4.

We define 3 judgements for bidirectional type checking:

Judgement	Description	Inputs	Outputs
$\Theta\|\Gamma \vdash A\,\mathsf{type} \rightsquigarrow (\Theta', \theta, B)$	check A is a type	Θ, Γ, A	Θ', θ, B
$\Theta\|\Gamma \vdash M \Leftarrow A \rightsquigarrow (\Theta', \theta, N)$	check M has type A	Θ, Γ, M, A	Θ', θ, N
$\Theta\|\Gamma \vdash M \Rightarrow A \rightsquigarrow (\Theta', \theta, N)$	infer type of M	Θ, Γ, M	A, Θ', θ, N

In these judgements, Γ is a context annotated with names (ie context extension is of the form $\Gamma.(x : A)$). When required, we implicitly erase the names from an annotated context. These judgements output the elaborated form of the preterm A (resp. M) — this is the type B (resp. term N) that is the fully explicit form of A (resp. M). Additionally, these judgements output a unifier (Θ', θ) which is required to make the elaborated type B or term N type-check. The type checking and type inference judgements satisfy the following specifications:

$$\frac{A \in \mathsf{Preterm} \qquad \Theta|\Gamma \vdash A\,\mathsf{type} \rightsquigarrow (\Theta', \theta, B)}{\Theta' \vdash \theta : \Theta \qquad \Theta'|\Gamma[\theta] \vdash B\,\mathsf{type}}$$

$$\frac{M \in \mathsf{Preterm} \qquad \Theta|\Gamma \vdash A\,\mathsf{type} \qquad \Theta|\Gamma \vdash M \Leftarrow A \rightsquigarrow (\Theta', \theta, N)}{\Theta' \vdash \theta : \Theta \qquad \Theta'|\Gamma[\theta] \vdash N : A[\theta]}$$

$$\frac{M \in \mathsf{Preterm} \qquad \Theta|\Gamma \vdash M \Rightarrow A \rightsquigarrow (\Theta', \theta, N)}{\Theta' \vdash \theta : \Theta \qquad \Theta'|\Gamma[\theta] \vdash A\,\mathsf{type} \qquad \Theta'|\Gamma[\theta] \vdash N : A}$$

We present selected rules below:

Type/Pi

$$\frac{\Theta_1|\Gamma \vdash A\,\mathsf{type} \rightsquigarrow (\Theta_2, \theta_1, A') \qquad \Theta_1|\Gamma[\theta_1].(x : A') \vdash B\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_2, B')}{\Theta_1|\Gamma \vdash \Pi(x : A)B\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_{12}, \Pi(A'[\theta_1], B'))}$$

Check/Lam

$$\frac{\Theta_1|\Gamma.(x : A) \vdash M \Leftarrow B \rightsquigarrow (\Theta_2, \theta_1, M')}{\Theta_1|\Gamma \vdash \lambda x.M \Leftarrow \Pi(A, B) \rightsquigarrow (\Theta_2, \theta_1, \lambda_A(M'))}$$

Infer/Var

$$\frac{\Gamma = \Gamma_1.(x : A).\Gamma_2 \qquad x \notin \mathsf{Subjects}(\Gamma_2)}{\Theta|\Gamma \vdash x \Rightarrow A \rightsquigarrow (\Theta, \mathbf{id}, \mathbf{v}_{|\Gamma_2|})}$$

Infer/J

$$\frac{\begin{array}{c}\Theta_1|\Gamma \vdash E \Rightarrow \mathbf{Id}_A(u, v) \rightsquigarrow (\Theta_2, \theta_1, E') \\ \Theta_2|\Gamma.(x : A).(y : A).(p : \mathbf{Id}_A(\mathbf{v}_1, \mathbf{v}_0)) \vdash M\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_2, M') \\ \Theta_3|\Gamma.(x : A) \vdash R : M'[\mathbf{id}\,.\,\mathbf{v}_0\,.\,\mathbf{v}_0\,.\,\mathbf{refl}(\mathbf{v}_0)] \rightsquigarrow (\Theta_4, \theta_3, R')\end{array}}{\begin{array}{l}\Theta_1|\Gamma \vdash \mathbf{J}(x\,y\,p.M; x.R; E) \Rightarrow M'[\theta_3][\mathbf{id}\,.u[\theta_{23}].v[\theta_{23}].E'[\theta_{23}]] \\ \qquad \rightsquigarrow (\Theta_4, \theta_{123}, \mathbf{J}(M'[\theta_3]; R'; E'[\theta_{23}]))\end{array}}$$

Change Direction

$$\frac{\Theta_1|\Gamma \vdash M \Rightarrow A \rightsquigarrow (\Theta_2, \theta_1, M') \qquad \Theta_2|\Gamma[\theta_1] \vdash A \stackrel{?}{=} B\,\mathsf{type} \rightsquigarrow (\Theta_3, \theta_2)}{\Theta_1|\Gamma \vdash M \Leftarrow B \rightsquigarrow (\Theta_3, \theta_{12}, M')}$$

Check/Hole

$$\frac{(\Theta_2, \theta_1, \alpha) = \mathsf{freshMeta}_{\Theta}(\Gamma, A)}{\Theta_1|\Gamma \vdash \_ \Leftarrow A \rightsquigarrow (\Theta_2, \theta_1, \alpha[\mathbf{id}])}$$

Infer/Hole[1]

$$\frac{\begin{array}{c}(\Theta_2, \theta_1, \alpha) = \mathsf{freshMeta}_{\Theta_1}(\Gamma, \mathbf{U}) \\ (\Theta_3, \theta_2, \beta) = \mathsf{freshMeta}_{\Theta_2}(\Gamma[\theta_1], \alpha)\end{array}}{\Theta_1|\Gamma \vdash \_ \Rightarrow \alpha[\theta_1][\mathbf{id}] \rightsquigarrow (\Theta_3, \theta_{12}, \beta[\mathbf{id}])}$$

[1]There are size issues with this rule, which can be solved by allowing metavariables to stand in for (large) types, not just terms.

The Infer/J rule does the following:

- Infer the type of the scrutinee, in order to find the underlying type of the identity type, and its endpoints. If the inferred type is not an identity type, then it can be unified with the identity type $\mathsf{Id}_\alpha(\beta, \gamma)$ for fresh metavariables α, β, γ.
- Check the motive is a type, in the appropriate context.
- Check the type of each case is the motive instantiated with the appropriate constructor. For **J**, the only case is **refl**.
- Return the infered type of the motive instantiated with the scrutinee.

Metavariables allow us to infer the type of more terms, such as lambdas without type annotations:

$$\frac{(\Theta_2, \theta_1, \alpha) = \mathsf{freshMeta}_{\Theta_1}(\Gamma, \mathbf{U}) \qquad \Theta_2|\Gamma[\theta_1].(x : \alpha) \vdash M \Rightarrow B \rightsquigarrow (\Theta_3, \theta_2, M')}{\Theta_1|\Gamma \vdash \lambda x.M \Rightarrow \Pi(\alpha[\theta_2], B) \rightsquigarrow (\Theta_3, \theta_{12}, \lambda_{\alpha[\theta_2]}(M'))}$$

3 Multimodal type theory

Multimodal type theory (MTT) extends MLTT with modalities. MTT is indexed by a strict 2-category $\mathcal{M}$:

- 0-cells of $\mathcal{M}$ are *modes*. Each mode is a copy of MLTT, and can be modelled differently.
- 1-cells of $\mathcal{M}$ are *modalities* and allow terms from one mode to be included in a different mode.
- a 2-cell $\alpha : \mu \Rightarrow \nu$ of $\mathcal{M}$ defines an operation $\langle \mu \mid A \rangle \to \langle \nu \mid A^\alpha \rangle$ and can be used to define the behaviour of each modality

Gratzer et al. [3] has some examples of mode theories that include guarded recursion and an S4 modality (ie an idempotent comonad).

All judgements are indexed by a mode (written $@\ m$ at the end of the judgement). This makes sure that 1-cells are the only way to move terms from one mode to another. Every variable in a context is annotated with a 1-cell: in named syntax, this is written as $x :_\mu A$, in nameless syntax as $(\mu|A)$.

For any 1-cell $\mu : m \to n$, we have an operation $-, \{\mu\}$ which takes a context at mode n to a context at mode n and is contravariantly functorial in μ. This operation is used to define modal types, since they require moving a context between modes.

$$\frac{\mu : m \to n \qquad \Gamma\ \mathsf{ctx}\ @\ n}{\Gamma.\{\mu\}\ \mathsf{ctx}\ @\ m}$$

The modality annotation of each variable, as well as any locks appearing after it, are used to decide when a variable is accessible — unlike non-modal DTT, variables are not always accessible. $|\Gamma_2|$ denotes the number of variables bound by Γ_2 and ignores any locks in Γ_2.

$$\frac{\mu : m \to n \qquad \Gamma\ \mathsf{ctx}\ @\ m \qquad \Gamma = \Gamma_1.(\mu|A).\Gamma_2 \qquad k = |\Gamma_2| \qquad \alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)}{\Gamma \vdash \mathbf{v}^\alpha_k : A^\alpha[\mathbf{p}^{k+1}]\ @\ m}$$

The operation A^α updates the 2-cells associated with variables in A so that A^α is a valid type in context Γ.

MTT includes a new connective: modal types. A 1-cell $\mu : m \to n$ defines a type-former that allows moving types from mode m to mode n:

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

The elimination rule for modal types is complicated, since we want to be able to define a function $\langle \mu \mid \langle \nu \mid A \rangle \rangle \to \langle \mu \circ \nu \mid A \rangle$. This requires eliminating modal types 'through' a 1-cell:

$$\frac{\begin{array}{c}\mu : n \to o \qquad \nu : m \to n \qquad \Gamma, \{\mu \circ \nu\} \vdash A\,\mathsf{type} \;@\; m \qquad \Gamma, \{\mu\} \vdash M : \langle \nu \mid A \rangle \;@\; n \\ \Gamma, y :_\mu \langle \nu \mid A \rangle \vdash B\,\mathsf{type} \;@\; o \qquad \Gamma, x :_{\mu\circ\nu} A \vdash N : B[\mathsf{mod}_\nu(x)/y] \;@\; o\end{array}}{\Gamma \vdash \mathbf{let}_\mu\,\mathsf{mod}_\nu(x) \leftarrow M\,\mathbf{at}\,y.B\,\mathbf{in}\,N : B[M/y] \;@\; o}$$

Each 2-cell $\alpha : \mu \Rightarrow \nu$ defines a substitution key^α, which is functorial in α:

$$\frac{\alpha : \mu \Rightarrow \nu}{\Gamma.\{\nu\} \vdash \mathsf{key}^\alpha_\Gamma : \Gamma.\{\mu\}}$$

This allows us to validate the variable rule, for example:

$$\frac{\alpha : \mu \Rightarrow \nu}{\Gamma.(\mu|A).\{\nu\} \vdash \mathbf{v}_0[\mathsf{key}^\alpha] : A[\mathsf{key}^\alpha \circ (\mathbf{p}\,.\{\nu\})]}$$

In order for $\mathbf{v}_0$ to be well-typed, we need the modality annotation and the locks to be the same; key^α adjusts the $\{\nu\}$ in the context to $\{\mu\}$ so this holds.

We also adjust the rules for dependent products to allow the argument to be behind a modality. This can be modelled by MTT without modal dependent product, by making the domain be a modal type, however that is less user friendly. We annotate applications and lambdas with the modality of the argument.

$$\frac{\mu : m \to n \qquad \Gamma, \{\mu\} \vdash A\,\mathsf{type} \;@\; m \qquad \Gamma, x :_\mu A \vdash B\,\mathsf{type} \;@\; n}{\Gamma \vdash (x :_\mu A) \to B\,\mathsf{type} \;@\; n}$$

$$\frac{\mu : m \to n \qquad \Gamma, \{\mu\} \vdash N : A \;@\; m \qquad \Gamma \vdash M : (x :_\mu A) \to B \;@\; n}{\Gamma \vdash M(\mu|N) : B[N/x] \;@\; n}$$

$$\frac{\mu : m \to n \qquad \Gamma, (x :_\mu A) \vdash M : B \;@\; n}{\Gamma \vdash \lambda(x :_\mu A).M : (x :_\mu A) \to B \;@\; n}$$

Example 1.

$$\begin{array}{l}\mathsf{comp}_{\mu,\nu} : (A :_{\mu\circ\nu} \mathbf{U}) \to \langle \mu \mid \langle \nu \mid A \rangle \rangle \to \langle \mu \circ \nu \mid A \rangle \\ \mathsf{comp}_{\mu,\nu}(A, x_1) = \mathbf{let}\,\mathsf{mod}_\mu(x_2) \leftarrow x_1\,\mathbf{in} \\ \qquad\qquad\qquad\quad \mathbf{let}_\mu\,\mathsf{mod}_\nu(x_3) \leftarrow x_2\,\mathbf{in} \\ \qquad\qquad\qquad\quad \mathsf{mod}_{\mu\circ\nu}(x_3) \\ \mathsf{coe}_{\alpha:\mu\Rightarrow\nu} : (A :_\mu \mathbf{U}) \to \langle \mu \mid A \rangle \to \langle \nu \mid A^\alpha \rangle \\ \mathsf{coe}_\alpha(A, x_1) = \mathbf{let}\,\mathsf{mod}_\mu(x_2) \leftarrow x_1\,\mathbf{in} \\ \qquad\qquad\qquad\; \mathsf{mod}_\nu(x_2^\alpha)\end{array}$$

We now show how to extend NbE, unification and bidirectional type checking to MTT restricted to preorder-enriched mode theories (ie there is at most one 2-cell between any pair of 1-cells). We show that bidirectional type checking and NbE can be easily extended to handle MTT, however elaboration (specifically metavariables) causes some issues. When considering a preorder-enriched mode theory, we don't need to annotate variables with the 2-cell that 'unlocks' them, since the choice of 2-cell is unique.

In Section 2 we associated metavariables with a delayed substitution, which, in non-modal DTT, is equivalent to a list of terms formed with the context extension rule. In MTT, substitutions are not, in general, equivalent to a list of terms, so we have to choose between associating metavariables with a list of terms (which we call a *non-modal substitution*), or a substitution. In Section 3.3 we demonstrate that there are issues with both options.

In Section 3.1, we demonstrate how to extend NbE to MTT, and demonstrate the problem that NbE causes if we try to associate metavariables with a substitution. Section 3.2 defines some typing judgements which we use in Section 3.3 to prove that the method `mitten` uses to solve metavariables produces well-typed terms. In Section 3.5, we explore an alternative conversion testing algorithm and see that it easily allows us to associate metavariables with a substitution.

3.1 Normalisation by Evaluation

We extend the definition of values and neutral forms. Note that the variable in neutrals is not annotated with a 2-cell, since the choice of 2-cell is unique.

$$\begin{array}{lll} \text{Values:} & A, u, v & ::= \cdots \mid \langle \mu \mid A \rangle \mid \mathsf{mod}_\mu(u) \\ \text{Neutrals:} & n & ::= x \mid \cdots \mid \mathbf{let}_\mu\, \mathsf{mod}_\nu(\_) \leftarrow n\, \mathbf{in}\, c\, \mathbf{at}\, c \end{array}$$

Definition 8. An *environment* σ for a context Γ is a list of values satisfying:

- If $\Gamma = \cdot$ then $\sigma = \cdot$
- If $\Gamma = \Delta.A$ then $\sigma = \delta.M$ for environment δ for Δ and value M
- If $\Gamma = \Delta.\{\mu\}$ then σ is an environment for Δ

$\mathsf{eval}^\sigma(M)$ is defined as in Section 2.3, and extended to handle modal connectives as follows:

$$\begin{aligned} \mathsf{eval}^\sigma(\langle \mu \mid A \rangle) &:= \langle \mu \mid \mathsf{eval}^\sigma(A) \rangle \\ \mathsf{eval}^\sigma(\mathsf{mod}_\mu(M)) &:= \mathsf{mod}_\mu(\mathsf{eval}^\sigma(M)) \\ \mathsf{eval}^\sigma(\mathbf{let}_\mu\, \mathsf{mod}_\nu(\_) \leftarrow M\, \mathbf{in}\, N\, \mathbf{at}\, A) &:= \mathbf{do\_letmod}_{\mu,\nu}(A \mathbin{@} \sigma, N \mathbin{@} \sigma, \mathsf{eval}^\sigma(M)) \\ \mathbf{do\_letmod}_{\mu,\nu}(A \mathbin{@} \sigma_1, N \mathbin{@} \sigma_2, \mathsf{mod}_\nu(u)) &:= \mathsf{eval}^{\sigma_2 . u}(N) \\ \mathbf{do\_letmod}_{\mu,\nu}((M \mathbin{@} \sigma), b, n : \langle \nu \mid B \rangle) &:= \mathbf{let}_\mu\, \mathsf{mod}_\nu(\_) \leftarrow n\, \mathbf{in}\, b\, \mathbf{at} (M \mathbin{@} \sigma) : \mathsf{eval}^{\sigma . n : \langle \mu | B \rangle}(M) \end{aligned}$$

Modal types don't have an η rule, so quoting modal types is straightforward:

$$\begin{aligned} \mathsf{quoteType}^s(\langle \mu \mid A \rangle) &:= \langle \mu \mid \mathsf{quoteType}^s(A) \rangle \\ \mathsf{quote}^s(\mathsf{mod}_\mu(M), \langle \mu \mid A \rangle) &:= \mathsf{mod}_\mu(\mathsf{quote}^s(M : A)) \\ \mathsf{quote}^s(\mathsf{Normal}(n, \langle \mu \mid A \rangle), \langle \mu \mid A \rangle) &:= \mathsf{quoteNeutral}^s(n) \end{aligned}$$

We must also consider evaluation for metavariables. Metavariables are associated with a delayed substitution and in non-modal DTT, this delayed substitution can always be written as a list of term, which makes evaluation straightforward. In MTT, substitutions are not, in general, just a list of terms so evaluating a metavariable is more complex.

3.2 Split typing judgements

In this section we show that type checking in MTT can be split into two parts: type checking as in non-modal DTT, and checking variables are accessible. This is required to prove that the approach `mitten` takes to unification and in particular solving metavariables produces well-typed terms.

We define a variant of the typing judgements from Section 3, written $\Theta|\Gamma \vdash_{\mathsf{type}} A\,\mathsf{type} \mathrel{@} m$ or $\Theta|\Gamma \vdash_{\mathsf{type}} M : A \mathrel{@} m$, which are defined the same as $\Theta|\Gamma \vdash A\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash M : A \mathrel{@} m$, except the variable rule doesn't require a 2-cell to make the variable accessible:

$$\frac{\mu : m \to n \qquad \Gamma = \Gamma_1.(\mu|A).\Gamma_2 \qquad |\Gamma_2| = k}{\Theta|\Gamma \vdash_{\mathsf{type}} \mathbf{v}_k : A[\mathbf{p}^{k+1}] \mathrel{@} m}$$

We define a variant of the definitional equality judgements: $\Theta|\Gamma \vdash_{\mathsf{type}} A = B\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash_{\mathsf{type}} M = N : A \mathrel{@} m$ that presuppose that $\Theta|\Gamma \vdash_{\mathsf{type}} A, B\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash_{\mathsf{type}} M, N : A \mathrel{@} m$ respectively and don't presuppose $\Theta|\Gamma \vdash A, B\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash M, N : A \mathrel{@} m$ respectively. We note that $\vdash_{\mathsf{type}}$ judgements still require variables are used at the correct mode.

Lemma 12. *If $\Theta|\Gamma \vdash A\,\mathsf{type} \mathrel{@} m$ then $\Theta|\Gamma \vdash_{\mathsf{type}} A\,\mathsf{type} \mathrel{@} m$, and if $\Theta|\Gamma \vdash M : A \mathrel{@} m$ then $\Theta|\Gamma \vdash_{\mathsf{type}} M : A \mathrel{@} m$.*

Proof. Induction on the derivation of $\Theta|\Gamma \vdash M : A \mathrel{@} m$. ☐

Lemma 13. *If $\Theta|\Gamma \vdash_{\mathsf{type}} A = B\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash A\,\mathsf{type} \mathrel{@} m$ and B is in β-normal η-long form, then $\Theta|\Gamma \vdash B\,\mathsf{type} \mathrel{@} m$ and $\Theta|\Gamma \vdash A = B\,\mathsf{type} \mathrel{@} m$. If $\Theta|\Gamma \vdash_{\mathsf{type}} M = N : A \mathrel{@} m$ and $\Theta|\Gamma \vdash M : A \mathrel{@} m$ and N is in β-normal η-long form, then $\Theta|\Gamma \vdash N : A \mathrel{@} m$ and $\Theta|\Gamma \vdash M = N : A \mathrel{@} m$.*

Proof. We demonstrate the proof for terms M and N; the proof for types is similar.

We recall that our type theory is parameterised by a preorder-enriched category $\mathcal{C}$. We define the preorder-enriched category $\mathcal{C}'$ such that $\mathcal{C}'$ has the same 0-cells and 1-cells as $\mathcal{C}$ and for any parallel 1-cells μ and ν of $\mathcal{C}'$ we have $\mu \leq \nu$. To distinguish judgements under $\mathcal{C}$ and $\mathcal{C}'$, we write $\vdash^{\mathcal{C}}$ or $\vdash^{\mathcal{C}'}$. Since inference rules for $\Theta|\Gamma \vdash^{\mathcal{C}}_{\mathsf{type}} N : A \mathrel{@} m$ don't depend on the 2-cells of $\mathcal{C}$, we have $\Theta|\Gamma \vdash^{\mathcal{C}'}_{\mathsf{type}} N : A \mathrel{@} m$ and $\Theta|\Gamma \vdash^{\mathcal{C}'}_{\mathsf{type}} M = N : A \mathrel{@} m$.

Variables are always accessible under $\mathcal{C}'$ so $\Theta|\Gamma \vdash^{\mathcal{C}'} N : A \mathrel{@} m$ and $\Theta|\Gamma \vdash^{\mathcal{C}'} M = N : A \mathrel{@} m$. Let M' be the β-normal, η-long form of M, then $\Theta|\Gamma \vdash^{\mathcal{C}'} M' = N : A$. Since β-normal, η-long forms are unique, we have $M' \equiv N$ and so $\Theta|\Gamma \vdash^{\mathcal{C}} N : A$. ☐

Definition 9. Variable accesses in a type or term M are valid when the judgement $\Theta|\Gamma \vdash_{\mathsf{var}} M \mathrel{@} m$ holds, given by the rules in Figure 2. Some rules have been omitted, these rules are the same as the rules for $\Theta|\Gamma \vdash M\,\mathsf{type} \mathrel{@} m$ or $\Theta|\Gamma \vdash M : A \mathrel{@} m$, except they ignore types and only track modalities. In this judgement, Γ is not a context, since it doesn't track types, however we may treat a context as a context without types.

$$\frac{\mu : m \to n \qquad \Gamma = \Gamma_1.(\mu|A).\Gamma_2 \qquad |\Gamma_2| = n \qquad \exists\alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)}{\Theta|\Gamma \vdash_{\mathsf{var}} \mathbf{v}_n \; @ \; m}$$

$$\frac{\mu : m \to n \qquad \Theta|\Gamma.\{\mu\} \vdash_{\mathsf{var}} A \; @ \; m \qquad \Theta|\Gamma.(\mu|A) \vdash_{\mathsf{var}} B \; @ \; n}{\Theta|\Gamma \vdash_{\mathsf{var}} \Pi_\mu(A, B) \; @ \; n}$$

$$\frac{\mu : m \to n \qquad \Theta|\Gamma \vdash_{\mathsf{var}} M \; @ \; n \qquad \Theta|\Gamma.\{\mu\} \vdash_{\mathsf{var}} N \; @ \; m}{\Theta|\Gamma \vdash_{\mathsf{var}} M(\mu|N) \; @ \; n}$$

$$\frac{\mu : m \to n \qquad \Theta|\Gamma.(\mu|A) \vdash_{\mathsf{var}} M \; @ \; n}{\Theta|\Gamma \vdash_{\mathsf{var}} \lambda_{(\mu|A)} M \; @ \; n}$$

Figure 2: Some rules for variable access judgement

Lemma 14. *If* $\Theta|\Gamma \vdash M : A \; @ \; m$ *then* $\Theta|\Gamma \vdash_{\mathsf{var}} M \; @ \; m$.

Proof. Induction on the derivation $\Theta|\Gamma \vdash M : A \; @ \; m$. □

Lemma 15 (Subject construction lemma)**.** *Any derivation of* $\Theta|\Gamma \vdash_{\mathsf{var}} \mathbf{v}_n : A \; @ \; m$ *is of the form:*

$$\frac{\mu : m \to n \qquad \Gamma = \Gamma_1.(\mu|A).\Gamma_2 \qquad |\Gamma_2| = k \qquad \exists\alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)}{\Theta|\Gamma \vdash \mathbf{v}_k : A \; @ \; m}$$

etc for other syntax forms and $\Theta|\Gamma \vdash_{\mathsf{type}} M : A \; @ \; m$.

Lemma 16. *If* $\Theta|\Gamma \vdash_{\mathsf{type}} A\,\mathsf{type} \; @ \; m$ *and* $\Theta|\Gamma \vdash_{\mathsf{var}} A \; @ \; m$ *then* $\Theta|\Gamma \vdash A\,\mathsf{type} \; @ \; m$ *and if* $\Theta|\Gamma \vdash_{\mathsf{type}} M : A \; @ \; m$ *and* $\Theta|\Gamma \vdash_{\mathsf{var}} M \; @ \; m$ *then* $\Theta|\Gamma \vdash M : A \; @ \; m$.

Proof. We procede by induction on A or $M : A$:

- $M = \mathbf{v}_k : A$: We have:

  $$\Theta|\Gamma \vdash_{\mathsf{type}} \mathbf{v}_k : A \; @ \; m$$
  $$\Theta|\Gamma \vdash_{\mathsf{var}} \mathbf{v}_k \; @ \; m$$

  By Lemma 15 we have:

  - $\mu : m \to n$
  - $\Gamma = \Gamma_1.(\mu|A).\Gamma_2$ with $|\Gamma_2| = k$
  - $\exists\alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)$

  Hence $\Theta|\Gamma \vdash \mathbf{v}_k : A \; @ \; m$.

- $M = \Pi_\mu(A, B)\,\mathsf{type}$ for $\mu : m \to n$: By Lemma 15:

$$\Theta|\Gamma.\{\mu\} \vdash_{\mathsf{type}} A\,\mathsf{type} \;@\; m$$
$$\Theta|\Gamma.(\mu|A) \vdash_{\mathsf{type}} B\,\mathsf{type} \;@\; n$$
$$\Theta|\Gamma.\{\mu\} \vdash_{\mathsf{var}} A \;@\; m$$
$$\Theta|\Gamma.(\mu|A) \vdash_{\mathsf{var}} B \;@\; n$$

By IH, $\Theta|\Gamma.\{\mu\} \vdash A\,\mathsf{type} \;@\; m$ and $\Theta|\Gamma.(\mu|A) \vdash B\,\mathsf{type} \;@\; n$ so $\Theta|\Gamma \vdash \Pi_\mu(A, B)\,\mathsf{type} \;@\; n$.

- ...

□

3.3 Unification

Modal types can be unified like other connectives:

$$\frac{\Theta_1|\Gamma.\{\mu\} \vdash A \overset{?}{=}_{\beta\eta} B\,\mathsf{type} \rightsquigarrow (\Theta_2, \theta_1)}{\Theta_1|\Gamma \vdash \langle\mu \mid A\rangle \overset{?}{=}_{\beta\eta} \langle\mu \mid B\rangle\,\mathsf{type} \rightsquigarrow (\Theta_2, \theta_1)}$$

$$\frac{\Theta_1|\Gamma.\{\mu\} \vdash M \overset{?}{=}_{\beta\eta} N : A \rightsquigarrow (\Theta_2, \theta_1)}{\Theta_1|\Gamma \vdash \mathsf{mod}_\mu(M) \overset{?}{=}_{\beta\eta} \mathsf{mod}_\mu(N) : \langle\mu \mid A\rangle \rightsquigarrow (\Theta_2, \theta_1)}$$

Unification of a term stuck on a metavariable and another term requires more significant changes. One approach to unification of these cases could be to treat pretend we are dealing with non-modal DTT, and use the unification algorithm from Section 2.4; we shall call this approach the non-modal approach. Variables that are accessible in one context maybe not be accessible in another context, so the solution generated by the non-modal approach may not generate a well-typed term.

For example, consider the following (preorder-enriched) mode theory:

$$\begin{tikzcd} * \arrow[loop above, "\mu"] \end{tikzcd} \qquad \begin{array}{c} \mu \circ \mu = \mu \\ 1 \le \mu \end{array}$$

When type checking the term

$$\mathbf{let}\, f : (\mu \mid \mathbf{Nat}) \to \mathbf{Nat} := \lambda_\mu x.\_\ \mathbf{in}\ \lambda_{\mathbf{id}} x.\, \mathsf{the}(\mathbf{Id}(x, f(x)))(\mathbf{refl}(x))$$

where $\mathsf{the} : (A : \mathbf{U}) \to A \to A$ is the polymorphic identity function, the hole is elaborated to a fresh metavariable $x :_\mu \mathbf{Nat} \vdash \alpha : \mathbf{Nat}$. The type-checker then has to unify the infered type of $\mathbf{refl}(x)$ (ie $\mathbf{Id}(x, x)$) and the expected type (ie $\mathbf{Id}(x, f(x))$), which results in unification problem $x :_{\mathbf{id}} \mathbf{Nat} \vdash \alpha[x] \overset{?}{=} x$. The non-modal approach will solve this is $x :_\mu \mathbf{Nat} \vdash \alpha := x : \mathbf{Nat}$, however there is no 2-cell $\mu \le \mathbf{id}$, so this solution is not well-typed.

To ensure unification generates well-typed terms, we can requires that variables in the solution are accessible. This is generally straightforward to check: we can maintain a context that only tracks the modality annotations of each variable and the locks and use that to check every variable in the solution. There are problems however when it comes to checking variables used in the delayed substitution of a metavariable β.

If we associate metavariables with a non-modal substitution, it is unclear what context we should check each term in the non-modal substitution, so when we solve β as t it is not necessarily the case that applying the non-modal substitution to t produces a well-typed term.

If we instead associate metavariables with a substitution, then we at least have meta-theoretic rules to check variable accesses in the substitution, however I do not know of an algorithm to check a substitution is well-formed. I also don't know how to compute the inverse to a substitution in MTT; since substitutions are not merely lists of terms, the approach used in Section 2.4 does not work.

In `mitten` we associate metavariables with a non-modal substitution and we delay checking variable accesses in delayed substitutions in the solution to a metavariable. More concretely, if we encounter an unsolved metavariable α in the solution to metavariable β, we record that β is solved in terms of α, and once we solve α, we re-check the solution to β. We now formally describe the approach used in `mitten` and prove that it produces well-typed terms. We note that the formal description below does not exactly what we do in `mitten`, since `mitten` does not deal with metasubstitutions explicitly.

Definition 10. A `mitten` type or term is an MTT type or term where the rule Meta/Intro has been replaced by:

$$\frac{\begin{array}{c}\textsc{Meta/Intro}\\ (\Gamma \vdash \alpha : A \mathbin{@} m) \in \Theta \qquad \Gamma = \{\mu_0\}.(x_1 :_{\nu_1} A_1).\{\mu_1\} \ldots (x_n :_{\nu_n} A_n).\{\mu_n\} \\ \forall 1 \leq i \leq n. \Theta|\Delta \vdash_{\mathsf{type}} M_i : A_i[\cdot.M_1 \ldots M_{i-1}] \mathbin{@} \mathsf{Dom}(\nu_i)\end{array}}{\Theta|\Delta \vdash \alpha[\cdot.M_1 \ldots M_n] : A[\cdot.M_1 \ldots M_n] \mathbin{@} m}$$

We must also update the $\vdash_{\mathsf{var}}$ and $\vdash_{\mathsf{type}}$ version of Meta/Intro and update the definitional equality rule for metavariables:

$$\frac{\begin{array}{c}(\Gamma \vdash \alpha : A \mathbin{@} m) \in \Theta \qquad \Gamma = \{\mu_0\}.(x_1 :_{\nu_1} A_1).\{\mu_1\} \ldots (x_n :_{\nu_n} A_n).\{\mu_n\} \\ \forall 1 \leq i \leq n. \Theta|\Delta \vdash_{\mathsf{type}} M_i : A_i[\cdot.M_1 \ldots M_{i-1}] \mathbin{@} \mathsf{Dom}(\nu_i)\end{array}}{\Theta|\Delta \vdash_{\mathsf{type}} \alpha[\cdot.M_1 \ldots M_n] : A[\cdot.M_1 \ldots M_n] \mathbin{@} m}$$

$$\frac{(\Gamma \vdash \alpha : A \mathbin{@} m) \in \Theta}{\Theta|\Delta \vdash_{\mathsf{var}} \alpha[\cdot.M_1 \ldots M_n] \mathbin{@} m}$$

$$\frac{\begin{array}{c}(\Gamma \vdash \alpha : A \mathbin{@} m) \in \Theta \qquad \Gamma = \{\mu_0\}.(x_1 :_{\nu_1} A_1).\{\mu_1\} \ldots (x_n :_{\nu_n} A_n).\{\mu_n\} \\ \forall 1 \leq i \leq n. \Theta|\Delta \vdash_{\mathsf{type}} M_i = N_i : A_i[\cdot.M_1 \ldots M_{i-1}] \mathbin{@} \mathsf{Dom}(\nu_i)\end{array}}{\Theta|\Delta \vdash \alpha[\cdot.M_1 \ldots M_n] = \alpha[\cdot.N_1 \ldots N_n] \mathbin{@} m}$$

Since the Meta/Intro rule doesn't check variables in the delayed substitution are accessible, applying metasubstitutions is fallible. Suppose metasubstitution $\Theta \vdash \theta : \Theta'$ solves $\Gamma \vdash \alpha : A$ as t. When we apply θ to an occurances of α of the form $\Theta|\Delta \vdash \alpha[\cdot.M_1 \ldots M_n] : A[\cdot.M_1 \ldots M_n]$, we must check $\Theta|\Delta \vdash_{\mathsf{var}} t[\cdot.M_1 \ldots M_n]$, then by Lemma 18, we have $\Theta|\Delta \vdash t[\cdot.M_1 \ldots M_n] : A[\cdot.M_1 \ldots M_n]$.

We note that since we don't need to check variables in delayed substitutions, it is straightforward to decide if $\Theta|\Gamma \vdash_{\mathsf{var}} M$ holds for type or term M.

Lemma 17. `mitten`*'s method to solve metavariables produces well-typed terms.* `mitten`*'s method is:*

- *Compute the non-modal solution, as in Section 2.4.*
- *Check variable accesses in the solution are valid.*

Proof. Consider unification problem that satisfies the pattern conditions:

$$\Theta|\Gamma \vdash \alpha[\cdot.x_1 \dots x_n] \overset{?}{=}_{\beta\eta} t : A[\cdot.x_1 \dots x_n]$$

for metavariables $(\Delta \vdash \alpha : A) \in \Theta$. By Lemma 26, the non-modal solution satisfies

$$\Theta|\Delta \vdash_{\mathsf{type}} \mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t)$$

. Suppose that the non-modal solution satisfies

$$\Theta|\Delta \vdash_{\mathsf{var}} \mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t)$$

Then by Lemma 18, $\Theta|\Delta \vdash \mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t) : A$. □

Lemma 18. *Lemma 16 holds with* `mitten`*'s* META/INTRO *rule.*

Proof. The only case we need to consider is metavariables, since other cases are proved by Lemma 16. Suppose:

$$\Theta|\Gamma \vdash_{\mathsf{type}} \alpha[\cdot.M_1 \dots M_n] : A[\cdot.M_1 \dots M_n] \mathbin{@} m$$
$$\Theta|\Gamma \vdash_{\mathsf{var}} \alpha[\cdot.M_1 \dots M_n] \mathbin{@} m$$

By Lemma 15, we have:

- $(\Gamma \vdash \alpha : A \mathbin{@} m) \in \Theta$
- $\Gamma = \{\mu_0\}.(x_1 :_{\nu_1} A_1).\{\mu_1\} \dots (x_n :_{\nu_n} A_n).\{\mu_n\}$
- $\forall 1 \leq i \leq n. \Theta|\Delta \vdash_{\mathsf{type}} M_i : A_i[\cdot.M_1 \dots M_{i-1}] \mathbin{@} \mathsf{Dom}(\nu_i)$

Hence we have $\Theta|\Gamma \vdash \alpha[\cdot.M_1 \dots M_n] : A[\cdot.M_1 \dots M_n] \mathbin{@} m$. □

We now define non-modal substitiutions in order to define $\mathsf{solve}(\alpha[\cdot.x_1 \dots x_n], t)$ for MTT. Many of these proofs are similar to the proofs from Section 2.4. Non-modal substitutions are written as $\overline{\theta}$ to distinguish them from substitutions.

Definition 11. A *non-modal substitution* $\Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Delta \mathbin{@} m$ is defined by the following rules:

$$\frac{}{\Theta|\Gamma \vdash_{\mathsf{type}} \cdot : \cdot \mathbin{@} m} \qquad \frac{\mu : m \to n \qquad \Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Delta \mathbin{@} n \qquad \Theta|\Gamma.\{\mu\} \vdash_{\mathsf{type}} M : A[\overline{\delta}] \mathbin{@} m}{\Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta}.M : \Delta.(\mu|A) \mathbin{@} m}$$

$$\frac{\mu, \nu : m \to n \qquad \Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Delta \mathbin{@} n}{\Theta|\Gamma.\{\mu\} \vdash_{\mathsf{type}} \overline{\delta} : \Delta.\{\nu\} \mathbin{@} m}$$

Definition 12. We define the action of a non-modal substitution $\overline{\gamma}$ on types and terms:

$$\begin{aligned} \mathbf{Nat}[\overline{\gamma}] &= \mathbf{Nat} & \mathbf{v}_k[\cdot.M_1 \dots M_n] &= M_k \\ \Pi_\mu(A, B)[\overline{\gamma}] &= \Pi_\mu(A[\overline{\gamma}], B[\overline{\gamma}.\mathbf{v}_0]) & (\lambda_{(\mu|A)} M) &= \lambda_{(\mu|A[\overline{\gamma}])}(M[\overline{\gamma}.\mathbf{v}_0]) \\ &\vdots & &\vdots \end{aligned}$$

Lemma 19. *Let* $\Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Delta$ *be a non-modal substitution. If* $\Theta|\Delta \vdash_{\mathsf{type}} A\,\mathsf{type}$ *then* $\Theta|\Gamma \vdash_{\mathsf{type}} A[\delta]\,\mathsf{type}$*, and if* $\Theta|\Delta \vdash_{\mathsf{type}} M : A$ *then* $\Theta|\Gamma \vdash_{\mathsf{type}} M[\delta] : A[\delta]$*.*

Proof. Induction on derivation $\Theta|\Delta \vdash_{\mathsf{type}} A\,\mathsf{type} \mathrel{@} m$ (resp. $\Theta|\Delta \vdash_{\mathsf{type}} M : A \mathrel{@} m$).

- $M = \mathbf{v}_k : A$: Then $\Delta = \Delta_1.(\mu|A').\Delta_2$ for $|\Delta_2| = k$ and $A = A'[\mathbf{p}^k]$. Since γ is a non-modal substitution, it can be expressed as $\gamma = \cdot.M_1 \dots M_n$ with $\Theta|\Gamma \vdash_{\mathsf{type}} M_k : A'[\cdot.M_1 \dots M_{n-k}]$. Since $A[\cdot.M_1 \dots M_n] = A'[\mathbf{p}^k][\cdot.M_1 \dots M_n] = A'[\cdot.M_1 \dots M_{n-k}]$ and $\mathbf{v}_k[\gamma] = M_k$, we have $\Theta|\Gamma \vdash_{\mathsf{type}} \mathbf{v}_k[\gamma] : A[\gamma]$.

□

Definition 13. A *renaming* is a non-modal substitution where the extension rule is restricted to variables.

Lemma 20. *Renamings and metasubstitutions commute.*

Definition 14. A *projection* is a substitution defined by the following rules:

$$\textsc{Empty}\quad \frac{}{\Theta|\Gamma \vdash \cdot : \cdot \mathrel{@} m}
\qquad
\textsc{Keep}\quad \frac{\mu : m \to n \qquad \Theta|\Gamma \vdash \delta : \Delta \mathrel{@} n \qquad \Theta|\Delta.\{\mu\} \vdash A\,\mathsf{type} \mathrel{@} m}{\Theta|\Gamma.(\mu|A[\delta]) \vdash \mathsf{keep}(\delta) \coloneqq (\delta \circ \mathbf{p}).\,\mathbf{v}_0 : \Delta.(\mu|A) \mathrel{@} n}$$

$$\textsc{Drop}\quad \frac{\mu : m \to n \qquad \Theta|\Gamma \vdash \delta : \Delta \mathrel{@} n \qquad \Theta|\Gamma.\{\mu\} \vdash A\,\mathsf{type} \mathrel{@} m}{\Theta|\Gamma.(\mu|A) \vdash \mathsf{drop}(\delta) \coloneqq \delta \circ \mathbf{p} : \Delta \mathrel{@} n}
\qquad
\textsc{Lock}\quad \frac{\mu : m \to n \qquad \Theta|\Gamma \vdash \delta : \Delta \mathrel{@} n}{\Theta|\Gamma.\{\mu\} \vdash \delta.\{\mu\} : \Delta.\{\mu\} \mathrel{@} m}$$

Definition 15. For projection $\Theta|\Gamma \vdash \delta : \Gamma \mathrel{@} m$, we define a non-modal substitution $\Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Gamma \mathrel{@} m$ by induction on γ:

$$\begin{aligned}
\overline{\cdot} &= \cdot \\
\overline{\mathsf{keep}(\delta)} &= \mathsf{weaken}(\overline{\delta}).\,\mathbf{v}_0 \\
\overline{\mathsf{drop}(\delta)} &= \mathsf{weaken}(\overline{\delta}) \\
\overline{\delta.\{\mu\}} &= \overline{\delta}
\end{aligned}$$

Where $\mathsf{weaken}(\cdot.M_1 \dots M_n) = \cdot.M_1[\mathbf{p}] \dots M_n[\mathbf{p}]$.

Lemma 21. *For projection* $\Theta|\Delta \vdash \gamma : \Gamma \mathrel{@} m$ *and type* $\Theta|\Gamma \vdash A\,\mathsf{type} \mathrel{@} m$ *(resp. term* $\Theta|\Gamma \vdash M : A \mathrel{@} m$*),* $A[\gamma] = A[\overline{\gamma}]$ *(resp.* $M[\gamma] = M[\overline{\gamma}]$*).*

Definition 16. The composition of non-modal substitutions $\Theta|\Gamma_2 \vdash_{\mathsf{type}} \overline{\gamma} : \Gamma_3 \mathrel{@} m$ and $\Theta|\Gamma_1 \vdash_{\mathsf{type}} \overline{\delta} : \Gamma_2 \mathrel{@} m$ is a non-modal substitution $\Theta|\Gamma_1 \vdash_{\mathsf{type}} \overline{\gamma} \circ \overline{\delta} : \Gamma_3 \mathrel{@} m$ and defined by:

$$\overline{\gamma} \circ \overline{\delta} \coloneqq \cdot.M_1[\overline{\delta}] \dots M_n[\overline{\delta}] \qquad \text{where} \quad \overline{\gamma} = \cdot.M_1 \dots M_n$$

Lemma 22. *For non-modal substitutions* $\Theta|\Gamma_2 \vdash_{\mathsf{type}} \overline{\gamma} : \Gamma_3 \mathrel{@} m$ *and* $\Theta|\Gamma_1 \vdash_{\mathsf{type}} \overline{\delta} : \Gamma_2 \mathrel{@} m$ *and type* $\Theta|\Gamma_3 \vdash_{\mathsf{type}} A\,\mathsf{type} \mathrel{@} m$ *or term* $\Theta|\Gamma_3 \vdash_{\mathsf{type}} M : A \mathrel{@} m$*:*

$$\begin{gathered}
\Theta|\Gamma_1 \vdash_{\mathsf{type}} A[\overline{\gamma} \circ \overline{\delta}] = A[\overline{\gamma}][\overline{\delta}]\,\mathsf{type} \mathrel{@} m \\
\Theta|\Gamma_1 \vdash_{\mathsf{type}} M[\overline{\gamma} \circ \overline{\delta}] = M[\overline{\gamma}][\overline{\delta}] : A[\overline{\gamma} \circ \overline{\delta}] \mathrel{@} m
\end{gathered}$$

Lemma 23. *Let* $\Theta|\Gamma \vdash_{\mathsf{type}} \overline{\delta} : \Delta$ *be a renaming and* $\Theta|\Delta \vdash_{\mathsf{type}} A\,\mathsf{type}$ *(resp.* $\Theta|\Delta \vdash_{\mathsf{type}} M : A$*). Then* $\mathsf{norm}(A)[\overline{\delta}] = \mathsf{norm}(A[\overline{\delta}])$ *(resp.* $\mathsf{norm}(M)[\overline{\delta}] = \mathsf{norm}(A[\overline{\delta}])$*).*

Lemma 24. *For any projection* $\Theta|\Gamma \vdash \delta : \Delta \text{ @ } m$*, renamings* $\Theta|\Delta \vdash_{\mathsf{type}} \overline{\gamma}_1, \overline{\gamma}_2 : \Delta' \text{ @ } m$*, types* $\Theta|\Delta \vdash_{\mathsf{type}} A, B\,\mathsf{type} \text{ @ } m$*, or terms* $\Theta|\Delta \vdash_{\mathsf{type}} M, N : A \text{ @ } m$*:*

$$\begin{aligned}
\overline{\gamma}_1 = \overline{\gamma}_2 &\iff \overline{\gamma}_1 \circ \overline{\delta} = \overline{\gamma}_2 \circ \overline{\delta} \\
A = B &\iff A[\delta] = B[\delta] \\
M = N &\iff M[\delta] = N[\delta]
\end{aligned}$$

Proof. First we show $\overline{\gamma}_1 = \overline{\gamma}_2 \iff \overline{\gamma}_1 \circ \overline{\delta} = \overline{\gamma}_2 \circ \overline{\delta}$. Suppose $\gamma_1 = \cdot.M_1 \ldots M_n$ and $\gamma_2 = \cdot.N_1 \ldots N_n$. Then

$$\begin{aligned}
\overline{\gamma}_1 = \overline{\gamma}_2 &\iff \forall 1 \le i \le n. M_i = N_i \\
&\iff \forall 1 \le i \le n. M_i[\delta] = N_i[\delta] \qquad \text{(IH)} \\
&\iff \forall 1 \le i \le n. M_i[\overline{\delta}] = N_i[\overline{\delta}] \\
&\iff \overline{\gamma}_1 \circ \overline{\delta} = \overline{\gamma}_2 \circ \overline{\delta}
\end{aligned}$$

WLoG we may assume A, B, M and N are in β-normal, η-long form. We show $A = B \iff A[\delta] = B[\delta]$ and $M = N \iff M[\delta] = N[\delta]$ by induction on A, B, M and N:

TODO ◻

Lemma 25. *Let* $\Theta|\Gamma \vdash \delta : \Delta \text{ @ } m$ *be a projection and* $\Theta|\Gamma \vdash t : A \text{ @ } m$ *be a term such that* $\mathsf{FV}(t) \subseteq \mathsf{vars}(\delta)$*. Then there is a unique type and term* $\Theta|\Delta \vdash t' : A' \text{ @ } m$ *(up to definitional equality) such that* $\Theta|\Gamma \vdash A'[\delta] = A\,\mathsf{type} \text{ @ } m$ *and* $\Theta|\Gamma \vdash t'[\delta] = t : A \text{ @ } m$*.*

Proof. Induction on the derivation $\Theta|\Gamma \vdash_{\mathsf{type}} t : A$. For non-variable cases see Lemma 8.

Consider the derivation

$$\frac{\Gamma = \Gamma_1.(\mu|A).\Gamma_2 \qquad |\Gamma_2| = k \qquad \exists\alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)}{\Theta|\Gamma \vdash \mathbf{v}_k : A[\mathbf{p}^{k+1}]}$$

$\mathsf{FV}(\mathbf{v}_k) = \{\mathbf{v}_k\}$, so $\mathbf{v}_k \in \mathsf{vars}(\delta)$.

We show $\forall\gamma, k.\ \mathbf{v}_k \in \mathsf{vars}(\gamma) \Rightarrow \exists\Delta_1, \Delta_2$ such that $\Delta = \Delta_1.(\mu|A').\Delta_2$ with $\mathsf{locks}(\Gamma_2) = \mathsf{locks}(\Delta_2)$ and $\mathbf{v}_{|\Delta_2|}[\delta] = \mathbf{v}_k$ by induction on δ and k:

- $\delta = \cdot$: then $\mathsf{vars}(\delta) = \emptyset$ but $\mathbf{v}_k \in \mathsf{vars}(\delta)$ so this case is impossible.
- $\delta = \mathsf{keep}(\delta')$: then $\Delta = \Delta'.(\nu|B)$ and $\Gamma = \Gamma_1.(\mu|A).\Gamma_2'.(\nu|B)$ for some projection $\Theta|\Gamma_1.(\mu|A).\Gamma_2' \vdash \delta' : \Delta'$.

  If $k = 0$ then $\mu = \nu$, so let $\Delta_1 = \Delta'$ and $\Delta_2 = \cdot$. Hence $\mathsf{locks}(\Delta_2) = \mathsf{locks}(\Gamma_2)$ and $\mathbf{v}_0[\mathsf{keep}(\delta')] = \mathbf{v}_0$.

  Otherwise $k = k' + 1$ so $\mathbf{v}_{k'+1} \in \mathsf{vars}(\delta) = \{\mathbf{v}_0\} \cup \{\mathbf{v}_{i+1} \,|\, \mathbf{v}_i \in \mathsf{vars}(\delta')\}$ so $\mathbf{v}_{k'} \in \mathsf{vars}(\delta')$. By IH, $\Delta' = \Delta_1.(\mu|A).\Delta_2'$ with $\mathsf{locks}(\Delta_2') = \mathsf{locks}(\Gamma_2')$ and $\mathbf{v}_{|\Delta_2'|}[\delta'] = \mathbf{v}_k$. Let $\Delta_2 = \Delta_2'.(\nu|B)$, then $\mathsf{locks}(\Delta_2) = \mathsf{locks}(\Delta_2') = \mathsf{locks}(\Gamma_2') = \mathsf{locks}(\Gamma_2)$ and $\mathbf{v}_{|\Delta_2|}[\mathsf{keep}(\delta')] = \mathbf{v}_{|\Delta_2'|}[\mathbf{p}] = \mathbf{v}_{k'}[\mathbf{p}] = \mathbf{v}_k$.
- $\delta = \mathsf{drop}(\delta')$: then $\Gamma = \Gamma_1.(\mu|A).\Gamma_2'.(\nu|B)$ for some projection $\Theta|\Gamma_1.(\mu|A).\Gamma_2' \vdash \delta' : \Delta$. $\mathbf{v}_k \in \mathsf{vars}(\delta) = \{\mathbf{v}_{i+1} \mid \mathbf{v}_i \in \mathsf{vars}(\delta')\}$ so $k = k' + 1$ for some $\mathbf{v}_{k'} \in \mathsf{vars}(\delta')$. So by IH, $\Delta = \Delta_1.(\mu|A').\Delta_2$ with $\mathbf{v}_{|\Delta_2|}[\delta'] = \mathbf{v}_{k'}$ and $\mathsf{locks}(\Delta_2) = \mathsf{locks}(\Gamma_2') = \mathsf{locks}(\Gamma_2)$. Hence $\mathbf{v}_{|\Delta_2|}[\delta] = \mathbf{v}_{k'}[\mathbf{p}] = \mathbf{v}_k$.

- $\delta = \delta'.\{\nu\}$: then $\Gamma = \Gamma_1.(\mu|A).\Gamma_2'.\{\nu\}$ and $\Delta = \Delta'.\{\nu\}$ for some projection $\Theta|\Gamma_1.(\mu|A).\Gamma_2' \vdash \delta' : \Delta'$. $\mathbf{v}_k \in \mathsf{vars}(\delta) = \mathsf{vars}(\delta')$ so by IH, $\Delta' = \Delta_1.(\mu|A').\Delta_2'$ and $\mathbf{v}_{|\Delta_2'|}[\delta'] = \mathbf{v}_k$. Let $\Delta_2 = \Delta_2'.\{\nu\}$, then $\mathbf{v}_{|\Delta_2|}[\delta] = \mathbf{v}_{|\Delta_2'|}[\delta'][\{\nu\}] = \mathbf{v}_k[\{\nu\}] = \mathbf{v}_k$ and $\mathsf{locks}(\Delta_2) = \mathsf{locks}(\Delta_2') \circ \nu = \mathsf{locks}(\Gamma_2') \circ \nu = \mathsf{locks}(\Gamma_2)$.

We now show uniqueness. Suppose there are two types or terms B and C such that $B[\delta] = C[\delta] = A$. Then by Lemma 24, $B = C$. □

Lemma 26. *Given metavariable $(\Delta \vdash \alpha : A) \in \Theta$ and unification problem*

$$\Theta|\Gamma \vdash \alpha[\overline{\gamma}] \overset{?}{=}_{\beta\eta} t : A[\overline{\gamma}]$$

that satisfies the pattern conditions. Then there is a non-modal solution that satisfies:

$$\Theta|\Delta \vdash_{\mathsf{type}} \mathsf{solve}(\alpha[\overline{\gamma}], t) : A$$
$$\Theta|\Gamma \vdash_{\mathsf{type}} \mathsf{solve}(\alpha[\overline{\gamma}], t)[\overline{\gamma}] = t : A[\overline{\gamma}]$$

Proof. Let $\overline{\gamma} = \cdot.x_1 \ldots x_n$. $\overline{\gamma}$ can be uniqely factored as $\overline{\gamma} = \overline{\iota} \circ \overline{\gamma}^p$ for renaming $\Theta|\Gamma \restriction \{x_1, \ldots, x_n\} \vdash_{\mathsf{type}} \overline{\iota} : \Delta$ and projection $\Theta|\Gamma \vdash \gamma^p : \Gamma \restriction \{x_1, \ldots, x_n\}$. By Lemma 25, there is a unique term $\Theta|\Gamma \restriction \{x_1, \ldots, x_n\} \vdash t' : A'$ such that $t = t'[\gamma^p]$. By Lemma 24, $\alpha[\overline{\gamma}] = t \iff \alpha[\overline{\iota}][\gamma^p] = t'[\gamma^p] \iff \alpha[\overline{\iota}] = t'$.

We can adapt the proof that ι is invertible in Lemma 11 to show that $\overline{\iota}$ is invertible.

By Lemma 19, we have $\Theta|\Delta \vdash_{\mathsf{type}} t'[\overline{\iota}^{-1}] : A$, so we define

$$\mathsf{solve}(\alpha[\overline{\gamma}], t) \coloneqq t'[\overline{\iota}^{-1}]$$

$$\begin{aligned}\mathsf{solve}(\alpha[\overline{\gamma}], t)[\overline{\gamma}] &= t'[\overline{\iota}^{-1}][\overline{\gamma}] \\ &= t'[\overline{\iota}^{-1} \circ \overline{\iota} \circ \overline{\gamma}^p] \\ &= t'[\gamma^p] \\ &= t\end{aligned}$$

□

Lemma 27. *Given metavariable $(\Delta \vdash \alpha : A \text{ @ } m) \in \Theta$ and unification problem*

$$\Theta|\Gamma \vdash \alpha[\overline{\gamma}] \overset{?}{=}_{\beta\eta} t : A[\overline{\gamma}] \text{ @ } m$$

that satisfies the pattern conditions. Suppose $\Delta \vdash_{\mathsf{var}} \mathsf{solve}(\alpha[\overline{\gamma}])$ @ m.
Then $\mathsf{solveMeta}_\Theta(\alpha, \mathsf{solve}(\alpha[\overline{\gamma}], t))$ is an MGU of $\alpha[\overline{\gamma}]$ and t.

Proof. First we show $\mathsf{solveMeta}_\Theta(\alpha, \mathsf{solve}(\alpha[\overline{\gamma}], t))$ is well-defined and is a unifier of $\alpha[\overline{\gamma}]$ and t. By Lemma 26, we have

$$\Theta|\Delta \vdash_{\mathsf{type}} \mathsf{solve}(\alpha[\overline{\gamma}], t) : A \text{ @ } m$$
$$\Theta|\Gamma \vdash_{\mathsf{type}} \mathsf{solve}(\alpha[\overline{\gamma}], t)[\overline{\gamma}] = t : A[\overline{\gamma}] \text{ @ } m$$

So by Lemma 13, $\Theta|\Gamma \vdash \mathsf{solve}(\alpha[\overline{\gamma}], t)[\overline{\gamma}] = t : A[\overline{\gamma}]$ @ m.

Define t', $\overline{\iota}$ and θ^p as in the proof for Lemma 26. By the pattern conditions, α doesn't occur in t, so $t[\theta] = t$. Since γ^p is a projection, α occurs in t iff α occurs in t', so α doesn't occur in t'. $\overline{\iota}$ is a renaming, so $\overline{\iota}^{-1}$ is a renaming, so α doesn't occur $t'[\overline{\iota}^{-1}] = \mathsf{solve}(\alpha[\overline{\gamma}], t)$.

Let $(\Theta', \theta) = \mathsf{solveMeta}_{\Theta}(\alpha, \mathsf{solve}(\alpha[\overline{\gamma}], t))$, then:

$$\begin{aligned} \Theta'|\Gamma \vdash \alpha[\overline{\gamma}][\theta] &= \alpha[\theta][\overline{\gamma}] \\ &= \mathsf{solve}(\alpha[\overline{\gamma}], t)[\overline{\gamma}] \\ &= t = t[\theta] : A[\overline{\gamma}] \end{aligned}$$

So (Θ', θ) is a unifier of $\alpha[\overline{\gamma}]$ and t.

TODO: show (Θ', θ) is the most general unifier □

3.4 Bidirectional type checking

Bidirectional type checking can now be easily extended to MTT. Each judgement is indexed by a mode, and some additional rules are added for modal types:

$$\frac{\mu : m \to n \qquad \Theta_1|\Gamma.\{\mu\} \vdash A\,\mathsf{type} \mathbin{@} m \rightsquigarrow (\Theta_2, \theta_1, A')}{\Theta_1|\Gamma \vdash \langle \mu \mid A \rangle\,\mathsf{type} \mathbin{@} n \rightsquigarrow (\Theta_2, \theta_1, \langle \mu \mid A' \rangle)}$$

$$\frac{\mu : m \to n \qquad \Theta_1|\Gamma.\{\mu\} \vdash M \Leftarrow A \mathbin{@} m \rightsquigarrow (\Theta_2, \theta_1, M')}{\Theta_1|\Gamma \vdash \mathsf{mod}_\mu(M) \Leftarrow \langle \mu \mid A \rangle \mathbin{@} n \rightsquigarrow (\Theta_2, \theta_1, \mathsf{mod}_\mu(M'))}$$

$$\frac{\mu : m \to n \qquad \Theta_1|\Gamma.\{\mu\} \vdash M \Rightarrow A \mathbin{@} m \rightsquigarrow (\Theta_2, \theta_1, M')}{\Theta_1|\Gamma \vdash \mathsf{mod}_\mu(M) \Rightarrow \langle \mu \mid A \rangle \mathbin{@} n \rightsquigarrow (\Theta_2, \theta_1, \mathsf{mod}_\mu(M'))}$$

Infer/Letmod

$$\frac{\begin{array}{c} \mu : n \to o \qquad \nu : m \to n \\ \Theta_1|\Gamma.\{\mu\} \vdash M \Rightarrow \langle \nu \mid A \rangle \mathbin{@} n \rightsquigarrow (\Theta_2, \theta_1, M') \\ \Theta_2|\Gamma[\theta_1].(y :_\mu \langle \nu \mid A \rangle) \vdash B\,\mathsf{type} \mathbin{@} o \rightsquigarrow (\Theta_3, \theta_2, B') \\ \Theta_3|\Gamma[\theta_{12}].(x :_{\mu \circ \nu} A[\theta_2]) \vdash N \Leftarrow B'[\mathsf{mod}_\nu(x)/y] \mathbin{@} o \rightsquigarrow (\Theta_4, \theta_3, N') \end{array}}{\begin{array}{c} \Theta_1|\Gamma \vdash \mathbf{let}_\mu\,\mathsf{mod}_\nu(x) \leftarrow M\,\mathbf{at}\,B\,\mathbf{in}\,N \Rightarrow B'[\theta_3][M'[\theta_{23}]/y] \mathbin{@} o \\ \rightsquigarrow (\Theta_4, \theta_{123}, \mathbf{let}_\mu\,\mathsf{mod}_\nu(\_) \leftarrow M'[\theta_{23}]\,\mathbf{at}\,B'[\theta_3]\,\mathbf{in}\,N') \end{array}}$$

The rule Infer/Letmod does the following:

1. Infer the type of scrutinee, and check it is a modal type. This is required to know what context to check the motive in. As in Infer/J from Section 2.5, if the infered type of the scrutinee is not a modal type, then it can be unified with the modal type $\langle \nu \mid \alpha \rangle$ for fresh metavariable α.
2. Check the motive is a type.
3. Check the body has the expected type (the motive instantiated with $\mathsf{mod}_\nu(x)$ for some x)
4. Output infered type of the motive instantiated with the scrutinee.

The Infer/Var rule has to check the variable is accessible.

Infer/Var

$$\frac{\mu : m \to n \qquad \Gamma = \Gamma_1, (x :_\mu A), \Gamma_2 \qquad x \notin \mathsf{Subjects}(\Gamma_2) \qquad \exists \alpha : \mu \Rightarrow \mathsf{locks}(\Gamma_2)}{\Theta|\Gamma \vdash x \Rightarrow A \mathbin{@} m \rightsquigarrow (\Theta, \mathbf{id}, \mathbf{v}_{|\Gamma_2|})}$$

In the core syntax there are a number of places that modality annotations are required including Π-types, applications and lambdas. When checking a λ against a Π-type, we can infer the modality from the type so we can add a form of λ to the pre-syntax that doesn't have a modality annotation, and similarly, if we infer the function in an application has a Π-type, we can infer the modality, so we can allow the user to omit the modality annotation. There are cases where modality annotations on λs and function applications are required, for example when infering the type of a λ, or when the infered type of a function is a metavariable. In the author's experience, modalities on λs and function applications can usually be infered, so long as the elaborator stays in 'checking' mode whenever possible, rather than switching to 'infering' mode.

3.5 Weak-head conversion checking

Abel et al. [1] define a conversion checking algorithm that uses explicit substitutions to evaluate terms to weak-head normal form. Gratzer [2] shows how to adapt this algorithm for MTT.

We define a deterministic weak-head reduction relation on types ($\Theta|\Gamma \vdash A \longrightarrow B\,\mathsf{type}$ @ m) and terms ($\Theta|\Gamma \vdash M \longrightarrow N : A$ @ m). These judgements presuppose $\Theta|\Gamma \vdash A\,\mathsf{type}$ @ m and $\Theta|\Gamma \vdash M : A$ @ m resepectively.

$$\frac{\mu : m \to n}{\Theta|\Gamma \vdash (\lambda_{(\mu|A)} N)(\mu|M) \longrightarrow N[\mathbf{1}.M] : B[\mathbf{1}.M] \ @\ n} \qquad \frac{\mu : m \to n \qquad \Theta|\Gamma \vdash M : \Pi(A, B) \ @\ n}{\Theta|\Gamma \vdash M(\mu|a) \longrightarrow N(\mu|a) : B[\mathbf{1}.a] \ @\ n}$$

$$\frac{}{\Theta|\Gamma\,\mathbf{let}_\mu\,\mathsf{mod}_\nu(\_) \leftarrow \mathsf{mod}_\nu(M)\,\mathbf{at}\,A\,\mathbf{in}\,N \longrightarrow N[\mathbf{1}.M] : A[\mathbf{1}.\mathsf{mod}_\nu(M)]}$$

$$\frac{\Theta|\Gamma \vdash M \longrightarrow M' : A}{\Theta|\Gamma \vdash \mathbf{let}_\mu\,\mathsf{mod}_\nu(\_) \leftarrow M\,\mathbf{at}\,B\,\mathbf{in}\,N \longrightarrow \mathbf{let}_\mu\,\mathsf{mod}_\nu(\_) \leftarrow M'\,\mathbf{at}\,B\,\mathbf{in}\,N : B[\mathbf{1}.M]}$$

A type or term written as $\overline{A}$ or $\overline{M}$ is in weak-head normal form. Generally, the type $\overline{A}$ denotes the weak-head normal form of A.

Lemma 28. *If* $\Theta|\Gamma \vdash M \longrightarrow N\,\mathsf{type}$ *then* $\Theta|\Gamma \vdash N\,\mathsf{type}$ *and* $\Theta|\Gamma \vdash M = N\,\mathsf{type}$*, and if* $\Theta|\Gamma \vdash M \longrightarrow N : A$ *then* $\Theta|\Gamma \vdash N : A$ *and* $\Theta|\Gamma \vdash M = N : A$*.*

Conversion checking computes the weak-head normal form, and if both weak-head normal forms use the same introduction form, it recurses on the arguments.

Definition 17. $\longrightarrow^*$ is the reflexive, transitive closure of $\longrightarrow$.

Definition 18. A type $\Theta|\Gamma \vdash M\,\mathsf{type}$ or term $\Theta|\Gamma \vdash M : A$ is in weak-head normal form when there doesn't exist N such that $\Theta|\Gamma \vdash M \longrightarrow N\,\mathsf{type}$ or $\Theta|\Gamma \vdash M \longrightarrow N : A$ respectively.

Lemma 29. *For any type* $\Theta|\Gamma \vdash M\,\mathsf{type}$ *or term* $\Theta|\Gamma \vdash M : A$*, there is a unique type or term* $\overline{M}$ *in weak-head normal form such that* $\Theta|\Gamma \vdash M \longrightarrow^* \overline{M}\,\mathsf{type}$ *or* $\Theta|\Gamma \vdash M \longrightarrow^* \overline{M} : A$ *respectively.*

For any type or term M, we denote the weak-head normal form of M by $\overline{M}$.

The conversion checking algorithm is defined using 6 judgements:

- $\Theta|\Gamma \vdash \overline{A} \Longleftrightarrow \overline{B}\,\mathsf{type}$ and $\Theta|\Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B\,\mathsf{type}$ which checks A and B are equal types,
- $\Theta|\Gamma \vdash \overline{M} \Longleftrightarrow \overline{N} : \overline{A}$ and $\Theta|\Gamma \vdash M \overset{\wedge}{\Longleftrightarrow} N : A$ which checks M and N are equal terms of type A, and

- $\Theta|\Gamma \vdash M \longleftrightarrow N : \overline{A}$ and $\Theta|\Gamma \vdash M \overset{\wedge}{\longleftrightarrow} N : A$ which checks neutral terms M and N are equal terms of type A.

The ‘hat’ version of each judgement computes the weak-head normal form of its arguments then calls the non-hat version:

$$\frac{\Theta|\Gamma \vdash A \longrightarrow \overline{A}\,\mathsf{type} \qquad \Theta|\Gamma \vdash B \longrightarrow \overline{B}\,\mathsf{type} \qquad \Theta|\Gamma \vdash \overline{A} \Longleftrightarrow \overline{B}\,\mathsf{type}}{\Theta|\Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B\,\mathsf{type}}$$

$$\frac{\Theta|\Gamma \vdash A \longrightarrow \overline{A}\,\mathsf{type} \qquad \Theta|\Gamma \vdash M \longrightarrow \overline{M} : A \qquad \Theta|\Gamma \vdash N \longrightarrow \overline{N} : A \qquad \Theta|\Gamma \vdash \overline{M} \Longleftrightarrow \overline{N} : \overline{A}}{\Theta|\Gamma \vdash M \overset{\wedge}{\Longleftrightarrow} N : A}$$

$$\frac{\Theta|\Gamma \vdash A \longrightarrow \overline{A}\,\mathsf{type} \qquad \Theta|\Gamma \vdash M \Longleftrightarrow N : \overline{A}}{\Theta|\Gamma \vdash M \overset{\wedge}{\longleftrightarrow} N : A}$$

TODO: finish this section

References

[1] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.*, 2(POPL), December 2017. doi: 10.1145/3158111. URL `https://doi.org/10.1145/3158111`.

[2] Daniel Gratzer. Weak-head conversion testing for mtt. Unpublished, 2024. URL `https://www.danielgratzer.com/papers/whct-for-mtt.pdf`.

[3] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. *Logical Methods in Computer Science*, Volume 17, Issue 3, July 2021. ISSN 1860-5974. doi: 10.46298/lmcs-17(3:11)2021. URL `http://dx.doi.org/10.46298/lmcs-17(3:11)2021`.

[4] András Kovács. Elaboration zoo, 2021. URL `https://github.com/AndrasKovacs/elaboration-zoo/`. Accessed: 2025-08-12.

[5] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 09 1991. ISSN 0955-792X. doi: 10.1093/logcom/1.4.497. URL `https://doi.org/10.1093/logcom/1.4.497`.

[6] Zoe Stafford. {mitten}: An elaborator for multimodal dependent type theory, 2025. URL `https://github.com/Z-snails/mitten_preorder/tree/zs-elaboration`.

[7] Philipp Stassen, Daniel Gratzer, and Lars Birkedal. {mitten}: A Flexible Multimodal Proof Assistant. In Delia Kesner and Pierre-Marie Pédrot, editors, *28th International Conference on Types for Proofs and Programs (TYPES 2022)*, volume 269 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:23, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-285-3. doi: 10.4230/LIPIcs.TYPES.2022.6. URL `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TYPES.2022.6`.