1 Setting the Scene

Thus far in the course, we have explored *closed term computation* and *open term calculation* for the simply typed lambda calculus (STLC) augmented with inductive types, with the goal to prove that all well-typed programs terminate as well as all well-typed terms normalize. We also have studied closed term computation for polymorphism, i.e., the type-level quantification. Our exploration led us to rediscover *Tait’s method* and *Girard’s method*.

From now on, we are going to study *equality*. Before developing *behavioral equality* (i.e., *exact equality*) in computational type theory, we first recall traditional *structural equality* (i.e., *definitional equality*) in formal type theory.

2 Structural Equality

Recall the STLC augmented with an answer type `ans` and a natural number type `nat`:

\[
A ::= ans \mid nat \mid A_1 \rightarrow A_2 \\
M ::= \uparrow \mid \downarrow \mid x \mid \lambda x : A.M \mid M_1 \ M_2 \mid z \mid s(M) \mid \text{rec}_A\{M_z ; x.M_s\}(M)
\]

The typing judgment is defined as follows:

\[
\Gamma \vdash \uparrow : ans \\
\Gamma, x : A \vdash x : A \\
\Gamma \vdash \lambda x : A_1.M : A_1 \rightarrow A_2 \\
\Gamma, x : A_1 \vdash M : A_2 \\
\Gamma, M_1 \vdash M_2 : A_1 \\
\Gamma \vdash M_1 \ M_2 : A_2 \\
\Gamma \vdash M : nat \\
\Gamma \vdash s(M) : nat \\
\Gamma, x : A \vdash M : A \\
\Gamma, x : A \vdash M_s : A \\
\Gamma \vdash \text{rec}_A\{M_z ; x.M_s\}(M) : A
\]

Structural equality for STLC, written \(\Gamma \vdash M \equiv M' : A\), is the strongest congruence respecting \(\beta\)-principles:

\[
\Gamma \vdash M : A \\
\Gamma \vdash M \equiv M' : A \\
\Gamma \vdash M \equiv M'' : A \\
\Gamma \vdash M \\
\Gamma \vdash M \equiv M' : A \\
\Gamma \vdash N \equiv N' : A_1 \\
\Gamma \vdash M \equiv M' : A_1 \rightarrow A_2 \\
\Gamma \vdash N \equiv N' : A_2 \\
\Gamma \vdash (\lambda x : A.M) \ N \equiv [N/x]M : A_2 \\
\Gamma, x : A \vdash M : A_1 \\
\Gamma \vdash s(M) : A_2 \\
\Gamma, x : A \vdash M_s : A_2 \\
\Gamma, x : A \vdash M_s \equiv M'_s : A_2 \\
\Gamma \vdash \text{rec}_A\{M_z ; x.M_s\}(M) \equiv \text{rec}_A\{M'_z ; x.M'_s\}(M') : A
\]
\[ \Gamma \vdash M : A \quad \Gamma, x : A \vdash M_s : A \]
\[ \Gamma \vdash \text{rec}_A(M_z : x.M_s)(z) \equiv M_z : A \]

\[ \Gamma \vdash M : \text{nat} \quad \Gamma \vdash M_z : A \quad \Gamma, x : A \vdash M_s : A \]
\[ \Gamma \vdash \text{rec}_A(M_z : x.M_s)(s(M)) \equiv |\text{rec}_A(M_z : x.M_s)(M)/x|M_s : A \]

Methodologically, structural equality is intended to be decidable. However, this approach never turns out to be efficient and the decidability could be very hard to prove.

### 3 Behavioral Equality

Under structural equality, one will not be able to prove \( x : \text{nat}, y : \text{nat} \vdash x + y \equiv y + x : \text{nat} \), where plus is defined by recursion on one of its arguments. In other words, structural equality is a very strong equality. Similar to the exploration of closed term computation, we want to develop a characterization of \textit{behavioral equality} in terms of program behaviors.

Intuitively, we want to “binarize logical relations”, written \( \Gamma \gg M \equiv M' \in A \), with the following fundamental theorem:

**Theorem 1** (FTLR). If \( \Gamma \vdash M \equiv M' : A \), then \( \Gamma \gg M \equiv M' \in A \).

Similar to \( \Gamma \gg M \in A \), \( \Gamma \gg M \equiv M' \in A \) is \textit{extensional} and \textit{functional}. In other words, suppose we have a behavioral equality relation for closed terms \( M \equiv M' \in A \) where \( M, M' \) are programs and \( A \) is a type, if \( \gamma = \gamma' \in \Gamma \) where \( \gamma, \gamma' \) are closed substitution mappings (i.e., for all \( x \in B \in \Gamma \), \( \gamma(x) = \gamma'(x) \in B \)), then \( \hat{\gamma}(M) \equiv \hat{\gamma'}(M') \in A \). Now we develop the logical relation \( \equiv \equiv M' \in A \), which can be seen as a binary version of hereditary termination relation.

**Definition 2.** \( M \equiv M' \in A \) is defined inductively on the structure of \( A \).

- \( M \equiv M' \in \text{ans} \) iff either \( M \downarrow \uparrow \) and \( M' \downarrow \downarrow \), or \( M \downarrow \downarrow \) and \( M' \downarrow \downarrow \).
- \( M \equiv M' \in \text{nat} \) iff either \( M \downarrow z \) and \( M' \downarrow z \), or \( M \downarrow s(N), M' \downarrow s(N') \), and \( N \equiv N' \in \text{nat} \) (following the horizontal induction principle).
- \( M \equiv M' \in A_1 \rightarrow A_2 \) iff \( M \downarrow \lambda x : A_1.M_1, M' \downarrow \lambda x : A_1.M'_1 \), and for all \( N, N' \) such that \( N \equiv N' \in A_1, [N/x]M_1 \equiv [N'/x]M'_1 \in A_2 \).

The first property we want to establish is that \( M \equiv M' \in A \) is symmetric and transitive.

**Lemma 3.** \textit{For any type} \( A \),

- \( M \equiv M' \in A \) \textit{implies} \( M' \equiv M \in A \).
- \( M \equiv M' \in A \) \textit{and} \( M' \equiv M'' \in A \) \textit{imply} \( M \equiv M'' \in A \).

**Proof.** By induction on the structure of \( A \):

- **Case** \( A = \text{ans} \):
  - Symmetry: By inversion we know that either \( M \downarrow \uparrow, M' \downarrow \uparrow \), or \( M \downarrow \downarrow, M' \downarrow \downarrow \). In either case it is straightforward to show \( M' \equiv M \in \text{ans} \).
  - Transitivity: By inversion on \( M \equiv M' \in \text{ans} \) we know that either \( M \downarrow \uparrow, M' \downarrow \uparrow \), or \( M \downarrow \downarrow, M' \downarrow \downarrow \). If \( M, M' \) evaluates to \( \uparrow \), then by the determinism of evaluation and inversion on \( M' \equiv M'' \in \text{ans} \), we know that \( M'' \) also evaluates to \( \uparrow \), hence by definition we know \( M \equiv M'' \in \text{ans} \). It is similar to prove for the case where \( M, M' \) evaluates to \( \downarrow \).
• Case $A = \text{nat}$:
  
  - Symmetry: By inversion and then a case analysis:
    * Subcase $M \Downarrow z, M' \Downarrow z$: By definition we know that $M' \Downarrow M \in \text{nat}$.
    * Subcase $M \Downarrow s(N), M' \Downarrow s(N'), N \Downarrow N' \in \text{nat}$: By induction hypothesis on $N \Downarrow N' \in \text{nat}$ we know that $N' \Downarrow N \in \text{nat}$. Thus by definition we know that $M' \Downarrow M \in \text{nat}$.
  
  - Transitivity: By inversion on $M \Downarrow M' \in \text{nat}$ and then a case analysis:
    * Subcase $M \Downarrow z, M' \Downarrow z$: By the determinism of evaluation and inversion on $M' \Downarrow M'' \in \text{nat}$, we know that $M''$ also evaluates to $z$. By definition we have $M \Downarrow M'' \in \text{nat}$.
    * Subcase $M \Downarrow s(N), M' \Downarrow s(N'), N \Downarrow N' \in \text{nat}$: By the determinism of evaluation and inversion on $M \Downarrow M'' \in \text{nat}$, we know that $M''$ evaluates to $s(N'')$ for some $N''$ and $N' \Downarrow N'' \in \text{nat}$. Then by induction hypothesis we know that $N \Downarrow N'' \in \text{nat}$. By definition we have $M \Downarrow M'' \in \text{nat}$.

• Case $A = A_1 \rightarrow A_2$:
  
  - Symmetry: By inversion we know that $M \Downarrow \lambda x : A_1, M_1, M' \Downarrow \lambda x : A_1, M_1'$ and for all $N, N'$ such that $N \Downarrow N' \in A_1$, $[N/x]M_1 = [N'/x]M_1' \in A_2$. We are supposed to show that for all $M', N$ such that $N \Downarrow N' \in A_1$, $[N'/x]M_1 \Downarrow [N'/x]M_1' \in A_2$. By induction hypothesis, we have $N \Downarrow N' \in A_1$, and it suffices to show that $[N/x]M_1 \Downarrow [N'/x]M_1' \in A_2$. It follows directly from assumptions.
  
  - Transitivity: By inversion on $M \Downarrow M' \in A_1 \rightarrow A_2, M' \Downarrow M'' \in A_1 \rightarrow A_2$, and determinism of evaluation, we know that $M \Downarrow \lambda x : A_1, M_1, M' \Downarrow \lambda x : A_1, M_1'$, $M'' \Downarrow \lambda x : A_1, M_1''$, for all $N, N'$ such that $N \Downarrow N' \in A_1, [N/x]M_1 = [N'/x]M_1' \in A_2$, and for all $N', N''$ such that $N' \Downarrow N'' \in A_1, [N'/x]M_1' = [N''/x]M_1'' \in A_2$. We are supposed to show that for all $N, N'$ such that $N \Downarrow N'' \in A_1, [N'/x]M_1' = [N''/x]M_1'' \in A_2$. By assumption, we know that $[N'/x]M_1' \Downarrow [N''/x]M_1'' \in A_2$. Then it suffices to show that $[N''/x]M_1'' \Downarrow [N''/x]M_1'' \in A_2$. It turns out that we need to prove $N'' \Downarrow N'' \in A_1$. By induction hypothesis on $N \Downarrow N'' \in A_1$ for symmetry, we know that $N'' \Downarrow N \in A_1$. Then by induction hypothesis for transitivity, we have $N'' \Downarrow N'' \in A_1$. Thus we conclude the proof.

The symmetry and transitivity can also be proved for open behavioral equality.

Lemma 4. For any typing context $\Gamma$ and type $A$,

- $\Gamma \triangleright M \Downarrow M' \in A$ implies $\Gamma \triangleright M' \Downarrow M \in A$.
- $\Gamma \triangleright M \Downarrow M' \in A$ and $\Gamma \triangleright M' \Downarrow M'' \in A$ imply $\Gamma \triangleright M \Downarrow M'' \in A$.

Proof. 
  
  Suppose $\gamma \Downarrow \gamma' \in \Gamma$. It suffices to show $\widehat{\gamma}(M') \Downarrow \widehat{\gamma}(M) \in A$. By symmetry we know that $\gamma' \Downarrow \gamma \in \Gamma$. By assumption we know that $\widehat{\gamma}(M) \Downarrow \widehat{\gamma}(M') \in A$. By symmetry again we conclude that $\widehat{\gamma}(M') \Downarrow \widehat{\gamma}(M) \in A$.

  Suppose $\gamma \Downarrow \gamma'' \in \Gamma$. It suffices to show $\widehat{\gamma}(M) \Downarrow \widehat{\gamma''}(M'') \in A$. By symmetry we know that $\gamma'' \Downarrow \gamma \in \Gamma$. By transitivity we know that $\gamma \Downarrow \gamma \in \Gamma$. By assumption we know that $\widehat{\gamma}(M') \Downarrow \widehat{\gamma''}(M'') \in A$. By assumption we know that $\widehat{\gamma}(M') \Downarrow \widehat{\gamma''}(M'') \in A$. By transitivity we conclude that $\widehat{\gamma}(M) \Downarrow \widehat{\gamma''}(M'') \in A$. 

\qed
Because behavioral equality reasons about behavioral equivalence of programs, and the evaluation of the STLC is deterministic, the equality relation should also be closed under reverse execution.

**Lemma 5** (Head expansion). If \( M \equiv M' \in A \), \( N \mapsto^* M \), and \( N' \mapsto^* M' \), then \( N \equiv N' \in A \).

**Proof.** By induction on the structure of \( A \):

- **Case \( A = \text{ans} \):** By inversion we know that either \( M \downarrow \uparrow \), \( M' \downarrow \uparrow \), or \( M \downarrow \downarrow \), \( M' \downarrow \downarrow \).
  
  If \( M, M' \) evaluates to \( \uparrow \), then \( N \mapsto^* M \) and \( N' \mapsto^* M' \) imply that \( N \downarrow \uparrow \) and \( N' \downarrow \uparrow \), thus \( N \equiv N' \in \text{ans} \). It is similar to prove for the case where \( M, M' \) evaluates to \( \downarrow \).

- **Case \( A = \text{nat} \):** By inversion and a then a case analysis:
  
  - Subcase \( M \downarrow z, M' \downarrow z \): \( N \mapsto^* M \) and \( N' \mapsto^* M' \) imply that \( N \downarrow z \) and \( N' \downarrow z \), thus \( N \equiv N' \in \text{nat} \).
  
  - Subcase \( M \downarrow s(M_1), M' \downarrow s(M'_1), M_1 \equiv M'_1 \in \text{nat} \): \( N \mapsto^* M \) and \( N' \mapsto^* M' \) imply that \( N \downarrow s(M_1) \) and \( N' \downarrow s(M'_1) \), thus by definition we have \( N \equiv N' \in \text{nat} \).

- **Case \( A = A_1 \rightarrow A_2 \):** By inversion we know that \( M \downarrow \lambda x : A_1.M_1, M' \downarrow \lambda x : A_1.M'_1 \), and for all \( M_2, M'_2 \) such that \( M_2 \equiv M'_2 \in A_1 \), \([M_2/x]M_1 \equiv [M'_2/x]M'_1 \in A_2 \). \( N \mapsto^* M \) and \( N' \mapsto^* M' \) imply that \( N \downarrow \lambda x : A_1.M_1 \) and \( N' \downarrow \lambda x : A_1.M'_1 \). By definition we know that \( N \equiv N' \in A_1 \rightarrow A_2 \).

Now we turn to prove the fundamental theorem to show that structural equality suffices for behavioral equality. We start with a lemma which justifies the positive definition of behavioral equality for function types.

**Lemma 6.** \( M \equiv M' \in A_1 \rightarrow A_2 \) and \( N \equiv N' \in A_1 \) imply \( M N \equiv M' N' \in A_2 \).

**Proof.** By inversion on \( M \equiv M' \in A_1 \rightarrow A_2 \) we know that \( M \downarrow \lambda x : A_1.M_1, M' \downarrow \lambda x : A_1.M'_1 \), and for all \( N, N' \) such that \( N \equiv N' \in A_1 \), \([N/x]M_1 \equiv [N'/x]M'_1 \in A_2 \). Thus we know that \([N/x]M_1 \equiv [N'/x]M'_1 \in A_2 \). Observe that \( M N \mapsto^* (\lambda x : A_1.M_1) N \mapsto [N/x]M_1 \), and \( M' N' \mapsto^* (\lambda x : A_1.M'_1) N' \mapsto [N'/x]M'_1 \). Thus by head expansion we conclude that \( M N \equiv M' N' \in A_2 \).

We show that well-typed terms are behaviorally equal to itself for its type.

**Lemma 7.** \( \Gamma \vdash M : A \) implies \( \Gamma \gg M \equiv M \in A \).

**Proof.** We are supposed to show for all \( \gamma, \gamma' \) such that \( \gamma \equiv \gamma' \in \Gamma \), \( \hat{\gamma}(M) \equiv \hat{\gamma'}(M) \in A \). The proof proceeds by induction on the derivation of \( \Gamma \vdash M : A \). We consider several nontrivial cases.

- \( \Gamma, x : A_1 \vdash M : A_2 \)

  \[
  \Gamma \vdash \lambda x : A_1.M : A_1 \rightarrow A_2
  \]

  Observe that \( \hat{\gamma}(\lambda x : A_1.M) = \lambda x : A_1.\hat{\gamma}(M) \). By definition we are supposed to show that for all \( N, N' \) such that \( N \equiv N' \in A_1 \), \([N/x]\hat{\gamma}(M) \equiv [N'/x]\hat{\gamma}(M) \in A_2 \). Let \( \gamma_x = \gamma[x \mapsto N] \) and \( \gamma'_x = \gamma'[x \mapsto N'] \). By assumption we know that \( \gamma_x \equiv \gamma'_x \in \Gamma, x : A_1 \).

  Thus by induction hypothesis we know that \( \hat{\gamma}_x(M) \equiv \hat{\gamma}'_x(M) \in A_2 \). Observe that \( \hat{\gamma}_x(M) = [N/x]\hat{\gamma}(M) \) and \( \hat{\gamma}'_x(M) = [N'/x]\hat{\gamma}(M) \). Thus we conclude this case.
Theorem 8 (FTLR). If $\Gamma \vdash M \equiv M' : A$, then $\Gamma \gg M \equiv M' \in A$.

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash M \equiv M' : A$. We consider several nontrivial cases.

Now we proceed to prove the fundamental theorem.
\[
\begin{align*}
\Gamma \vdash M : A \\
\Gamma \vdash M \equiv M : A
\end{align*}
\]
We conclude this case by Lemma \[7\].

\[
\begin{align*}
\Gamma \vdash M' \equiv M : A \\
\Gamma \vdash M \equiv M' : A
\end{align*}
\]
We conclude this case by induction hypothesis and then Lemma \[4\].

\[
\begin{align*}
\Gamma \vdash M \equiv M' : A \\
\Gamma \vdash M' \equiv M'' : A
\end{align*}
\]
We conclude this case by induction hypothesis and then Lemma \[4\].

\[
\Gamma, x : A_1 \vdash M : A_2 \quad \Gamma \vdash N : A_1
\]
\[
\Gamma \vdash (\lambda x : A_1.M) N \equiv [N/x]M : A_2
\]
Suppose \(\gamma \vdash \gamma' \in \Gamma\). It suffices to show that \((\lambda x : A_1.\tilde{\gamma}(M)) \tilde{\gamma}(N) \equiv [\tilde{\gamma}(N)/x]\tilde{\gamma'}(M) \in A_2\). By Lemma \[7\] on \(\Gamma \vdash N : A_1\) we know that \(\tilde{\gamma}(N) \equiv \tilde{\gamma}'(N) \in A_1\). Let \(\gamma_x = \gamma[x \mapsto \tilde{\gamma}(N)]\) and \(\gamma'_x = \gamma'[x \mapsto \tilde{\gamma}'(N)]\). By assumption we know that \(\gamma_x \equiv \gamma'_x \in \Gamma, x : A_1\).

By Lemma \[7\] on \(\Gamma, x : A_1 \vdash M : A_2\) we know that \(\tilde{\gamma}_{\tilde{x}}(M) \equiv \tilde{\gamma}'_{\tilde{x}}(M) \in A_2\). Thus \([\tilde{\gamma}(N)/x]\tilde{\gamma}(M) \equiv [\tilde{\gamma}(N)/x]\tilde{\gamma}(M) \in A_2\). Observe that

\[
(\lambda x : A_1.\tilde{\gamma}(M)) \tilde{\gamma}(N) \mapsto [\tilde{\gamma}(N)/x]\tilde{\gamma}(M).
\]

Thus by head expansion we conclude this case.

\[
\begin{align*}
\Gamma \vdash M \equiv M' : A_1 \rightarrow A_2 \quad \Gamma \vdash N \equiv N' : A_1
\end{align*}
\]
\[
\Gamma \vdash M N \equiv M' N' : A_2
\]
We conclude this case by induction hypothesis and then Lemma \[6\].

\[
\begin{align*}
\Gamma \vdash \lambda x : A_1.M \equiv \lambda x : A_1.M' : A_1 \rightarrow A_2
\end{align*}
\]
Suppose \(\gamma \vdash \gamma' \in \Gamma\). It suffices to show that \(\lambda x : A_1.\tilde{\gamma}(M) \equiv \lambda x : A_1.\tilde{\gamma}'(M) \in A_2\). By definition we are supposed to show that for all \(N, N'\) such that \(N \equiv N' \in A_1\), \([N/x]\tilde{\gamma}(M) \equiv [N'/x]\tilde{\gamma}'(M) \in A_2\). Let \(\gamma_x = \gamma[x \mapsto N]\) and \(\gamma'_x = \gamma'[x \mapsto N']\).

By assumption we know that \(\gamma_x \equiv \gamma'_x \in \Gamma, x : A_1\). Thus by induction hypothesis we know that \(\tilde{\gamma}_{\tilde{x}}(M) \equiv \tilde{\gamma}'_{\tilde{x}}(M) \in A_2\). Observe that \(\tilde{\gamma}_{\tilde{x}}(M) = [N/x]\tilde{\gamma}(M)\) and \(\tilde{\gamma}'_{\tilde{x}}(M) = [N'/x]\tilde{\gamma}(M)\). Hence we conclude this case.

\[
\begin{align*}
\Gamma \vdash M \equiv M' : \text{nat} \quad \Gamma \vdash M_2 \equiv M' : A \quad \Gamma, x : A \vdash M_s \equiv M'_s : A
\end{align*}
\]
\[
\Gamma \vdash \text{rec}_A\{M_s ; x. M_s\}(M) \equiv \text{rec}_A\{M'_s ; x. M'_s\}(M') : A
\]
Suppose \(\gamma \vdash \gamma' \in \Gamma\). By induction hypothesis on \(\Gamma \vdash M \equiv M' : \text{nat}\) we know that \(\tilde{\gamma}(M) \equiv \tilde{\gamma}'(M) \in \text{nat}\). By horizontal induction on natural numbers:

- Subcase \(\tilde{\gamma}(M) \downarrow z, \tilde{\gamma}'(M') \downarrow z\): Observe that

\[
\begin{align*}
\text{rec}_A\{\tilde{\gamma}(M_s) ; x. \tilde{\gamma}(M_s)\}(\tilde{\gamma}(M)) & \rightarrow^* \tilde{\gamma}(M_2) \\
\text{rec}_A\{\tilde{\gamma}'(M'_s) ; x. \tilde{\gamma}'(M'_s)\}(\tilde{\gamma}'(M')) & \rightarrow^* \tilde{\gamma}'(M'_2)
\end{align*}
\]

By induction hypothesis on \(\Gamma \vdash M_2 \equiv M'_s : A\) we know that \(\tilde{\gamma}(M_2) \equiv \tilde{\gamma}'(M'_2) \in A\).

Thus by head expansion we conclude this subcase.

- Subcase \(\tilde{\gamma}(M) \downarrow s(N), \tilde{\gamma}'(M') \downarrow s(N'), N \equiv N' \in \text{nat}\) assuming that

\[
\begin{align*}
\text{rec}_A\{\tilde{\gamma}(M_s) ; x. \tilde{\gamma}(M_s)\}(N) \equiv \text{rec}_A\{\tilde{\gamma}'(M'_s) ; x. \tilde{\gamma}'(M'_s)\}(N') \in A:
\end{align*}
\]
Observe that
\[ \text{rec}_A \{ \gamma(M_z) \} \rightarrow^* \text{rec}_A \{ \gamma(M_z) \} \]
\[ \text{rec}_A \{ \gamma(M'_z) \} \rightarrow^* \text{rec}_A \{ \gamma(M'_z) \} . \]

Let
\[ \gamma_x = \gamma[x \mapsto \text{rec}_A \{ \gamma(M_z) \}] \]
and
\[ \gamma'_x = \gamma'[x \mapsto \text{rec}_A \{ \gamma(M'_z) \}] . \]

By assumption we know that \( \gamma_x \models \gamma'_x \in \Gamma, x : A \). Thus by induction hypothesis on \( \Gamma, x : A \vdash M_x \equiv M'_x : A \) we know that \( \gamma_x(M) \equiv \gamma'_x(M'_x) \in A \). Observe that \( \gamma_x(M_z) = \text{rec}_A \{ \gamma(M_z) \} \) and \( \gamma'_x(M'_z) = \text{rec}_A \{ \gamma(M'_z) \} \). Hence by head expansion we conclude this subcase.

\[ \Gamma \vdash M : \text{nat} \quad \Gamma \vdash M_z : A \quad \Gamma, x : A \vdash M_x : A \]

Suppose \( \gamma \equiv \gamma' \in \Gamma \). It suffices to show that \( \text{rec}_A \{ \gamma(M_z) \} \equiv \text{rec}_A \{ \gamma(M'_z) \} \in A \). By assumption we can derive \( \Gamma \vdash \text{rec}_A \{ M_z : x.M_z \} \) : \( A \). Thus by Lemma 7 we know that \( \Gamma \vdash \text{rec}_A \{ M_z : x.M_z \} \equiv \text{rec}_A \{ M'_z : x.M'_z \} \) : \( A \).

Hence \( \text{rec}_A \{ \gamma(M_z) \} \equiv \text{rec}_A \{ \gamma(M'_z) \} \) : \( A \). Let \( \gamma_x = \gamma[x \mapsto \text{rec}_A \{ \gamma(M_z) \}] \) and \( \gamma'_x = \gamma'[x \mapsto \text{rec}_A \{ \gamma(M'_z) \}] \). By assumption we know that \( \gamma_x \models \gamma'_x \in \Gamma, x : A \). By Lemma 7 on \( \Gamma, x : A \vdash M_x : A \) we know that \( \gamma_x(M_z) \equiv \gamma'_x(M'_z) \in A \).

Observe that
\[ \text{rec}_A \{ \gamma(M_z) \} \equiv \text{rec}_A \{ \gamma(M'_z) \} \]
\[ \text{rec}_A \{ \gamma(M'_z) \} \equiv \text{rec}_A \{ \gamma(M_z) \} \]
\[ \text{rec}_A \{ \gamma(M'_z) \} \equiv \text{rec}_A \{ \gamma(M_z) \} \]

Hence by head expansion we conclude this subcase.

\[ \square \]

4 Discussion

Suppose we want to prove \( x : \text{nat} \vdash M \equiv M' \in A \) in a proof assistant such as RedPRL. In other words, we want to show that if \( N \equiv N' \in \text{nat} \), then \( [N/x]M \equiv [N'/x]M' \in A \). By horizontal induction on natural numbers, it suffices to show

1. If \( N \vdash z, N' \vdash z \), then \( [N/x]M \equiv [N'/x]M' \in A \).
2. If \( N \vdash s(P), N' \vdash s(P') \), \( P \equiv P' \in \text{nat} \), and \( [P/x]M \equiv [P'/x]M' \in A \), then \( [N/x]M \equiv [N'/x]M' \in A \).

Intuitively, the second condition can be expressed using the following “judgment”:
\[ [P/x]M \equiv [P'/x]M' \in A \]
\[ \vdash_{p \in \text{nat}} [s(P)/x]M \equiv [s(P')/x]M' \in A \]

That is, we can derive a proof of equality from a proof of equality. We can even rewrite it as follows informally:
\[ P \in \text{nat}, P' \in \text{nat}, y : \text{Eq}_{\text{nat}}(P, P'), z : \text{Eq}_A([P/x]M, [P'/x]M') \vdash \cdots \in \text{Eq}_A([s(P)/x]M, [s(P')/x]M') \]

\( \text{Eq}_A(M, M') \) can be seen as an “equality type” whose inhabitants are proofs for equality of \( M \) and \( M' \) for type \( A \). This fact motivates the “propositions-as-types” principle, which leads to the exploration of dependent type theory.