Computational Higher Type Theory (CHTT)

Robert Harper

Lecture Notes of Week 4 by Yue Niu and Charles Yuan

Thus far in the course, we have covered a typed lambda calculus augmented with inductively defined positive types such as natural numbers, booleans, products, and sums, as well as how they may be formulated in a negative fashion. Instead of delving deeper into negatively defined coinductive types, we will now examine quantification.

1 Quantification

Type-level quantification, as seen in System F and its variants, allow us to write types and expressions containing type variables, which assert validity over all types α, or over some type α.

$$\forall X.A \qquad \exists X.A$$

The theory of type polymorphism was largely developed by Gordon, Milner [1978], and Wadsworth, and has been adopted in languages such as ML. It enables innovations such as the polymorphic identity function:

$$\texttt{id} : \forall X.X \to X$$

To begin, consider the following syntax for types, incorporating variables, a positive answer type ans, functions, and a universal type quantifier.

$$A ::= X \mid \mathsf{ans} \mid A_1 \to A_2 \mid \forall X.A$$

The expressions are defined correspondingly, with a mechanism for type lambdas:

$$M ::= x \mid M_1\, M_2 \mid \lambda x{:}A.\, M \mid \Lambda X.\, M \mid M\,[A] \mid \uparrow \mid \downarrow$$

The typing judgment is defined as follows:

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \vdash \uparrow : \mathsf{ans}} \qquad \frac{}{\Gamma \vdash \downarrow : \mathsf{ans}} \qquad \frac{\Gamma, x : A_1 \vdash A_2 : M_2}{\Gamma \vdash \lambda x{:}A_1.\, M_2 : A_1 \to A_2}$$

$$\frac{\Gamma \vdash M_1 : A_1 \to A_2 \qquad \Gamma \vdash M_2 : A_1}{\Gamma \vdash M_1\, M_2 : A_2} \qquad \frac{\Gamma, X\ \mathsf{type} \vdash M : A}{\Gamma \vdash \Lambda X.\, M : \forall X.A}$$

$$\frac{\Gamma \vdash M : \forall X.A \qquad \Gamma \vdash B\ \mathsf{type}}{\Gamma \vdash M\,[B] : [B/X]A}$$

1.1 Context Splitting

Now that both expressions and types are allowed to vary, we split the judgment context into two components: Θ, a finite set of type variables, and Γ, a finite set of variables as before. We use both contexts in a judgment, as:

$$\Theta, \Gamma \vdash M : A$$

1.2 Erasure

So far, we have expressed the claim that

$$M : A \text{ implies } M\ \mathsf{term}_\beta$$

that is,

$$(\Gamma \vdash M : A \text{ and } \mathsf{HT}_\Gamma(\gamma)) \text{ imply } \mathsf{HT}_A(\hat{\gamma}(M))$$

Now that we have type lambdas and variables, we recognize that semantics about expressions should not need to specially account for type lambda abstraction and application. We should instead *erase* this polymorphic type information before reasoning about hereditary termination, etc. There are two logical approaches to performing erasure:

1. Full erasure. Here, we simply elide all type lambda application and abstraction from the expression. We will use the notation $|M|^{\mathrm{I}}$ for this, i.e.:

$$\begin{aligned} &\dots \\ |\Lambda X.\, M|^{\mathrm{I}} &= |M|^{\mathrm{I}} \\ |M\,[A]|^{\mathrm{I}} &= |M|^{\mathrm{I}} \end{aligned}$$

   where every other erasure rule simply erases each component of M in the expected fashion.

2. Delayed erasure. Here, we replace type lambdas with classical lambdas and type applications with classical applications. The expression argument is a dummy, and given a choice for its type we arbitarily select the answer type (and $\uparrow$ as its value).

$$\begin{aligned} &\dots \\ |\Lambda X.\, M|^{\mathrm{II}} &= \lambda\_.\, |M|^{\mathrm{II}} \\ |M\,[A]|^{\mathrm{II}} &= |M|^{\mathrm{II}}\,(\uparrow) \end{aligned}$$

Note that in full erasure, it is possible for a value (a type lambda) to become a non-value after erasure (the body of the lambda), whereas this is impossible in delayed erasure. Allowing values to become non-values has negative consequences later, and can make the result unsound. From now we will use the notation for erasure, $|M|$, to refer to delayed erasure $|M|^{\mathrm{II}}$.

2 Termination

Our goal is to prove, for the language augmented with quantifiers:

$$M : A \implies |M|\ \mathsf{term}_\beta$$

Let us begin with the obvious attempt. We will use hereditary termination, $\mathsf{HT}_A(M)$, taking erasure into account.

Proof Attempt.

- $\mathsf{HT}_X(M)$. We're not sure how to proceed here yet. Moving on for a moment...
- $\mathsf{HT}_{\mathsf{ans}}(M) \iff M \mapsto_\beta^* \uparrow$ or $M \mapsto_\beta^* \downarrow$. This is straightforward.

- $\mathsf{HT}_{A_1 \to A_2}(M) \iff \mathsf{HT}_{A_1}(M_1) \implies \mathsf{HT}_{A_2}(M\,M_2)$. We've seen this before.
- $\mathsf{HT}_{\forall X.A}(M) \iff$ for all closed B, we have $\mathsf{HT}_{[B/X]A}(M\,[B])$.

  Let's think about this case. First, observe that $M\,[B]$ should actually be written as $M(\downarrow)$, due to erasure. Next, there is a central issue! The substitution of B into A may result in a type that is structurally larger than $\forall X.A$. For example, let $B = (\forall X.X \to A) \to (\forall X.X \to X)$, which clearly will not yield a result that is smaller than A. This scheme is *not inductive*, and it fails.

X

At this point, we may be frustrated enough to try a desperate move: defining some alternative measure of size that means the substitution may not become larger than A, and would allow us to move forward with the previous attempt. This would be difficult, as we would not know where to start in defining such a measure, and it likely does not exist at all.

2.1 Next Attempt

Having been thwarted by the substitution, let us try to factor it out. Define

$$\mathsf{HT}_A(M)[\delta : \Theta]$$

as the hereditary termination of M at A with a *closing type substitution* δ with respect to Θ. Now A can be open (which also allows us to deal with the first case from earlier). In particular, $\Theta \vdash A\ \mathsf{type}$. We restate the problematic cases from before:

- $\mathsf{HT}_X(M)[\delta : \Theta] \iff \mathsf{HT}_{\delta(X)}(M)$.

  This relates the new definition to the old one for type variables, given some closing substitution.
- $\mathsf{HT}_{\forall X.A}(M)[\delta : \Theta] \iff$ for all closed B, we have $\mathsf{HT}_A(M(\downarrow))[\delta[X \mapsto B] : \Theta, X]$.

  Now, we allow X to be free in A, and record the substitution and X into the context. Surely the proof will go through this time?

As pleased as we are that we have reorganized the proof, have we really done anything? B is still a menace as before, and even if we first allow X to be free, eventually we must attempt to close over it, at which point B resurfaces. We must find some way of truly ridding ourselves of it.

2.2 A Step Further

Here is an idea: instead of dealing with B directly, define some *interpretation* of a type variable. Namely, try:

$$\mathsf{HT}_A(M)[\delta; \eta : \Theta]$$

where η is an interpretation, or a set of predicates, of variables, given δ, and $\eta(X)$ is an interpretation of X. From here on, when Θ is clear from context we will leave it out.

We restate the cases again:

- $\mathsf{HT}_X(M)[\delta[X \mapsto A]; \eta[X \mapsto T]] \iff T(M)$, where T is a predicate. Letting $\eta' = \eta[X \mapsto T]$, we have $T = \eta'(X)$.

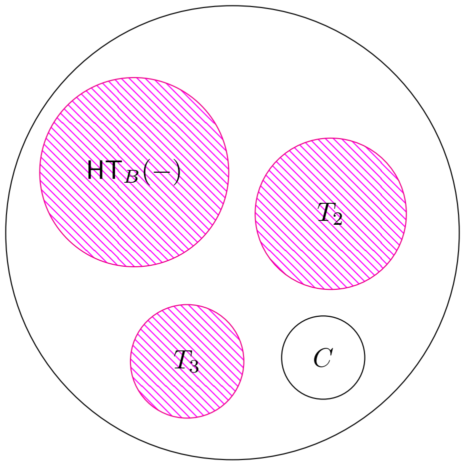
- $\mathsf{HT}_{\forall X.A}(M)[\delta;\eta] \iff$ for all closed B, we have $\mathsf{HT}_A(M(\downarrow))[\delta[M \mapsto B];\eta[X \mapsto \mathsf{HT}_B(-)[\delta;\eta]]]$.

  So the predicate we wish to use is $\mathsf{HT}_B(-)[\delta;\eta]$, which we denote the *standard interpretation* of B. In the proof checking, we eventually check X for hereditary termination under B.

We now seem to be much closer, but the proof is still flawed because checking $\mathsf{HT}_B(-)[\delta;\eta]$ is still not inductive. This framework requires only one final leap before it is correct. The final leap is a statement not only about our claim, but about the logic in which we are trying to prove it!

2.3 A Leap of Faith

We said that $\mathsf{HT}_B(-)[\delta;\eta]$ is the standard interpretation of B, and that checking it is not inductive in nature. Throughout this whole time we have been pestered by the fact that we need to pick some specific B, and admit that it may be larger than A in the first place. For every type B there is a corresponding standard interpretation. That leaves the question open: do there exist non-standard interpretations?



We know that a type is a collection of terms, and indeed so is a predicate like $\mathsf{HT}_B(-)$. In the diagram above, each circle represents some collection of terms. Some of the circles, including T_1 and T_2, are standard interpretations of types. On the other hand, C is not. Let us call collections of terms *type candidates*, or *possible types*. (We will need to impose some other conditions later, but in general not every type candidate is actually a type). So each circle above is a type candidate. Of these, the shaded ones are actually standard interpretations, though some are not inductively checkable. Now that we have this larger picture of the world, can we somehow indirectly reach $\mathsf{HT}_B(-)$?

Instead of pursuing $\mathsf{HT}_B(-)$ directly, we can recognize *any type candidate* as the predicate. Since one of the type candidates is $\mathsf{HT}_B(-)$, this is at least as strong as the original claim. And instead of ever explicitly substituting B, we need only to define convenient properties holding over all type candidates to use in the proof.

However, this means we need to quantify over *all* type candidates, appealing to the existence of the set of all type candidates. This is an appeal to a higher-ordered logic, which we have not seen so far in the proof!

Let's step back for a minute. Types themselves are "sets" (quantifications) over terms. Previously we proved the termination of Gödel's T using the hereditary termination predicates, which are second-order quantifications over terms, each corresponding to a type in T. Now, according to Gödel's second incompleteness theorem, first-order Peano arithmetic (and System T, which contains it) cannot prove its own consistency (computationally speaking, its termination), so the second-order proof was the best we could do.

By analogy, System F operates on types the way that System T operates on terms, allowing us to abstract over and apply them. It has become clear that proving termination for System F requires us to move one level higher, to third-order comprehension. We must produce the set of all type candidates in order to finish the proof, which can only be done in third-order logic.

There is a question of whether this is bending the axioms. Were we examining the logical system itself, this might be a sore point, as a higher-order comprehension principle is a controversial assumption. However, having already completed the termination proof for System T, we are content with accepting it. We are computer scientists seeking to prove a fact about the real System F, and will be happy with a proof in third-order logic.

With this machinery in place, we are finally ready to redefine the problematic case one last time:

$\mathsf{HT}_{\forall X.A}(M)[\delta;\eta] \iff$ for all closed B, for all type candidates T, $\mathsf{HT}_A(M(\downarrow))[\delta[X \mapsto B];\eta[X \mapsto T]]$.

Here, we quantify over all type candidates, among which is $\mathsf{HT}_B(-)$, the type candidate corresponding to the interpretation of B. Note the consequence of this claim on the nature of a polymorphic computation. For an expression to be polymorphic it must be uniform in *non-expressible properties* of the system. Some of these properties, like quantification over all type candidates, can only be stated in a higher-order logic. We *demand more* of programs than is evidently required. This makes our understanding of polymorphism much richer than "generics" in conventional programming languages.

3 Girard's Method

The new definition of hereditary termination, as we discovered through several trials, is:

$$\mathsf{HT}_A(M)[\delta;\eta:\Theta]$$

read as: M is hereditarily terminating at A, where δ is a closing type variable substitution and η is a candidate assignment, all with respect to Θ. Since it's usually clear from context, we drop Θ unless it's required.

To make precise the argument of type candidates, we need to make precise what they mean:

Type candidates are any collection $\mathcal{C}$ of *closed, erased* terms such that:

1. $\mathcal{C}$ is closed under head expansion, and possibly
2. $T \in \mathcal{C}$ and $T(M) \implies M\ \mathsf{term}_\beta$

Before presenting the FTLR for system F, we will need the following lemma:

Lemma 1 (Compositionality). $\mathsf{HT}_{[A/X]B}(M)[\delta;\eta]$ *iff* $\mathsf{HT}_B(M)[\delta[X \mapsto A];\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$

Proof. Induction on B.

- $B = X$
  For the forward direction, suppose $\mathsf{HT}_{[A/X]X}(M)[\delta;\eta]$. We need to show that $\mathsf{HT}_X(M)[\delta[X \mapsto A];\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. By definition, it suffices to show $(\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]])(X)(M)$, or that $\mathsf{HT}_A(M)[\delta;\eta]$, which follows from the assumption.

For the backward direction, suppose $\mathsf{HT}_X(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. We need to show that $\mathsf{HT}_{[A/X]X}(M)[\delta;\eta]$, or $\mathsf{HT}_A(M)[\delta;\eta]$, which follows by unfolding the assumption.

- $B = X'$ where $X \neq X'$
  For the forward direction, suppose $\mathsf{HT}_{[A/X]X'}(M)[\delta;\eta]$. We need to show that $\mathsf{HT}_{X'}(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. By definition, it suffices to show $(\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]])(X')(M)$. Since $X \neq X'$, it suffices to show $\eta(X')(M)$, which holds by assumption.

  For the backward direction, suppose $\mathsf{HT}_{X'}(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. We need to show that $\mathsf{HT}_{[A/X]X'}(M)[\delta;\eta]$, or $\mathsf{HT}_{X'}(M)[\delta;\eta]$. By definition, it suffices to show that $\eta(X')(M)$. By assumption, we know $(\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]])(X')(M)$ holds. Since $X \neq X'$, $(\eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]])(X') = \eta(X')$, and we have that $\eta(X')(M)$ holds.

- $B = A_1 \to A_2$
  For the forward direction, suppose $\mathsf{HT}_{[A/X]B}(M)[\delta;\eta]$. We need to show that $\mathsf{HT}_B(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. Suppose $\mathsf{HT}_{A_1}(N)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. It suffices to show then $\mathsf{HT}_{A_2}(M\,N)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. By IH, we have $\mathsf{HT}_{[A/X]XA_1}(N)[\delta;\eta]$. Further, from assumption we get $\mathsf{HT}_{[A/X]A_1 \to [A/X]A_2}(M)[\delta;\eta]$. By the definition of hereditary termination, we get $\mathsf{HT}_{[A/X]A_2}(M\,N)[\delta;\eta]$, and the result follows from the IH.

  For the backward direction, suppose $\mathsf{HT}_{A_1 \to A_2}(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. We need to show that $\mathsf{HT}_{[A/X]A_1 \to A_2}(M)[\delta;\eta]$, or $\mathsf{HT}_{[A/X]A_1 \to [A/X]A_2}(M)[\delta;\eta]$. Suppose $\mathsf{HT}_{[A/X]A_1}(N)[\delta;\eta]$. It suffices to show $\mathsf{HT}_{[A/X]A_2}(M\,N)[\delta;\eta]$. By IH, we have $\mathsf{HT}_{A_1}(N)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. Along with the assumption, we get $\mathsf{HT}_{A_2}(M\,N)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$, and the result follows from IH.

- $B = \forall Z.B'$
  For the forward direction, suppose $\mathsf{HT}_{[A/X]\forall Z.B'}(M)[\delta;\eta]$. We need to show that $\mathsf{HT}_{\forall Z.B'}(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_a(-)[\delta;\eta]]]$. Let C be a closed type, and T a type candidate. Letting $\delta' = \delta[X \mapsto A]$ and $\eta' = \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]$, it suffices to show $\mathsf{HT}_{B'}(|M|\downarrow)[\delta'[Z \mapsto C]; \eta'[Z \mapsto T]]$. Reordering the mappings, we have to show that $\mathsf{HT}_{B'}(|M|\downarrow)[\delta''[X \mapsto A]; \eta''[X \mapsto \mathsf{HT}_a(-)[\delta''; \eta'']]]]$, where $\delta'' = \delta[Z \mapsto C]$ and $\eta'' = \eta[Z \mapsto T]$. By the assumption (and capture-avoiding substitution), we have $\mathsf{HT}_{\forall Z.[A/X]B'}(M)[\delta;\eta]$. Now instantiate the RHS definition with C and T, obtaining $\mathsf{HT}_{[A/X]B'}(|M|\downarrow)[\delta[Z \mapsto C]; \eta[Z \mapsto T]]$. By IH, we have $\mathsf{HT}_{B'}(|M|\downarrow)[\delta''[X \mapsto A]; \eta''[X \mapsto \mathsf{HT}_A(-)[\delta''; \eta'']]]$, which is what was needed.

  For the backward direction, suppose $\mathsf{HT}_{\forall Z.B'}(M)[\delta[X \mapsto A]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]]]$. We need to show that $\mathsf{HT}_{[A/X]\forall Z.B'}(M)[\delta;\eta]$, or $\mathsf{HT}_{\forall Z.[A/X]B'}(M)[\delta;\eta]$. Let C be a closed type, and T a type candidate. By definition, it suffices to show $\mathsf{HT}_{[A/X]B'}(|M|\downarrow)[\delta[Z \mapsto C]; \eta[Z \mapsto T]]$. Instantiating the assumption with C and T, we have $\mathsf{HT}_{B'}(|M|\downarrow)[\delta[X \mapsto A][Z \mapsto C]; \eta[X \mapsto \mathsf{HT}_A(-)[\delta;\eta]][Z \mapsto T]]$. Swapping X and Z in δ and η as in the forward direction and applying the IH, we have $\mathsf{HT}_{[A/X]B'}(|M|\downarrow)[\delta[Z \mapsto C]; \eta[Z \mapsto T]]$.

□

Lemma 2 (Head expansion)**.** *If* $\Theta \vdash A$ type*,* $\delta : \Theta$*,* $\eta : \Theta$*, then* $\mathsf{HT}_A(-)[\delta;\eta]$ *is a type candidate.*

Proof. Induction on A.

- $A = X$
  Suppose $\mathsf{HT}_A(M')[\delta;\eta]$ and $M \mapsto M'$. We need to show that $\mathsf{HT}_A(M)[\delta;\eta]$. By

assumption, we have $\eta(X)(M')$. Since η is a candidate assignment, $\eta(X)$ is closed under head expansion, and so $\eta(X)(M)$ holds.

- $A = A_1 \to A_2$
  Suppose $\mathsf{HT}_{A_1 \to A_2}(M')[\delta;\eta]$ and $M \mapsto M'$. We need to show that $\mathsf{HT}_{A_1 \to A_2}(M)[\delta;\eta]$. Suppose $\mathsf{HT}_{A_1}(N)[\delta;\eta]$. It suffices to show that $\mathsf{HT}_{A_2}(M\,N)[\delta;\eta]$. Instantiating the assumption, we get $\mathsf{HT}_{A_2}(M'\,N)[\delta;\eta]$. Since $M\,N \mapsto M'\,N$, by IH, we have our result.

- $A = \forall X.B$
  Suppose $\mathsf{HT}_{\forall X.B}(M')[\delta;\eta]$ and $M \mapsto M'$. We need to show $\mathsf{HT}_{\forall X.B}(M)[\delta;\eta]$. Let C be a closed type, T a type candidate. It suffices to show $\mathsf{HT}_B(|M| \downarrow)[\delta[X \mapsto C];\eta[X \mapsto T]]$. Instantiating the assumption with C and T, we have $\mathsf{HT}_B(|M'| \downarrow)[\delta[X \mapsto C];\eta[X \mapsto T]]$. Since $|M| \downarrow \mapsto |M'| \downarrow$, by IH, we have our result.

□

Theorem 3 (FTLR for System F). *If* $\Theta;\Omega \vdash M : A$*, and*

1. $\delta : \Theta$ *is a closing type substitution*
2. $\eta : \Theta$ *is a candidate assignment*
3. $\gamma : \cdot \to \Gamma$ *is a closing term substitution*
4. $\mathsf{HT}_\Gamma(\gamma)[\delta;\eta]$

then $\mathsf{HT}_A(\hat{\gamma}(|M|))[\delta;\eta]$.

Proof. Induction on typing.

- $\dfrac{\Theta, X;\Gamma \vdash M : B}{\Theta;\Gamma \vdash \Lambda X.\,M : \forall X.B}$
  Fix $\delta : \Theta, \eta : \Theta$ and $\mathsf{HT}_\Gamma(\gamma)[\delta;\eta]$. We need to show that $\mathsf{HT}_{\forall X.B}(\hat{\gamma}(|\Lambda X.\,M|))[\delta;\eta : \Theta]$, or that $\mathsf{HT}_{\forall X.B}(\lambda\_.\,\hat{\gamma}(|M|))[\delta;\eta : \Theta]$. Let C be a closed type, T a type candidate, $\delta' = \delta[X \mapsto C]$, $\eta' = \eta[X \mapsto T]$, and $\Theta' = \Theta, X$. We need to show that $\mathsf{HT}_B(\lambda\_.\,\hat{\gamma}(|M|) \downarrow)[\delta';\eta' : \Theta']$. By head expansion, it suffices to show that $\mathsf{HT}_B(\hat{\gamma}(|M|))[\delta';\eta' : \Theta']$, which follows from IH.

- $\dfrac{\Theta;\Gamma \vdash M : \forall X.B \qquad \Theta \vdash C \; \mathsf{type}}{\Theta;\Gamma \vdash M\,[C] : [C/X]B}$
  Fix $\delta : \Theta, \eta : \Theta$ and $\mathsf{HT}_\Gamma(\gamma)[\delta;\eta]$. We need to show that $\mathsf{HT}_{[C/X]B}(\hat{\gamma}(|M\,[C]|))[\delta;\eta]$, or that $\mathsf{HT}_{[C/X]B}(\hat{\gamma}(|M|) \downarrow)[\delta;\eta]$. By compositionality, it suffices to show that $\mathsf{HT}_B(\hat{\gamma}(|M|) \downarrow)[\delta[X \mapsto C];\eta[X \mapsto \mathsf{HT}_C(-)[\delta;\eta]]]$. By IH, we have $\mathsf{HT}_{\forall X.B}(\hat{\gamma}(|M|))[\delta;\eta]$. Instantiating the RHS with C and $\mathsf{HT}_C(-)[\delta;\eta]$ (which by Lemma 2 is a type candidate), we get $\mathsf{HT}_B(\hat{\gamma}(|M|) \downarrow)[\delta[X \mapsto C];\eta[X \mapsto \mathsf{HT}_C(-)[\delta;\eta]]]$.

□

Corollary 4. $\mathsf{HT}_{\mathsf{ans}}(M)[\emptyset;\emptyset] \implies M\ \mathsf{term}_\beta$.

In order to extend this to all types, we need to stipulate termination for all type candidates and formulate hereditary termination positively.

4 Equality

The plan from now on:

1. Equality: formal vs semantic
2. Parametricity / data abstraction by "binarizing" Girard's method
3. Propositions as types and dependent types
4. CTT / HDTT

Recall definitional (structural) equality (equivalence) in formal type theory, originating from Gentzen's inversion principles. D.E. can be charaterized as follows:

1. RST (is a equivalence relation)
2. Compatible with the term formers (is a congruence)
3. "Calculates" via beta-reduction (but is not directed)

D.E. is a very strong "equality", and doesn't support hypothetical reasoning. For instance, if we defined plus in Gödel's T (by recursion on one of the arguments), we will not be able to prove $x : \mathsf{nat}, y : \mathsf{nat} \vdash x + y \equiv y + x : \mathsf{nat}$, even though for all numerals $\bar{n}, \bar{m}$, we *can* derive $\vdash \bar{n} + \bar{m} \equiv \bar{m} + \bar{n} : \mathsf{nat}$.

References

Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 1978.