Semantic Equality for Typed λ-Calculus*

Robert Harper

Spring, 2025

1 Introduction

The unary logical relations developed in Harper (2025b) may be extended from unary predicates to binary relations. In binary form these relations define *exact equality* at each type. Unlike axiomatic accounts (those given by rules) exact equality defines when two terms of a type are *semantically equal*. For example, the "add-to-self" and "doubling" functions on the natural numbers are exactly equal, because they have the same I/O behavior. This formulation—the standard one for equality of functions—is called *extensional equality*.

This note defines exact equality for terms of each type, and establishes some basic properties of it, in particular that it is an equivalence relation that is compatible with the term-forming operations, and that it respects—and thus contains—evaluation.

2 Exact Equality

The definition of exact equality is very similar to the definition of hereditary termination given in Harper (2025b), here written $M \doteq M' \in A$:[1]

$$M \doteq M' \in \mathtt{ans} \text{ iff } M, M' \longmapsto^* \mathtt{yes} \text{ or } M, M' \longmapsto^* \mathtt{no}$$

$$M \doteq M' \in \mathtt{unit} \text{ iff } M, M' \longmapsto^* \langle\rangle$$

$$M \doteq M' \in A_1 \times A_2 \text{ iff } M \longmapsto^* \langle M_1, M_2\rangle, M' \longmapsto^* \langle M_1', M_2'\rangle, \text{ and}$$
$$M_1 \doteq M_1' \in A_1 \text{ and } M_2 \doteq M_2' \in A_2$$

$$M \doteq M' \in A_1 \to A_2 \text{ iff } M \longmapsto^* \lambda(x.N), M' \longmapsto^* \lambda(x.N'), \text{ and}$$
$$\text{if } M_1 \doteq M_1' \in A_1 \text{ then } [M_1/x]N \doteq [M_1'/x]N' \in A_2$$

The judgment $M \in A$ is defined to mean $M \doteq M \in A$.

If exact equality is to be so-called, it ought to be symmetric and transitive.

Lemma 1. *1. If $M \doteq M' \in A$ then $M' \doteq M \in A$.*

[1] The closed binary relation is often written $\mathsf{HE}_A(M, M', -NoValue-)$, for "hereditary extensionality."

2. *If* $M \doteq M' \in A$ *and* $M' \doteq M'' \in A$*, then* $M \doteq M'' \in A$*.*

Proof. Symmetry and transitivity must be proved simultaneously by structural induction on the type A. Consider the case $A = A_1 \to A_2$.

1. Suppose that $M \doteq M' \in A$ with the goal to show that $M' \doteq M \in A$. By assumption $M \longmapsto^* \lambda(x.N)$ and $M' \longmapsto^* \lambda(x.N')$. Assume that $M_1' \doteq M_1 \in A_1$, with the intent to show that $[M_1'/x]N' \doteq [M_1/x]N \in A_2$. A direct application of the outer assumption yields $[M_1/x]N' \doteq [M_1'/x]N \in A_2$, which is not what is required. However, exact equality at both A_1 and A_2 is symmetric. First, appealing to symmetry at A_1, from the assumption $M_1' \doteq M_1 \in A_1$ it follows that $M_1 \doteq M_1' \in A_1$, and hence by the outer assumption $[M_1/x]N \doteq [M_1'/x]N' \in A_2$. Then, applying symmetry at A_2, the desired result follows.

2. Suppose that $M \doteq M' \in A$ and $M' \doteq M'' \in A$ with the goal to show that $M \doteq M'' \in A$. By the definition of exact equality at function type, the two assumptions imply that $M \longmapsto^* \lambda(x.N)$, $M' \longmapsto^* \lambda(x.N')$, and $M'' \longmapsto^* \lambda(x.N'')$. Now suppose that $M_1 \doteq M_1'' \in A_1$ with the intent to show that $[M_1/x]N \doteq [M_1''/x]N'' \in A_2$. Here again a direct application of the outer assumptions does not seem to help, obtaining

   (a) $[M_1/x]N \doteq [M_1''/x]N' \in A_2$, and

   (b) $[M_1/x]N' \doteq [M_1''/x]N'' \in A_2$.

   Note that a symmetric and transitive relation is reflexive on related elements: if $R(M, M')$ then $R(M', M)$ by symmetry, and so $R(M, M)$ and $R(M', M')$ by transitivity. By the inductive assumptions equality at type A_1 is symmetric and transitive, and so $M \doteq M \in A_1$ follows from the inner assumption. Then, by the first outer assumption, $[M/x]N \doteq [M/x]N' \in A_2$. Applying the second displayed equation above, and the transitivity of equality at A_2, the result follows.

□

Symmetric and transitive relations are called *partial equivalence relations*, or *p.e.r.'s*. The remark in the proof about deriving reflexivity for related elements, called the *p.e.r. trick*, will be of further use below.

Exercise 1. *Check the remaining cases of symmetry and transitivity.*

The analogue of the fundamental theorem in Harper (2025b) is the reflexivity of exact equality. Define $\gamma \doteq \gamma' \in \Gamma$ variable-by-variable and define $\Gamma \gg M \doteq M' \in A$ to mean if $\gamma \doteq \gamma' \in \Gamma$, then $\widehat{\gamma}(M) \doteq \widehat{\gamma'}(M') \in A$.

Exact equality is *behavioral* in that it expresses execution properties of programs. It is codified by the notion of "head expansion," a term that is widely used in the literature to express it.

Lemma 2 (Head Expansion)**.** *If* $M \doteq M' \in A$ *and* $N \longmapsto M$*, then* $N \doteq M' \in A$*.*

The analogous propery for the right-hand side of the equation follows from symmetry, or may be proved separately by an analogous argument. Thus, for example, a notion of "equality" of programs that is sensitive to their *form*, rather than their *behavior*, would not be amenable to the type-directed treatment considered here.

Theorem 3 (Reflexivity). *If* $\Gamma \vdash M : A$, *then* $\Gamma \gg M \in A$.

Proof. By induction on typing derivations, proceeding analogously to the proof given in Harper (2025b). □

Observe that the full meaning of reflexivity of open terms involves disparate substitution instances of them. This is necessitated by the definition of computability at function types.

The fundamental theorem tells us that well-typed terms are exactly equal to themselves. At first this may sound trivial, but because exact equality is a *behavioral* condition on evaluation, it requires proof, and can even fail when a type system is not properly designed. By Lemma 1 exact equality for *closed* terms is symmetric and transitive. However, this does not immediately imply that the same is true for *open* terms!

Lemma 4 (Symmetry and Transitivity). 1. *If* $\Gamma \gg M \doteq M' \in A$, *then* $\Gamma \gg M' \doteq M \in A$.

2. *If* $\Gamma \gg M \doteq M' \in A$ *and* $\Gamma \gg M' \doteq M'' \in A$, *then* $\Gamma \gg M \doteq M'' \in A$.

Proof. Note, first of all, that these properties are not immediate consequences of exact equality being symmetric and transitive—some argumentation is required because of the disparate instantiation of the variables in Γ.

1. Assume that $\Gamma \gg M \doteq M' \in A$, and suppose that $\gamma' \doteq \gamma \in \Gamma$, with the intent to show that $\widehat{\gamma'}(M') \doteq \widehat{\gamma}(M) \in A$. Simply instantiating the assumption yields $\widehat{\gamma'}(M) \doteq \widehat{\gamma}(M') \in A$, which is neither the intended result, nor is it its symmetric form. Instead, by the symmetry of closed exact equality, $\gamma \doteq \gamma' \in \Gamma$ follows from $\gamma' \doteq \gamma \in \Gamma$, then instantiating the hypothesis accordingly yields the desired result.

2. Assume the two premises, and suppose that $\gamma \doteq \gamma'' \in \Gamma$, with the intent to show $\widehat{\gamma}(M) \doteq \widehat{\gamma''}(M'') \in A$. Instantiating the two premises directly yields

   (a) $\widehat{\gamma}(M) \doteq \widehat{\gamma''}(M') \in A$, and

   (b) $\widehat{\gamma}(M') \doteq \widehat{\gamma''}(M'') \in A$.

   Here again these two facts do not yield the desired result—the disparate substitutions preclude direction application of transitivity of exact equality at type A. Instead, observe that by symmetry and transitivity of exact equality of substitutions for Γ, it follows that $\gamma \doteq \gamma \in \Gamma$, which can be applied to the first inductive hypothesis to obtain $\widehat{\gamma}(M) \doteq \widehat{\gamma}(M') \in A$. This, together with the second equation above implies, by transitivity, the desired result.

□

The disparity between the two sides of an equation induced by application of disparate substitutions gives rise to the use of the "p.e.r. trick" to obtain a reflexive instance that mediates the proof. An alternative approach is developed below that embraces, rather than evades, the disparities, which proves to be useful in the theory of parametricity.

The rules in Figure 1 may be validated as expressing true exact equations.

Theorem 5 (Equational Validity). *If* $\Gamma \vdash M \equiv N : A$, *then* $\widehat{\gamma}(M) \doteq \widehat{\gamma'}(N) \in A$ *for all* $\gamma \doteq \gamma' \in \Gamma$.

$$\text{REFL}\ \frac{\Gamma \vdash M : A}{\Gamma \vdash M \equiv M : A} \qquad \text{SYM}\ \frac{\Gamma \vdash M \equiv N : A}{\Gamma \vdash N \equiv M : A} \qquad \text{TRANS}\ \frac{\Gamma \vdash M \equiv N : A \qquad \Gamma \vdash N \equiv P : A}{\Gamma \vdash M \equiv P : A} \qquad 1\text{-}\eta\ \frac{\Gamma \vdash M : 1}{\Gamma \vdash M \equiv \langle\rangle : 1}$$

$$\times\text{-I}\ \frac{\Gamma \vdash M_1 \equiv N_1 : A_1 \qquad \Gamma \vdash M_2 \equiv N_2 : A_2}{\Gamma \vdash \langle M_1, M_2\rangle \equiv \langle N_1, N_2\rangle : A_1 \times A_2} \qquad \times\text{-E-L}\ \frac{\Gamma \vdash M \equiv N : A_1 \times A_2}{\Gamma \vdash M \cdot 1 \equiv N \cdot 1 : A_1} \qquad \times\text{-E-R}\ \frac{\Gamma \vdash M \equiv N : A_1 \times A_2}{\Gamma \vdash M \cdot 2 \equiv N \cdot 2 : A_1}$$

$$\times\text{-}\beta\text{-L}\ \frac{\Gamma \vdash M_1 : A_1 \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2\rangle \cdot 1 \equiv M_1 : A_1} \qquad \times\text{-}\beta\text{-R}\ \frac{\Gamma \vdash M_1 : A_1 \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2\rangle \cdot 2 \equiv M_2 : A_2}$$

$$\times\text{-}\eta\ \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash M \equiv \langle M \cdot 1, M \cdot 2\rangle : A_1 \times A_2} \qquad \rightarrow\text{-I}\ \frac{\Gamma, x : A_1 \vdash M_2 \equiv N_2 : A_2}{\Gamma \vdash \lambda(x.M_2) \equiv \lambda(x.N_2) : A_1 \rightarrow A_2}$$

$$\rightarrow\text{-E}\ \frac{\Gamma \vdash M \equiv N : A_1 \rightarrow A_2 \qquad \Gamma \vdash M_1 \equiv N_1 : A_1}{\Gamma \vdash \mathtt{ap}(M\,;M_1) \equiv \mathtt{ap}(N\,;N_1) : A_2} \qquad \rightarrow\text{-}\beta\ \frac{\Gamma, x : A_1 \vdash M_2 : A_2 \qquad \Gamma \vdash M_1 : A_1}{\Gamma \vdash \mathtt{ap}(\lambda(x.M_2)\,;M_1) \equiv [M_1/x]M_2 : A_2}$$

$$\rightarrow\text{-}\eta\ \frac{\Gamma \vdash M : A_1 \rightarrow A_2}{\Gamma \vdash M \equiv \lambda(x.\,\mathtt{ap}(M\,;x)) : A_1 \rightarrow A_2}$$

Figure 1: Definitional Equivalence for Products and Functions

Proof. The proof is by induction on the derivation of the equation, making use of the lemmas given above, including head expansion and reflexivity lemmas. The rules are formulated with typing premises that are essential to the argument. In particular the rule for β-equivalence for function types relies on the combination of the two typing premises to obtain an equation between two instances of the right-hand side of the equation, with the result then following by head expansion. □

Exercise 2. *Complete the proof of Theorem 5 in the indicated manner. Which typing premises, if any, are needed to complete the proof?*

Exercise 3. *Give β- and η equations for the type* `conat` *of co-natural numbers defined in Harper (2025a).*

Exercise 4. *Define exact equality for co-natural numbers, and prove that the β- and η equations are valid as principles of exact equality.*

3 Sums and (Co-)Natural Numbers

Extending the equational theory to account for (empty and non-empty) sums, and other inductive types, raises some important questions. Although the β-like rules of equality are only to be expected, what

would be the analog of an η-like rule? These rules for products and function types are straightforward, amounting to the assertion that every element of a product type is a pair, and every element of a function type is a λ-abstraction. But what would be an η-like rule for sums? In the binary case it is clear that there must be a way to state that every element of a sum is either a first- or second-injection of an element of the summand type, but how is one to state that? And in the nullary case? Or the infinitary case of an inductive type such as the type `nat` of natural numbers? Or any other form of inductive type? There is a further difficulty arising from the fact that the elimination forms for sums "reach in" to a third type, and hence influence equality at all types.

A formulation of equality for sums is given in Figure 2. The β rules are as would be expected, capturing the computational behavior of case analysis on injections. The η rules amount to "reconstructing" a term of sum type by case analysis and injection, there being two cases for sums and no cases for the empty type. Finally, the *commutation equations* state that (nullary and binary) case analyses may be "lifted" from inside a term, and correspondingly substituting the correct summand data in each case.

Exact equality is extended to sums relative to the dynamics given in Harper (2025b) as follows:

$$\begin{aligned} M \doteq M' \in \mathtt{void} \text{ iff } &\textit{(never)} \\ M \doteq M' \in A_1 + A_2 \text{ iff } &M \longmapsto^* 1 \cdot M_1,\ M' \longmapsto^* 1 \cdot M_1' \text{ and } M_1 \doteq M_1' \in A_1 \text{ or} \\ &M \longmapsto^* 2 \cdot M_2,\ M' \longmapsto^* 2 \cdot M_2' \text{ and } M_2 \doteq M_2' \in A_1 \end{aligned}$$

Exercise 5. *Prove the soundness of the equations given in Figure 2 relative to these equations.*

The analogous equations for the type of natural numbers are given in Figure 3. Whereas the β rules are as would be expected, the analogue of the η and commutation rules are codified as a principle of induction. Specifically, the term P computes a result of type C from an input x of type `nat`. The given recursor defines another such computation that is equivalent to P iff (a) it behaves the same as P on `zero`, and, assuming that it behaves the same as P on x, show that it behaves the same as P on $\mathtt{succ}(x)$ (implicitly making use of the β rule for the recursor on a successor).

Exercise 6. *Show that* $x : \mathtt{nat} \vdash \mathtt{rec}(x; \mathtt{zero}; y.\, \mathtt{succ}(y)) \equiv x : \mathtt{nat}$ *using the rules in Figure 3.*

Exact equality of natural numbers relative to the dynamics given in Harper (2025a) is defined as follows:

$$\begin{aligned} M \doteq M' \in \mathtt{nat} \text{ iff } &\mathcal{N}(M, M'), \text{ where } \mathcal{N} \text{ is the strongest relation such that } \mathcal{N}(M, M') \text{ if} \\ &M, M' \longmapsto^* \mathtt{zero}, \text{ or } M \longmapsto^* \mathtt{succ}(N),\ M' \longmapsto^* \mathtt{succ}(N') \text{ and } \mathcal{N}(N, N'). \end{aligned}$$

NB: The induction principle induced by this definition (Harper, 2025c) is *not* the familiar principle of mathematical induction, though it is obviously closely related to it. It pertains to closed values of type `nat`, which are constructed from `zero` and $\mathtt{succ}(-)$, interposing evaluation to evaluate intermediate computations.

Exercise 7. *Prove the soundness of the equations given in Figure 3 relative to the definition of exact equality for the type of natural numbers.*

Exercise 8. *Prove that* $\mathtt{nat} \to \mathtt{nat} \equiv \lambda(x.x + x) : \lambda(x.2 \times x)$*, for suitable definitions of the two functions in terms of iteration on the argument* x*.*

$$\textsc{void-}\eta\ \frac{\Gamma \vdash M : \mathtt{void}}{\Gamma \vdash M \equiv \mathtt{absurd}(M) : \mathtt{void}} \qquad \textsc{void-com}\ \frac{\Gamma, x : \mathtt{void} \vdash N : C \qquad \Gamma \vdash M : \mathtt{void}}{\Gamma \vdash [\mathtt{absurd}(M)/x]N \equiv \mathtt{absurd}(M) : C}$$

$$+\text{-}\beta\text{-}i\ \frac{\Gamma \vdash M_i : A_i \qquad \Gamma, x_1 : A_1 \vdash N_1 : C \qquad \Gamma, x_2 : A_2 \vdash N_2 : C}{\Gamma \vdash \mathtt{case}\ i \cdot M_i \{ x_1.N_1 \mid x_2.N_2 \} \equiv [M_i/x_i]N_i : C}$$

$$+\text{-}\eta\ \frac{\Gamma \vdash M : A_1 + A_2}{\Gamma \vdash M \equiv \mathtt{case}\ M \{ x_1.1 \cdot x_1 \mid x_2.2 \cdot x_2 \} : A_1 + A_2}$$

$$+\textsc{-com}\ \frac{\Gamma, x : A_1 + A_2 \vdash N : C \qquad \Gamma \vdash M : A_1 + A_2}{\Gamma \vdash [\mathtt{case}\ M \{ x_1.M_1 \mid x_2.M_2 \}/x]N \equiv \mathtt{case}\ M \{ x_1.[M_1/x]N \mid x_2.[M_2/x]N \} : C}$$

Figure 2: Definitional Equivalence for Sums

$$\mathtt{nat}\text{-}\beta\text{-}\mathtt{zero}\ \frac{\Gamma \vdash N_0 : C \qquad \Gamma, y : C \vdash N_1 : C}{\Gamma \vdash \mathtt{rec}(\mathtt{zero}; N_0; y.N_1) \equiv N_0 : C} \qquad \mathtt{nat}\text{-}\beta\textsc{-succ}\ \frac{\Gamma \vdash M : \mathtt{nat} \qquad \Gamma \vdash N_0 : C \qquad \Gamma, y : C \vdash N_1 : C}{\Gamma \vdash \mathtt{rec}(\mathtt{succ}(M); N_0; y.N_1) \equiv [\mathtt{rec}(M; N_0; y.N_1)/y]N : C}$$

$$\textsc{nat-ind}\ \frac{\Gamma \vdash M : \mathtt{nat} \qquad \Gamma, x : \mathtt{nat} \vdash P : C \qquad \Gamma \vdash N_0 : C \qquad \Gamma, y : C \vdash N_1 : C \qquad \Gamma \vdash N_0 \equiv [\mathtt{zero}/x]P : C \qquad \Gamma, x : \mathtt{nat} \vdash [P/y]N \equiv [\mathtt{succ}(x)/x]P : C}{\Gamma \vdash \mathtt{rec}(M; N_0; y.N_1) \equiv [M/x]P : C}$$

Figure 3: Definitional Equivalence for Natural Numbers

Exercise 9. *Formulate definitional equality for the coinductive type* `conat` *considered in Harper (2025a). Extend exact equality to this case, and prove that the equational theory is sound with respect to that interpretation.*

References

Robert Harper. Termination for natural and unnatural numbers. (Unpublished lecture note), January 2025a. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/notes/natco.pdf`.

Robert Harper. How to (re)invent Tait's method. Unpublished lecture note, January 2025b. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/tait.pdf`.

Robert Harper. Tarski's fixed point theorem. (Unpublished lecture note), January 2025c. URL https://www.cs.cmu.edu/~rwh/courses/atpl/notes/tarski.pdf.