The Relational Action of Type Constructors*

Robert Harper

Spring, 2025

1 Introduction

The termination proof given in Harper (2025b) and the normalization proof given in Harper (2025a) are similar in that they make use of the *action* of type constructors on *candidates (for computability)*. By defining these actions appropriately, the proof of the fundamental theorem "writes itself" in that the cases for the introduction and elimination rules for each type are automatic. This formulation and consolidation is the foundation for the method of *logical relations* used widely in semantics.

2 Hereditary Termination, Reformulated

Definition 1 (Computability Candidate)**.** *A* computability candidate $\mathcal{C}$ *for a type* A *is a predicate on closed terms of type* A *that is* closed under head expansion *in that, for closed* M *and* M' *of type* A*, if* $M \in \mathcal{C}$ *and* $M' \longmapsto M$*, then* $M' \in \mathcal{C}$*.*

Each of the type constructors has an action on closed candidates that suffices for the proof of the Fundamental Theorem.

Definition 2 (Action of Type Constructors on Computability Candidates)**.**

$$\begin{aligned}
0 &\triangleq \emptyset \\
1 &\triangleq \{M \mid M : 1 \text{ and } M \longmapsto^* \langle\rangle\} \\
\mathcal{C}_1 + \mathcal{C}_2 &\triangleq \{M \mid M : A_1 + A_2 \text{ and } M \longmapsto^* 1 \cdot M_1 \text{ and } M_1 \in \mathcal{C}_1\} \\
&\quad \cup \{M \mid M : A_1 + A_2 \text{ and } M \longmapsto^* 2 \cdot M_2 \text{ and } M_2 \in \mathcal{C}_2\} \\
\mathcal{C}_1 \times \mathcal{C}_2 &\triangleq \{M \mid M : A_1 \times A_2 \text{ and } M \cdot 1 \in \mathcal{C}_1 \text{ and } M \cdot 2 \in \mathcal{C}_2\} \\
\mathcal{C}_1 \to \mathcal{C}_2 &\triangleq \{M \mid M : A_1 \to A_2 \text{ and if } M_1 \in \mathcal{C}_1 \text{ then } \mathtt{ap}(M\,;M_1) \in \mathcal{C}_2\}
\end{aligned}$$

Exercise 1. *Verify that each of the actions results in a computability candidate, given that its arguments are computability candidates of appropriate type.*

Exercise 2. *Give the definitions of computability candidates* N *and* $\mathcal{N}$ *for the types of natural and conatural numbers, using Tarski's Theorem.*

Hereditary termination may be defined succinctly using the action of type constructors on computability candidates:

Definition 3 (Hereditary Termination)**.**

$$\begin{aligned} HT_0 &\triangleq 0 \\ HT_1 &\triangleq 1 \\ HT_{A_1+A_2} &\triangleq HT_{A_1} + HT_{A_2} \\ HT_{A_1\times A_2} &\triangleq HT_{A_1} \times HT_{A_2} \\ HT_{A_1\to A_2} &\triangleq HT_{A_1} \to HT_{A_2} \end{aligned}$$

The semantic membership judgment, $\Gamma \gg M \in A$, is defined to mean $\mathsf{HT}_\Gamma(\gamma)$ implies $\mathsf{HT}_A(\hat{\gamma}(M))$, and the fundamental theorem states that if $\Gamma \vdash M : A$, then $\Gamma \gg M \in A$.

3 Hereditary Extensionality, Reformulated

The reformulation of hereditary termination extends to hereditary extensionality, or exact equality, for closed terms.

Definition 4 (Extensionality Candidate)**.** *An* extensionality candidate *for a type, A, is a binary relation, $\mathcal{E}$, on closed terms of type A that is closed under head expansion.*

Definition 5 (Action of Type Constructors on Extensionality Candidates)**.**

$$\begin{aligned} 0 &\triangleq \emptyset \\ 1 &\triangleq \{(M,M') \mid M,M' : 1 \text{ and } M \longmapsto^* \langle\rangle\} \\ \mathcal{C}_1 + \mathcal{C}_2 &\triangleq \{(M,M') \mid M,M' : A_1 + A_2,\ M \longmapsto^* 1\cdot M_1,\ M' \longmapsto^* 1\cdot M_1',\ \text{and } (M_1,M_1') \in \mathcal{C}_1\} \\ &\quad \cup \{(M,M') \mid M,M' : A_1 + A_2,\ M \longmapsto^* 2\cdot M_2,\ M' \longmapsto^* 2\cdot M_2',\ \text{and } (M_2,M_2') \in \mathcal{C}_2\} \\ \mathcal{C}_1 \times \mathcal{C}_2 &\triangleq \{(M,M') \mid M,M' : A_1\times A_2 \text{ and } (M\cdot 1, M'\cdot 1) \in \mathcal{C}_1 \text{ and } (M\cdot 2, M'\cdot 2,) \in \mathcal{C}_2\} \\ \mathcal{C}_1 \to \mathcal{C}_2 &\triangleq \{M \mid M,M' : A_1 \to A_2 \text{ and if } (M_1M_1',) \in \mathcal{C}_1 \text{ then } (\mathtt{ap}(M\,;M_1), \mathtt{ap}(M'\,;M_1')) \in \mathcal{C}_2\} \end{aligned}$$

Definition 6 (Hereditary Extensionality)**.**

$$\begin{aligned} HE_0 &\triangleq 0 \\ HE_1 &\triangleq 1 \\ HE_{A_1+A_2} &\triangleq HE_{A_1} + HE_{A_2} \\ HE_{A_1\times A_2} &\triangleq HE_{A_1} \times HE_{A_2} \\ HE_{A_1\to A_2} &\triangleq HE_{A_1} \to HE_{A_2} \end{aligned}$$

4 Hereditary Normalization, Reformulated

In the case of the normalization proof candidates are re-defined as *families* of sets indexed by the pre-order on contexts. Each context Δ determines a set (predicate) in such a way that if $\Delta' \leq \Delta$, then the set assigned to Δ must be contained in the set assigned to Δ'. This requirement is in line with weakening for the typing judgment: if $\Delta \vdash M : A$, then $\Delta' \vdash M : A$ for any $\Delta' \leq \Delta$. Put in other terms, the set assignment cannot depend on what variables are *not* present in the context, only on those that *are*.

Definition 7 (Normalization Candidate)**.** *An* normalization candidate*, $\mathcal{C}$, for type A over context Δ is a set of terms $\Delta \vdash M : A$ that is closed under head expansion: if $M \in \mathcal{C}$, and $M' \longmapsto_\beta M$, then $M' \in \mathcal{C}$.*

Definition 8 (Family of Normalization Candidates)**.** *A* family of normalization candidates*, $\mathcal{F}$, for a type A is an assignment of a normalization candidate $\mathcal{F}(\Delta)$ for A over Δ to each context Δ such that if $\Delta' \leq \Delta$, then $\mathcal{F}(\Delta) \subseteq \mathcal{F}(\Delta')$.*

As with closed candidates, each type constructor acts on candidate families in such a way as to ensure that the FTLR holds.

Definition 9 (Action of Type Constructors on Normalization Candidates)**.**

$$
\begin{aligned}
0(\Delta) &\triangleq \{M \mid \Delta \vdash M : 0 \text{ and } \mathsf{norm}_\beta(M)\} \\
1(\Delta) &\triangleq \{M \mid \Delta \vdash M : 1 \text{ and } \mathsf{norm}_\beta(M)\} \\
(\mathcal{F}_1 + \mathcal{F}_2)(\Delta) &\triangleq \{M \mid \Delta \vdash M : A_1 + A_2 \text{ and} \\
&\qquad \text{if } M \to_\beta^* 1 \cdot M_1 \text{ then } M_1 \in \mathcal{F}_1(\Delta) \text{ and} \\
&\qquad \text{if } M \to_\beta^* 2 \cdot M_2 \text{ then } M_2 \in \mathcal{F}_2(\Delta)\} \\
(\mathcal{F}_1 \times \mathcal{F}_2)(\Delta) &\triangleq \{M \mid \Delta \vdash M : A_1 \times A_2 \text{ and } M \cdot 1 \in \mathcal{F}_1(\Delta) \text{ and } M \cdot 2 \in \mathcal{F}_2(\Delta)\} \\
(\mathcal{F}_1 \to \mathcal{F}_2)(\Delta) &\triangleq \{M \mid \Delta \vdash M : A_1 \to A_2 \text{ and} \\
&\qquad \text{if } \Delta' \leq \Delta \text{ and } M_1 \in \mathcal{F}_1(\Delta') \text{ then } \mathsf{ap}(M\,;M_1) \in \mathcal{F}_2(\Delta')\}
\end{aligned}
$$

Hereditary normalization may be succinctly defined using these operations in formally the same way as before, albeit with the right-hand now being a family of normalization candidates in each case.

Definition 10 (Hereditary Normalization)**.**

$$
\begin{aligned}
HN_0 &\triangleq 0 \\
HN_1 &\triangleq 1 \\
HN_{A_1+A_2} &\triangleq HN_{A_1} + HN_{A_2} \\
HN_{A_1\times A_2} &\triangleq HN_{A_1} \times HN_{A_2} \\
HN_{A_1\to A_2} &\triangleq HN_{A_1} \to HN_{A_2}
\end{aligned}
$$

5 Inductive and Coinductive Types, In General

One use of computability candidates is in the semantics of general inductive and coinductive types, which define the least and greatest solutions to a type equation given by a monotone type operator.

Another use is in the formulation of parametricity, as described in Harper (2025d). Here we consider the definition of hereditary termination for general inductive and coinductive types.

First, a *type abstractor* $t.A$ is an otherwise-closed type expression with a distinguished variable, t, that may occur within it. As regards termination, we may define $\mathsf{HT}_{t.A}$ as a mapping sending a computability predicate, $\mathcal{C}$, over a type B to the computability predicate induced by $[B/t]A$, taking computability at type t to be defined by $\mathcal{C}$:

$$\mathsf{HT}_{t.A} : \mathcal{C} \longmapsto [\mathcal{C}/t]\,\mathsf{HT}_A$$

The pseudo-substitution notation, $[\mathcal{C}/t]\,\mathsf{HT}_A$, means to regard HT_t to be equal to $\mathcal{C}$ on the right-hand side. When $t.A$ is a *positive* type operator, then this assignment is *monotone* in that if $\mathcal{C} \subseteq \mathcal{C}'$ are computability candidates, then $[\mathcal{C}/t]\,\mathsf{HT}_{t.A} \subseteq [\mathcal{C}'/t]\,\mathsf{HT}_{t.A}$ as computability candidates for $[B/t]A$.

Exercise 3. *Prove that $\mathsf{HT}_{t.A}$ is monotone in the above sense when $t.A$ is a polynomial type operator when A is otherwise closed (contans no other free type variables than t.)*

Second, given that positive type operators in syntax induce monotone operators on candidates, hereditary termination for the *inductive* and *coinductive* may be defined by appeal to Tarski's Theorem:

$$\mathsf{HT}_{\mathtt{ind}(t.A)} \triangleq \bigcap_{\mathcal{C}} [\mathcal{C}/t]\,\mathsf{HT}_{t.A}$$
$$\mathsf{HT}_{\mathtt{coi}(t.A)} \triangleq \bigcup_{\mathcal{C}} [\mathcal{C}/t]\,\mathsf{HT}_{t.A}$$

where $\mathcal{C}$ ranges over computability predicates of type $\mathtt{ind}(t.A)$ and $\mathtt{coi}(t.A)$, respectively. The intersection and union are taken over all candidates for the inductive and coinductive types, respectively.

This definition provides what is necessary to prove the computability of the introductory and eliminatory forms for inductive and coinductive types.

Exercise 4. *Prove the fundamental theorem for the case of inductive and coinductive types in the above sense by appeal to Tarksi's Theorem (Harper, 2025c). As a special case, re-derive computability for the types* `nat` *and* `conat` *defined as the inductive- and coinductive types determined by the type operator $t.\,1+t$.*

References

Robert Harper. Kripke-style logical relations for normalization. Unpublished lecture note, January 2025a. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/kripke.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`.

Robert Harper. Variables types for genericity and abstraction. Unpublished lecture note, January 2025d. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/variabletypes.pdf`.