Higher Order Computability

John Longley & Dag Normann

March 21, 2022

Contents

1 Theory of Computability Models

1.1 Computational Structure in Higher-Order Models

1.1.1 Combinatory Completeness

Combinatory completeness can be seen as a syntactic counterpart to the notion of weakly cartesian closed model. In essence, combinatory completeness asserts that any operation definable by means of a formal expression over A (constructed using application) is representable by an element of A itself.

Definition 1.1. 1. A **partial applicative structure A** consists of

- an inhabited family $|\mathbf{A}|$ of datatypes $A, B, \ldots$ (indexed by some set T)
- a (right-associative) binary operation $\Rightarrow$ on $|\mathbf{A}|$
- for each $A, B \in |\mathbf{A}|$, a partial function $\cdot_{AB} : (A \Rightarrow B) \times A \rightharpoonup B$

2. A **typed partial combinatory algebra** (TPCA) is a partial applicative structure **A** satisfying the following conditions

   (a) For any $A, B \in |\mathbf{A}|$, there exists $k_{AB} \in A \Rightarrow B \Rightarrow A$ s.t.

   $$\forall a.k \cdot a \downarrow, \quad \forall a, b.k \cdot a \cdot b = a$$

   (b) For any $A, B, C \in |\mathbf{A}|$, there exists $s_{ABC} \in (A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C)$ s.t.

   $$\forall f, g.s \cdot f \cdot g \downarrow, \quad \forall f, g, a.s \cdot f \cdot g \cdot a \simeq (f \cdot a) \cdot (g \cdot a)$$

   A **lax TPCA** is obtained from a TPCA change '$\simeq$' to '$\succeq$' in the axiom s

3. If $\mathbf{A}^{\circ}$ denotes a partial applicative structure, a **partial applicative substructure** $\mathbf{A}^{\sharp}$ of $\mathbf{A}^{\circ}$ consists of a subset $A^{\sharp} \subseteq A$ for each $A \in |\mathbf{A}^{\circ}|$ s.t.

   - if $f \in (A \Rightarrow B)^{\sharp}$, $a \in A^{\sharp}$ and $f \cdot a \downarrow$ in $\mathbf{A}^{\circ}$, then $f \cdot a \in B^{\sharp}$

   such a pair $(\mathbf{A}^{\circ}; \mathbf{A}^{\sharp})$ is called a **relative partial applicative structure**

4. A **relative TPCA** is a relative partial applicative structure $(\mathbf{A}^{\circ}, \mathbf{A}^{\sharp})$ s.t. there exist elements k_{AB}, s_{ABC} in $\mathbf{A}^{\sharp}$ witnessing that $\mathbf{A}^{\circ}$ is a TPCA

Definition 1.2. Suppose **A** is a relative partial applicative structure over T

1. The set of well-typed **applicative expressions** $e : \sigma$ over **A** is defined inductively as follows

   - for each $\sigma \in \mathsf{T}$, we have an unlimited supply of variables $x^{\sigma} : \sigma$
   - for each $\sigma \in \mathsf{T}$ and $a \in \mathbf{A}^{\sharp}(\sigma)$, we have a **constant** symbol $c_a : \sigma$ (we shall often write c_a simply as a)
   - If $e : \sigma \to \tau$ and $e' : \sigma$ are applicative expressions, then ee' is an applicative expression of type τ.

   We write $V(e)$ for the set of variables appearing in e

2. A **valuation** in **A** is a function v assigning to certain variables x^{σ} an element $v(x^{\sigma}) \in \mathbf{A}^{\circ}(\sigma)$. Given an applicative expression e and a valuation v covering $V(e)$, the value $[\![e]\!]_v$, when defined, is given inductively by

   $$[\![x^{\sigma}]\!]_v = v(x), \quad [\![c_a]\!]_v = a, \quad [\![ee']\!]_{\nu} \simeq [\![e]\!]_v \cdot [\![e']\!]_v$$

   Note that if $e : \tau$ and $[\![e]\!]_v$ is defined then $[\![e]\!]_v \in \mathbf{A}^{\circ}(\tau)$.

Note that for any v with $\mathrm{ran}(v) \in \mathbf{A}^\sharp$, we can prove $[\![e : \tau]\!]_v \in \mathbf{A}^\sharp(\tau)$ by induction:

1. If e is of the form x^τ
2. If e is of the form c_a where $a \in \mathbf{A}^\sharp(\tau)$
3. If e is of the form $e'e''$ where $e' : \sigma \to \tau$ and $e'' : \sigma$.
   $[\![e]\!]_v = [\![e']\!]_v \cdot [\![e'']\!]_v$ where $[\![e']\!]_v \in \mathbf{A}^\sharp(\sigma \to \tau)$ and $[\![e'']\!]_v \in \mathbf{A}^\sharp(\sigma)$. Since $\mathbf{A}^\sharp$ is a substructure of $\mathbf{A}^\circ$, if $[\![e']\!]_v \cdot [\![e'']\!]_v \downarrow$, then $[\![e]\!] \in \mathbf{A}^\sharp(\tau)$

Definition 1.3. Let $\mathbf{A}$ be a relative partial applicative structure. We say $\mathbf{A}$ is **lax combinatory complete** if for every applicative expression $e : \tau$ over $\mathbf{A}$ and every variable x^σ, there is an applicative expression $\lambda^* x^\sigma.e$ with $V(\lambda^* x^\sigma.e) = V(e) - \{x^\sigma\}$ s.t. for any valuation v covering $V(\lambda^* x^\sigma.e)$ and any $a \in \mathbf{A}^\circ(\sigma)$ we have

$$[\![\lambda^* x^\sigma.e]\!]_v \downarrow, \quad [\![\lambda^* x^\sigma.e]\!]_v \cdot a \succeq [\![e]\!]_{v,x\mapsto a}$$

We say $\mathbf{A}$ is **strictly combinatory complete** if this holds with '$\simeq$' in place of '$\succeq$'

Theorem 1.4. *A (relative) partial applicative structure **A** is a lax (relative) TPCA iff it is lax combinatory complete*

Proof. If $\mathbf{A}$ is lax combinatory complete, then for any ρ, σ, τ we may define

$$\begin{aligned} k_{\sigma\tau} &= [\![\lambda^* x^\sigma.(\lambda^* y^\tau.x)]\!]_\emptyset \\ s_{\rho\sigma\tau} &= [\![\lambda^* x^{\rho\to\sigma\to\tau}.(\lambda^* y^{\rho\to\sigma}.(\lambda^* z^\rho.xz(yz)))]\!]_\emptyset \end{aligned}$$

Conversely, if $\mathbf{A}$ is a lax TPCA, then given any suitable choice of elements k and s for $\mathbf{A}$, we may define $\lambda^* x^\sigma.e$ by induction on the structure of e:

$$\begin{aligned} \lambda^* x^\sigma.x &= s_{\sigma(\sigma\to\sigma)} k_{\sigma(\sigma\to\sigma)} k_{\sigma\sigma} && \\ \lambda^* x^\sigma.a &= k_{\tau\sigma} a && \text{for each } a \in \mathbf{A}^\sharp(\tau) \\ \lambda^* x^\sigma.ee' &= s_{\sigma\tau\tau'}(\lambda^* x^\sigma.e)(\lambda^* x^\sigma.e') && \text{if } e : \tau \to \tau', e' : \tau \text{ and } ee' \text{ contains } x \end{aligned}$$

□

The same argument shows that $\mathbf{A}$ is a strict TPCA iff it is strictly combinatory complete we often tacitly suppose that a TPCA **A** comes equipped with some choice of k and s drawn from A♯, and in this case we shall use the notation $\lambda^* x.e$ for the applicative expression given by the above proof. Since all the constants appearing in e are drawn from $A^\sharp$, the same will be true for $\lambda^* x.e$.

In TPCAs constructed as syntactic models for untyped or typed λ-calculi (as in Example 3.1.6 or Section 3.2.3), the value of $\lambda^* x.e$ coincides with $\lambda x.e$. However, the notational distinction is worth retaining, since the term $\lambda^* x.e$ as defined above is not syntactically identical to $\lambda x.e$.

More generally, we may consider terms of the λ-calculus as **meta-expressions** for applicative expressions. Specifically any such λ-term M can be regarded as denoting an applicative expression $M^\dagger$ as follows:

$$x^\dagger = x, \quad c_a^\dagger = c_a, \quad (MN)^\dagger = M^\dagger N^\dagger, \quad (\lambda x.M)^\dagger = \lambda^* x.(M^\dagger)$$

Some caution is needed here, however, because β-equivalent meta-expressions do not always have the same meaning

Example 1.1. Consider the two meta-expressions $(\lambda x.(\lambda y.y)x)$ and $\lambda x.x$. Although these are β-equivalent, the first expands to $s(ki)i$ and the second to i, where $i \equiv skk$.

The moral here is that β-reductions are not valid underneath λ^*-abstractions: in this case, the reduction $(\lambda^* y.y)x \rightsquigarrow x$ is not valid underneath λ^*. However at least for the definition of λ^* given above, β-reductions at top level are valid.

Proposition 1.5. *1. If M is a meta-expression, x is a variable and a is a constant or variable, then $[\![((\lambda x.M)a)^\dagger]\!]_v \succeq [\![M[x \mapsto a]^\dagger]\!]$*

2. If M, N are meta-expressions, $x \notin FV(N)$, no free occurrence of x in M occurs under a λ, and $[\![N^\dagger]\!]_v \downarrow$, then $[\![((\lambda x.M)N)^\dagger]\!]_v \succeq [\![M[x \mapsto N]^\dagger]\!]_v$

Proof. Longley's PhD thesis □

From now on, we will not need to distinguish formally between meta-expressions and the applicative expressions they denote. For the remainder of this chapter we shall use the λ^* notation for such (meta-)expressions, retaining the asterisk as a reminder that the usual rules of λ-calculus are not always valid.

1.1.2 Pairing

Definition 1.6. 1. A **type world** is simply a set T of **type names** σ, optionally endowed with any or all of the following:

   (a) a **fixing map**, assigning a set $\mathsf{T}[\sigma]$ to certain type names $\sigma \in \mathsf{T}$

   (b) a **product structure**, consisting of a total binary operation $(\sigma, \tau) \mapsto \sigma \times \tau$

   (c) an **arrow structure**, consisting of a total binary operation $(\sigma, \tau) \mapsto \sigma \to \tau$

2. A **computability model over** a type world T is a computability model **C** with index set T (so that $|\mathbf{C}| = \{\mathbf{C}(\sigma) \mid \sigma \in \mathsf{T}\}$) subject to the following conventions

   (a) If T has a fixing map, then $\mathbf{C}(\sigma) = T[\sigma]$ whenever $\mathsf{T}(\sigma)$ is defined

   (b) If T has a product structure, then **C** has weak products and for any $\sigma, \tau \in \mathsf{T}$ we have $\mathbf{C}(\sigma \times \tau) = \mathbf{C}(\sigma) \bowtie \mathbf{C}(\tau)$

   (c) If T has an arrow structure, then **C** is a higher-order model and for any $\sigma, \tau \in \mathsf{T}$ we have $\mathbf{C}(\sigma \to \tau) = \mathbf{C}(\sigma) \Rightarrow \mathbf{C}(\tau)$

   (d) If T has both a product and an arrow structure, then **C** is weakly cartesian closed

Theorem 1.7. *There is a canonical bijection between higher-order models and relative TPCAs*

Let **A** be a relative TPCA (which is combinatory complete) over a type world T with arrow structure, and suppose that **A** (considered as a higher-order model) has weak products, inducing a product structure $\times$ on T. This means that for any $\sigma, \tau \in \mathsf{T}$ there are elements

$$fst \in \mathbf{A}^\sharp((\sigma \times \tau) \to \sigma), \quad snd \in \mathbf{A}^\sharp((\sigma \times \tau) \to \tau)$$

And for each $\sigma, \tau \in \mathsf{T}$ a **paring** operation

$$pair \in \mathbf{A}^\sharp(\sigma \to \tau \to (\sigma \times \tau))$$

s.t.

$$\forall a \in \mathbf{A}^\circ(\sigma), b \in \mathbf{A}^\circ(\tau).\ fst \cdot (pair \cdot a \cdot b) = a \wedge snd \cdot (pair \cdot a \cdot b) = b$$

Proposition 1.8. *A higher-order model with weak products has pairing iff it is weakly cartesian closed*

Lemma 1.9 (**??**). *Suppose $m, n > 0$. Given*

$$f_j \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow B_j)^\sharp, \quad (j = 0, \ldots, n-1),$$
$$g \in (B_0 \Rightarrow \cdots \Rightarrow B_{n-1} \Rightarrow C)^\sharp$$

there exists $h \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow C)^\sharp$ s.t.

$$\forall a_0, \ldots, a_{m-1}. h \cdot a_0 \cdot \cdots \cdot a_{m-1} \simeq g \cdot (f_0 \cdot a_0 \cdot \cdots \cdot a_{m-1}) \cdot \cdots \cdot (f_{n-1} \cdot a_0 \cdot \cdots \cdot a_{m-1})$$

Proof. The binary partial functions representable in $\mathbf{A}^\sharp((\rho \times \sigma) \to \tau)$ are exactly those representable in $\mathbf{A}^\sharp(\rho \to \sigma \to \tau)$

Given $f \in \mathbf{A}^\sharp((\rho \times \sigma) \to \tau)$, by Proposition **??**, we have $h \in \mathbf{A}^\sharp(\rho \to \sigma \to \tau)$ where

$$\forall a, b.\ h \cdot a \cdot b \simeq f \cdot (pair\ \cdot a \cdot b)$$

Given $f \in \mathbf{A}^\sharp(\rho \to \sigma \to \tau)$, by the same Proposition, we have $h \in \mathbf{A}^\sharp((\rho \times \sigma) \to \tau)$ where

$$\forall a, b.\ h \cdot c \simeq f \cdot (fst \cdot c) \cdot (snd \cdot c)$$

□

Henceforth we shall generally work with pair in preference to the 'external' pairing of operations, and will write $pair \cdot a \cdot b$ when there is no danger of confusion.

In untyped models, pairing is automatic

$$pair = \lambda^* xyz.zxy, \quad fst = \lambda^* p.p(\lambda^* xy.x), \quad snd = \lambda^* p.p(\lambda^* xy.y)$$

1.1.3 Booleans

Definition 1.10. A model $\mathbf{A}$ has **booleans** if for some type $\mathtt{B}$ there exist elements

$$\mathit{tt}, \mathit{ff} \in \mathbf{A}^\sharp(\mathtt{B})$$
$$\mathit{if}_\sigma \in \mathbf{A}(\mathtt{B}, \sigma, \sigma \to \sigma) \text{ for each } \sigma$$

s.t. for all $x, y \in \mathbf{A}^\circ(\sigma)$ we have

$$\mathit{if}_\sigma \cdot \mathit{tt} \cdot x \cdot y = x, \quad \mathit{if}_\sigma \cdot \mathit{ff} \cdot x \cdot y = y$$

Note that tt, ff need not be the sole element of $\mathbf{A}^\sharp(\mathbf{B})$

Alternatively, we may define a notion of having booleans in the setting of computability model **C** with weak products: replace if_σ with $\mathit{if}'_\sigma \in \mathbf{C}[\mathbb{B} \times \sigma \times \sigma, \sigma]$. In a TPCA with products and pairing the two definitions coincide

In untyped models, the existence of booleans is automatic: $\mathit{tt} = \lambda^* xy.x$, $\mathit{ff} = \lambda^* xy.y$ and $\mathit{if} = \lambda^* zxy.zxy$

Obviously, the value of an expression $\mathit{if}_\sigma \cdot b \cdot e \cdot e'$ cannot be defined unless the values of both e and e' are defined. However, there is a useful trick that allows us to build conditional expressions whose definedness requires only that the chosen branch of the conditional is defined. This trick is specific to the higher-order setting, and is known as **strong definition by cases**:

Proposition 1.11. *Suppose* **A** *has booleans as above. Given applicative expressions* $e, e' : \sigma$ *there is an applicative expression* $(e \mid e') : \mathtt{B} \to \sigma$ *s.t. for any valuation* v *covering* $V(e)$ *and* $V(e')$ *we have*

$$[\![(e \mid e')]\!]_v \downarrow, \quad [\![(e \mid e') \cdot \mathit{tt}]\!]_v \succeq [\![e]\!]_v, \quad [\![(e \mid e') \cdot \mathit{ff}]\!]_v \succeq [\![e']\!]_v$$

Proof. Let ρ be any type s.t. $\mathbf{A}^\circ(\rho)$ is inhabited by some element a, and define

$$(e \mid e') = \lambda^* z^{\mathtt{B}} \cdot (\mathit{if}_\sigma z(\lambda^* r^\rho.e)(\lambda^* r^\rho.e')c_a)$$

where z, r are fresh variables

$[\![(e \mid e')]\!]_v \downarrow$ since by lax combinatory completeness

$[\![(e \mid e') \cdot \mathit{tt}]\!]_v \succeq [\![e]\!]_v$ by 1.5 □

The expressions $\lambda^* r.e$, $\lambda^* r.e'$ in the above proof are known as **suspensions** or **thunks**: the idea is that $[\![\lambda^* r.e]\!]_v$ is guaranteed to be defined, but the actual evaluation of e_v (which may be undefined) is 'suspended' until the argument c_a is supplied.

1.1.4 Numerals

Definition 1.12. A model **A** has **numerals** if for some type $\mathtt{N}$ there exist

$$\hat{0}, \hat{1}, \hat{2}, \ldots \in \mathbf{A}^\sharp(\mathtt{N})$$
$$\mathit{suc} \in \mathbf{A}^\sharp(\mathtt{N} \to \mathtt{N})$$

and for any $x \in \mathbf{A}^\sharp(\sigma)$ and $f \in \mathbf{A}^\sharp(\mathtt{N} \to \sigma \to \sigma)$ an element

$$\mathit{Rec}_\sigma(x, f) \in \mathbf{A}^\sharp(\mathtt{N} \to \sigma)$$

s.t. for all $x \in \mathbf{A}^\sharp(\sigma)$, $f \in \mathbf{A}^\sharp(\mathtt{N} \to \sigma \to \sigma)$ and $n \in \mathbb{N}$ we have

$$\begin{aligned} suc \cdot \hat{n} &= \widehat{n+1} \\ Rec_\sigma(x, f) \cdot \hat{0} &= x \\ Rec_\sigma(x, f) \cdot \widehat{n+1} &\succeq f \cdot \hat{n} \cdot (Rec_\sigma(x, f) \cdot \hat{n}) \end{aligned}$$

The above definition has the advantage that it naturally adapts to the setting of a computability model C with products: just replace the types of f and $Rec_\sigma(x, f)$ above with $\mathbf{C}[\mathtt{N} \times \sigma, \sigma]$ and $\mathbf{C}[\mathtt{N}, \sigma]$ respectively.

Proposition 1.13. *A model **A** has numerals iff it has elements $\hat{n}$, suc as above and*

$$rec_\sigma \in \mathbf{A}^\sharp(\sigma \to (N \to \sigma \to \sigma) \to N \to \sigma) \quad \text{for each } \sigma$$

s.t. for all $x \in \mathbf{A}^\circ(\sigma)$, $f \in \mathbf{A}^\circ(N \to \sigma \to \sigma)$ and $n \in \mathbb{N}$ we have

$$\begin{aligned} suc \cdot \hat{n} &= \widehat{n+1} \\ rec_\sigma \cdot x \cdot f \cdot \hat{0} &= x \\ rec_\sigma \cdot x \cdot f \cdot \widehat{n+1} &\succeq f \cdot \hat{n} \cdot (rec_\sigma \cdot x \cdot f \cdot \hat{n}) \end{aligned}$$

Proof. $\Leftarrow$: Let $Rec_\sigma(x, f) = rec_\sigma \cdot x \cdot f$

$\Rightarrow$: define

$$rec_\sigma = Rec_{\sigma \to (\mathtt{N} \to \sigma \to \sigma) \to \sigma}(\lambda^* xf.x, \lambda^* nr.\lambda^* xf.fn(rxf))$$

? $\square$

Exercise 1.1.1. Show that **A** has numerals, then **A** has booleans

Proposition 1.14. *Every untyped model has numerals*

Proof. Using the encodings for pairings and booleans given above, we may define the **Curry numerals** $\hat{n}$ in any untyped models as follows:

$$\hat{0} = \langle tt, tt \rangle, \quad \widehat{n+1} = \langle ff, \hat{n} \rangle$$

and $suc = \lambda^* x.\langle ff, x \rangle$. We also have elements for the zero testing and predecessor operations: take $iszero = fst$ and $pre = \lambda^* x.if(iszero\ x)\hat{0}(snd\ x)$ $\square$

In any model with numerals, a rich class of functions $\mathbb{N}^r \to \mathbb{N}$ is representable. For example, the (first-order) primitive recursive functions on $\mathbb{N}$

Proposition 1.15. *For any primitive recursive $f : \mathbb{N}^r \to \mathbb{N}$ there is an applicative expression $e_f : \mathbb{N}^{(r)} \to \mathbb{N}$ (involving constants 0, suc, $rec_{\mathtt{N}}$) s.t. in any model $(\boldsymbol{A}^\circ; \boldsymbol{A}^\sharp)$ with numerals we have $[\![e_f]\!]_v \in \boldsymbol{A}^\sharp$ (where v is the obvious valuation of the constants) and*

$$\forall n_0, \dots, n_{r-1}, m. f(n_0, \dots, n_{r-1}) = m \Rightarrow [\![e_f]\!]_v \cdot \hat{n}_0 \cdot \hat{n}_{r-1} = \hat{m}$$

Proof. □

1.1.5 Recursion and Minimization

Definition 1.16.
1. A total model **A has general recursion**, or **has fixed points**, if for every element $f \in \mathbf{A}^\sharp(\rho \to \rho)$ there is an element $Fix_\rho(f) \in \mathbf{A}^\sharp(\rho)$ s.t. $Fix_\rho(f) = f \cdot Fix_\rho(f)$
2. An arbitrary model **A has guarded recursion**, or **guarded fixed points**, if for every element $f \in \mathbf{A}^\sharp(\rho \to \rho)$ where $\rho = \sigma \to \tau$ there is an element $GFix_\rho(f) \in \mathbf{A}^\sharp(\rho)$ s.t. $GFix_\rho(f) \cdot x \succeq f \cdot GFix_\rho(f) \cdot x$ for all $x \in \mathbf{A}^\circ(\sigma)$

Proposition 1.17.
1. *A total model $\boldsymbol{A}$ has general recursion iff for every type ρ there is an element $Y_\rho \in \boldsymbol{A}^\sharp((\rho \to \rho) \to \rho)$ s.t. for all $f \in \boldsymbol{A}^\circ(\rho \to \rho)$ we have*
$$Y_\rho \cdot f = f \cdot (Y_\rho \cdot f)$$
2. *$\boldsymbol{A}$ has guarded recursion iff for every type $\rho = \sigma \to \tau$ there is an element $Z_\rho \in \boldsymbol{A}^\sharp((\rho \to \rho) \to \rho)$ s.t. for all $f \in \boldsymbol{A}^\circ(\rho \to \rho)$ and $x \in \boldsymbol{A}^\circ(\sigma)$ we have*
$$Z_\rho \cdot f \downarrow, \quad Z_\rho \cdot f \cdot x \succeq f \cdot (Z_\rho \cdot f) \cdot x$$

Proof. Define

$$Y_\rho = Fix_{(\rho \to \rho) \to \rho}(\lambda^* y. \lambda^* f. f(yf)), \quad Z_\rho = GFix_{(\rho \to \rho) \to \rho}(\lambda^* z. \lambda^* fx. f(zf)x)$$

□

Not all models of interest possess such recursion operators. Clearly, if **A** is a **total** model with $\mathbf{A}(\mathtt{N}) = \mathbb{N}$ a type of numerals as above, then **A** cannot have general or even guarded recursion: if $\rho = \mathtt{N} \to \mathtt{N}$ and $f = \lambda^* gx.suc(gx)$ then we would have $Z \cdot f \cdot \hat{n} = suc \cdot Z \cdot f \cdot \hat{n}$, which is impossible. However, many models with $\mathbf{A}(\mathtt{N}) = \mathbb{N}_\perp$ will have general recursion

Any **untyped** total model has general recursion, since we may take

$$W = \lambda^* wf.f(wwf), \quad Y = WW$$

(This element Y is known as the **Turing fixed point combinator**). Likewise, every untyped model, total or not, has guarded recursion, since we may take

$$V = \lambda^* vfx.f(vvf)x, \quad Z = VV$$

Note in passing that Kleene's **second recursion theorem** from classical computability theory is tantamount to the existence of a guarded recursion operator in K_1

We can now prove 1.14. In any untyped model, let Z be a guarded recursion operator, define

$$R = \lambda^* rxfm.\mathit{if}(\mathit{iszero}\ m)(kx)(\lambda^* y.f(\mathit{pre}\ m))(rxf(\mathit{pre}\ m)\hat{0})$$

and take $rec = \lambda^* xfm.(ZR)xfmi$.

Definition 1.18. A model **A** with numerals **has minimization** if it contains an element $min \in \mathbf{A}^\sharp((\mathtt{N} \to \mathtt{N}) \to \mathtt{N})$ s.t. whenever $\hat{g} \in \mathbf{A}^\circ(\mathtt{N} \to \mathtt{N})$ represents some total $g : \mathbb{N} \to \mathbb{N}$ and m is the least number s.t. $g(m) = 0$, we have $min \cdot \hat{g} = \hat{m}$

Proposition 1.19. *There is an applicative expression Min involving constants $\hat{0}$, suc, iszero, if and Z s.t. in any model with numerals and guarded recursion, $[\![Min]\!]_v$ is a minimization operator*

Proof. Take $Min = Z(\lambda^* M.\lambda^* g.\mathit{if}(\mathit{iszero}(g\ \hat{0}))\hat{0}(M(\lambda^* n.g(\mathit{suc}\ n))))$ □

Proposition 1.20. *For any partial computable $f : \mathbb{N}^r \rightharpoonup \mathbb{N}$ there is an applicative expression $e_f : \mathtt{N}^{(r)} \to \mathtt{N}$ (involving constants 0, suc, $rec_{\mathtt{N}}$, min) s.t. in any model **A** with numerals and minimization we have $[\![e_f]\!]_v \in \mathbf{A}^\sharp$ (with the obvious valuation v) and*

$$\forall n_0, \dots, n_{r-1}, m. f(n_0, \dots, n_{r-1}) = m \Rightarrow [\![e_f]\!]_v \cdot \hat{n}_0 \cdot \dots \cdot \hat{n}_{r-1} = \hat{m}$$

Proof. Since our definition of minimization refers only to total functions $g : \mathbb{N} \to \mathbb{N}$, we appeal to the *Kleene normal form* theorem: there are primitive recursive functions $T : \mathbb{N}^{r+2} \to \mathbb{N}$ and $U : \mathbb{N} \to \mathbb{N}$ such that any partial computable f has an 'index' $e \in \mathbb{N}$ such that $f(\bar{n}) \simeq U(\mu y.T(e, \bar{n}, y) = 0)$ for all $\bar{n}$. Using this, the result follows easily from Propositions 1.15 and 1.19. □

1.1.6 The Category of Assemblies

Definition 1.21. Let **C** be a lax computability model over T. The **category of assemblies over C**, written $\mathcal{A}\mathrm{sm}(\mathbf{C})$ is defined as follows:

- Objects X are triples $(|X|, \rho_X, \Vdash_X)$ where $|X|$ is a set, $\rho_X \in T$ names some type, and $\Vdash_X \subseteq \mathbf{C}(\rho_X) \times |X|$ is a relation s.t. $\forall x \in |X|. \exists a \in \mathbf{C}(\rho_X). a \Vdash_X x$ (The formula $a \Vdash_X x$ may be read as 'a **realizes** x')
- A morphism $f : X \to Y$ is a function $f : |X| \to |Y|$ that is **tracked** by some $\overline{f} \in \mathbf{C}[\rho_X, \rho_Y]$, in the sense that for any $x \in |X|$ and $a \in \mathbf{C}(\rho_X)$ we have
$$a \Vdash_X x \Rightarrow \overline{f}(a) \Vdash_Y f(x)$$

An assembly X is called **modest** if $a \Vdash_X x \wedge a \Vdash_X x'$ implies $x = x'$. We write $\mathcal{M}\mathrm{od}(\mathbf{C})$ for the full subcategory of $\mathcal{A}\mathrm{sm}(\mathbf{C})$ consisting of modest assemblies

Intuitively, we regard an assembly X as an "abstract datatype" for which we have a concrete implementation on the "machine" **C**. The underlying set $|X|$ is the set of values of the abstract type, and for each $x \in |X|$, the elements $a \Vdash_X x$ are the possible machine representations of this abstract value. (Note that an abstract value x may have many possible machine representations a.) The morphisms $f : X \to Y$ may then be regarded as the "computable mappings" between such datatypes

In the case that **C** is a lax TPCA **A**, we may also denote the above categories $\mathcal{A}\mathrm{sm}(\mathbf{A})$, $\mathcal{M}\mathrm{od}(\mathbf{A})$, or by $\mathcal{A}\mathrm{sm}(\mathbf{A}^\circ; \mathbf{A}^\sharp)$, $\mathcal{M}\mathrm{od}(\mathbf{A}^\circ; \mathbf{A}^\sharp)$. Note that realizers for elements $x \in |X|$ may be arbitrary elements of $\mathbf{A}^\circ(\rho_X)$, whereas a morphism $f : X \to Y$ must be tracked by an element of $\mathbf{A}^\sharp(\rho_X \to \rho_Y)$

Viewed in this way, all the datatypes we shall typically wish to consider in fact live in the subcategory $\mathcal{M}\mathrm{od}(\mathbf{C})$: an abstract data value is uniquely determined by any of its machine representations. Note also that if Y is modest, a morphism $f : X \to Y$ is completely determined by any $\overline{f}$ that tracks it.

Definition 1.22. Let the category **C** have binary products. An **exponential** of objects B and C consists of an object C^B and an arrow $\epsilon : C^B \times B \to C$ s.t. for any object A and arrow $f : A \times B \to C$ there is a unique arrow $\tilde{f} : A \to C^B$ s.t. $\epsilon \circ (\tilde{f} \times 1_B) = f$

$$\begin{tikzcd} C^B & C^B \times B \arrow[r, "\epsilon"] & C \\ A \arrow[u, "\tilde{f}"'] & A \times B \arrow[u, "\tilde{f} \times 1_B"] \arrow[ur, "f"'] & \end{tikzcd}$$

Theorem 1.23. *Let **C** be a lax computability model*

1. *If **C** has a weak terminal, then $\mathcal{A}\mathrm{sm}(\boldsymbol{C})$ has a terminal object 1*
2. *If **C** has weak products, then $\mathcal{A}\mathrm{sm}(\boldsymbol{C})$ has binary cartesian products*
3. *If **C** weakly cartesian closed, then $\mathcal{A}\mathrm{sm}(\boldsymbol{C})$ is cartesian closed*
4. *If **C** has a weak terminal and booleans, $\mathcal{A}\mathrm{sm}(\boldsymbol{C})$ has the coproduct $1+1$*
5. *If **C** has a weak terminal and numerals, $\mathcal{A}\mathrm{sm}(\boldsymbol{C})$ has a natural number object*

Proof. 1. If (I, i) is a weak terminal, define $1 = (\{i\}, I, \Vdash_1 = \{(i, i)\})$. Then for any $X \in \mathcal{A}\mathrm{sm}(\mathbf{C})$, $f = \Lambda x.i$ is the unique morphism where $\overline{f} = \Lambda x.i$.

2. If X and Y are assemblies and ρ is a weak product of ρ_X and ρ_Y, define the assembly $X \times Y$ by

$$|X \times Y| = |X| \times |Y|, \quad \rho_{X\times Y} = \rho, \quad a \Vdash_{X\times Y} (x, y) \text{ iff } \pi_X(a) \Vdash_X x \wedge \pi_Y(a) \Vdash_Y y$$

3. If X and Y are assemblies, let us say an element $t \in \mathbf{C}(\rho_X \to \rho_Y)$ **tracks** a function $f : |X| \to |Y|$ if

$$\forall x \in |X|, a \in \mathbf{C}(\rho_X).\ a \Vdash_X x \Rightarrow t \cdot_{XY} a \Vdash_Y f(x)$$

Now define the assembly Y^X as follows:

$$\left|Y^X\right| = \{f : |X| \to |Y| \mid f \text{ is tracked by some } t \in \mathbf{C}(\rho_X \to \rho_Y)\}$$
$$\rho_{Y^X} = \rho_X \to \rho_Y$$
$$t \Vdash_{Y^X} f \Leftrightarrow t \text{ tracks } f$$

□

Theorem **??** also holds with $\mathcal{M}\mathrm{od}(\mathbf{C})$, and the inclusion $\mathcal{M}\mathrm{od}(\mathbf{C}) \hookrightarrow \mathcal{A}\mathrm{sm}(\mathbf{C})$ preserves all the relevant structure