Higher Order Computability

John Longley & Dag Normann

March 6, 2022

Contents

1 Theory of Computability Models

- $e \downarrow$ 'the value of e is defined'
- $e \uparrow$ 'the value of e is undefined'
- $e = e'$ 'the values of both e and e' are defined and they are equal'
- $e \simeq e'$ 'if either e or e' is defined then so is the other and they are equal'
- $e \succeq e'$ 'if e' is defined then so it e and they are equal'

if e is a mathematical expression possibly involving the variable x, we write $\Lambda x.e$ to mean the ordinary (possibly partial) function f defined by $f(x) \simeq e$

Finite sequences of length n starts from index 0.

1.1 Higher-Order Computability Models

1.1.1 Computability Models

Definition 1.1. A **computability model C** over a set T of **type names** consists of

- an indexed family $|\mathbf{C}| = \{\mathbf{C}(\tau) \mid \tau \in \mathsf{T}\}$ of sets, called the **datatypes** of **C**
- for each $\sigma, \tau \in \mathsf{T}$, a set $\mathbf{C}[\sigma, \tau]$ of partial functions $f : \mathbf{C}(\sigma) \to \mathbf{C}(\tau)$, called the **operations** of **C**

s.t.

1. for each $\tau \in \mathsf{T}$, the identity function $\mathrm{id} : \mathbf{C}(\tau) \to \mathbf{C}(\tau)$ is in $\mathbf{C}(\tau, \tau)$
2. for any $f \in \mathbf{C}[\rho, \sigma]$ and $g \in \mathbf{C}[\sigma, \tau]$ we have $g \circ f \in \mathbf{C}[\rho, \tau]$ where $\circ$ denotes ordinary composition of partial functions

We shall use uppercase letters $A, B, C, ...$ to denote **occurrences** of sets within $|\mathbf{C}|$: that is, sets $\mathbf{C}(\tau)$ implicitly tagged with a type name τ. We shall write $\mathbf{C}[A, B]$ for $\mathbf{C}[\sigma, \tau]$ if $A = \mathbf{C}(\sigma)$ and $B = \mathbf{C}(\tau)$

In typical cases of interest, the operations of **C** will be 'computable' maps of some kind between datatypes

Definition 1.2. A computability model **C** is **total** if every operation $f \in \mathbf{C}[A, B]$ is a total function $f : A \to B$

Definition 1.3. A computability model **C** has **weak (binary cartesian) products** if there is an operation assigning to each $A, B \in |\mathbf{C}|$ a datatype $A \bowtie B \in |\mathbf{C}|$ along with operations $\pi_A \in \mathbf{C}[A \bowtie B, A]$ and $\pi_B \in \mathbf{C}[A \bowtie B, B]$ (known as **projections**) s.t. for any $f \in \mathbf{C}[C, A]$ and $g \in \mathbf{C}[C, B]$ there exists $\langle f, g \rangle \in \mathbf{C}[C, A \bowtie B]$ satisfying the following for all $c \in C$

1. $\langle f, g \rangle(c) \downarrow$ iff $f(c) \downarrow$ and $g(c) \downarrow$
2. $\pi_A(\langle f, g \rangle(c)) = f(c)$ and $\pi_B(\langle f, g \rangle(c)) = g(c)$

We say that $d \in A \bowtie B$ **represents** the pair (a, b) if $\pi_A(d) = a$ and $\pi_B(d) = b$

In contrast to the usual definition of categorical products, the operation $\langle f, g \rangle$ need not be unique, since many elements of $A \bowtie B$ may represent the same pair (a, b). We do not formally require that every (a, b) is represented in $A \bowtie B$, though in all cases of interest this will be so. The reader is also warned that $\pi_A \circ \langle f, g \rangle$ will not in general coincide with f .

TODO: examples different bijections from $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

Definition 1.4. A **weak terminal** in a computability model $\mathbf{C}$ consists of a datatype $I \in |\mathbf{C}|$ and an element $i \in I$ s.t. for any $A \in |\mathbf{C}|$ the constant function $\Lambda a.i$ is in $\mathbf{C}[A, I]$

If $\mathbf{C}$ has weak products and a weak terminal (I, i), then for any $A \in |\mathbf{C}|$ there is an operation $t_A \in \mathbf{C}[A, I \bowtie A]$ s.t. $\pi_A \circ t_A = \mathrm{id}_A$

1.1.2 Examples of Computability Models

Example 1.1. Model with single datatype $\mathbb{N}$ and whose operations $\mathbb{N} \rightharpoonup \mathbb{N}$ are precisely the Turing-computable partial functions. The model has standard products, since the well-known computable pairing operation

$$\langle m, n \rangle = (m + n)(m + n + 1)/2 + m$$

defines a bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Any element $i \in \mathbb{N}$ may serve as a weak terminal, since $\Lambda n.i$ is computable

Example 1.2. untyped λ-calculus

Terms M of the λ-calculus are generated from a set of variable symbols x by means of the following grammar:

$$M ::= x \mid MM' \mid \lambda x.M$$

Writing L for the quotient set $\Lambda / =_\beta$

We write $M[x \mapsto N]$ for the result of substituting N for all free occurrences of x within M

We define Λ to be the set of untyped λ-terms modulo α-equivalence.

Let $\sim$ be any equivalence relation on Λ with the following properties:

$$(\lambda x.M)N \sim M[x \mapsto N], \quad M \sim N \Rightarrow PM \sim PN$$

1. $(\lambda x.x)M \sim M$
2. If $M \sim N$, then $(\lambda x.N)M \sim (\lambda x.M)N$ and hence $N \sim M$.
3. If $M \sim N$ and $N \sim O$, then

Then we have $M \sim N \Rightarrow MP \sim NP$ since $(\lambda y.yP)M \sim (\lambda y.yP)N \Rightarrow MP \sim NP$.

As a example, we may define $=_\beta$ to be the smallest equivalence relation $\sim$ satisfying the above properties and also

$$M \sim N \Rightarrow \lambda x.M \sim \lambda x.N$$

Writing $[M]$ for the $\sim$-equivalence class of M, any term $P \in A$ induces a well-defined mapping $[M] \mapsto [PM]$ on $\Lambda/\sim$. The mappings induced by some P in this way are called **λ-definable**

We may regard $\Lambda/\sim$ as a total computability model: the sole datatype is $\Lambda/\sim$ itself, and the operations on it are exactly the λ-definable mappings. It also has weak products: a pair (M, N) may be represented by the term $pair\ M\ N$ where $pair = \lambda xyz.zxy$ the terms $fst = \lambda p.p(\lambda xy.x)$ and $snd = \lambda p.p(\lambda xy.y)$. We can check that $fst(pair\ M\ N) \sim M$ and $snd(pair\ M\ N) \sim N$

We can also obtain a submodel $\Lambda^0/\sim$ consisting of the equivalence classes of closed terms M

Example 1.3. Let B be any family of **base sets**, and let $\langle B\rangle$ denote the family of sets generated from B by adding the singleton set $1 = \{()\}$ and closing under binary products $X \times Y$ and set-theoretic function spaces Y^X. We shall consider some computability models whose family of datatypes is $\langle B\rangle$

First we may define a computability model $\mathsf{S}(B)$ with $|\mathsf{S}(B)| = \langle B\rangle$ (often called the **full set-theoretic model over** B) by letting $\mathsf{S}(B)[X, Y]$ consist of all set-theoretic functions $X \to Y$ for $X, Y \in \langle B\rangle$; that is, we consider all functions to be computable. However this model is of limited interest since it does not represent an interesting concept of computability

To do better we may start by noting that whatever the 'computable' functions between these sets are supposed to be, it is reasonable to expect that they will enjoy the following closure properties

1. For any $X \in \langle B\rangle$, the unique function $X \to 1$ is computable
2. For any $X, Y \in \langle B\rangle$, the projections $X \times Y \to X$, $X \times Y \to Y$ is computable
3. For any $X, Y \in \langle B\rangle$, the application function $Y^X \times X \to Y$ is computable
4. If $f : Z \to X$ and $g : Z \to Y$ is computable, so is their pairing $(f, g) : Z \to X \times Y$

5. If $f : X \to Y$ and $g : Y \to Z$ are computable, so is their composition $g \circ f : X \to Z$

6. If $f : Z \times X \to Y$ is computable, so is its transpose $\hat{f} : Z \to Y^X$

One possible approach is therefore to start by specifying some set C of functions between out datatypes that we wish to regard as "basic computable operations", and define a computability model $\mathsf{K}(B; C)$ over $\langle B \rangle$ whose operations are exactly the functions generated from C under the above closure conditions

Take $B = \{\mathbb{N}\}$; we shall often denote $\mathsf{S}(\{\mathbb{N}\})$ by S. Let C consist of the following basic operations: the zero function $\Lambda x.0 : \mathbb{N} \to 1$, the successor function $suc : \mathbb{N} \to \mathbb{N}$; and for each $X \in \langle B \rangle$, the primitive recursion operator $rec_X : (X \times X^{X \times \mathbb{N}} \times \mathbb{N}) \to X$ defined by

$$rec_X(x, f, 0) = 0$$
$$rec_X(x, f, n+1) = f(rec_X(x, f, n), n)$$

1.1.3 Weakly Cartesian Closed Models

Definition 1.5. Suppose **C** has weak products and a weak terminal. We say **C** is **weakly cartesian closed** if it is endowed with the following for each $A, B \in |\mathbf{C}|$:

- a choice of datatype $A \Rightarrow B \in |\mathbf{C}|$
- a partial function $\cdot_{AB} : (A \Rightarrow B) \times A \rightharpoonup B$, external to the structure of **C**

s.t. for any partial function $f : C \times A \rightharpoonup B$ the following are equivalent

1. f is represented by some $\bar{f} : \mathbb{C}[C \bowtie A, B]$, in the sense that if d represents (c, a) then $\bar{f}(d) \simeq f(c, a)$

2. f is represented by some total operation $\hat{f} : \mathbf{C}[C, A \Rightarrow B]$, in the sense that
$$\forall c \in C, a \in A \quad \hat{f}(c) \cdot_{AB} a \simeq f(c, a)$$

$\cdot_{AB}$ is represented by an operation $app_{AB} \in \mathbb{C}[(A \Rightarrow B) \bowtie A, B]$

Crucially, and in contrast to the definition of cartesian closed category, there is no requirement that f is unique. This highlights an important feature of our framework: in many models of interest, elements of $A \Rightarrow B$ will be **intensional** objects (programs or algorithms), and there may be many intensional objects giving rise to the same partial function $A \to B$

Example 1.4. Consider again the model of Example 1.1, comprising the partial Turing-computable functions $\mathbb{N} \rightharpoonup \mathbb{N}$. Here $\mathbb{N} \Rightarrow \mathbb{N}$ can only be $\mathbb{N}$, so we must provide a suitable operation $\cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. This is done using the concept of a **universal Turing machine**. Let $T_0, T_1, ...$ be some sensibly chosen enumeration of all Turing machines for computing partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$. Then there is a Turing machine that accepts two inputs e, a and returns the result of applying the machine T_e to the single input a. We may therefore take $\cdot$ to be the partial function computed by U

Clearly the partial functions $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ representable within the model via the pairing operation from Example 1.1 are just the partial computable ones. We may also see that these coincide exactly with those represented by some total computable $\bar{f} : \mathbb{N} \to \mathbb{N}$, in the sense that $f(c, a) \simeq \tilde{f}(c) \cdot a$.

$\Leftarrow$: Given a computable $\tilde{f}$ the operation $\Lambda(c, a).\tilde{f}(c) \cdot a$ is clearly computable

$\Rightarrow$: s-m-n theorem

When endowed with this weakly cartisian closed structure, this computability model is known as **Kleene's first model** of K_1

Example 1.5. Now consider the model $\Lambda/\sim$; we shall write L for the set $\Lambda/\sim$ considered as the sole datatype in this model. Set $\mathsf{L} \Rightarrow \mathsf{L} = \mathsf{L} \bowtie \mathsf{L} = \mathsf{L}$. We may obtain a weakly cartesian closed structure by letting $\cdot$ be given by application. If $M \in \Lambda$ induces an operation in $[\mathsf{L} \bowtie \mathsf{L}, \mathsf{L}]$ representing some $f : \mathsf{L} \times \mathsf{L} \to \mathsf{L}$, then $\lambda x.\lambda y.M(pair\ x\ y)$ induces the corresponding operation in $[\mathsf{L}, \mathsf{L} \Rightarrow \mathsf{L}]$; conversely if N induces an operation in $[\mathsf{L}, \mathsf{L} \Rightarrow \mathsf{L}]$ then $\lambda z.N(fst\ z)(snd\ z)$ induces the corresponding one in $[\mathsf{L} \bowtie \mathsf{L}, \mathsf{L}]$

TODO

Example 1.6.

1.1.4 Higher-Order Models

Definition 1.6. A **higher-order structure** is a computability model $\mathbf{C}$ possessing a weak terminal (I, i) and endowed with the following for each $A, B \in |\mathbf{C}|$

- a choice of datatype $A \Rightarrow B \in |\mathbf{C}|$
- a partial function $\cdot_{AB} : (A \Rightarrow B) \times A \rightharpoonup B$

We treat $\Rightarrow$ as right-associative and $\cdot$ as left-associative

The significance of the weak terminal (I, i) here is that it allows us to pick out a subset $A^\sharp$ of each $A \in |\mathbf{C}|$, namely the set of elements of the form $f(i)$ where $f \in \mathbf{C}[I, A]$ and $f(i)\downarrow$.

This is independent of the choice of (I, i): if $a = f(i)$ and (J, j) is another weak terminal, then composing f with $\Lambda x.i \in \mathbf{C}[J, I]$ gives $f' \in \mathbf{C}[J, A]$ with $f'(j) = a$.

Intuitively, we think of $A^\sharp$ as playing the role of the 'computable' elements of A, and i as some generic computable element. On the one hand, if $a \in A$ were computable, we would expect each $\Lambda x.a$ to be computable so that $a \in A^\sharp$; on the other hand, the image of a computable element under a computable operation should be computable, so that every element of $A^\sharp$ is computable.

Any weakly cartesian closed model $\mathbf{C}$ is a higher-structure.

Definition 1.7. A **higher-order (computability) model** is a higher-order structure $\mathbf{C}$ satisfying the following conditions for some (or equivalently any) weak terminal (I, i)

1. A partial function $f : A \rightharpoonup B$ is present in $\mathbf{C}[A, B]$ iff there exists $\hat{f} \in \mathbf{C}[I, A \Rightarrow B]$ s.t.

$$\hat{f}(i)\downarrow, \quad \forall a \in A. \hat{f}(i) \cdot a \simeq f(a)$$

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

$$\forall a. k_{AB} \cdot a \downarrow, \quad \forall a, b. k_{AB} \cdot a \cdot b = a$$

3. For any $A, B, C \in |\mathbf{C}|$ there exits

$$s_{ABC} \in ((A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C))^\sharp$$

s.t.

$$\forall f, g. s_{ABC} \cdot f \cdot g \downarrow, \quad \forall f, g, a. s_{ABC} \cdot f \cdot g \cdot a \simeq (f \cdot a) \cdot (g \cdot a)$$

The elements k and s correspond to combinators from combinatory logic.

k allows us to construct **constant** maps in a computable way

A possible intuition for s is that it somehow does duty for an application operation $(B \Rightarrow C) \times B \rightharpoonup C$ within $\mathbf{C}$ itself, where the application may be performed uniformly in a parameter of type A.p

Proposition 1.8. *Suppose* **C** *is a higher-order model*

1. *for any* $j < m$*, there exists* $\pi_j^m \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow A_j)^\sharp$ *s.t.*

$$\forall a_0, \ldots, a_{m-1}.\pi_j^m \cdot a_0 \cdot \cdots \cdot a_{m-1} = a_j$$

2. *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.*

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

3. *Suppose* $m > 0$*. For any element* $f \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow B)^\sharp$*, there exists* $f^\dagger \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow B)^\sharp$ *s.t.*

$$\forall a_0, \ldots, a_{m-1}.f^\dagger \cdot a_0 \cdot \cdots \cdot a_{m-1} \simeq f \cdot a_0 \cdot \cdots \cdot a_{m-1}$$
$$\forall k < m.\forall a_0, \ldots, a_{k-1}.f^\dagger \cdot a_0 \cdot \cdots \cdot a_{k-1}$$

If **C**, **D** are higher-order structures, we say **C** is a **full substructure** of **D** if

- $|\mathbf{C}| \subseteq |\mathbf{D}|$
- $\mathbf{C}[A, B] = \mathbf{D}[A, B]$ for all $A, B \in |\mathbf{C}|$
- some (or equivalently any) weak terminal in **C** is also a weak terminal in **D**
- the meaning of $A \Rightarrow B$ and $\cdot_{AB}$ in **C** and **D** coincide

Note that if (I, i) and (J, j) are weak terminals in **C** then $\Lambda x.j \in \mathbf{C}[I, J]$, so if (I, i) is a weak terminal in **D** then so is (J, j)

Theorem 1.9. *A higher-order structure is a higher-order model iff it is a full substructure of a weakly cartesian closed model*

Proof. Let **C** be a higher-order structure.

$\Leftarrow$: suppose **D** is weakly cartesian closed and **C** is a full substructure of **D** with a weak terminal (I, i)

1. For any $f \in \mathbf{C}[A, B] = \mathbf{D}[A, B]$ we have that $f \circ \pi_A \in \mathbf{D}[I \bowtie A, B]$ represents $\Lambda(x, a).f(a)$, which by definition 1.5 is in turn represented by some total $\hat{f} \in \mathbf{D}[I, A \Rightarrow B]$.

   Conversely, given $f : A \rightharpoonup B$ and $\hat{f} \in \mathbf{C}[I, A \Rightarrow B]$ with $\hat{f}(i) \downarrow$ and $\hat{f}(i) \cdot a \simeq f(a)$ for all a, take $\hat{g} = \hat{f} \circ (\Lambda x.i) \in \mathbf{C}[I, A \Rightarrow B] = \mathbf{D}[I, A \Rightarrow B]$ so that $\hat{g}$ is total and represents $g = \Lambda(x, a).f(a) : I \times A \rightharpoonup B$. Now let $\bar{g} \in \mathbf{D}[I \bowtie A, B]$ also represents g. Then $\bar{g} \circ \langle \Lambda a.i, \mathrm{id}_A \rangle \in \mathbf{D}[A, B] = \mathbf{C}[A, B]$ and it is routine to check that $\bar{g} \circ \langle \Lambda a.i, \mathrm{id}_A \rangle = f$

2. Suppose $A, B \in |\mathbf{C}|$. Let $k' \in \mathbf{D}[A, B \Rightarrow A]$ correspond to $\pi_A \in \mathbf{D}[A \bowtie B, A]$ as in definition 1.5, then $k'(a) \cdot b \simeq \pi_A(d)$. Let $\hat{k}' \in \mathbf{D}[I, A \Rightarrow (B \Rightarrow A)]$ correspond to $k' \circ \pi'_A \in \mathbf{D}[I \bowtie A, B \Rightarrow A]$ where $\pi'_A \in \mathbf{D}[I \bowtie A, A]$ and take $k = \hat{k}'(i)$ $k \cdot a \cdot b = \hat{k}'(i) \cdot a \cdot b = (k' \circ \pi'_A(i, a)) \cdot b = k'(a) \cdot b = a$

3.

$\Rightarrow$: Suppose $\mathbf{C}$ is a higher-order model, with (I, i) a weak terminal. We build a weakly cartesian closed model $\mathbf{C}^\times$ into which $\mathbf{C}$ embeds fully as follows:

- Datatypes of $\mathbf{C}^\times$ are sets $A_0 \times \cdots \times A_{m-1}$, where $m > 0$ and $A_0, \ldots, A_{m-1} \in |\mathbf{C}|$
- If $D = A_0 \times \cdots \times A_{m-1}$ and $E = B_0 \times \cdots \times B_{n-1}$ where $m, n > 0$ the operations in $\mathbf{C}^\times[D, E]$ are those partial functions $f : D \rightharpoonup E$ of the form

  $$f = \Lambda(a_0, \ldots, a_{m-1}).(f_0 \cdot a_0 \cdot \cdots \cdot a_{m-1}, \ldots, f_{n-1} \cdot a_0 \cdot \cdots \cdot a_{m-1})$$

  where $f_j \in (A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow B_j)^\sharp$ for each j; we say that $f_0, \ldots, f_{n-1}$ **witness** the operation f. Note that for $(f_0 \cdot a_0 \cdot \cdots \cdot a_{m-1}, \ldots, f_{n-1} \cdot a_0 \cdot \cdots \cdot a_{m-1})$ to be defined, it is necessary that all its components be defined

It remains to check the relevant properties of $\mathbf{C}^\times$. That $\mathbf{C}^\times$ is a computability model is straightforward: the existence of identities follows from part 1 of Proposition 1.8 and composition from part 2. $\mathbf{C}^\times$ has standard products and that (I, i) is a weak terminal in $\mathbf{C}^\times$.

Now let's show that $\mathbf{C}^\times$ is weakly cartesian closed. Given $D = A_0 \times \cdots \times A_{m-1}$ and $E = B_0 \times \cdots \times B_{n-1}$ with $m, n > 0$, take $C_j = A_0 \Rightarrow \cdots \Rightarrow A_{m-1} \Rightarrow$

B_j for each j, and let $D \Rightarrow E$ be the set of tuples $(f_0, \dots, f_{n-1}) \in C_0 \times \dots \times C_{n-1}$ witnessing operations in $\mathbf{C}^\times[D, E]$. The application $\cdot_{DE}$ is then given by

$$(f_0, \dots, f_{n-1}) \cdot_{DE} (a_0, \dots, a_{m-1}) \simeq (f_0 \cdot a_0 \cdot \dots \cdot a_{m-1}, \dots, f_{n-1} \cdot a_0 \cdot \dots \cdot a_{m-1})$$

Next, given an operation $g \in \mathbf{C}^\times[G \times D, E]$ witnessed by operations $g_0, \dots, g_{n-1}$ in $\mathbf{C}$, take $g_0^\dagger, \dots, g_{n-1}^\dagger$ as in Proposition 1.8 (3); then $g_0^\dagger, \dots, g_{n-1}^\dagger$ witness the corresponding total operation $\hat{g} \in \mathbf{C}^\times[G, D \Rightarrow E]$. Conversely, the witnesses for any such total $\hat{g}$ also witness the corresponding g $\square$

1.1.5 Typed Partial Combinatory Algebras

The following definition captures roughly what is left of a higher-order model once the operations are discarded

Definition 1.10. 1. A **partial applicative structure A** consists of

- an inhabited family $|\mathbf{A}|$ of datatypes $A, B, \dots$ (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 $\mathbf{A}$ 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 TPCA is **total** if all the application operations $\cdot_{AB}$ are total