

Appendix A: W-continuity

Recall our definition (2.5.1) of continuity of *partial functions*: $f : X \dashrightarrow Y$ is continuous iff for every open $V \subseteq Y$, $f^{-1}[V]$ is open in X .

This is not the only reasonable definition. Another definition, used in [Wei00, Bra96, Bra99] (henceforth “W-continuity”), amounts to saying that f is continuous iff its restriction to its domain

$$f \upharpoonright \mathbf{dom}(f) : \mathbf{dom}(f) \rightarrow Y$$

is continuous (as a total function), where $\mathbf{dom}(f)$ has the topology as a subspace of A ; or, equivalently, iff for every open $V \subseteq Y$, $f^{-1}[V]$ is open in $\mathbf{dom}(f)$.

The following is easily checked:

Proposition A.1. f is continuous $\iff f$ is W-continuous and $\mathbf{dom}(f)$ is open.

It is instructive to express these two notions of continuity in terms of metric spaces (cf. Remark 2.5.2). Suppose $f : X \dashrightarrow Y$ where X and Y are metric spaces. Then

(a) f is continuous iff

$$\forall a \in \mathbf{dom}(f) \forall \epsilon > 0 \exists \delta > 0 \forall x \in \mathbf{B}(a, \delta) (x \in \mathbf{dom}(f) \wedge f(x) \in \mathbf{B}(f(a), \epsilon)).$$

(b) f is W-continuous iff

$$\forall a \in \mathbf{dom}(f) \forall \epsilon > 0 \exists \delta > 0 \forall x \in \mathbf{B}(a, \delta) (x \in \mathbf{dom}(f) \rightarrow f(x) \in \mathbf{B}(f(a), \epsilon)).$$

Here $\mathbf{B}(a, \delta)$ is the open ball with centre a and radius δ .

Example A.2. Consider the partial function $f : \mathbb{R} \dashrightarrow \mathbb{N}$ defined by

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is an integer} \\ \uparrow & \text{otherwise.} \end{cases}$$

Then f is W-continuous, but not continuous. The intuition here is that for continuity (and certainly for computability), we would want a finite approximation to the input (however defined exactly) to produce a finite approximation to the output (in this case, the output itself). However that would not be the case here, since no finite approximation to the input will tell us whether the input is in $\mathbf{dom}(f)$ (i.e., an exact integer) or not.

Analogously, we can consider another notion of continuity for *many-valued* functions $f : X \rightrightarrows Y$ by modifying Definition 5.1.3(b); namely, f is W-continuous iff for all open $V \subseteq Y$, $f^{-1}[V]$ is open in $\mathbf{dom}(f)$. Note that Lemma 5.1.7, and the equivalences given in Remark 5.1.9, also hold for W-continuity.

Finally, Theorem 5.3.1 holds for W-continuity, without the assumption that $\mathbf{dom}(f)$ is open:

Theorem A.3. *Let A be a metric Σ -algebra, and $f: A^u \rightarrow A^v$. If f is **WhileCC*** approximable on A then f is continuous.*

The proof is similar to that for Theorem 5.3.1. In fact, Theorem 5.3.1 follows immediately from Theorem A.3 and Proposition A.1.

Appendix B: Computation tree with infinite paths, but no recursive infinite paths

Here we prove:

Proposition 3.4.3. *There is a **WhileCC***(\mathcal{N}) procedure P such that its computation tree has infinite paths, but no recursive infinite paths.*

Proof: Our construction of P is based on the construction of a recursive tree with infinite paths, but no recursive infinite paths [Odi99, V.5.25].

Let A and B be two disjoint r.e., recursively inseparable sets, and suppose $A = \mathbf{ran}(f)$ and $B = \mathbf{ran}(g)$ where f and g are total recursive functions. The procedure P can be written in pseudo-code as:

```

func aux n, k : nat,
    choices* : nat*, { array recording all choices up to present stage n }
    halt : bool
begin
    n := 0;
    choices* := Null;
    halt := false;
    while not halt do
        n := n + 1;
        choices* := Newlength(choices*, n + 1);
        choices*[n] := choose z : (z = 0 or z = 1);
        for k := 0 to n - 1 do
            if (choices*[k] = 0 and k ∈ { f(0), ..., f(n - 1) }) or
                (choices*[k] = 1 and k ∈ { g(0), ..., g(n - 1) })
            then halt := true
        od
    od
end.

```

Let $\alpha_0, \alpha_1, \alpha_2, \dots$ be the successive values (0 or 1) given by the ‘choose’ operator in some given implementation of P . Note that at stage n ,

$$\text{choices}^*[k] = \alpha_k \quad \text{for } k = 0, \dots, n-1.$$

Further, the execution diverges if, and only if, the set $C =_{df} \{k \mid \alpha_k = 1\}$ separates A and B (i.e., $A \subseteq C$ and $C \cap B = \emptyset$), in which case C , and hence its characteristic function $\alpha = (\alpha_0, \alpha_1, \alpha_2, \dots)$, are *non-recursive*.

Note finally that for any given sequence α of choices, α is effectively obtainable from the corresponding computation sequence or path, i.e., α is recursive in that path (with a standard coding of the computation tree). Hence, since any infinite sequence α is non-recursive, so is the corresponding infinite path. \square

Appendix C: Proofs for Section 5.1

This section contains mainly technical results relating to the continuity of countably-many-valued functions. The proofs are collected here.

Lemma 5.1.7. *Let $f: X \rightrightarrows Y$ and $g: X \rightrightarrows^+ Y^\uparrow$ be any two functions such that*

$$f \sqsubseteq g \sqsubseteq f^\uparrow,$$

i.e., for all $x \in X$, $g(x) \neq \emptyset$, and either $g(x) = f(x)$ or $g(x) = f(x) \cup \{\uparrow\}$. Then

$$f \text{ is continuous} \iff g \text{ is continuous.}$$

Proof: (\Rightarrow) Suppose f is continuous. We must show g is continuous. Let V be an open subset of Y^\uparrow . We must show $g^{-1}[V]$ is open in X . There are two cases, according as \uparrow is in V or not.

Case 1: $\uparrow \notin V$, i.e., $V \subseteq Y$. Then V is also open in Y (by definition of the topology on Y^\uparrow). Hence $f^{-1}[V]$ is open in X , and hence

$$\begin{aligned} g^{-1}[V] &= \{x \in X \mid g(x) \cap V \neq \emptyset\} \\ &= \{x \in X \mid f(x) \cap V \neq \emptyset\} \quad \text{since } \uparrow \notin V \\ &= f^{-1}[V] \end{aligned}$$

is open in X .

Case 2: $\uparrow \in V$. Then $V = Y^\uparrow$ (by definition of the topology on Y^\uparrow). Hence

$$g^{-1}[V] = g^{-1}[Y^\uparrow] = X \quad (\text{since } g \text{ is total}),$$

which is open in X .

(\Leftarrow) Suppose g is continuous. We must show f is continuous. Let V be an open subset of Y . We must show $f^{-1}[V]$ is open in X . Since V is also open in Y^\uparrow (by definition of the topology on Y^\uparrow), $g^{-1}[V]$ is open in X . Hence

$$\begin{aligned} f^{-1}[V] &= \{x \in X \mid f(x) \cap V \neq \emptyset\} \\ &= \{x \in X \mid g(x) \cap V \neq \emptyset\} \quad \text{since } \uparrow \notin V \\ &= g^{-1}[V] \end{aligned}$$

is open in X . \square

Corollary 5.1.8. *Suppose $f : X \rightrightarrows^+ Y^\uparrow$ (i.e., f is total). Then*

$$f \text{ is continuous} \iff f^- \text{ is continuous} \iff f^\uparrow \text{ is continuous.}$$

Proof: Apply Lemma 5.1.7 twice: once with f^- and f , and once with f^- and f^\uparrow . \square

Lemma 5.1.11. *Given $f : X \rightrightarrows Y^\uparrow$, extend it to $\tilde{f} : X^\uparrow \rightrightarrows Y^\uparrow$ by stipulating that $\tilde{f}(\uparrow) = \uparrow$. If f is continuous and total, then \tilde{f} is continuous.*

Proof: Let V be an open subset of Y^\uparrow . We must show $\tilde{f}^{-1}[V]$ is open in X^\uparrow . There are two cases:

Case 1: $\uparrow \notin V$, i.e., $V \subseteq Y$. Then $\tilde{f}^{-1}[V] = f^{-1}[V]$, which is open in X , and hence in X^\uparrow .

Case 2: $\uparrow \in V$. Then $V = Y^\uparrow$ (by definition of the topology on Y^\uparrow). Hence

$$\begin{aligned} \tilde{f}^{-1}[V] &= \tilde{f}^{-1}[Y^\uparrow] \\ &= \mathbf{dom}(f) \cup \{\uparrow\} \\ &= X \cup \{\uparrow\} \quad (\text{since } f \text{ is total}) \end{aligned}$$

which is open in X^\uparrow . \square

Proposition 5.1.13 (Continuity of composition).

(a) *If $f : X \rightrightarrows Y$ and $g : Y \rightrightarrows Z$ are continuous, then so is $g \circ f : X \rightrightarrows Z$.*

(b) *If $f : X \rightrightarrows^+ Y^\uparrow$ and $g : Y \rightrightarrows^+ Z^\uparrow$ are continuous, then so is $g \circ f : X \rightrightarrows^+ Z^\uparrow$.*

Proof: (a) Just note that for $W \subseteq Z$,

$$(g \circ f)^{-1}[W] = f^{-1}[g^{-1}[W]].$$

(b) We give two proofs: (i) Note that

$$(g \circ f)^- = g^- \circ f^- : X \rightrightarrows Z$$

and use part (a) and Corollary 5.1.8.

(ii) Note that for $W \subseteq Z^\uparrow$,

$$(g \circ f)^{-1}[W] = f^{-1}[\tilde{g}^{-1}[W]]$$

(in the notation of Lemma 5.1.11), and apply Lemma 5.1.11. \square

Lemma 5.1.15. *If $f_i : X \rightrightarrows Y^\uparrow$ is continuous for all $i \in I$, then so is $\bigsqcup_{i \in I} f_i$.*

Proof: This follows from the fact that for $V \subseteq Y^\uparrow$,

$$\left(\bigsqcup_{i \in I} f_i\right)^{-1}[V] = \bigcup_{i \in I} f_i^{-1}[V]. \quad \square$$

Appendix D: Proof of the Soundness Theorem \mathbf{A}_0

We re-state this theorem (*cf.* §6.2).

Theorem \mathbf{A}_0 (Soundness for countable algebras). *Let (A, β) be an enumerated N -standard Σ -algebra such that β is strictly Σ -effective. If $f : A^u \longrightarrow A_s$ is **WhileCC*** computable on A , then f is strictly β -computable on A .*

Assume that (A, β) is an enumerated N -standard Σ -algebra and β is strictly Σ -effective.

We will show that each of the semantic functions listed in §3.2(a)–(g) has a computable strict tracking function. More precisely, we will work, not with the semantic functions themselves, but “localised” functions representing them (*cf.* [TZ00, §4]).

First we will prove a series of results of the form:

Lemma Scheme 6.3.1. *For each semantic representing function*

$$\Phi : A^u \rightrightarrows^+ A^{v\uparrow}$$

*representing one of the **WhileCC** semantic functions listed in §3.2(a)–(g), there is a computable tracking function w.r.t. β , i.e., a function*

$$\varphi : \Omega_\beta^u \longrightarrow \Omega_\beta^v$$

which commutes the diagram

$$\begin{array}{ccc} A^u & \xrightleftharpoons{\Phi} & + A^{v\uparrow} \\ \beta^u \uparrow & & \uparrow \beta^v \\ \Omega_\beta^u & \xrightarrow{\varphi} & \Omega_\beta^v \end{array}$$

in the sense that for all $k, l \in \Omega_\beta^u$:

$$\begin{aligned}\varphi(k) \downarrow l &\implies \beta^v(l) \in \Phi(\beta^u(k)), \\ \varphi(k) \uparrow &\implies \uparrow \in \Phi(\beta^u(k)).\end{aligned}$$

Proof: We proceed to prove this lemma scheme by constructing *concrete strict tracking functions for the semantic functions* in §3.2.

Let \mathbf{x} be a u -tuple of variables, where $u = s_1 \times \cdots \times s_m$. Let $\mathbf{PTerm}_{\mathbf{x}} = \mathbf{PTerm}_{\mathbf{x}}(\Sigma)$ be the class of all Σ -terms with variables among \mathbf{x} only, and for all sorts s of Σ , let $\mathbf{PTerm}_{\mathbf{x},s} = \mathbf{PTerm}_{\mathbf{x},s}(\Sigma)$ be the class of such terms of sort s .

We consider in turn the semantic functions in §3.2, or rather versions of these *localised to \mathbf{x}* , *i.e.*, defined only in terms of the state values on \mathbf{x} (*cf.* [TZ00, §4]). For example, we localise the set $\mathbf{State}(A)$ of states on A to the set

$$\mathbf{State}_{\mathbf{x}}(A) =_{df} A^u$$

of u -tuples of elements of A , where a tuple $a \in A^u$ represents a state σ (relative to \mathbf{x}) if $\sigma[\mathbf{x}] = a$. The set A^u is, in turn, represented (relative to β) by the set Ω_β^u .

We assume an effective coding, or Gödel numbering, of the syntax of Σ . We use the notation

$$\ulcorner \mathbf{PTerm}_s \urcorner =_{df} \{\ulcorner t \urcorner \mid t \in \mathbf{PTerm}_s\},$$

etc., for sets of Gödel numbers of syntactic expressions.

(a) Tracking of term evaluation.

The function

$$\mathbf{PTE}_{\mathbf{x},s}^A : \mathbf{PTerm}_{\mathbf{x},s} \times \mathbf{State}_{\mathbf{x}}(A) \rightrightarrows^+ A_s^\uparrow$$

defined by

$$\mathbf{PTE}_{\mathbf{x},s}^A(t, a) = \llbracket t \rrbracket^A \sigma$$

for any state σ on A such that $\sigma[\mathbf{x}] = a$, is strictly tracked by a computable function

$$\mathbf{pte}_{\mathbf{x},s}^{A,\beta} : \ulcorner \mathbf{PTerm}_{\mathbf{x},s} \urcorner \times \Omega_\beta^u \longrightarrow \Omega_{\beta,s}$$

so that the following diagram commutes:

$$\begin{array}{ccc} \mathbf{PTerm}_{\mathbf{x},s} \times \mathbf{State}_{\mathbf{x}}(A) & \xrightarrow{\mathbf{PTE}_{\mathbf{x},s}^A} & A_s^\uparrow \\ \uparrow \langle \mathbf{enum}, \beta^u \rangle & & \uparrow \beta_s \\ \ulcorner \mathbf{PTerm}_{\mathbf{x},s} \urcorner \times \Omega_\beta^u & \xrightarrow{\mathbf{pte}_{\mathbf{x},s}^{A,\beta}} & \Omega_{\beta,s} \end{array}$$

(where *enum* is the inverse of the Gödel numbering function), in the sense that

$$\begin{aligned} \mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k) \downarrow l &\implies \beta_s(l) \in \mathbf{PTE}_{\mathbf{x},s}^A(t, \beta^u(k)), \\ \mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k) \uparrow &\implies \uparrow \in \mathbf{PTE}_{\mathbf{x},s}^A(t, \beta^u(k)). \end{aligned} \tag{1}$$

In order to construct such a representing function, we first define the *state variant representing function*, i.e., a (primitive) recursive function

$$\mathit{vart}_{\mathbf{x}}^\beta : \Omega_\beta^u \times \ulcorner \mathbf{Var}_s \urcorner \times \Omega_{\beta,s} \rightarrow \Omega_{\beta,s}$$

such that

$$\beta^u(\mathit{vart}_{\mathbf{x}}^\beta(k, \ulcorner \mathbf{y} \urcorner, k_0)) = \beta^u(e)\{\mathbf{y}/\beta_s(k_0)\}.$$

for $k \in \Omega_\beta^u$, $\mathbf{y} \in \mathbf{Var}_s$ and $k_0 \in \Omega_{\beta,s}$ (cf. Definition 3.2.3(b)).

We turn to the definition of $\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k)$. This is by induction on $\ulcorner t \urcorner$, or structural induction on $t \in \mathbf{PTerm}_{\mathbf{x}}$, over all Σ -sorts s . The cases are:

- $t \equiv c$, a primitive constant. Then define

$$\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k) = k_0 \quad \text{where} \quad \beta(k_0) = c^A.$$

(Such a k_0 exists by the strict Σ -effectivity of β).

- $t \equiv \mathbf{x}_i$ for some $i = 1, \dots, m$, where $\mathbf{x} \equiv \mathbf{x}_1, \dots, \mathbf{x}_m$. Note that $k = (k_1, \dots, k_m) \in \Omega_\beta^u$. So define

$$\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k) = k_i.$$

- $t \equiv F(t_1, \dots, t_m)$. Let φ be a computable strict tracking function for F , which exists by the strict Σ -effectivity of β . Then define

$$\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k) \simeq \varphi(\mathit{pte}_{\mathbf{x},s_1}^{A,\beta}(\ulcorner t_1 \urcorner, k), \dots, \mathit{pte}_{\mathbf{x},s_m}^{A,\beta}(\ulcorner t_m \urcorner, k)).$$

From the induction hypothesis applied to t_1, \dots, t_m , the definition of \mathbf{PTE} (§3.2(a)) and the fact that φ *strictly* tracks F , we can infer (1) for t .

- $t \equiv \text{if}(b, t_1, t_2)$. Define

$$\mathit{pte}_{\mathbf{x},s}^{A,\beta}(t, k) \simeq \begin{cases} \mathit{pte}_{\mathbf{x},s}^{A,\beta}(t_1, k) & \text{if } \mathit{pte}_{\mathbf{x},\text{bool}}^{A,\beta}(b, k) \downarrow 1 \\ \mathit{pte}_{\mathbf{x},s}^{A,\beta}(t_2, k) & \text{if } \mathit{pte}_{\mathbf{x},\text{bool}}^{A,\beta}(b, k) \downarrow 0 \\ \uparrow & \text{if } \mathit{pte}_{\mathbf{x},\text{bool}}^A(b, k) \uparrow. \end{cases}$$

From the induction hypothesis applied to b, t_0 and t_1 , and the definition of \mathbf{PTE} , we can infer (1) for t .

- $t \equiv (\text{choose } \mathbf{z} : t_0)$. We define $\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t \urcorner, k)$ by specifying its computation: find, by dovetailing (recall the discussion in §4.1!) some n such that

$$\mathit{pte}_{\mathbf{x},s}^{A,\beta}(\ulcorner t_0 \urcorner, \mathit{vart}_{\mathbf{x}}^\beta(k, \ulcorner \mathbf{z} \urcorner, n)) \downarrow 1$$

(remember, $\beta(1) = \mathbf{\#}$, by Remark 6.1.4(b)), so that $\mathbf{pte}_{x,s}^{A,\beta}(\ulcorner t^\neg, k) =$ some such n , if it exists, and \uparrow otherwise. From the induction hypothesis applied to t_0 , and the definition of \mathbf{PTE} , we can infer (1) for t .

(b) Tracking of atomic statement evaluation.

Let \mathbf{AtSt}_x be the class of atomic statements with variables among x only. The atomic statement evaluation function on A localised to x ,

$$\mathbf{AE}_x^A : \mathbf{AtSt}_x \times \mathbf{State}_x(A) \rightrightarrows^+ \mathbf{State}_x(A)^\uparrow,$$

defined by

$$\mathbf{AE}_x^A(S, a) = \langle S \rangle^A \sigma$$

for any state σ such that $\sigma[x] = a$, is strictly tracked by a computable function

$$\mathbf{ae}_x^{A,\beta} : \ulcorner \mathbf{AtSt}_x^\neg \times \Omega_\beta^u \longrightarrow \Omega_\beta^u$$

so that the following diagram commutes:

$$\begin{array}{ccc} \mathbf{AtSt}_x \times \mathbf{State}_x(A) & \xrightleftharpoons{\mathbf{AE}_x^A} & \mathbf{State}_x(A)^\uparrow \\ \langle \mathbf{enum}, \beta^u \rangle \uparrow & & \uparrow \beta^u \\ \ulcorner \mathbf{AtSt}_x^\neg \times \Omega_\beta^u & \xrightarrow{\mathbf{ae}_x^{A,\beta}} & \Omega_\beta^u \end{array}$$

in the sense that

$$\begin{aligned} \mathbf{ae}_x^{A,\beta}(\ulcorner S^\neg, k) \downarrow l &\implies \beta(l) \in \mathbf{AE}_x^A(S, \beta(k)), \\ \mathbf{ae}_x^{A,\beta}(\ulcorner S^\neg, k) \uparrow &\implies \uparrow \in \mathbf{AE}_x^A(S, \beta(k)). \end{aligned} \tag{2}$$

The definition of $\mathbf{ae}_x^{A,\beta}(\ulcorner S^\neg, k)$ is given by:

$$\begin{aligned} \mathbf{ae}_x^{A,\beta}(\ulcorner \mathbf{skip}^\neg, k) &\downarrow k \\ \mathbf{ae}_x^{A,\beta}(\ulcorner \mathbf{div}^\neg, k) &\uparrow \\ \mathbf{ae}_x^{A,\beta}(\ulcorner y := t^\neg, k) &\simeq \begin{cases} \mathbf{vart}_x^\beta(k, y, l) & \text{if } \mathbf{pte}_{x,s}^{A,\beta}(s^\ulcorner t^\neg, k) \downarrow l \\ \uparrow & \text{if } \mathbf{pte}_{x,s}^{A,\beta}(\ulcorner t^\neg, k) \uparrow. \end{cases} \end{aligned}$$

Using (1) and the definition of \mathbf{AE}_x^A (§3.2(b)), we can infer (2).

(c) Tracking of *First* and *Rest* operations.

Let \mathbf{Stmt}_x be the class of statements with variables among x only. Consider the functions \mathbf{First} and \mathbf{Rest}^A (§3.2(c)). Then \mathbf{First} is strictly tracked by a computable function

$$\mathbf{first} : \ulcorner \mathbf{Stmt}^\neg \rightarrow \ulcorner \mathbf{AtSt}^\neg$$

defined on Gödel numbers in the obvious way, so that the following diagram commutes:

$$\begin{array}{ccc}
 \mathit{Stmt} & \xrightarrow{\mathit{First}} & \mathit{AtSt} \\
 \mathit{enum} \uparrow & & \uparrow \mathit{enum} \\
 \ulcorner \mathit{Stmt} \urcorner & \xrightarrow{\mathit{first}} & \ulcorner \mathit{AtSt} \urcorner
 \end{array}$$

(Note that first , unlike most of the other representing functions here, does not depend on $\mathit{State}_x(A)$, or, indeed, on A or x .) Next, the localised version of Rest^A :

$$\mathit{Rest}_x^A : \mathit{Stmt}_x \times \mathit{State}_x(A) \rightrightarrows^+ \mathit{Stmt}_x$$

defined by

$$\mathit{Rest}_x^A(S, a) = \mathit{Rest}^A(S, \sigma)$$

for any state σ such that $\sigma[x] = a$, is strictly tracked by a computable function

$$\mathit{rest}_x^{A,\beta} : \ulcorner \mathit{Stmt}_x \urcorner \times \Omega_\beta^u \longrightarrow \ulcorner \mathit{Stmt}_x \urcorner$$

so that the following diagram commutes:

$$\begin{array}{ccc}
 \mathit{Stmt}_x \times \mathit{State}_x(A) & \xrightarrow{\mathit{Rest}_x^A} & \mathit{Stmt}_x \\
 \langle \mathit{enum}, \beta^u \rangle \uparrow & & \uparrow \mathit{enum} \\
 \ulcorner \mathit{Stmt}_x \urcorner \times \Omega_\beta^u & \xrightarrow{\mathit{rest}_x^{A,\beta}} & \ulcorner \mathit{Stmt}_x \urcorner
 \end{array}$$

in the sense that

$$\begin{aligned}
 \mathit{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k) \downarrow \ulcorner S' \urcorner &\implies S' \in \mathit{Rest}^A(S, \beta(k)), \\
 \mathit{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k) \uparrow &\implies \mathit{div} \in \mathit{Rest}^A(S, \beta(k))
 \end{aligned} \tag{3}$$

The definition of $\mathit{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k)$, as well as the proof of (3), are by induction on $\ulcorner S^\urcorner$, or structural induction on S .

- S is atomic. Then

$$\mathit{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k) = \ulcorner \mathit{skip} \urcorner.$$

- $S \equiv S_1; S_2$. Then

$$\mathit{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k) = \begin{cases} \ulcorner S_2^\urcorner & \text{if } S_1 \text{ is atomic} \\ \ulcorner \mathit{rest}_x^{A,\beta}(S_1, k); S_2^\urcorner & \text{otherwise} \end{cases}$$

- $S \equiv \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi.}$ Then

$$\text{rest}_x^{A,\beta}(\ulcorner S^\urcorner, k) \simeq \begin{cases} \ulcorner S_1^\urcorner & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \downarrow 1 \\ \ulcorner S_2^\urcorner & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \downarrow 0 \\ \uparrow & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \uparrow. \end{cases}$$

- $S \equiv \text{while } b \text{ do } S_0 \text{ od.}$ Then

$$\text{rest}_x^{A,\beta}(S, k) \simeq \begin{cases} \ulcorner S_0; S^\urcorner & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \downarrow 1, \\ \ulcorner \text{skip}^\urcorner & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \downarrow 0, \\ \uparrow & \text{if } \mathit{pte}_{\text{bool},s}^{A,\beta}(b, k) \uparrow. \end{cases}$$

(d) Tracking of a computation step.

The computation step function (§3.2(d)) localised to x :

$$\mathit{CompStep}_x^A : \mathit{Stmt}_x \times \mathit{State}_x(A) \rightrightarrows^+ \mathit{State}_x(A)^\uparrow$$

defined by

$$\mathit{CompStep}_x^A(S, a) = \mathit{CompStep}^A(S, \sigma)$$

for any state σ such that $\sigma[x] = a$, is represented by the computable function

$$\mathit{compstep}_x^{A,\beta} : \ulcorner \mathit{Stmt}_x^\urcorner \times \Omega_\beta^u \longrightarrow \Omega_\beta^u$$

defined by

$$\mathit{compstep}_x^{A,\beta}(\ulcorner S^\urcorner, k) \simeq \mathit{ae}_x^{A,\beta}(\mathit{first}(\ulcorner S^\urcorner), k).$$

This makes the following diagram commute:

$$\begin{array}{ccc} \mathit{Stmt}_x \times \mathit{State}_x(A) & \xrightarrow{\mathit{CompStep}_x^A} & \mathit{State}_x(A)^\uparrow \\ \uparrow \langle \mathit{enum}, \beta^u \rangle & & \uparrow \beta^u \\ \ulcorner \mathit{Stmt}_x^\urcorner \times \Omega_\beta^u & \xrightarrow{\mathit{compstep}_x^{A,\beta}} & \Omega_\beta^u \end{array}$$

in the sense that

$$\begin{aligned} \mathit{compstep}_x^{A,\beta}(\ulcorner S^\urcorner, k) \downarrow l &\implies \beta(l) \in \mathit{CompStep}_x^A(S, \beta(k)), \\ \mathit{compstep}_x^{A,\beta}(\ulcorner S^\urcorner, k) \uparrow &\implies \uparrow \in \mathit{CompStep}_x^A(S, \beta(k)). \end{aligned} \tag{4}$$

This is proved easily from the definitions and (2).

(e) **Tracking of a computation sequence.**

Now consider localised versions of the computation tree stage and computation tree of §3.2.(e):

$$\begin{aligned} \mathbf{CompTreeStage}_x^A &: \mathbf{Stmt}_x \times \mathbf{State}_x(A) \times \mathbb{N} \rightarrow \mathcal{P}((\mathbf{State}_x(A)^\uparrow)^{<\omega}) \\ \mathbf{CompTree}_x^A &: \mathbf{Stmt}_x \times \mathbf{State}_x(A) \rightarrow \mathcal{P}((\mathbf{State}_x(A)^\uparrow)^{\leq\omega}) \end{aligned}$$

We will define a function which *selects a path through the computation tree*:

$$\mathbf{compseq}_x^{A,\beta} : \ulcorner \mathbf{Stmt}_x \urcorner \times \Omega_\beta^u \times \mathbb{N} \longrightarrow \Omega_\beta^u \cup \{\ulcorner * \urcorner\}$$

(where ‘*’ is a symbol meaning “already terminated”) by recursion on n :

$$\begin{aligned} \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, 0) &= k \\ \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n+1) &\simeq \begin{cases} \ulcorner * \urcorner & \text{if } S \text{ is atomic and } n > 0 \text{ and } \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n) \downarrow \\ \uparrow & \text{if } S \text{ is atomic and } n > 0 \text{ and } \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n) \uparrow \\ \mathbf{compseq}_x^{A,\beta}(\mathbf{rest}_x^{A,\beta}(\ulcorner S \urcorner, k), \mathbf{compstep}_x^{A,\beta}(\ulcorner S \urcorner, k), n) & \\ \text{otherwise.} & \end{cases} \end{aligned}$$

(This is a “tail recursion”: compare definition of \mathbf{Comp}_1^A in [TZ00, §3.4].)

Writing $k_n = \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n)$, this defines a (concrete) *computation sequence*

$$\bar{k} = k_0, k_1, k_2, \dots$$

for S from the initial state $k = k_0$. (Our notation here includes the possibility that some of the k_i may be $\ulcorner * \urcorner$ or \uparrow .) As can easily be checked, there are three possibilities for \bar{k} (compare the discussion in §3.2.(e)):

- (i) For some n , $k_i \in \Omega_\beta^u$ for all $i \leq n$ and $k_i = *$ for all $i > n$. This represents a computation which *terminates* at stage n , with *final state* k_n .
- (ii) For some n , $k_i \in \Omega_\beta^u$ for all $i < n$ and $k_i = \uparrow$ for all $i \geq n$. This represents a *non-terminating* computation, with *local divergence* at stage n .
- (iii) For all i , $k_i \in \Omega_\beta^u$. This represents *non-terminating* computation, with *global divergence*.

We write $\bar{k}[n] =$ the initial segment k_0, k_1, \dots, k_n , with length $\mathbf{lgth}(\bar{k}[n]) = n + 1$. We put $\mathbf{lgth}(\bar{k}) = \infty$. The k_i are called *components* of \bar{k} , and of $\bar{k}[n]$, for all $i \leq n$.

The computation sequence \bar{k} then has the following connection with the computation tree $\mathbf{CompTree}_x^A$. Extend (for now) the definition of β by $\beta(\ulcorner * \urcorner) = *$, $\beta(\uparrow) = \uparrow$, and

$$\begin{aligned} \beta(\bar{k}) &=_{df} \beta(k_0), \beta(k_1), \beta(k_2), \dots \\ \beta(\bar{k}[n]) &=_{df} \beta(k_0), \beta(k_1), \beta(k_2), \dots, \beta(k_n). \end{aligned}$$

Lemma. Let $\tau = \mathbf{CompTree}_x^A(S, \beta(k))$. Then

- (i) If the computation sequence \bar{k} terminates at stage n , then $\beta(\bar{k}[n])$ is a path through τ from the root to a leaf ($= \beta(k_0)$, the final state).
- (ii) If for some (smallest) n , $k_n = \uparrow$, then $\beta(\bar{k}[n])$ is a path through τ from the root to a leaf ($= \uparrow$, local divergence).
- (iii) If for all n , $k_n \in \Omega_\beta^u$, then $\beta(\bar{k})$ is an infinite path through τ (global divergence).

To prove this, we first define an initial segment of \bar{k} (including \bar{k} itself) to be *acceptable* if (i) no component is equal to $*$, and (ii) no component, except possibly the last, is equal to \uparrow . Further, an acceptable initial segment of \bar{k} is *maximal (acceptable)* if it has no acceptable extension. Thus if \bar{k} is acceptable, it is automatically maximal. If $\bar{k}[n]$ is acceptable, it is maximal acceptable provided either $k_{n+1} = *$ or $k_n = \uparrow$. We then show:

Sublemma. Given a computation sequence $\bar{k} = k_0, k_1, \dots$ for $\ulcorner S \urcorner$ from k , where $k_n = \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n)$, let $\tau = \mathbf{CompTree}_x^A(S, \beta(k))$. Then with every acceptable initial segment $\bar{k}[n]$ of \bar{k} , $\beta(\bar{k}[n])$ is a path through τ from the root. If $\bar{k}[n]$ is maximal, then $\beta(k_n)$ is a leaf.

Proof of sublemma: Put $\tau[n] = \mathbf{CompTreeStage}_x^A(S, \beta(k_0), n)$. The proof is by induction on n , comparing the inductive definitions of k_n and $\tau[n]$.

Basis: $n = 0$. This is immediate from the definitions: $k_0 = k$, and $\tau[0] = \{\beta(k_0)\}$.

Induction step: Assume the induction hypothesis holds for the initial segment of length n of the computation sequence for $\ulcorner S' \urcorner$ from k_1 , where

$$\begin{aligned} S' &= \mathbf{rest}_x^{A,\beta}(\ulcorner S \urcorner, \beta(k)), \\ e_1 &= \mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, 1) \\ &\simeq \mathbf{compseq}_x^{A,\beta}(\mathbf{rest}_x^{A,\beta}(\ulcorner S \urcorner, k), \mathbf{compstep}_x^{A,\beta}(\ulcorner S \urcorner, k), 0) \\ &\simeq \mathbf{compstep}_x^{A,\beta}(\ulcorner S \urcorner, e) \end{aligned}$$

i.e., assume the induction hypothesis for the segment \bar{l} of length n :

$$l_0, l_1, l_2, \dots, l_n$$

where $l_i = e_{i+1}$ ($i = 1, \dots, n$). Now apply the inductive definitions for $\mathbf{compseq}_x^{A,\beta}(\ulcorner S \urcorner, k, n+1)$ (above) and $\mathbf{CompTreeStage}_x^A(S, \beta(k), n+1)$ (§3.2(e)), and use (3) and (4). This proves the sublemma, and hence the lemma.

(f) Tracking of statement evaluation.

First we need a constructive computation length function

$$\mathbf{complength}_x^{A,\beta} : \ulcorner \mathbf{Stmt}_x \urcorner \times \Omega_\beta^u \longrightarrow \mathbb{N}$$

by (cf. [TZ00, §3.4])

$$\mathbf{complength}_x^{A,\beta}(\ulcorner S^\top, k) \simeq \mu n[\mathbf{compseq}_x^{A,\beta}(\ulcorner S^\top, k, n+1) \downarrow *]$$

i.e., the least n (if it exists) such that for all $i \leq n$, $\mathbf{compseq}_x^{A,\beta}(\ulcorner S^\top, k, i) \downarrow \neq *$ and $\mathbf{compseq}_x^{A,\beta}(\ulcorner S^\top, k, n+1) \downarrow *$.

Thus $\mathbf{complength}_x^{A,\beta}(\ulcorner S^\top, k)$ is undefined (\uparrow) in the case of local or global divergence of the computation sequence for $\ulcorner S^\top$ from k .

Now the statement evaluation function (§3.2(f)) localised to x :

$$\mathbf{SE}_x^A : \mathbf{Stmt}_x \times \mathbf{State}_x(A) \rightrightarrows^+ \mathbf{State}_x(A)^\uparrow$$

defined by

$$\mathbf{SE}_x^A(S, a) = \llbracket S \rrbracket^A(\sigma)$$

for any state σ such that $\sigma[x] = a$, is strictly tracked by the computable function

$$\mathbf{se}_x^{A,\beta} : \ulcorner \mathbf{Stmt}_x^\top \times \Omega_\beta^u \longrightarrow \Omega_\beta^u$$

defined by

$$\mathbf{se}_x^{A,\beta}(\ulcorner S^\top, k) \simeq \mathbf{compseq}_x^{A,\beta}(\ulcorner S^\top, k, \mathbf{complength}_x^{A,\beta}(\ulcorner S^\top, k)).$$

This makes the following diagram commute:

$$\begin{array}{ccc} \mathbf{Stmt}_x \times \mathbf{State}_x(A) & \xrightarrow{\mathbf{SE}_x^A} & \mathbf{State}_x(A)^\uparrow \\ \langle \mathbf{enum}, \beta^u \rangle \uparrow & & \uparrow \beta^u \\ \ulcorner \mathbf{Stmt}_x^\top \times \Omega_\beta^u & \xrightarrow{\mathbf{se}_x^{A,\beta}} & \Omega_\beta^u \end{array}$$

in the sense that

$$\begin{aligned} \mathbf{se}_x^{A,\beta}(\ulcorner S^\top, k) \downarrow l &\implies \beta(l) \in \mathbf{SE}_x^A(S, \beta(k)), \\ \mathbf{se}_x^{A,\beta}(\ulcorner S^\top, k) \uparrow &\implies \uparrow \in \mathbf{SE}_x^A(S, \beta(k)). \end{aligned} \tag{5}$$

This result is clear from the definition of $\mathbf{complength}$ and the lemma in (e).

(g) Tracking of procedure evaluation.

For a specific triple of lists of variables $\mathbf{a} : u, \mathbf{b} : v, \mathbf{c} : w$, let $\mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}}$ be the class of all $\mathbf{WhileCC}^*$ procedures of type $u \rightarrow v$, with declaration ‘in \mathbf{a} out \mathbf{b} aux \mathbf{c} ’. The procedure evaluation function (§3.2(g)) localised to this declaration:

$$\mathbf{PE}_{\mathbf{a},\mathbf{b},\mathbf{c}}^A : \mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}} \times A^u \rightrightarrows^+ A^v$$

defined by

$$\mathbf{PE}_{\mathbf{a},\mathbf{b},\mathbf{c}}^A(P, a) = P^A(a),$$

is strictly tracked by the computable function

$$\mathbf{pe}_{\mathbf{a},\mathbf{b},\mathbf{c}}^{A,\beta} : \lceil \mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}} \rceil \times \Omega_\beta^u \longrightarrow \Omega_\beta^v$$

defined by the following algorithm. Let $P \in \mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}}$; say

$$P \equiv \text{proc in } \mathbf{a} \text{ out } \mathbf{b} \text{ aux } \mathbf{c} \text{ begin } S \text{ end}$$

and let $k_0 \in \Omega_\beta^u$. Take any $k_1 \in \Omega_\beta^v$ and $k_2 \in \Omega_\beta^w$. (The choice of k_1 and k_2 is irrelevant, by Remark 3.2.4.) Put $k \equiv k_0, k_1, k_2$ and put $\mathbf{x} \equiv \mathbf{a}, \mathbf{b}, \mathbf{c}$. Compute $\mathbf{se}_{\mathbf{x}}^{A,\beta}(\lceil S \rceil, k)$. Suppose this converges to $l \equiv l_0, l_1, l_2$, where $l_0 \in \Omega_\beta^u$, $l_1 \in \Omega_\beta^v$ and $l_2 \in \Omega_\beta^w$. Then we define $\mathbf{pe}_{\mathbf{a},\mathbf{b},\mathbf{c}}^{A,\beta}(\lceil P \rceil, k_0) \downarrow l_1$. The following diagram then commutes:

$$\begin{array}{ccc} \mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}} \times A^u & \xrightarrow{\mathbf{PE}_{\mathbf{a},\mathbf{b},\mathbf{c}}^A} & A^{v\uparrow} \\ \langle \mathit{enum}, \beta^u \rangle \uparrow & & \uparrow \beta^v \\ \lceil \mathbf{Proc}_{\mathbf{a},\mathbf{b},\mathbf{c}} \rceil \times \Omega_\beta^u & \xrightarrow{\mathbf{pe}_{\mathbf{a},\mathbf{b},\mathbf{c}}^{A,\beta}} & \Omega_\beta^v \end{array}$$

in the sense that

$$\begin{aligned} \mathbf{pe}_{\mathbf{a},\mathbf{b},\mathbf{c}}^{A,\beta}(\lceil P \rceil, k) \downarrow l &\implies \beta(l) \in \mathbf{PE}_{\mathbf{a},\mathbf{b},\mathbf{c}}^A(P, \beta(k)), \\ \mathbf{pe}_{\mathbf{a},\mathbf{b},\mathbf{c}}^{A,\beta}(\lceil P \rceil, k) \uparrow &\implies \uparrow \in \mathbf{PE}_{\mathbf{a},\mathbf{b},\mathbf{c}}^A(P, \beta(k)). \end{aligned} \tag{6}$$

This is proved from (5) and the definitions of \mathbf{PE} and \mathbf{pe} .

This concludes the proof of Lemma Scheme 6.3.1. \square

Proof of Theorem A₀ (conclusion): Suppose $f : A^u \longrightarrow A_s$ is **WhileCC*** computable on A . Then there is a deterministic **WhileCC*** procedure (Definitions 3.2.5/6)

$$P: u \rightarrow s$$

such that for all $a \in A^u$,

$$\begin{aligned} f(x) \downarrow y &\implies P^A(x) = \{y\}, \\ f(x) \uparrow &\implies P^A(x) = \{\uparrow\}. \end{aligned}$$

Hence by (g) (above) there is a computable (partial) function

$$\varphi: \Omega_\beta^u \longrightarrow \Omega_{\beta,s}$$

which strictly tracks f , as required. \square

Note that this last step implicitly uses the following (the proof of which is omitted):

Lemma (Canonical extensions of numberings). *A numbering β of A can be canonically extended to a numbering β^* of A^* , such that if β is strictly Σ -effective, then β^* is strictly Σ^* -effective.*

References

- [Bra96] V. Brattka. Recursive characterisation of computable real-valued functions and relations. *Theoretical Computer Science*, 162:45–77, 1996.
- [Bra99] V. Brattka. *Recursive and computable operations over topological structures*. Ph.d. thesis, FernUniversität Hagen, Fachbereich Informatik, Hagen, Germany, 1999. Informatik Berichte 255, FernUniversität Hagen, July 1999.
- [Odi99] P. Odifreddi. *Classical Recursion Theory* (21nd ed.). North Holland, 1999.
- [TZ00] J.V. Tucker and J.I. Zucker. Computable functions and semicomputable sets on many-sorted algebras. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 5, pages 317–523. Oxford University Press, 2000.
- [Wei00] K. Weihrauch. *Computable Analysis: An Introduction*. Springer-Verlag, 2000.