Characterizing Programming Systems Allowing Program Self-Reference*

John Case and Samuel E. Moelius III

Department of Computer & Information Sciences
University of Delaware
103 Smith Hall
Newark, DE 19716
{case,moelius}@cis.udel.edu

March 6, 2007

Abstract. The interest is in characterizing insightfully the power of program self-reference in effective programming systems (epses), the computability-theoretic analogs of programming languages. In an eps in which the *constructive* form of Kleene's Recursion Theorem (**KRT**) holds, it is possible to construct, algorithmically, from an arbitrary algorithmic task, a self-referential program that, in a sense, creates a self-copy and then performs that task on the self-copy. In an eps in which the *not-necessarily*-constructive form of Kleene's Recursion Theorem (**krt**) holds, such self-referential programs exist, but cannot, in general, be found algorithmically.

In an earlier effort, Royer proved that there is *no* collection of recursive denotational control structures whose implementability *characterizes* the epses in which **KRT** holds. One main result herein, proven by a finite injury priority argument, is that the epses in which **krt** holds are, similarly, *not* characterized by the implementability of some collection of recursive denotational control structures.

On the positive side, however, a characterization of such epses of a rather different sort *is* shown herein. Though, perhaps not the insightful characterization sought after, this surprising result reveals that a hidden and inherent constructivity is always present in **krt**.

Know thyself.
– Greek proverb

Keywords: Computability Theory, Programming Language Semantics, Self-Reference.

1 Introduction

The first author has, for some time, been interested in the difficult problem of understanding and insightfully characterizing the power of *program or machine self-reference* (synonym: *program self-reflection*).[1] Initial mathematical attempts

* This is a slightly expanded version of a paper to appear in *CiE'07*. This paper received support from NSF Grant CCR-0208616.

[1] This paper does not address *linguistic* self-reference, e.g., in arithmetic [16].

on this subject [10, 11, 14] were based on conceptualizing the *constructive* form of *Kleene's Recursion Theorem* (**KRT**) (Property 2 below) in terms of an associated *non-denotational control structure* [14] (synonym: *connotational control structure* [14]). Beginning with the next subsection, we explain what we mean by program self-reference and self-knowledge, what **KRT** has to do with these, and how we model control structures. Then, we briefly highlight some relevant results from the prior literature. Finally, we summarize and relevantly interpret the main results of the present paper — which results are the recent progress on the still difficult problem mentioned at the beginning of this paragraph.

1.1 Kleene's Recursion Theorems

Let $\mathbb{N}$ be the set of natural numbers, $\{0, 1, 2, \ldots\}$. Let $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any fixed, 1-1, onto, computable mapping [13]. The function $\langle \cdot, \cdot \rangle$ enables us to restrict our attention to one-argument partial functions and still handle, with coding by $\langle \cdot, \cdot \rangle$, multiple argument cases.

For all one-argument partial functions ψ and all $p \in \mathbb{N}$, $\psi_p \stackrel{\text{def}}{=} \psi(\langle p, \cdot \rangle)$. A one-argument *partial computable* function ψ is an *effective programming system (*eps*)* $\stackrel{\text{def}}{\Leftrightarrow}$ for every one-argument partial computable function α, there exists p, such that $\psi_p = \alpha$ [12, 13, 9–11, 14].[2] Informally, one can think of ψ as a programming language (e.g., C++, Java, Haskell) and of p as a *program* within that language. In this sense, ψ_p is the partial computable function coded by ψ-program p. Thus, for all $p, x \in \mathbb{N}$, $\psi(\langle p, x \rangle)$ is the (coded) output on input (coded by) x of the program (coded by) p in that language.

For the remainder of the present subsection (1.1), let ψ be any fixed eps. The following property (Property 1) is the not-necessarily-constructive form of Kleene's Recursion Theorem *for the ψ-system*.[3]

*Property 1 (***krt** *for* eps ψ*).* $(\forall p)(\exists e)(\forall x)[\psi_e(x) = \psi_p(\langle e, x \rangle)]$.

One way to interpret Property 1 is as follows. ψ-program p represents an arbitrary preassigned, algorithmic task to perform with a self-copy; e represents a ψ-program that

1. creates a copy of itself, external to itself, and, *then*,
2. runs the preassigned task p on the pair consisting of this self-copy and e's input x.

The 'e' on the right-hand side of the equation in Property 1 *is* the self-copy of the original 'e' on the left-hand side of this equation. Thus, in an important

[2] In much of the literature on epses, e.g., [12, 7, 8], they are called *numberings* since, in such systems, programs are conveniently named by numbers. In learning theory contexts, e.g., [6, 18, 5], they are also referred to as *hypothesis spaces*.

[3] Rogers [13] popularized a fixed-point variant of Property 1: for all computable $f : \mathbb{N} \to \mathbb{N}$, there exists e such that $\psi_e = \psi_{f(e)}$. His variant should *not* be confused with Property 1. Riccardi [10] explored their interconnections.

sense, e is a program that *creates* complete (low level) *self-knowledge*. The way in which e uses this self-knowledge is according to how the preassigned task p says to.[4] Infinite regression is *not* needed since e projects its self-copy *externally* to itself [4]. We say above that this self-knowledge is complete since it *is* e's syntactic code-script, wiring/flow diagram, etc. For higher level knowledge about, say, e's behavioral propensities, e.g., ψ-program e runs in polynomial time, p can run a safe theorem prover on e perchance to prove such things about e, but e having access to e itself is more basic and fundamental than e merely having access to facts such as that it runs in polynomial time.

Self-knowledgeable programs have long been known to be an elegant theoretical tool in computability theory. Such programs can, when relevant, provide *succinct* solutions to problems "that would otherwise require extensive, complex treatment" [13] (see also [15]). Self-knowledge can also serve as a useful game-theoretic aid to strategy [4], e.g., in the *game* played between a robot and its environment [1, 3].

Of course, Property 1 asserts that, given p, there merely *exists* an e satisfying the equation in Property 1 for p. It is another problem to *find* such an e algorithmically *from* p. Here, then, is Property 2, the *constructive* form of Kleene's Recursion Theorem for the ψ-system, which makes this stronger assertion.

*Property 2 (***KRT** *for* eps ψ*).* There exists computable $r : \mathbb{N} \to \mathbb{N}$ such that $(\forall p, x)\big[\psi_{r(p)}(x) = \psi_p\big(\langle r(p), x\rangle\big)\big]$.

In Property 2, $r(p)$ plays the role of e in Property 1. Since r is computable, $r(p)$ can be found algorithmically from p.

1.2 Control Structures

From a programming languages standpoint, the r in Property 2 represents an *instance* (or *implementation*) of a *control structure* [10, 11, 14, 8, 5]. In the context of epses, an instance of a control structure provides a means of forming a composite program from given constituent programs and/or data. For comparison, an *instance in an* eps ψ *of the control structure* **if-then-else** is (by definition [10, 11]) a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for all a, b, c, and x,

$$\psi_{f(\langle a,b,c\rangle)}(x) = \begin{cases} \psi_b(x), & \text{if } \psi_a(x) \text{ converges}^5 \text{ and } \psi_a(x) > 0; \\ \psi_c(x), & \text{if } \psi_a(x) \text{ converges and } \psi_a(x) = 0; \\ \text{divergent, otherwise.} \end{cases} \tag{1}$$

An instance such as f above of **if-then-else** combines three ψ-programs, a, b, and c (and *no* data) to form a fourth (composite) ψ-program $f(\langle a, b, c\rangle)$.

[4] We care, of course, that **krt** provides not only self-knowledgeable programs, but also, self-knowledgeable programs that can *use* that knowledge in any preassigned algorithmic way. *Usable*, as opposed to empty, self-knowledge is what we care about.

[5] For all one-argument partial functions ψ and $x \in \mathbb{N}$, $\psi(x)$ *converges* iff there exists $y \in \mathbb{N}$ such that $\psi(x) = y$; $\psi(x)$ *diverges* iff there is *no* $y \in \mathbb{N}$ such that $\psi(x) = y$. If ψ is partial computable, and x is such that $\psi(x)$ diverges, then one can imagine that a program associated with ψ *goes into an infinite loop* on input x.

if-then-else is an example of a *nonrecursive denotational control structure* (synonym: *nonrecursive extensional control structure*). A *non*recursive denotational control structure is one for which the I/O behavior of a composite program may depend *only* upon the I/O behavior of the constituent programs and upon the data (see (a) of Definition 1 below). So, for example, the I/O behavior of such a composite program can*not* depend upon the number of symbols in, or the run-time complexity of, a constituent program.

A *recursive denotational control structure* (synonym: *recursive extensional control structure*) is like a *non*recursive denotational control structure where the I/O behavior of a composite program may depend, additionally, upon the I/O behavior of the composite program itself (see (b) of Definition 1 below). Consider the following example, chosen for illustrative purposes. Let an effective instance in an eps ψ of **recursive unbounded minimization** be a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for all a, b, and x,

$$\psi_{f(\langle a,b\rangle)}(x) = \begin{cases} \psi_b(x), & \text{if } \psi_a(x) \text{ converges and } \psi_a(x) > 0; \\ \psi_{f(\langle a,b\rangle)}(x+1), & \text{if } \psi_a(x) \text{ converges and } \psi_a(x) = 0; \\ \text{divergent}, & \text{otherwise.} \end{cases} \tag{2}$$

Note the use of $\psi_{f(\langle a,b\rangle)}$ in the second if-clause in (2). This use of $\psi_{f(\langle a,b\rangle)}$ is what makes **recursive unbounded minimization** a *recursive* denotational control structure.

For many recursive denotational control structures, there is *wiggle room* in how they may be implemented. For **recursive unbounded minimization**, this wiggle room manifests itself in the extreme diversity of the functions f that satisfy (2). For example, suppose that $f_1 : \mathbb{N} \to \mathbb{N}$ is computable and that, for all a, b, and x,

$$\psi_{f_1(\langle a,b\rangle)}(x) = \begin{cases} \psi_b(z), & \text{where } z \text{ is } \textit{least} \text{ such that } z \geq x, \\ & \quad (\forall y \in \{x, ..., z\})[\psi_a(y) \text{ converges}], \\ & \quad (\forall y \in \{x, ..., z-1\})[\psi_a(y) = 0], \\ & \quad \text{and } \psi_a(z) > 0, \text{ if such a } z \text{ exists;} \\ \text{divergent, otherwise.} \end{cases} \tag{3}$$

Then, $f = f_1$ is a solution of (2).

Next, suppose that a_0 is a ψ-program such that, for all x, $\psi_{a_0}(x)$ converges and $\psi_{a_0}(x) = 0$. Note that when $a = a_0$ in (2), (2) merely insists that $\psi_{f(\langle a,b\rangle)}(0) = \psi_{f(\langle a,b\rangle)}(1) = ...$, for any b. Thus, if $f_2 : \mathbb{N} \to \mathbb{N}$ is computable and, for all a, b, and x,

$$\psi_{f_2(\langle a,b\rangle)}(x) = \begin{cases} 5, & \text{if } a = a_0; \\ \psi_{f_1(\langle a,b\rangle)}(x), & \text{otherwise;} \end{cases} \tag{4}$$

then, $f = f_2$ is also a solution of (2). (In (4), the number 5 was chosen arbitrarily.)[6]

[6] Readers familiar with denotational semantics may recognize that f_1 provides a minimal fixed-point solution of (2); whereas, f_2 provides a *non*-minimal fixed-point solution of (2) [14, 17].

Of course, a composite program produced by a recursive denotational control structure *may* choose to *ignore* its own behavior. In this sense, recursive denotational control structures are a generalization of *non*recursive denotational control structures.

KRT, when viewed as a control structure, is *not* denotational in any sense.[7] So, a problem that the first author posed to Royer was to find a collection of denotational control structures whose implementability characterizes the epses in which **KRT** holds. The thinking was that *denotational control structures are easier to understand*, and such a collection would be a decomposition of **KRT** into more easily understood components. Royer proved that *no* such characterization exists, even if one allows the collection to contain *recursive* denotational control structures [14].

1.3 Summary of Results

krt is the focus in the present paper as it ostensibly involves *pure* self-reference without the required constructivity of **KRT**. So, a question we had is whether Royer's negative result mentioned above still holds if one replaces **KRT** by **krt**, i.e., whether there exists a collection of (possibly) recursive denotational control structures whose implementability characterizes the epses in which **krt** holds. One of our main results, Corollary 1 in Section 3, says that *no* such characterization exists. The proof is by a finite injury priority argument.[8]

In Section 3, we also consider a *relatively constructive* variant of **krt**. Suppose ξ is an eps and that ψ is partial computable, but not-necessarily an eps. We say that ξ-**KRT** *holds in* $\psi \stackrel{\text{def}}{\Leftrightarrow}$ there exists computable $r : \mathbb{N} \to \mathbb{N}$ such that $(\forall p, x)\big[\psi_{r(p)}(x) = \xi_p\big(\langle r(p), x\rangle\big)\big]$. Here, $r(p)$ is a self-knowledgeable ψ-program, where the preassigned task for $r(p)$ to employ on its self-copy is ξ-program p.

Theorem 2, our other main result, says: ψ is an eps in which **krt** holds $\Leftrightarrow$ $(\exists$ eps $\xi)[\xi$-**KRT** holds in $\psi]$. This implies that, if, for some eps ξ, ξ-**KRT** holds in merely partial computable ψ, then both ψ is an eps and **krt** holds in ψ. It also surprisingly implies that, if **krt** holds in an eps ψ, then it holds with *some degree of constructivity* — constructivity with respect to *some* eps ξ.

Section 2 just below provides notation and preliminaries.

Complete proofs of all theorems can be found in the appendix.

2 Notation and Preliminaries

Computability-theoretic concepts not explained below are treated in [13]. $\mathbb{N}$ denotes the set of natural numbers. $2\mathbb{N}$ and $2\mathbb{N}+1$ denote the sets of even and odd natural numbers, respectively. Lowercase Roman letters, with or without decorations, range over elements of $\mathbb{N}$ unless stated otherwise.

[7] Such control structures are called *connotational* [14].

[8] Rogers [13] explains priority arguments.

The pairing function $\langle \cdot, \cdot \rangle$ was introduced in Section 1. For all x, $\langle x \rangle \stackrel{\text{def}}{=} x$. For all $x_1, ..., x_n$, where $n > 2$, $\langle x_1, ..., x_n \rangle \stackrel{\text{def}}{=} \big\langle x_1, \langle x_2, ..., x_n \rangle \big\rangle$.

$\mathcal{P}$ denotes the collection of all one-argument partial functions. α, ξ, Ξ, σ, ψ, and Ψ, with or without decorations, range over elements of $\mathcal{P}$. We use Church's lambda-notation [13] to name partial functions, including total functions and predicates, as is standard in many programming languages. For example, $\lambda x\,.\,(x+1)$ denotes the one-argument (total) function that maps a natural number to its successor.

For all α and x, $\alpha(x)\!\downarrow$ denotes that $\alpha(x)$ converges; $\alpha(x)\!\uparrow$ denotes that $\alpha(x)$ diverges. We use $\uparrow$ in expressions to indicate divergence. For example, $\lambda x\,.\,\uparrow$ denotes the everywhere divergent partial computable function. For all α, $\text{dom}(\alpha) \stackrel{\text{def}}{=} \{x : \alpha(x)\!\downarrow\}$ and $\text{rng}(\alpha) \stackrel{\text{def}}{=} \{y : (\exists x)[\alpha(x) = y]\}$. We identify a partial function with its graph, e.g., we identify α with the set $\{(x, y) : \alpha(x) = y\}$. As noted in the introduction, for all ψ and p, $\psi_p \stackrel{\text{def}}{=} \psi_p(\langle p, \cdot \rangle)$.

$F_0, F_1, ...$ denotes a fixed, canonical enumeration of all one-argument finite functions [13, 9].

φ denotes a fixed, acceptable eps.[9] Φ denotes a fixed Blum complexity measure for φ [2].[10] For all p and t, φ_p^t and W_p^t are as follows.

$$\varphi_p^t \stackrel{\text{def}}{=} \{(x, y) : x \leq t \ \wedge \ \Phi_p(x) \leq t \ \wedge \ \varphi_p(x) = y\}. \tag{5}$$

$$W_p^t \stackrel{\text{def}}{=} \text{dom}(\varphi_p^t). \tag{6}$$

Γ and Θ, with or without decorations, range over mappings of type $\mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$, where $m + n > 0$.

For all $\Gamma : \mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$, where $m + n > 0$, Γ is a *computable operator*[11] $\stackrel{\text{def}}{\Leftrightarrow}$ there exists p such that, for all $x_1, ..., x_m$, $\alpha_1, ..., \alpha_n$, y, and z,

$$\begin{array}{c} \Gamma(x_1, ..., x_m, \alpha_1, ..., \alpha_n)(y) = z \\ \Leftrightarrow \\ (\exists i_1, ..., i_n, t)\big[(\forall j \in \{1, ..., n\})[F_{i_j} \subseteq \alpha_j] \\ \qquad \wedge \ \langle x_1, ..., x_m, i_1, ..., i_n, y, z \rangle \in W_p^t\big]. \end{array} \tag{7}$$

Intuitively, Γ is a computable operator iff there exists an algorithm for listing the *graph of* the partial function $\Gamma(x_1, ..., x_m, \alpha_1, ..., \alpha_n)$ from $x_1, ..., x_m$ and the *graphs of* the partial functions $\alpha_1, ..., \alpha_n$ — independently of the enumeration order chosen for each of $\alpha_1, ..., \alpha_n$. $\mathcal{C}$ ranges over collections of computable operators. For all computable operators $\Gamma : \mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$, where $m + n > 0$, and for

[9] An eps ψ is *acceptable* $\stackrel{\text{def}}{\Leftrightarrow}$ ($\forall$ epses ξ)($\exists$ computable $t : \mathbb{N} \to \mathbb{N}$)($\forall p$)$[\psi_{t(p)} = \xi_p]$ [12, 13, 9–11, 14]. Thus, the acceptable epses are exactly those epses into which every eps can be compiled. Any eps corresponding to a real-world, general purpose programming language (e.g., C++, Java, Haskell) is acceptable.

[10] For any partial computable function, e.g., an eps, many such measures exist.

[11] Rogers [13] calls the computable operators, *recursive operators*. We have chosen to use the former term so that we may reserve the term *recursive* for something that *refers to itself*.

all t, $\Gamma^t : \mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$ is the computable operator such that, for all $x_1, ..., x_m$, $\alpha_1, ..., \alpha_n$, y, and z,

$$\begin{array}{c} \Gamma^t(x_1, ..., x_m, \alpha_1, ..., \alpha_n)(y) = z \\ \Leftrightarrow \\ (\exists i_1, ..., i_n)\big[(\forall j \in \{1, ..., n\})[F_{i_j} \subseteq \alpha_j] \\ \wedge\ \langle x_1, ..., x_m, i_1, ..., i_n, y, z\rangle \in W_p^t\big], \end{array} \tag{8}$$

where p is any fixed φ-program as in (7) above for Γ. Clearly, for all computable operators $\Gamma : \mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$, where $m+n > 0$, there exists an algorithm for finding j from t, $x_1, ..., x_m$, and $i_1, ..., i_n$, such that $F_j = \Gamma^t(x_1, ..., x_m, F_{i_1}, ..., F_{i_n})$.

Definition 1. For all epses ψ, and all $f : \mathbb{N} \to \mathbb{N}$, (a) and (b) below.

(a) For all computable operators $\Gamma : \mathbb{N}^m \times \mathcal{P}^n \to \mathcal{P}$, where $m + n > 0$, f *is an effective instance in* ψ *of the* non*recursive denotational control structure determined by* $\Gamma \Leftrightarrow f$ is computable and, for all $x_1, ..., x_{m+n}$,

$$\psi_{f(\langle x_1, ..., x_{m+n}\rangle)} = \Gamma(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}). \tag{9}$$

(b) For all computable operators $\Theta : \mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, where $m + n > 0$, f *is an effective instance in* ψ *of the recursive denotational control structure determined by* $\Theta \Leftrightarrow f$ is computable and, for all $x_1, ..., x_{m+n}$,

$$\psi_{f(\langle x_1, ..., x_{m+n}\rangle)} = \Theta(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}, \psi_{f(\langle x_1, ..., x_{m+n}\rangle)}). \tag{10}$$

3 Results

Royer [14] proved that there is *no* collection of recursive denotational control structures whose implementability characterizes the epses in which **KRT** holds. Corollary 1, below, proves the analogous result for **krt**. Thus, even the *pure* self-reference embodied by **krt** cannot be decomposed into recursive denotational control structures.[12] Our proof is by a finite injury priority argument.

Definition 2. For all computable operators $\Theta : \mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, where $m+n > 0$, Θ *is recursively denotationally omnipresent* $\Leftrightarrow$ for all epses ψ, there exists an effective instance in ψ of the recursive denotational control structure determined by Θ.

[12] N.B. Our result does *not* subsume Royer's.

Theorem 1. Suppose that computable operator $\Theta : \mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, where $m + n > 0$, is *not* recursively denotationally omnipresent. Then, there exists an eps ψ such that (a) and (b) below.

(a) **krt** holds in ψ.
(b) There is *no* effective instance in ψ of the recursive denotational control structure determined by Θ.

Proof. See the appendix.

Corollary 1. There is *no* collection of computable operators $\mathcal{C}$ such that (a) *and* (b) below.

(a) For each $\Theta \in \mathcal{C}$, Θ has type $\mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, for some m and n, where $m + n > 0$.[13]
(b) For all epses ψ, **krt** holds in $\psi \Leftrightarrow (\forall \Theta \in \mathcal{C})$[there exists an effective instance in ψ of the recursive denotational control structure determined by Θ].

Proof. See the appendix.

Theorem 2, below, is our other main result. It reveals that a hidden and inherent constructivity is always present in **krt**.

Definition 3. For all epses ξ and partial computable ψ, ξ-**KRT** *holds in* $\psi \Leftrightarrow$ $(\exists$ computable $r : \mathbb{N} \to \mathbb{N})(\forall p, x)\big[\psi_{r(p)}(x) = \xi_p\big(\langle r(p), x\rangle\big)\big]$.

Theorem 2. For all partial computable ψ, ψ is an eps in which **krt** holds $\Leftrightarrow$ $(\exists$ eps $\xi)$[ξ-**KRT** holds in ψ].

Proof. See the appendix.

References

1. C. Adami. What do robots dream of? *Science*, 314:1093–1094, 2006.
2. M. Blum. A machine independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.
3. J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314:1118–1121, 2006.
4. J. Case. Infinitary self-reference in learning theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:3–16, 1994.
5. J. Case, S. Jain, and M. Suraj. Control structures in hypothesis spaces: The influence on learning. *Theoretical Computer Science*, 270(1-2):287–308, 2002.
6. R. Freivalds, E. Kinber, and R. Wiehagen. Inductive inference and computable one-one numberings. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 28:463–479, 1982.
7. S. Goncharov and A. Sorbi. Generalized computable numberings and non-trivial Rogers semilattices. *Algebra and Logic*, 36:359–369, 1997.

[13] (a) ensures that each $\Theta \in \mathcal{C}$ determines a recursive denotational control structure (see (b) of Definition 1).

8. S. Jain and J. Nessel. Some independence results for control structures in complete numberings. *Journal of Symbolic Logic*, 66(1):357–382, 2001.
9. M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms*. North Holland, New York, 1978.
10. G. Riccardi. *The Independence of Control Structures in Abstract Programming Systems*. PhD thesis, SUNY Buffalo, 1980.
11. G. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22:107–143, 1981.
12. H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.
13. H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967. Reprinted, MIT Press, 1987.
14. J. Royer. *A Connotational Theory of Program Structure*. Lecture Notes in Computer Science 273. Springer-Verlag, 1987.
15. J. Royer and J. Case. *Subrecursive Programming Systems: Complexity and Succinctness*. Research monograph in *Progress in Theoretical Computer Science*. Birkhäuser Boston, 1994.
16. C. Smorynski. Fifty years of self-reference in arithmetic. *Notre Dame Journal of Formal Logic*, 22(4):357–374, 1981.
17. G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. Foundations of Computing Series. MIT Press, 1993.
18. T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In Klaus P. Jantke and Steffen Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer-Verlag, 1995.

Appendix

Definition 4 (Rogers [12, 13]). For all epses ξ and ψ, (a) and (b) below.

(a) $\xi \leq_R \psi$ (pronounced: ξ *is Rogers reducible to* ψ) $\stackrel{\text{def}}{\Leftrightarrow}$

$$(\exists \text{ computable } t : \mathbb{N} \to \mathbb{N})(\forall p)[\psi_{t(p)} = \xi_p]. \tag{11}$$

(b) ψ is *acceptable* $\stackrel{\text{def}}{\Leftrightarrow}$

$$(\forall \text{ epses } \xi)[\xi \leq_R \psi]. \tag{12}$$

Thus, $\xi \leq_R \psi$ whenever it is possible to compile ξ-programs into ψ-programs. Furthermore, the acceptable epses are exactly those epses into which every eps can be compiled.

The following lemma is used in later parts of the appendix.

Lemma 1. Suppose that ψ partial computable, and that ξ is an eps Further suppose that $A \subseteq \mathbb{N}$ is such that, for all p, there exists $a \in A$ such that

$$\psi_a = \xi_p(\langle a, \cdot \rangle). \tag{13}$$

Then, (a) and (b) below.

(a) For all partial computable α, there exists $a \in A$ such that $\psi_a = \alpha$.
(b) ψ is an eps in which **krt** holds.

Proof. Suppose the hypotheses. To see (a), let partial computable α be fixed, and let p be such that, for all a and x,

$$\xi_p(\langle a, x \rangle) = \alpha(x). \tag{14}$$

Let $a \in A$ be as in (13) for p. Then, for all x,

$$\begin{aligned} \psi_a(x) &= \xi_p(\langle a, x \rangle) \ \{\text{by (13)}\} \\ &= \alpha(x) \qquad \{\text{by (14)}\}. \end{aligned}$$

To see (b), note that (a) implies that ψ is an eps Next, let ψ-program b be fixed, and let p be such that

$$\xi_p = \psi_b. \tag{15}$$

Let $a \in A$ be as in (13) for p. Then, for all x,

$$\begin{aligned} \psi_a(x) &= \xi_p(\langle a, x \rangle) \ \{\text{by (13)}\} \\ &= \psi_b(\langle a, x \rangle) \ \{\text{by (15)}\}. \end{aligned}$$

□ *(Lemma 1)*

The following lemma is used in the proof of Theorem 1.

Lemma 2. Suppose that computable operator $\Theta : \mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, where $m+n > 0$, is fixed. Further suppose that ψ is partial computable, and that Ψ is a Blum complexity measure for ψ. For a and t, let

$$\psi_a^t = \{(x, y) : x \leq t \ \wedge \ \Psi_a(x) \leq t \ \wedge \ \psi_a(x) = y\}. \tag{16}$$

Then, for all $g : \mathbb{N} \to \mathbb{N}$, (a) and (b) below are equivalent.

(a) For all $x_1, ..., x_{m+n}$,

$$\psi_{g(\langle x_1,...,x_{m+n}\rangle)} = \Theta(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}, \psi_{g(\langle x_1,...,x_{m+n}\rangle)}). \tag{17}$$

(b) For all s, there exists t such that, for all $\langle x_1, ..., x_{m+n}\rangle < s$, (i) and (ii) below.
  (i) $\psi^s_{g(\langle x_1,...,x_{m+n}\rangle)} \subseteq \Theta^t(x_1, ..., x_m, \psi^t_{x_{m+1}}, ..., \psi^t_{x_{m+n}}, \psi^t_{g(\langle x_1,...,x_{m+n}\rangle)})$.
  (ii) $\Theta^s(x_1, ..., x_m, \psi^s_{x_{m+1}}, ..., \psi^s_{x_{m+n}}, \psi^s_{g(\langle x_1,...,x_{m+n}\rangle)}) \subseteq \psi^t_{g(\langle x_1,...,x_{m+n}\rangle)}$.

Proof. A straightforward argument using the monotonicity and continuity properties of Θ [13, page 147]. □ *(Lemma 2)*

Theorem 1. Suppose that computable operator $\Theta : \mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, where $m+n > 0$, is *not* recursively denotationally omnipresent. Then, there exists an eps ψ such that (a) and (b) below.

(a) **krt** holds in ψ.
(b) There is *no* effective instance in ψ of the recursive denotational control structure determined by Θ.

Proof. Let Θ be as stated. Since Θ is *not* recursively denotationally omnipresent, there exists an eps ξ such that there is *no* effective instance in ξ of the recursive denotational control structure determined by Θ.

ψ is constructed via a finite injury priority argument. The requirements, in order of *decreasing* priority, are: $S, R_0, R_1, ...$, where, for all q, R_q and S are as follows.

$$\begin{aligned} R_q &\Leftrightarrow (\exists a)[\psi_a = \varphi_q(\langle a, \cdot\rangle)]. \\ S &\Leftrightarrow (\forall \text{ computable } g : \mathbb{N} \to \mathbb{N})(\exists x_1, ..., x_{m+n}) \\ &\qquad [\psi_{g(\langle x_1,...,x_{m+n}\rangle)} \neq \Theta(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}, \psi_{g(\langle x_1,...,x_{m+n}\rangle)})]. \end{aligned}$$

The satisfaction of R_q, for all q, ensures that ψ is an eps in which **krt** holds. The satisfaction of S ensures that there is *no* effective instance in ψ of the recursive denotational control structure determined by Θ.

Our strategy for satisfying S is as follows. We construct ψ so that:

- $\xi \leq_R \psi$, and
- for all computable $g : \mathbb{N} \to \mathbb{N}$, if g were a *counterexample* to S, then there would exist partial computable τ such that $\big(\forall c \in \mathrm{rng}(g)\big)[\tau(c)\!\downarrow \wedge\, \xi_{\tau(c)} = \psi_c]$.

Given that ψ has these properties, the existence of a counterexample to S would imply the existence of an effective instance in ξ of the recursive denotational control structure determined by Θ — a contradiction. Thus, ensuring the above will guarantee that S is satisfied.

ψ is constructed in stages. For all a and t, ψ_a^t denotes ψ_a at the beginning of stage t. For all a, $\psi_a^0 = \lambda x.\uparrow$. For all a, t, and x, $\psi_a^{t+1}(x) = \psi_a^t(x)$ unless stated otherwise.

In conjunction with ψ, a partial computable σ and a limit-computable d are constructed. σ and d are used to help satisfy the S requirement. For all t, σ^t and d^t denote σ and d, respectively, at the beginning of stage t. For all a, σ^0 and d^0 are as follows.

$$\sigma^0(a) = \begin{cases} (a-1) \div 2, \text{ if } a \in 2\mathbb{N}+1; \\ \uparrow, \qquad\qquad \text{otherwise.} \end{cases} \tag{18}$$

$$d^0(a) = 0. \tag{19}$$

For all t and a, $\sigma^{t+1}(a) = \sigma^t(a)$ unless stated otherwise. Similarly, for all t and a, $d^{t+1}(a) = d^t(a)$ unless stated otherwise. The following will be clear, by construction.

$$(\forall t)[\mathrm{dom}(\sigma^t) \cap 2\mathbb{N} \text{ is finite}]. \tag{20}$$

$$\lambda t, a.[\sigma^t(a)\downarrow] \text{ is a computable predicate.} \tag{21}$$

Let r be such that, for all t and q,

$$r^t(q) = \begin{cases} 2\langle q, i\rangle, \text{ where } i \text{ is } \textit{least} \text{ such that } \sigma^t(2\langle q, i\rangle)\uparrow \\ \quad \text{and } (\forall p < q)[2\langle q, i\rangle > r^t(p)]. \end{cases} \tag{22}$$

r is used to help satisfy the R requirements. It can be shown, by a straightforward induction, that, if (20) holds as claimed, then, for all t, r^t is total and monotonically increasing. Furthermore, if (21) holds as claimed, then $\lambda t, q.r^t(q)$ is computable. Clearly, by (22), if t, q, and i are such that $r^t(q) = 2\langle q, i\rangle$, then $\sigma^t(2\langle q, i\rangle)\uparrow$. It follows that, for all t, $\mathrm{dom}(\sigma^t) \cap \mathrm{rng}(r^t) = \emptyset$.

For all q and t, it can be seen that R_q is injured in stage t whenever $r^{t+1}(q) \neq r^t(q)$. There are two ways that this can occur. The first is when $[(\sigma^t \circ r^t)(q)\uparrow \wedge (\sigma^{t+1} \circ r^t)(q)\downarrow]$, equivalently, $(\sigma \circ r^t)(q)$ becomes defined in stage $t+1$. The second is when, for some $p < q$, $[r^t(p) < r^t(q) \ \wedge \ r^{t+1}(p) \geq r^t(q)]$. In this latter case, R_q is injured as a result of a *cascading effect*. Clearly, either condition causes $r^{t+1}(q) \neq r^t(q)$.

Let Ξ be a Blum complexity measure for ξ. For all p and t, let

$$\xi_p^t = \{(x, y) : x \leq t \ \wedge \ \Xi_p(x) \leq t \ \wedge \ \xi_p(x) = y\}. \tag{23}$$

Construct ψ, σ, and d by executing successive stages $t = 0, 1, ...$ as follows.

STAGE $t = \langle a, i \rangle$.

CASE $\sigma^t(a)\downarrow$. Let $p = \sigma^t(a)$ and, for all $x \leq t+1$ such that $[\psi_a^t(x)\uparrow \ \wedge \ \xi_p^t(x)\downarrow]$, set $\psi_a^{t+1}(x) = \xi_p^t(x)$.

CASE $a \in \mathrm{rng}(r^t)$. Perform steps (1) and (2) below.

(1) Let $s = d^t(a)$ and determine whether conditions (a) and (b) below are satisfied.

(a) $\psi_a^t(s)\downarrow$.

(b) For all $\langle x_1, ..., x_{m+n} \rangle < s$ and b such that $\psi_a^t(\langle x_1, ..., x_{m+n} \rangle) = b$, (i) and (ii) below.

(i) $\psi_b^s \subseteq \Theta^t(x_1, ..., x_m, \psi_{x_{m+1}}^t, ..., \psi_{x_{m+n}}^t, \psi_b^t)$.

(ii) $\Theta^s(x_1, ..., x_m, \psi_{x_{m+1}}^s, ..., \psi_{x_{m+n}}^s, \psi_b^s) \subseteq \psi_b^t$.

If conditions (a) and (b) are satisfied, then perform substeps $(*)$ and $(**)$ below.

$(*)$ Let $c = \psi_a^t(s)$. If $[c > a \ \wedge \ \sigma^t(c)\uparrow]$, then find any p such that $\psi_c^t \subseteq \xi_p$ and set $\sigma^{t+1}(c) = p$.

$(**)$ Set $d^{t+1}(a) = s + 1$.

(2) Let q be be such that $r^t(q) = a$ and, for all $x \leq t+1$ such that $[\psi_a^t(x)\uparrow \ \wedge \ \varphi_q^t(\langle a, x \rangle)\downarrow]$, set $\psi_a^{t+1}(x) = \varphi_q^t(\langle a, x \rangle)$.

End of construction of ψ, σ, and d.

Claim 1. For all a, t, and x, if $\psi_a^t(x)\downarrow$, then $x \leq t$.

Proof of Claim. Clear by the construction of ψ. □ *(Claim 1)*

Claim 2. Let Ψ be such that, for all a and x,

$$\Psi_a(x) = \begin{cases} t, \text{ where } t \text{ is } \textit{least} \text{ such that } \psi_a^t(x)\downarrow, \\ \qquad \text{if such a } t \text{ exists;} \\ \uparrow, \text{ otherwise.} \end{cases} \tag{24}$$

Then, Ψ is a Blum complexity measure for ψ. Moreover, for all a and t,

$$\psi_a^t = \{(x, y) : x \leq t \ \wedge \ \Psi_a(x) \leq t \ \wedge \ \psi_a(x) = y\}. \tag{25}$$

Proof of Claim. Follows from the construction of ψ and Claim 1. □ *(Claim 2)*

Claim 3. For all $a \in \mathrm{dom}(\sigma)$, $\psi_a = \xi_{\sigma(a)}$.

Proof of Claim. Let $a \in \mathrm{dom}(\sigma)$ be fixed. Let t be *least* such that $\sigma^t(a)\downarrow$, and let $p = \sigma(a)$. Note that, if $t = 0$, then ($a \in 2\mathbb{N}+1$ and) $\psi_a^t = \lambda x.\uparrow \subseteq \xi_p$. On the other hand, if $t > 0$, then, clearly, p was chosen so that $\psi_a^t \subseteq \xi_p$. Thus, in either case, $\psi_a^t \subseteq \xi_p$. Next, note that, for infinitely many $u \geq t$, $\psi_a^{u+1}(x)$ is set equal to $\xi_p^u(x)$ for each $x \leq u+1$ such that $[\psi_a^u(x)\uparrow \ \wedge \ \xi_p^u(x)\downarrow]$. Furthermore, beginning with stage t, this is the only way that pairs are inserted into the graph of ψ_a. Clearly, then, in the limit, $\psi_a = \xi_p$. □ *(Claim 3)*

Claim 4. For all p, $\psi_{2p+1} = \xi_p$.

Proof of Claim. Immediate by Claim 3 and (18). □ *(Claim 4)*

Claim 5. For all t, a, and s, if $[d^t(a) = s \ \wedge \ d^{t+1}(a) = s+1]$, then (a)-(c) below.

(a) For all $\langle x_1, ..., x_{m+n} \rangle \leq s$, $\psi_a^t(\langle x_1, ..., x_{m+n} \rangle)\downarrow$.
(b) For all $\langle x_1, ..., x_{m+n} \rangle < s$ and b such that $\psi_a^t(\langle x_1, ..., x_{m+n} \rangle) = b$, (i) and (ii) below.
  (i) $\psi_b^s \subseteq \Theta^t(x_1, ..., x_m, \psi^t_{x_{m+1}}, ..., \psi^t_{x_{m+n}}, \psi_b^t)$.
  (ii) $\Theta^s(x_1, ..., x_m, \psi^s_{x_{m+1}}, ..., \psi^s_{x_{m+n}}, \psi_b^s) \subseteq \psi_b^t$.
(c) For all $\langle x_1, ..., x_{m+n} \rangle \leq s$ and c such that $[\psi_a^t(\langle x_1, ..., x_{m+n} \rangle) = c \ \wedge \ c > a]$, $\sigma^{t+1}(c)\downarrow$.

Proof of Claim. (a) and (c) are each proven by a straightforward induction. (b) is clear by the construction of ψ and d. □ *(Claim 5)*

Claim 6. For all a, there exists t such that, for all $u > t$, $d^u(a) = d^t(a)$.

Proof of Claim. By way of contradiction, let a be such that, for infinitely many t, $d^{t+1}(a) = d^t(a) + 1$. Then, (a)-(c) below.

(a) ψ_a is total.
(b) For all $x_1, ..., x_{m+n}$,

$$\psi_{\psi_a(\langle x_1, ..., x_{m+n} \rangle)} = \Theta(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}, \psi_{\psi_a(\langle x_1, ..., x_{m+n} \rangle)}). \quad (26)$$

(c) For all $c \in \text{rng}(\psi_a)$ such that $c > a$, $\sigma(c)\downarrow$.

(a) and (c) follow immediately from (a) and (c), respectively, of Claim 5. (b) follows from Claim 2, (b) of Claim 5, and the right-to-left direction of Lemma 2.

Let $p_0, ..., p_a$ be such that, for all $c \in \{0, ..., a\}$,

$$\xi_{p_c} = \psi_c. \quad (27)$$

Let τ be such that, for all c,

$$\tau(c) = \begin{cases} p_c, & \text{if } c \leq a; \\ \sigma(c), & \text{otherwise.} \end{cases} \quad (28)$$

Clearly, τ is partially computable. By (27) and Claim 3,

$$\big(\forall c \in \text{dom}(\tau)\big)[\xi_{\tau(c)} = \psi_c]. \quad (29)$$

Clearly, $\{0, ..., a\} \subseteq \text{dom}(\tau)$. Furthermore, by (c) above,

$$\text{rng}(\psi_a) \subseteq \text{dom}(\tau). \quad (30)$$

Let $f : \mathbb{N} \to \mathbb{N}$ be such that, for all $x_1, ..., x_{m+n}$,

$$f(\langle x_1, ..., x_{m+n} \rangle) = (\tau \circ \psi_a)(\langle x_1, ..., x_m, 2x_{m+1} + 1, ..., 2x_{m+n} + 1 \rangle). \quad (31)$$

Clearly, f is computable. Furthermore, for all $x_1, ..., x_{m+n}$,

$$\begin{aligned}
&\xi_{f(\langle x_1,...,x_{m+n}\rangle)} \\
&= \xi_{(\tau\circ\psi_a)(\langle x_1,...,x_m,2x_{m+1}+1,...,2x_{m+n}+1\rangle)} && \{\text{by (31)}\} \\
&= \psi_{\psi_a(\langle x_1,...,x_m,2x_{m+1}+1,...,2x_{m+n}+1\rangle)} && \{\text{by (29) and (30)}\} \\
&= \Theta(x_1, ..., x_m, \psi_{2x_{m+1}+1}, ..., \psi_{2x_{m+n}+1}, \\
&\quad \psi_{\psi_a(\langle x_1,...,x_m,2x_{m+1}+1,...,2x_{m+n}+1\rangle)}) && \{\text{by (b) above}\} \\
&= \Theta(x_1, ..., x_m, \psi_{2x_{m+1}+1}, ..., \psi_{2x_{m+n}+1}, \\
&\quad \xi_{(\tau\circ\psi_a)(\langle x_1,...,x_m,2x_{m+1}+1,...,2x_{m+n}+1\rangle)}) && \{\text{by (29) and (30)}\} \\
&= \Theta(x_1, ..., x_m, \psi_{2x_{m+1}+1}, ..., \psi_{2x_{m+n}+1}, \xi_{f(\langle x_1,...,x_{m+n}\rangle)}) && \{\text{by (31)}\} \\
&= \Theta(x_1, ..., x_m, \xi_{x_{m+1}}, ..., \xi_{x_{m+n}}, \xi_{f(\langle x_1,...,x_{m+n}\rangle)}) && \{\text{by Claim 4}\}.
\end{aligned}$$

Thus, f is an effective instance in ξ of the recursive denotational control structure determined by Θ — a contradiction. □ *(Claim 6)*

Claim 7. For all q, there exists t such that, for all $u > t$, $r^u(q) = r^t(q)$.

Proof of Claim. By way of contradiction, let q be *least* such that, for infinitely many t, $r^{t+1}(q) \neq r^t(q)$. By the choice of q, there exists t such that, for all $p < q$ and $u \geq t$, $r^u(p) = r^t(p)$. By Claim 6, there exists $u \geq t$ such that, for all $p < q$ and $v \geq u$, $(d^v \circ r^t)(p) = (d^u \circ r^t)(p)$. Clearly, for all $p < q$ and $v \geq u$,

$$(d^{v+1} \circ r^v)(p) = (d^{v+1} \circ r^t)(p) = (d^v \circ r^t)(p) = (d^v \circ r^v)(p). \tag{32}$$

Let $v \geq u$ be such that $r^{v+1}(q) \neq r^v(q)$. Clearly, by the construction of ψ and d, there must exist $a < r^v(q)$ such that $d^{v+1}(a) \neq d^v(a)$. Furthermore, there must exist p such that $r^v(p) = a$. Finally, since r^v is monotonically increasing and $r^v(p) = a < r^v(q)$, it must be the case that $p < q$. To summarize: there exists $p < q$ and $v \geq u$ such that

$$(d^{v+1} \circ r^v)(p) \neq (d^v \circ r^v)(p). \tag{33}$$

But this contradicts (32). □ *(Claim 7)*

Claim 8. For all q, there exists $a \in \mathrm{rng}(r)$ such that $\psi_a = \varphi_q(\langle a, \cdot\rangle)$, i.e., R_q is satisfied.

Proof of Claim. Let q be fixed. By Claim 7, there exists $a = r(q)$. Note that, for infinitely many t, $\psi_a^{t+1}(x)$ is set equal to $\varphi_q^t(\langle a, x\rangle)$ for each $x \leq t+1$ such that $[\psi_a^t(x)\!\uparrow \ \wedge \ \varphi_q^t(\langle a, x\rangle)\!\downarrow]$. Furthermore, this is the only way that pairs are inserted into the graph of ψ_a. Clearly, then, in the limit, $\psi_a = \varphi_q(\langle a, \cdot\rangle)$.
□ *(Claim 8)*

Claim 9.

(a) For all partial computable α, there exists $a \in \mathrm{rng}(r)$ such that $\psi_a = \alpha$.
(b) ψ is an eps in which **krt** holds.

Proof of Claim. Immediate by Claim 8 and Lemma 1. □ *(Claim 9)*

Claim 10. There is *no* effective instance in ψ of the recursive denotational control structure determined by Θ, i.e., S is satisfied.

Proof of Claim. Suppose, by way of contradiction, otherwise. Let $g : \mathbb{N} \to \mathbb{N}$ be an effective instance in ψ of the recursive denotational control structure determined by Θ, i.e., for all $x_1, ..., x_{m+n}$,

$$\psi_{g(\langle x_1,...,x_{m+n}\rangle)} = \Theta(x_1, ..., x_m, \psi_{x_{m+1}}, ..., \psi_{x_{m+n}}, \psi_{g(\langle x_1,...,x_{m+n}\rangle)}). \quad (34)$$

By (a) of Claim 9, there exists $a \in \mathrm{rng}(r)$ such that $\psi_a = g$. Since ψ_a is total, for all s, there exists t such that $\psi_a^t(s)\downarrow$. Furthermore, by Claim 2 and the left-to-right direction of Lemma 2, for all s, there exists t such that, for all $\langle x_1, ..., x_{m+n}\rangle < s$, (i) and (ii) below.

(i) $\psi^s_{\psi_a(\langle x_1,...,x_{m+n}\rangle)} \subseteq \Theta^t(x_1, ..., x_m, \psi^t_{x_{m+1}}, ..., \psi^t_{x_{m+n}}, \psi^t_{\psi_a(\langle x_1,...,x_{m+n}\rangle)})$.
(ii) $\Theta^s(x_1, ..., x_m, \psi^s_{x_{m+1}}, ..., \psi^s_{x_{m+n}}, \psi^s_{\psi_a(\langle x_1,...,x_{m+n}\rangle)}) \subseteq \psi^t_{\psi_a(\langle x_1,...,x_{m+n}\rangle)}$.

Thus, it must be the case that, for infinitely many t, $d^{t+1}(a) \neq d^t(a)$. But this contradicts Claim 6. □ *(Claim 10)*

□ *(Theorem 1)*

Corollary 1. There is *no* collection of computable operators $\mathcal{C}$ such that (a) *and* (b) below.

(a) For each $\Theta \in \mathcal{C}$, Θ has type $\mathbb{N}^m \times \mathcal{P}^{n+1} \to \mathcal{P}$, for some m and n, where $m + n > 0$.[14]
(b) For all epses ψ, **krt** holds in ψ iff there exists an effective instance in ψ of the recursive denotational control structure determined by Θ, for each $\Theta \in \mathcal{C}$.

Proof of Corollary. Suppose, by way of contradiction, that such an $\mathcal{C}$ exists. Consider the following cases.

> Case $(\forall \Theta \in \mathcal{C})[\Theta$ is recursively denotationally omnipresent]. Clearly, this would imply that **krt** holds in every eps — a contradiction.
> Case $(\exists \Theta \in \mathcal{C})[\Theta$ is *not* recursively denotationally omnipresent]. Let Θ be a computable operator asserted to exist by the case. Let ψ be an eps as in Theorem 1 for Θ. Then, **krt** holds in ψ but there is *no* effective instance in ψ of the recursive denotational control structure determined by Θ — a contradiction.

□ *(Corollary 1)*

[14] (a) ensures that each $\Theta \in \mathcal{C}$ determines a recursive denotational control structure (see (b) of Definition 1).

Theorem 2. For all partial computable ψ, ψ is an eps in which **krt** holds $\Leftrightarrow$ $(\exists\ \mathsf{eps}\ \xi)[\xi$-**KRT** holds in $\psi]$.

Proof.

($\Leftarrow$) Immediate by (b) of Lemma 1.

($\Rightarrow$) Let ψ be as stated. Let ξ be such that, for all a, b, x_1, and x_2,

$$\xi_{\langle a,b\rangle}(\langle x_1, x_2\rangle) = \begin{cases} \psi_a(x_2), & \text{if } x_1 = a; \\ \psi_b(\langle x_1, x_2\rangle), & \text{otherwise.} \end{cases} \tag{35}$$

Claim 1. ξ is an eps

Proof of Claim. Clearly, ξ is partial computable. To complete the proof of the claim, it suffices to show that, for all b, there exists a such that $\xi_{\langle a,b\rangle} = \psi_b$. Let ψ-program b be fixed. By **krt** in ψ, there exists a such that, for all x,

$$\psi_a(x) = \psi_b(\langle a, x\rangle). \tag{36}$$

For all x_1 and x_2, consider the following cases.

Case $x_1 = a$. Then,

$$\begin{aligned} \xi_{\langle a,b\rangle}(\langle a, x_2\rangle) &= \psi_a(x_2) && \{\text{by (35)}\} \\ &= \psi_b(\langle a, x_2\rangle) && \{\text{by (36)}\}. \end{aligned}$$

Case $x_1 \neq a$. Then, by (35), $\xi_{\langle a,b\rangle}(\langle x_1, x_2\rangle) = \psi_b(\langle x_1, x_2\rangle)$.

□ *(Claim 1)*

Claim 2. ξ-**KRT** holds in ψ.

Proof of Claim. Let $r : \mathbb{N} \to \mathbb{N}$ be such that, for all a and b, $r(\langle a, b\rangle) = a$. Clearly, r is computable. Furthermore, for all a, b, and x,

$$\begin{aligned} \psi_{r(\langle a,b\rangle)}(x) &= \psi_a(x) && \{\text{by the choice of } r\} \\ &= \xi_{\langle a,b\rangle}(\langle a, x\rangle) && \{\text{by (35)}\} \\ &= \xi_{\langle a,b\rangle}\big(\big\langle r(\langle a, b\rangle), x\big\rangle\big) && \{\text{by the choice of } r\}. \end{aligned}$$

□ *(Claim 2)*

□ *(Theorem 2)*