Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 202 (2008) 171–189
www.elsevier.com/locate/entcs

Sequential Real Number Computation and Recursive Relations

J. Raymundo Marcial-Romero[1,2]

División de Computación UAEM
Ciudad Universitaria S/N, 50040
Toluca, Estado de México México

M. Andrew Moshier[3]

Department of Math and Computer Science
Chapman University
Orange CA 92867 USA

Abstract

In the first author's thesis [9], a sequential language, LRT, for real number computation is investigated. The thesis includes a proof that all polynomials are programmable, but that work comes short of giving a complete characterization of the expressive power of the language even for first-order functions. The technical problem is that LRT is non-deterministic. So a natural characterization of its expressive power should be in terms of relations rather than functions. In [2], Brattka investigates a formalization of recursive relations in the style of Kleene's recursive functions on the natural numbers. This paper establishes the expressive power of LRT_p, a variant of LRT, in terms of Brattka's recursive relations. Because Brattka already did the work of establishing the precise connection between his recursive relations and Type 2 Theory of Effectivity, we thus obtain a complete characterization of first-order definability in LRT_p.

Keywords: exact real-number computation, sequential computation, recursive relations, semantics, non-determinism, PCF

1 Introduction

In the literature on real number computation, several papers follow an idea originally due to Scott [17] of interpreting a type for real numbers in the domain of compact intervals (for simplicity, often restricted to the closed unit interval). In particular, extensions to PCF following this approach are investigated at length in

[1] This work was performed during the first author's visit to Chapman University in the Fall of 2006. Dr. Marcial-Romero's visit was funded by a grant from La Academia Mexicana de Ciencias y la Fundación México-Estados Unidos para la Ciencia (AMC-FUMEC) and the project PROMEP/103.5/05/1696.

[2] Email: rmarcial@fi.uaemex.mx

[3] Email:moshier@chapman.edu

doi:10.1016/j.entcs.2008.03.014

Escardó's thesis [5]. See Di Gianantonio [3], Potts et. al. [16] and Farjudian [7] for other examples of this approach. One of the most striking results along this line is Escardó, Hofmann and Streicher's proof [6] that "parallel if" can be implemented in a language that includes addition extended canonically to the domain of partial reals. This means that in order to have a reasonably expressive language with sequential interpretation, one must give up the canonical extension of addition. One way to do this is to introduce non-deterministic choice into the language. In [10,11], the sequential, non-deterministic language LRT is defined. In those papers, it is also shown that the non-determinism must be interpreted via the Hoare power domain. So, the ground types of the language are interpreted as Hoare power domains. It is the interaction of partiality and non-determinism that characterizes the basic idea of LRT.

LRT, with its sequential, non-deterministic semantics, seems naturally suited to a relational view of computation. And yet, all investigations into its expressive power [9,10,11] concentrate exclusively on functions. In particular, the operational concept of "strong convergence" is really useful only for analysis of programs that denote functions. The non-determinism of a strongly convergent program comes in only as the program produces partial results. In the limit, the non-deterministically produced partial outputs converge to a total (determined) value. Thus strong convergence, as it stands, is not useful for analyzing programs that are intended to produce non-deterministic (yet total) outputs. Furthermore, strong convergence is tied closely to the call-by-name semantics of LRT, which is forced by operational considerations. In particular, call-by-value simply makes no sense for the real number type in LRT because a "value" only corresponds to a converging sequence of partial results. Since our goal is to understand relations, we extend the language to include a `let` construct, and propose a generalization of strong convergence and a family of semantic interpretations of `let` parameterized by a positive real number p. The idea is to allow for reduction of a let term when a partial value is known within precision p. We call this reduction strategy *call-by-partial-value*. The corresponding denotational semantics employs several ideas familiar to domain theorists, including measurement as defined by Martin in [12] and a monadic treatment of the distinction between value and computation as in Moggi [14]. Furthermore, because we are interested in relations, say, on product types, we also extend the language to have explicit products of ground types.

In domain theoretic approaches to real number computation, one usually interprets the real number type as a domain D into which $\mathbb{R}$ embeds topologically. Typically, $\mathbb{R}$ embeds as the subspace of maximal elements of D, [17]. Thus, elements of D are taken to denote *partial numbers*, or *information* about real numbers. More generally, given a topological space X, a *domain model for* X [12] is a continuous domain D equipped with an embedding e of X *onto* $\max D$, the set of maximal elements of D.

LRT interprets the real number type as the Hoare power domain ($\mathcal{P}^H$) of a model of $[0,1]$. In other words, the elements of this type are not merely partial real numbers, but are non-deterministic computations of partial real numbers. The reals

still embed in the resulting domain, but not as maximal elements. This introduces an extra layer of indirection with respect to the "actual" real numbers. The other basic types of LRT are treated exactly the same way. Namely, we fix models for the natural numbers and the Booleans, and deal with the Hoare power domains of these models.

The Hoare power domain itself constitutes a strong monad. Thus, one can adopt methods of Moggi [14], or better yet, Levy's call-by-push-value [8] generalization of Moggi's method, to talk about the relation between computations (elements of a powerdomain) and values (generators of that powerdomain). But in the case of LRT, the *values* are already indirect ways of talking about real numbers, natural numbers and Booleans.

For the purposes of this paper, a *(total) value* is a datum drawn from a complete metric space. We are concerned mostly with values from $\mathbb{R}$, from the two discrete spaces $\mathbb{N}$ and $\mathbb{T}$ and from finite products of these. A *partial value* is a datum drawn from a designated corresponding domain model. Specifically, an element of $\mathcal{R}_\perp$ is a "partial real", an element of the flat domain of Booleans $\mathbb{T}_\perp$ is a "partial Boolean", and an element of the flat domain of natural numbers $\mathbb{N}_\perp$ is a "partial natural number." Elements of a product of these domains are partial values for the analogous product of metric spaces. We are justified in thinking of a tuple of partial values as a partial tuple because of the easily checked fact that the property "D is a domain model for X" is preserved by taking finite products.

A "real", "Boolean" or "natural number" *computation* belongs to $\mathcal{P}^H(\mathcal{R}_\perp)$, $\mathcal{P}^H(\mathbb{T}_\perp)$ or $\mathcal{P}^H(\mathbb{N}_\perp)$, respectively. For tuples, we must face an important distinction between computations of tuples and tuples of computations. That is, $\mathcal{P}^H(D) \times \mathcal{P}^H(E)$ and $\mathcal{P}^H(D \times E)$ simply are not isomorphic. We avoid the isomorphism requirement in the denotational semantics of the language by interpreting product types in a special way.

The paper is organized as follows: after the basics, in Section 3 we present the Recursive Space System introduced by Brattka. In Section 4, we introduce the language LRT_p and prove adequacy of the language. In Section 5, we present the translation between Brattka's relations and LRT_p programs. Finaly, Section 6 is devoted to Conclusions.

2 Basics

2.1 Continuous relations

In [2], Brattka extends Kleene's system of recursive partial functions on the natural numbers to other metric spaces, particularly to $\mathbb{R}$. Continuity is a necessary condition for effectiveness, and yet the fact that $\mathbb{R}$ is connected means there are no non-constant continuous functions, e.g., from $\mathbb{R}$ to the discrete space $\mathbb{N}$. So Brattka gives up functionality and retains a generalization of continuity to relations.

Definition 2.1 For binary relation R between sets X and Y and element $x \in X$, define $R(x) := \{y \in Y \mid xRy\}$. For $B \subseteq Y$, define $R^{-1}(B) := \{x \in X : R(x) \cap B \neq$

$\emptyset\}$, and let $\mathsf{dom}(R) = R^{-1}(Y) = \{x \in X \mid R(x) \neq \emptyset\}$. Thus we think of a relation as a partial function from X to non-empty subsets of Y. For this reason, we follow Brattka by usually writing f, g, etc., as names for binary relations. Binary relations from X to Y will be indicated by $f\colon X \leftrightarrow Y$.

If X and Y are topological spaces, then f is said to be *continuous* if and only if $f^{-1}(V)$ is open in X whenever V is open in Y. Also f is said to have *closed images* if $f(x)$ is closed in Y for every $x \in X$.

If X and Y are topological (or metric) spaces, then $X \times Y$ denotes the standard topological (metric) product.

Clearly, for a function h between spaces, the graph of h is a continuous relation if and only if h is continuous in the usual sense. In particular, the graphs of projection maps for cartesian products are continuous. If the codomain is T_1, graphs of functions also have closed images. Furthermore, any relation f is continuous if and only if $f(\overline{A}) \subseteq \overline{f(A)}$ for every $A \subseteq \mathsf{dom}(f)$. Note that A ranges only over subsets of $\mathsf{dom}(f)$, not over all subsets of X. This jibes with our interpretation of $f(x) = \emptyset$ as meaning that f is undefined at x.

Continuous relations are not closed under the usual relational composition. On the other hand, for continuous relations $f\colon X \leftrightarrow Y$ and $g\colon Y \leftrightarrow Z$, define $g \odot f$ by

$$(g \odot f)(x) := \begin{cases} \overline{(g \circ f)(x)}, & \text{if } f(x) \subseteq \mathsf{dom}(g); \\ \emptyset, & \text{otherwise.} \end{cases}$$

where $g \circ f$ is the usual relational composition. A simple exercise shows that continuous relations are closed under $\odot$.

By definition, $g \odot f$ has closed images. The graph of the identity function on a space Y satisfies $f = I \odot f$ if and only if f has closed images, and similarly for $g = g \odot I$. So $\odot$ defines composition for a category of topological (or metric) spaces in which the morphisms are continuous relations with closed images. This can be taken to be the ambient category for Brattka's recursive relations. Note that the graphs of projections on products of T_1 spaces are continuous with closed image. So the category can be given a monoidal structure.

2.2 The interval domain

The ideas discussed in this section are considered in more detail in [5].

The set $\mathcal{R}$ of non-empty connected compact subsets of the Euclidean real line forms a continuous dcpo when ordered by reverse inclusion:

$x \sqsubseteq y$ iff $x \supseteq y$.

We regard elements of $\mathcal{R}$ as "partial real numbers"; the $\sqsubseteq$-maximal intervals are singletons, corresponding to "total numbers". That is, the continuous map $x \mapsto \{x\}$ embeds $\mathbb{R}$ as maximal elements, making $\mathcal{R}$ into a domain model for $\mathbb{R}$. The dcpo $\mathcal{R}$, however, does not have a least element. By adding a least element, corresponding to the completely under-specified partial real number $\mathbb{R}$, we obtain a bounded complete

continuous domain $\mathcal{R}_\perp$.

For any $x \in \mathcal{R}_\perp$, we write

$\underline{x} = \inf x$ and $\overline{x} = \sup x$

so that $x = [\underline{x}, \overline{x}]$, and define

$$\kappa_x := \overline{x} - \underline{x}.$$

The upper bound of a subset $A \subseteq \mathcal{R}_\perp$ is $\bigcap A$ when this is not empty. Alternatively,

$$\bigsqcup A = \bigcap A = \left[\sup_{x\in A} \underline{x}, \inf_{x\in A} \overline{x}\right].$$

The way-below relation of $\mathcal{R}_\perp$ is given by

$x \ll y$ iff $\underline{x} < \underline{y}$ and $\overline{y} < \overline{x}$.

This amounts to y being a subset of the interior of x. Of course $\mathbb{R} = \perp \ll a$ for any compact interval a. The intervals with distinct rational end-points form a basis for $\mathcal{R}_\perp$.

For basis element a, consider the partial function $x \mapsto a \sqcup x$ defined when a and x are consistent. This join map has a total continuous extension:

$$\mathsf{join}_a x = \begin{cases} a \sqcup x, \ a \text{ and } x \text{ are consistent;} \\ \{\underline{a}\}, \quad \overline{x} < \underline{a}; \\ \{\overline{a}\}, \quad \overline{a} < \underline{x}. \end{cases}$$

Also, $\mathsf{join}_a \perp = a$. The map join_a can be regarded as a partial output:

Lemma 2.2 *For basis elements a and b,*

(i) $\mathsf{join}_a\mathsf{join}_b = \mathsf{join}_{a\sqcup b}$ *if $a \sqcup b$ exists;*

(ii) $\mathsf{join}_a\mathsf{join}_b = k_{\underline{a}}$ *if $\overline{b} < \underline{a}$;*

(iii) $\mathsf{join}_a\mathsf{join}_b = k_{\overline{a}}$ *if $\overline{a} < \underline{b}$;*

where k_z denotes the constant map $x \mapsto \{z\}$. Thus $\mathsf{join}_a \sqsubseteq \mathsf{join}_a\mathsf{join}_b$ always holds.

Each basis element a is also associated with a positive affine map $\mathsf{rrcons}_a : \mathbb{R} \to \mathbb{R}$ given by $x \mapsto \kappa_a x + \underline{a}$. Taking images, rrcons_a extends to a strict continuous map on $\mathcal{R}_\perp$. These maps form a left group action on $\mathcal{R}_\perp$. Because of this, we will think of the basis of $\mathcal{R}_\perp$ as itself forming a group, writing ab for concatenation, a^{-1} for inverse and I for the identity (that is, the interval $[0, 1]$, corresponding to the identity affine map).

Composites of joins and affine transformations interact as follows:

Lemma 2.3 *For basis elements a and b,*

(i) $\mathsf{rrcons}_a\mathsf{join}_b = \mathsf{join}_{ab}\mathsf{rrcons}_a$;

(ii) $\mathsf{rrcons}_a\mathsf{rrcons}_b = \mathsf{rrcons}_{ab}$;

Proof. Straightforward. □

The functions rrcons_a and join_a are said to be *strongly convergent*, meaning that they send maximal elements to maximal elements. In addition, the functions rrcons_a are all homeomorphisms ($\mathsf{rrcons}_a \mathsf{rrcons}_{a^{-1}} \mathsf{rrcons}_I = \mathsf{id}$), so they also send non-maximal elements to non-maximal elements.

2.3 The Hoare powerdomain

In [9,10,11], the first author shows that under certain reasonable assumptions, a suitable semantics for sequential, non-deterministic real number computation requires the Hoare powerdomain ($\mathcal{P}^H$). That is, starting from the assumption that *some* functorial powerdomain is needed to model non-determinism, general considerations about continuity show that the Hoare powerdomain is the only one that can be used. We refer the reader to the cited work for an explanation. In that work, however, the fact that $\mathcal{P}^H$ is actually a free construction is not used explicitly (though certain definitions in the semantics depend on it implicitly). In this section, we review the basic facts about $\mathcal{P}^H$ as the construction of free inflationary semi-lattices. The reader may consult [1] for a general theory of free domain constructions defined by inequalities.

A *semi-lattice* in the category of domains is simply a domain equipped with a continuous binary operation $\uplus : X \times X \to X$ that satisfies the usual semi-lattice laws. Such a semi-lattice is *inflationary* if $x \sqsubseteq x \uplus y$. It is not hard to see that idempotency is equivalent to $\uplus \circ \delta = \mathsf{id}_X$, and inflationarity to $\mathsf{id}_{X\times X} \sqsubseteq \delta \circ \uplus$, where $\delta\colon X \to X \times X$ is the diagonal map. Since these two conditions constitute a Galois connection between $\uplus$ and δ, if $\uplus$ exists it is unique.

The Hoare powerdomain is the free construction for inflationary semi-lattices [1]. If $f\colon X \to Y$ is a continuous map and $(Y, \uplus)$ is an inflationary semi-lattice, then there is a unique continuous map $\overline{f}\colon \mathcal{P}^H(X) \to Y$ that preserves $\uplus$ for which $f = \overline{f}\eta$, where η is the unit of the powerdomain monad. There is also a unique continuous map $\widehat{f}\colon \mathcal{P}^H(X) \to \mathcal{P}^H(Y)$ defined by $\widehat{f} := \mathcal{P}^H(f)$.

In domains, the binary formal join of an inflationary semi-lattice extends automatically to formal joins of non-empty sets: For $A \subseteq X$, take the closure of A under $\uplus$. This is automatically a directed set and hence has a supremum, which we denote by $\bigcup A$. If the generating domain has a least element, then so does $\mathcal{P}^H(X)$. So $\bigcup$ is defined for all subsets of $\mathcal{P}^H(X)$.

Concretely, elements of $\mathcal{P}^H(X)$ are non-empty Scott closed subsets of X, the unit sends $x \in X$ to the closure of $\{x\}$. Also, $\uplus$ is simply binary union, and $\bigcup$ is closure of union.

To mediate between products and powerdomains, we exploit the fact that the Hoare powerdomain is a monoidal monad with natural transformation $m\colon \mathcal{P}^H(X) \times \mathcal{P}^H(Y) \to \mathcal{P}^H(X \times Y)$ satisfying the usual coherence conditions. In concrete terms, $m(A, B) := A \times B$.

Thus the relevant structure of the Hoare powerdomain, for our purposes, is given

by the functor $\mathcal{P}^H$ itself, the unit $\eta : X \to \mathcal{P}^H(X)$, the formal union $\uplus\colon \mathcal{P}^H(X) \times \mathcal{P}^H(X) \to \mathcal{P}^H(X)$ and the transformation $m : \mathcal{P}^H(X) \times \mathcal{P}^H(Y) \to \mathcal{P}^H(X \times Y)$.

A continuous map f between inflationary semi-lattice domains X and Y *preserves* $\uplus$ if $f(x \uplus y) = f(x) \uplus f(y)$.

2.4 Hoare power domains of domain environments

If D and E are domain environments for spaces X and Y, we can ask when a continuous function $F\colon \mathcal{P}^H(D) \to \mathcal{P}^H(E)$ corresponds naturally to a continuous relation from X to Y.

Definition 2.4 Suppose X is a topological space, E_X is a domain model for X with embedding e_X and $d \in \mathcal{P}^H(E_X)$. Let

$$u_X(d) := \{x \in X \mid \nu_X(x) \sqsubseteq d\} = (\nu_X)^{-1}(\downarrow d)$$

$$\nu_X := \eta \circ e_X$$

As usual, we omit the subscripts when possible.

Furthermore, suppose that Y is a second space and E_Y is a corresponding domain model. Say that a relation f from X to Y is *captured by* $F\colon \mathcal{P}^H(E_X) \to \mathcal{P}^H(E_Y)$ (written $f \sim F$) if and only if for each $x \in \mathsf{dom}(f)$, $f(x) = u(F(\nu_X(x)))$. Say that f is *exactly captured by* F (written $f \simeq F$) if and only if $f \sim F$ and

$$\mathsf{dom}(f) = \{x \in X \mid u(F(\nu(x))) \neq \emptyset\}\,.$$

Say that $d \in \mathcal{P}^H(E_X)$ *is convergent* provided that $d = \bigsqcup \{\nu(x) \mid x \in u(d)\}$. Notice that by definition $\nu_X(x)$ is convergent. Also say that d is *divergent* provided that $\nu(x) \not\sqsubseteq d$ for all x. Also, say that continuous $F\colon \mathcal{P}^H(E_X) \to \mathcal{P}^H(E_Y)$ is *disciplined* provided that it preserves $\uplus$ and for each $x \in X$, $F(\nu_X(x))$ is either convergent or divergent.

For the remainder of this section, we assume that X, Y, E_X, E_Y, and embeddings e_X, e_Y are fixed.

Lemma 2.5 *For any closed $A \subseteq X$, $A = u(\bigsqcup_{x\in A} \nu(x))$.*

Proof. Let $x \in A$, $\nu(x)$ is a directed set with supremum x, hence $x \in u(\bigsqcup_{x\in A} \nu(x))$, for all $x \in A$. The converse is straightforward. □

Lemma 2.6 *For any $F\colon \mathcal{P}^H(E_X) \to \mathcal{P}^H(E_Y)$, there is a unique relation that is exactly captured by F. If F is disciplined, the exactly captured relation is a continuous relation with closed images.*

Proof. Proof omitted □

This leads to the following fundamental connection between continuous relations with closed images and disciplined functions.

Theorem 2.7 *Disciplined functions are closed under composition. Moreover, if F and G are disciplined, f and g are continuous with closed images, $f \sim F$ and $g \sim G$ and these "type check" in the obvious way, then $(g \odot f) \sim (G \circ F)$.*

Proof. Closure under composition for preservation of $\uplus$ follows from the general theory of power domains. And $G(F(\nu(x))) \supseteq \biguplus_{z \in m(G(F(\nu(x))))} \nu(z)$ is immediate from the definition. Conversely, $z \in u(G(\nu(y)))$ and $y \in u(F(\nu(x)))$ implies that $\nu(z) \sqsubseteq G(\nu(y)) \sqsubseteq G(F(\nu(x)))$, and by preservation of $\uplus$,

$$G(F(\nu(x))) = \bigsqcup_{y \in u(F(\nu(x)))} \bigsqcup_{z \in u(G(\nu(y)))} \nu(z)$$

So $G \circ F$ is disciplined.

The second statement is now routinely checked. □

Discipline is closely related to the operational concept of strong convergence discussed at length in [11]. There a closed *term* of ground type is strongly convergent if it denotes $\nu(x)$ for some x in the modeled space (although the definition is given operationally and adequacy of the operational semantics justifies the present characterization). A closed first-order term is strongly convergent if it preserves strong convergence of inputs. The reason an operational definition is given is that proof of strong convergence typically involves the operational semantics. The reader may consult [9], [10] or [11] for discussion and examples. For the purposes of the earlier work, preservation of strong convergence makes sense because the authors are interested primarily in defining *functions*. In the present setting, definability of *relations* is our main concern. So discipline can be seen as a generalization of strong convergence to a relational setting.

The ground spaces about which we are concerned have additional structure that allow a form of call-by-value, which we refer to as *call-by-partial-value*. In [13], Martin introduces the concept of a *measurement* on a continuous domain, D, as a Scott continuous function $M : D \to ([0, \infty], \geq)$. That is, M assigns a positive extended real to each element of D so that $M(\bigsqcup A) = \inf_{a \in A} M(a)$ for directed A. A measurement is also required to satisfy $M(a) = 0$ if and only if $a \in \max D$.

The domains $\mathcal{R}_\perp$, $\mathbb{T}_\perp$ and $\mathbb{N}_\perp$ clearly can be equipped with measurements: in $\mathcal{R}_\perp$, $M(a) = \kappa_a$; in $\mathbb{T}_\perp$, $M(\mathsf{true}) = M(\mathsf{false}) = 0$; in $\mathbb{N}_\perp$, $M(n) = 0$; and in all of these $M(\perp) = \infty$. In a finite product of domains with measurements, a measurement on a tuple is obtained by taking the minimum measurement of the components. For any positive p, any domain D with least element and with measurement M, the function $\mathsf{pv}_p : D \to D$ given by $\mathsf{pv}_p(a) = a$ if $M(a) < p$ and $\mathsf{pv}_p(a) = \perp$ otherwise is continuous. Its extension to $\mathcal{P}^H D$ satisfies $\widehat{\mathsf{pv}_p}(d) \sqsubseteq d$ and allows us to isolate the maximal part of an element $d \in \mathcal{P}^H(D)$ that can be written $\biguplus_{a \in A} \eta(a)$ where all elements of A have "small" measurement. As p decreases, pv_p decreases as well. Importantly, each $\widehat{\mathsf{pv}_p}(d)$ is the identity map when restricted to convergent d, and $\bigsqcup_p \widehat{\mathsf{pv}_p}$ is the identity on $\mathcal{R}_\perp$.

3 The Recursive Space System $(\mathbb{R}, \mathbb{N}, \mathbb{T}, \rho_{\mathbb{R}})$

Using continuous relations with closed images as his ambient category, Brattka defines a combinatorial notion of *recursive relation* over a fixed (finite) collection of (complete, separable) metric spaces $\{X_0, \ldots, X_n\}$ that includes $\mathbb{N}$. In this section, we provide a brief overview of Brattka's approach where $\mathbb{N}$, $\mathbb{T}$ and $\mathbb{R}$ are the given spaces, $\mathbb{N}$ and $\mathbb{T}$ have the discrete topology, and $\mathbb{R}$ has the Euclidean topology.

As a guide to intuition, think of "effectivity" for a relation $f\colon X \leftrightarrow Y$ as meaning that for each $a \in X$, the set $f(a)$ is recursively enumerable uniformly in a. This description is far too vague and cannot actually be the whole story when Y is uncountable, but it is of some help in justifying Brattka's approach and, once separability is taken into account, can be justified formally. In the next few paragraphs, we discuss Brattka's combinators. The idea is to capture methods of constructing "effective" relations from "effective" relations.

To make the exposition of these ideas a bit clearer, and to come closer to the type system assumed in the PCF family of languages, we modify Brattka's original definitions to include the two element discrete space $\mathbb{T} = \{\mathsf{true}, \mathsf{false}\}$. This extension is easily seen to be conservative in that every relation definable in Brattka's original setting is definable here, and for X and Y being products that do not involve $\mathbb{T}$, every relation $f\colon X \leftrightarrow Y$ definable here is definable in Brattka's original setting.

Definition 3.1 In addition to $\odot$-composition, define the following combinators on binary relations:

Juxtaposition Given $f\colon X \leftrightarrow Y$ and $g\colon X \leftrightarrow Z$, define $(f, g)\colon X \leftrightarrow Y \times Z$ by

$$(f, g)(a) = \{(b, c) \in Y \times Z \colon b \in f(a) \text{ and } c \in g(a)\}$$

This generalizes the pairing of functions to the relational setting.

Iteration Given $f\colon X \leftrightarrow X$, define $f^*\colon X \times \mathbb{N} \leftrightarrow X$ so that $f^*(x, n)$ is the n-fold composition of f. That is,

$$f^*(x, 0) := \{x\}$$
$$f^*(x, n+1) := f \odot f^*(x, n)$$

Minimization Given a relation $f\colon X \times \mathbb{N} \leftrightarrow Y \times \mathbb{T}$ we think of a value $(y, b) \in Y \times \mathbb{T}$ as a "possible result" y together with a "status flag" b. Beginning with $n = 0$, minimization dovetails the enumerations of $f(x, 0)$, $f(x, 1)$, $\ldots$, $f(x, n)$ and collects the pairs (y, i) where $(y, \mathsf{true}) \in f(x, i)$, incrementing n (that is the range over which dovetailing occurs) whenever a pair (y, false) is produced by $f(x, n)$. So define $\min f\colon X \leftrightarrow \mathbb{N} \times Y$ by

$$\min f(x) := \{(n, y) \colon (y, \mathsf{true}) \in f(x, n) \text{ and } \forall m < n \exists y' \in Y.(y', \mathsf{false}) \in f(x, m)\}$$

The reader may wish to show that Kleene's minimization combinator is definable using this min (together with composition, juxtaposition and the standard basic primitive recursive functions on $\mathbb{N}$).

Limitation In a (complete) metric space, a sequence $\{B_n\}_n$ of subsets is *strongly Cauchy* provided that for each i and each $b_i \in B_i$, b_i is the i-th element of a strong Cauchy sequence $\{b_n\}_n$ for which $b_n \in B_n$ for each n. In other words, all elements of B_i participate in *some* strong Cauchy sequence obtained from the sets B_n. For such a sequence of subsets, define $\lim_{i\to\infty} B_i$ to consist of all limits (in the usual sense) of all strong Cauchy sequences $\langle b_n\rangle_n$ such that $b_n \in B_n$.

For a relation $C\colon X \times N \leftrightarrow Y$, the limitation combinator is defined by

$$\lim[C](a) := \begin{cases} \lim_{n\to\infty} C(a,n), & C(a,n)_n \text{ is strongly Cauchy;} \\ \emptyset, & \text{otherwise.} \end{cases}$$

All of the above combinators (along with $\odot$-composition) preserve continuity, the property of having closed images and the property of being the graph of a function ([2]: Lemma 11).

Definition 3.2 Complete metric spaces $(X_0, \ldots, X_{n-1}, \mathbb{N}, \mathbb{T})$ together with a collection ρ of continuous relations between products formed from the given spaces is called a *recursive space system* provided that ρ includes the graphs of projections and is closed under the combinators of $\odot$-composition, juxtaposition, iteration, minimization and limitation.

Consider the collection β_K consisting of the following relations (all graphs of familiar basic primitive recursive functions):

- $\mathsf{succ}\colon \mathbb{N} \leftrightarrow \mathbb{N}$ – graph of the successor function.
- $0_{\mathbb{N}}\colon \mathbb{N} \leftrightarrow \mathbb{N}$ – graph of the constant zero function.
- $\mathsf{iszero}\colon \mathbb{N} \leftrightarrow \mathbb{T}$ – graph of the zero test, $n \mapsto \mathsf{true}$ iff $n = 0$.
- $\mathsf{nt}\colon \mathbb{T} \leftrightarrow \mathbb{N}$ – graph of the right inverse of iszero where $\mathsf{true} \mapsto 0$, $\mathsf{false} \mapsto 1$.

Let ρ'_K be the least set of relations between products of $\mathbb{N}$ and $\mathbb{T}$ that includes β_K and the graphs of projections, and is closed under Brattka's combinators except for lim. It is not difficult to see that for relations between discrete spaces $\odot$ is simply the usual relational composition. So every relation $f\colon \mathbb{N}^k \leftrightarrow \mathbb{N}$ in ρ'_K is the graph of a Kleene recursive (partial) function. Likewise, the graph of every such function appears in ρ'_K. In this sense, Brattka's recursive relation spaces generalize Kleene's combinatorial approach to recursive functions on $\mathbb{N}$.

In this paper, we are primarily concerned with the recursive space system obtained by including $\mathbb{R}$ with its field structure. Since there are no continuous functions from $\mathbb{R}$ to $\mathbb{N}$ or to $\mathbb{T}$, a non-deterministic test is included. That is, let $\beta_{\mathbb{R}}$ consist of β_K plus the following relations:

- $0_{\mathbb{R}}\colon \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of constant zero.
- $1_{\mathbb{R}}\colon \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of constant one.
- $+\colon \mathbb{R} \times \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of the addition.
- $*\colon \mathbb{R} \times \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of multiplication.

- $\mathsf{neg}\colon \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of negation.
- $\mathsf{inv}\colon \mathbb{R} \leftrightarrow \mathbb{R}$ – the graph of reciprocation, $x \in \mathsf{inv}(y)$ iff $xy = 1$.
- $\mathsf{ord}\colon \mathbb{R} \leftrightarrow \mathbb{T}$ – $\mathsf{true} \in \mathsf{ord}(x)$ iff $x < 0$, $\mathsf{false} \in \mathsf{ord}(x)$ iff $0 < x + 1$.

Let $\rho_{\mathbb{R}}$ be the least recursive space system including $\beta_{\mathbb{R}}$.

For $X = \mathbb{R}, \mathbb{N}, \mathbb{T}$, we assume given a partial continuous function δ_X from Baire space $\mathbb{B}$ to X that is surjective. For X being a finite product of these spaces, δ_X is also definable inductively by pairing the functions defined on the three basic spaces. We omit the details of this, referring the reader to [2].

A partial computable function $F\colon \mathbb{B} \to \mathbb{B}$ determines a partial computable function $\hat{F}\colon \mathbb{B} \times \mathbb{N} \to \mathbb{B}$ by $\hat{F}(p,n)(k) = F(p)\langle n, k\rangle$ where $\langle\rangle\colon \mathbb{N}^2 \to \mathbb{N}$ is Cantor's (computable) bijective pairing function. The function $\hat{F}$ has the property that $\hat{F}(p,n)$ is defined if and only if $F(p)$ is defined.

Brattka defines a relation $f\colon X \leftrightarrow Y$ to be *densely computable* if there exists a partial computable $F\colon \mathbb{B} \to \mathbb{B}$ so that for all $p \in \mathbb{B}$ if $\delta_X(p) \in \mathsf{dom}(f)$, then $f(\delta_X(p)) = \overline{\left\{\delta_Y(\hat{F}(p,n)) \mid n \in \mathbb{N}\right\}}$. Furthermore, he defines f to be *strongly densely computable* if, in addition, $p \in \mathsf{dom}(F)$ implies $\delta_X(p) \in \mathsf{dom}(f)$.

The main result of Brattka's paper for our purposes is the following characterization of the relations in $\rho_{\mathbb{R}}$ in terms of Type 2 Theory of Effectivity [18].

Theorem 3.3 (Corollary 33 and Theorem 34 in [2]) *If $f \in \rho_{\mathbb{R}}$, then f is densely computable. If f is strongly densely computable, then $f \in \rho_{\mathbb{R}}$.*

4 LRT_p

The language LRT is a modification of RealPCF considered by Escardó [4] for real number computation. In LRT, parallel conditional `pif` is replaced by a nondeterministic test $\mathtt{rtest}_{l,r}$. In this section, we describe a variant of LRT suitable for comparison to the recursive space system described above. The language LRT_p differs from LRT in three ways: products of ground types are made explicit, the type I for the compact interval $[0,1]$ is eliminated in favor of a type corresponding to $\mathbb{R}$, and a `let` construct is introduced that provides for call-by-partial-value semantics.

4.1 Syntax

Syntactically, the type system for LRT_p is given by

$$\gamma := \mathtt{nat} \mid \mathtt{bool} \mid \mathtt{real}$$
$$\beta := \gamma \mid \gamma \times \beta$$
$$\tau := \beta \mid (\tau \to \tau)$$

Types in the first clause are *ground types*; in the second clause, *basic types*; and in the third clause, *general types*. As usual, we associate $\to$ right to left, and omit parentheses when we can.

The raw syntax of the language is given by

$$
\begin{aligned}
& x \in \mathit{Variable},\\
P ::= {} & x \mid \mathtt{n} \mid \mathtt{true} \mid \mathtt{false} \mid (+\mathtt{1})(P) \mid (-\mathtt{1})(P) \mid\\
& (=\mathtt{0})(P) \mid \mathtt{if}\, P\, \mathtt{then}\, P\, \mathtt{else}\, P \mid \mathtt{rrcons}_a(P) \mid\\
& \mathtt{join}_a(P) \mid \mathtt{rtest}_{l,r}(P) \mid \lambda x^\tau.P \mid PP \mid \mathtt{Y}P \mid\\
& \mathtt{let}\ x = P\ \mathtt{in}\ P \mid\\
& \mathtt{pr}_i P \mid \langle P_0, \ldots, P_n \rangle
\end{aligned}
$$

where $(+\mathtt{1})(P), (-\mathtt{1})(P)$ and $(=\mathtt{0})(P)$ amount for successor, predecessor and equality for zero respectively; the subscripts of the constructs `rrcons` and `join` are proper rational intervals and those of `rtest` are rational numbers. In the `let` construct, the first term P must be of basic type.

In addition, we allow ourselves the syntactic sugar of writing

$$\mathtt{let}\ \langle x_0, \ldots, x_n \rangle = P_1\ \mathtt{in}\ P_2$$

where the notation $\langle x_0, \ldots, x_n \rangle$ stands for a variable of the appropriate product type and where free occurrences of x_i in P_2 abbreviate $\mathtt{pr}_i\langle x_0, \ldots, x_n \rangle$.

Terms can be associated with types in the familiar style by proof rules and judgements, but in the interest of brevity, we trust the reader to fill in the details.

4.2 Denotational Semantics

We define denotational semantics $[\![-]\!]^p$ for LRT_p subject to a positive real number parameter p in such a way that $[\![M]\!]^p$ is semi-continuous in p and $\bigsqcup_p [\![M]\!]^p$ corresponds to call-by-name interpretation. The idea is to employ $\mathtt{pv}_p$ (see page 8) in the interpretation of the `let` construct to ignore differences due to "badly" divergent behavior. As p increases, the semantics ignores less. We use $B[\![]\!]$ to denote basic types, which includes ground types and product types.

The ground types `bool`, `nat` and `real` are interpreted, first, as the domains of booleans ($\mathbb{T}_\perp$), natural numbers ($\mathbb{N}_\perp$) and compact intervals ($\mathcal{R}_\perp$), respectively. That is,

$$B[\![\mathtt{bool}]\!] := \mathbb{T}_\perp, \qquad B[\![\mathtt{nat}]\!] := \mathbb{N}_\perp, \qquad B[\![\mathtt{real}]\!] := \mathcal{R}_\perp.$$

Finite products are interpreted the usual way: $B[\![\gamma \times \beta]\!] := B[\![\gamma]\!] \times B[\![\beta]\!]$. Basic types are interpreted as Hoare powerdomains of finite products:

$$[\![\beta]\!] := \mathcal{P}^H(B[\![\beta]\!]).$$

Function types are interpreted as function spaces in the category of dcpos:

$$[\![\sigma \to \tau]\!] := [\![\sigma]\!] \to [\![\tau]\!].$$

These definitions reflect a call-by-name semantics in which product types are interpreted as consisting of computations of tuples, rather than tuples of computations.

The interpretation of constants is given as follows:

$$[\![\mathtt{true}]\!]^p = \eta(\mathsf{true}), \quad [\![\mathtt{false}]\!]^p = \eta(\mathsf{false}), \quad [\![\mathtt{n}]\!]^p = \eta(\mathsf{n}),$$

$$[\![(+1)]\!]^p = \widehat{(+1)}, \quad [\![(-1)]\!]^p = \widehat{(-1)}, \quad [\![(=0)]\!]^p = \widehat{(0=)},$$
$$[\![\mathtt{join}_a]\!]^p = \widehat{\mathsf{join}_a}, \quad [\![\mathtt{rrcons}_a]\!]^p = \widehat{\mathsf{rrcons}_a},$$
$$[\![\mathtt{rtest}_{l,r}]\!]^p = \overline{\mathsf{rtest}}_{l,r}, \qquad [\![\mathtt{Y}]\!]^p(F) = \bigsqcup_{n\geq 0} F^n(\bot),$$

$$[\![\mathtt{if}]\!]^p(B,X,Y) \begin{cases} X, & \text{if } B = \eta(\mathsf{true}), \\ Y, & \text{if } B = \eta(\mathsf{false}), \\ X \uplus Y, & \text{if } B = \eta(\mathsf{true}) \uplus \eta(\mathsf{false}), \\ \bot, & \text{if } B = \bot, \end{cases}$$

with syntactic sugar,

$$[\![\mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ P]\!]^p_\rho := [\![\mathtt{if}]\!]^p([\![M]\!]^p_\rho, [\![N]\!]^p_\rho, [\![P]\!]^p_\rho)$$

$$[\![\mathtt{pr}_i]\!]^p = \widehat{\pi_i},$$

where π_i is the usual projection map. Tuples are interpreted by

$$[\![\langle X_1, \ldots, X_n\rangle]\!]^p_\rho := m([\![X_1]\!]^p_\rho, \ldots, [\![X_n]\!]^p_\rho)$$

Note that so far, none of these definitions depend on the parameter p. The `let` construct enforces what we refer to as *call-by-partial-value*.

$$[\![\mathtt{let}\ x = M\ \mathtt{in}\ N]\!]^p_\rho := [\![N]\!]^p \rho(x/\widehat{\mathsf{pv}_p}([\![M]\!]^p_\rho))$$

Here the symbols $\eta, \widehat{\ }, \overline{\ }$ and m derive from the Hoare powerdomain monad: η is the unit, $\widehat{f} := \mathcal{P}^H(f)$, $\overline{f}$ denotes the transpose of $f\colon X \to \mathcal{P}^H(Y)$, m is the natural transformation $\mathcal{P}^H(X_0) \times \ldots \mathcal{P}^H(X_n) \to \mathcal{P}^H(X_0 \times \ldots \times X_n)$. The functions $(+1)$, (-1), $(=0)$ are the standard interpretations in the Scott model of PCF [15], the functions $\mathsf{join}_a, \mathsf{rrcons}_a$ are defined in section 2.2, and the function $\mathsf{rtest}_{l,r}$ is defined by:

$$\mathsf{rtest}_{l,r}(x) \begin{cases} \eta(\mathsf{true}) \uplus \eta(\mathsf{false}), & \text{if } l < \underline{x} < \overline{x} < r; \\ \eta(\mathsf{true}), & \text{if } \overline{x} \leq r; \\ \eta(\mathsf{false}), & \text{if } \underline{x} \geq l; \\ \bot, & \text{otherwise.} \end{cases}$$

Lemma 4.1 *The constants* $(+1)$, (-1), $(=0)$, $\mathtt{rtest}_{l,r}$, $\mathtt{join}_a$, $\mathtt{rrcons}_a$ *and* $\mathtt{pr}_i$ *denote disciplined functions.*

Proof. Proof omitted □

Lemma 4.2 *The semantics $[\![-]\!]^p$ is semi-continuous in p: for bounded $A \subseteq \mathbb{R}^+$,*

$$\bigsqcup_{p \in A} [\![M]\!]^p = [\![M]\!]^{\sup A}.$$

Moreover, define $[\![]\!]^\infty$ exactly as $[\![]\!]^p$ for all cases except

$$[\![\texttt{let } x = M \texttt{ in } N]\!]^\infty_\rho := [\![N]\!]^\infty \rho(x/[\![M]\!]^\infty_\rho))$$

Then $[\![M]\!]^\infty = \bigsqcup_p [\![M]\!]^p$.

4.3 Operational Semantics

We now develop single-step operational semantics, also parametric in p, so that the "p-th" operational interpretation is complete for $[\![-]\!]^p$. We do not need an operational semantics corresponding to $[\![-]\!]^\infty$.

Definition 4.3 For each basic type β, we define a subset of the closed terms to be *output terms*, and for each output term M we define it's *output* $o(M)$ to be a value in $B[\![\beta]\!]$.

For `real`, a term of the form $\texttt{join}_a M$ is an output term, and $o(\texttt{join}_a M) := a$. For `nat`, a term of the form `n` is an output term, and $o(\texttt{n}) = n$. For `bool`, a term of the form `true` or `false` is an output term, and o is defined obviously. For $\gamma \times \beta$, a term of the form $\langle M, N\rangle$ is an output term provided M and N are output terms, and $o(\langle M, N\rangle) = \langle o(M), o(N)\rangle$.

Lemma 4.4 *For an output term M and $p > 0$,*

$$\eta(o(M)) \sqsubseteq [\![M]\!]^p \sqsubseteq \bigsqcup \{\nu(x) \mid o(M) \sqsubseteq \nu(x)\}.$$

Proof. For an output term of type `real`, let $x \in \eta(o(\texttt{join}_a(M')))$, hence $x \in \eta(a) \sqsubseteq \widehat{\texttt{join}}_a [\![M']\!]^p = [\![\texttt{join}_a(M')]\!]^p$. The second inequality is derived from the fact that $\nu(x) = \eta \circ e(x)$. The proof for the other output terms is similar. □

We define $\rightarrow_p$ to be the least relation that includes single-step reduction rules for PCF [15] and is closed under rules for the type `real` and for `let` as follows.

(1)	$\texttt{rrcons}_a(\texttt{rrcons}_b M) \to_p \texttt{rrcons}_{ab} M$	
(2)	$\texttt{join}_a\texttt{join}_b M \to_p \texttt{join}_{a \sqcup b} M$	if $\overline{b} > \underline{a}$ or $\overline{a} > \underline{b}$
(3)	$\texttt{join}_a\texttt{join}_b M \to_p \texttt{rrcons}_a Y(\texttt{rrcons}_L\texttt{join}_I)$	if $\overline{b} \leq \underline{a}$
(4)	$\texttt{join}_a\texttt{join}_b M \to_p \texttt{rrcons}_a Y(\texttt{rrcons}_R\texttt{join}_I)$	if $\overline{a} \leq \underline{b}$
(5)	$\texttt{rrcons}_a(\texttt{join}_b M) \to_p \texttt{join}_{ab}(\texttt{rrcons}_a M)$	
(6)	$\texttt{rtest}_{l,r}\texttt{join}_a M \to_p \texttt{true}$	$\overline{a} < r$
(7)	$\texttt{rtest}_{l,r}\texttt{join}_a M \to_p \texttt{false}$	$l < \underline{a}$
(8)	$\texttt{if true then } M \texttt{ else } M' \to_p M$	
(9)	$\texttt{if false then } M \texttt{ else } M' \to_p M'$	
(10)	$\texttt{pr}_i\langle M_0, \ldots, M_n\rangle \to_p M_i$	
(11)	$\texttt{let } x = M \texttt{ in } N \to_p [M/x]N$	M is an output term and $\mu(o(M)) < p$
(12)	$\dfrac{N \to_p N'}{MN \to_p MN'}$	if M is $\texttt{join}_a$, $\texttt{rrcons}_a$, $\texttt{rtest}_{l,r}$, if, $\texttt{pr}_i$, let and none of the above hold.

Definition 4.5 We define the operational meaning of a closed term M of basic type β in i steps of computation, written $[M]^p_i \in [\![\beta]\!]$.

For a closed term of basic type β, define $[M]_i$ as follows:

$$[M]^p_i = \bigsqcup \{\eta(o(M')) \mid \exists M' \exists k \leq i, M' \text{ is an output term and } M \xrightarrow{k}_p M' \},$$

where an empty formal join is $\perp$, and $\xrightarrow{k}_p$ denotes the k-fold composition of the relation $\to_p$.

Finally,

$$[M]^p = \bigsqcup_i [M]^p_i.$$

which is justified by the obvious fact that $[M]^p_i \sqsubseteq [M]^p_{i+1}$.

Note that implicit in this definition is the fact that the operational rules are such that $M \xrightarrow{k}_p M'$ can only hold for finitely many output terms M'. This can be established easily by induction on the operational rules.

The operational interpretation of closed terms is adequate with respect to the denotational semantics.

Lemma 4.6 $[\![M]\!]^p = \bigcup \{[\![N]\!]^p \mid M \to_p N\}$ *(this is a finite union).*

Proof. Most of the details of the proof duplicate the analogous result in [9], except for let.

For $\texttt{let}\ x = M\ \texttt{in}\ N \to_p N[x/M]$, the restriction on M implies that $\widehat{\mathsf{pv}_p}([\![M]\!]^p) = [\![M]\!]^p$. So the remaining argument for this case reduces to the argument for β-reduction. □

Lemma 4.7 *For all closed terms M of ground type,*

$$[M]^p \sqsubseteq [\![M]\!]^p.$$

Proof. Proof omitted □

Definition 4.8 A closed term is said to be p-computable as follows:

(i) A closed term M of basic type is p-computable whenever $[\![M]\!]^p \sqsubseteq [M]^p$,

(ii) A closed term $M : \sigma \to \tau$ is p-computable whenever $MQ : \tau$ is p-computable for every closed p-computable term Q of type σ,

An open term $M : \sigma$ with free variables $x_1, \ldots, x_n$ of type $\sigma_1, \ldots, \sigma_n$ is p-computable whenever $[N_1/x_1] \cdots [N_n/x_n]M$ is p-computable for every family $N_i : \sigma_i$ of closed p-computable terms.

Lemma 4.9 *Every term of LRT_p is p-computable.*

Proof. Proof omitted □

Theorem 4.10 *$[M]^p = [\![M]\!]^p$, for all closed LRT_p terms M and all positive reals p.*

Proof. Lemma 4.7 and Lemma 4.9. □

5 Translations

The main aim of this section is to establish that all recursive relations in Brattka's system $(\mathbb{R}, \mathbb{N}, \mathbb{T}, \rho_{\mathbb{R}})$ are programmable in LRT_p and that all first-order closed terms of LRT_p correspond to recursive relations. A recursive relation is (forgetfully) a function from a set X to a powerset $\mathcal{P}(Y)$, but generally first-order terms of LRT_p do not correspond to relations between the underlying domains. So the familiar technique of establishing a logical relation will not work here unmodified. For a first-order function $f \in [\![\beta_1 \to \beta_2]\!]$, the composite $f \circ \eta$ is a map from the underlying domain $B[\![\beta_1]\!]$ to $\mathcal{P}^H(B[\![\beta_2]\!])$. In other words, we can think of f as an "implementation of" a certain kind of relation from $B[\![\beta_1]\!]$ to $B[\![\beta_2]\!]$. In order to distinguish the domain-theoretical interpretation of basic types from the set-theoretic interpretation in LRT_p, we use $R[\![]\!]$ for the latter.

Definition 5.1 For basic LRT_p types, we define a set-theoretic interpretation as follows:

- $R[\![\texttt{nat}]\!] = \mathbb{N}$,
- $R[\![\texttt{bool}]\!] = \mathbb{T}$,
- $R[\![\texttt{real}]\!] = \mathbb{R}$,

- $R[\![\gamma \times \beta]\!] = R[\![\gamma]\!] \times R[\![\beta]\!]$,

Theorem 2.7 establishes that composition in LRT_p corresponds to $\odot$-composition. That is, if P and Q are closed terms of type $\beta_1 \to \beta_2$ and $\beta_2 \to \beta_3$, both are disciplined and $f \sim [\![F]\!]$ and $g \sim [\![G]\!]$, then $g \odot f \sim [\![\lambda x.\texttt{let } y = F(x) \texttt{ in } G(y)]\!]^p$. We extend this to the other recursion operators.

Definition 5.2 Define the following closed terms of LRT_p:

$$\begin{aligned}
\texttt{Jx}[F,G](x) &:= \langle F(x), G(x)\rangle \\
\texttt{Iter}[F](z) &:= \texttt{let } \langle x,n\rangle = z \texttt{ in if } (=\texttt{0})n \texttt{ then } x \texttt{ else } F(\texttt{Iter}[F](\langle x,(-\texttt{1})n\rangle)) \\
\texttt{Min}[F](x) &:= \texttt{Min}'[F](x,0) \\
\texttt{Lim}[F](x) &:= \texttt{aux_lim } F(x,0) \texttt{ id}
\end{aligned}$$

where

$$\begin{aligned}
&\texttt{Min}'[F](x,n) := \\
&\quad \texttt{let } \langle y,b\rangle = F(x,n) \texttt{ in if } b \texttt{ then } \langle y,n\rangle \texttt{ else } \texttt{Min}'[F](\langle x,(+\texttt{1})n\rangle)
\end{aligned}$$

$$\begin{aligned}
&\texttt{aux_lim } F\ (x,n)\ G := \\
&\quad \texttt{let } r = G(F(x,n)) \texttt{ in} \\
&\quad\ \texttt{if } \texttt{rtest}_{-1,1}(r) \\
&\quad\ \texttt{then if } \texttt{rtest}_{-1,-1/2}(r) \\
&\quad\quad\ \texttt{then } \texttt{join}_{[-2,-1]}(\texttt{aux_lim } F\ (x,n)\ (\texttt{rrcons}_{[1,2]} \circ\ G)) \\
&\quad\quad\ \texttt{else } \texttt{aux_lim}'\ F\ (x,n)\ G \\
&\quad\ \texttt{else if } \texttt{rtest}_{1/2,1}(r) \\
&\quad\quad\ \texttt{then } \texttt{aux_lim}'\ F\ (x,n)\ G \\
&\quad\quad\ \texttt{else } \texttt{join}_{[1,2]}(\texttt{aux_lim } F\ (x,n)\ (\texttt{rrcons}_{[-2,-1]} \circ\ G))
\end{aligned}$$

$$\begin{aligned}
&\texttt{aux_lim}'\ F\ (x,n)\ G := \\
&\quad \texttt{let } r = G(F(x,n)) \texttt{ in} \\
&\quad\ \texttt{if } \texttt{rtest}_{-5/16,5/16}(r) \\
&\quad\ \texttt{then if } \texttt{rtest}_{-5/16,-4/16}(r) \\
&\quad\quad\ \texttt{then } \texttt{cons}_L(\texttt{aux_lim } F\ (x,(+\texttt{1})(n))\ (\texttt{tail}_L \circ\ G)) \\
&\quad\quad\ \texttt{else } \texttt{cons}_C(\texttt{aux_lim } F\ (x,(+\texttt{1})(n))\ (\texttt{tail}_C \circ\ G)) \\
&\quad\ \texttt{else if } \texttt{rtest}_{4/16,5/16}(r) \\
&\quad\quad\ \texttt{then } \texttt{cons}_C(\texttt{aux_lim } F\ (x,(+\texttt{1})(n))\ (\texttt{tail}_C \circ\ G)) \\
&\quad\quad\ \texttt{else } \texttt{cons}_R(\texttt{aux_lim } F\ (x,(+\texttt{1})(n))\ (\texttt{tail}_R \circ\ G))
\end{aligned}$$

$$\begin{aligned}
\texttt{cons}_a &:= \texttt{join}_A\ \texttt{rrcons}_a \\
\texttt{tail}_a &:= \texttt{join}_A\ \texttt{rrcons}_{a^{-1}} \\
A &:= [-1,1] \\
L &:= [-1/2,0] \\
C &:= [-1/4,1/4] \\
R &:= [0,1/2]
\end{aligned}$$

In these definitions we understand Jx, for example, to be a third-order term, where F and G are arguments. We set them apart for readability using square

brackets.

Lemma 5.3 *For any $p < 1/4$, in the semantics $[\![-]\!]^p$, the terms* `Jx`, `Iter`, `Min` *and* `Lim` *preserve discipline. Moreover, if $f \sim F$ and $g \sim G$ and these "type check" in the obvious way, then*

(i) $(f, g) \sim [\![\texttt{Jx}]\!]^p(F, G)$;

(ii) $f^* \sim [\![\texttt{Iter}]\!]^p F$;

(iii) $\min f \sim [\![\texttt{Min}]\!]^p F$;

(iv) $\lim f \sim [\![\texttt{Lim}]\!]^p F$.

Proof. The proof is straightforward, except for `Lim`. For that, the assumption $p < 1/4$ is needed to ensure that the set of limits of strong Cauchy sequences in which r_i appears as the i-th term is bounded within a distance of $2^{-(i+2)}$ from r_i. In fact, this is the only point at when the assumption that p is small is required. □

Theorem 5.4 *For every relation in $f\colon R[\![\beta_1]\!] \leftrightarrow R[\![\beta_2]\!]$ belonging to $\rho_{\mathbb{R}}$, there is a closed first-order LRT_p term $P_f\colon \beta_1 \to \beta_2$ and $p > 0$ for which $[\![P_f]\!]^p$ is disciplined and $f \sim [\![P_f]\!]^p$. Furthermore, for every closed first-order LRT_p term $P\colon \beta_1 \to \beta_2$ and $p > 0$ such that $[\![P]\!]^p$ is disciplined, there is a relation $f_P\colon R[\![\beta_1]\!] \leftrightarrow R[\![\beta_2]\!]$ belonging to $\rho_{\mathbb{R}}$ for which $f_P \simeq [\![P]\!]^p$.*

Proof. [Sketch] The basic relations in $\rho_{\mathbb{R}}$ are programmable in LRT_p because the language supports general recursion on `nat`, $\mathsf{ord} \sim [\![\texttt{rtest}_{0,1}]\!]$ and the arithmetic functions are easily programmed in LRT_p. The proof that each of these is disciplined is implicit in their proofs of correctness. The first claim holds for non-basic relations by induction using Lemma 5.3.

For the second claim, one can devise a Gödel numbering system $\ulcorner\ \urcorner$ for LRT_p terms along with the standard primitive recursive apparatus for substitution, type checking and so on. □

6 Conclusion

In this paper, we consider the relational semantics of non-deterministic, sequential exact real number computation in the style of RealPCF and LRT. The semantic interpretation, both denotational and operational, is parameterized by a positive real number that allows for control of a limited form of call-by-value, which we refer to as call-by-partial-value. Although our original goal was to characterize the expressive power of LRT with respect to Brattka's recursive relations, we have found that in a non-deterministic setting, the call-by-name semantics of LRT is inadequate. Nevertheless, for exact real number computation, a *value*, i.e., a real number, is only understood to be a limit of finitely computable partial values. Operationally, no term corresponds to a total real number value. Even if we were to include countably many explicit names (say for the rationals), uncountably many values will be missed, so call-by-value semantics is out of the question. Nevertheless, by defining a parameterized family of interpretations in which partial reals of a specified

precision are treated as values, a middle ground is struck between call-by-value and call-by-name.

References

[1] Abramsky, S., and A. Jung, *Domain Theory*, in: S. Abramsky and D. Gabbay and T. S. E. Maibaum, eds., Handbook of Logic in Computer Science **3** (Oxford University Press, 1994) 1–168.

[2] Brattka, V., *Recursive characterization of computable real-valued functions and relations*, Theoretical Computer Science **162 (1)** (1996) 45–77.

[3] Di Gianantonio, P., "A Functional Approach to Computability on Real Numbers ", PhD thesis, (Udine, 1993).

[4] Escardó, M. H., *PCF Extended with Real Numbers*, Theoretical Computer Science **162 (1)** (1996) 79–115.

[5] Escardó, M. H., "PCF extended with real numbers: A domain-theoretic approach to higher-order exact real number computation", PhD thesis at Imperial College of the University of London, 1997.

[6] Escardó, M. H., M. Hofmann., and Th. Streicher, *On the non-sequential nature of the interval-domain model of exact real-number computation*, Mathematical Structures in Computer Science **14 (6)** (2004) 803–814.

[7] Farjudian, A., "Sequentiality in Real Number Computation", PhD thesis at the University of Birmingham, 2004.

[8] Levy, P.B., *Call-by-push-value: A subsuming paradigm*, in: Typed Lambda Calculus and Aplications, (1999) 228–242

[9] Marcial-Romero, J. R., "Semantics of a sequential language for exact real-number computation", PhD thesis at the University of Birmingham, 2004).

[10] Marcial-Romero, J. R. and M. Escardó, *Semantics of a sequential language for exact real-number computation*, in: Harald Ganzinger, editor., Proceedings of the 19^{th} Anual IEEE Symposium on Logic in Computer Science, (IEEE Computer Society press, July 2004) 426–435.

[11] Marcial-Romero, J. R. and M. Escardó, *Semantics of a sequential language for exact real-number computation*, Theoretical Computer Science **379 (1-2)** (2007) 120–141.

[12] Martin, K., *Domain theoretic models of topological spaces*, in: A. Edalat and A. Jung and K. Keimel and M. Kwiatkowska, eds., Proceedings of Complox III, ENTCS (Elsevier, 1998) **13** 173–181.

[13] Martin, K., *The measurement process in domain theory*, in: Automata, Languages and Programming, (2000) 116–126.

[14] Moggi, E., *Notions of Computations and Monads*, Information and Computation **93 (1)** (1991) 55–92.

[15] Plotkin, G. D., *LCF Considered as a Programming Language*, Theoretical Computer Science **5 (1)** (1977) 223–255.

[16] Potts, P. J., A Edalat. and M. H. Escardó, *Semantics of Exact real arithmetic*, Proceedings 12^{th} IEEE Symposium on Logic in Computer Science (IEEE Computer Society press, July 1997) 248–257.

[17] Scott, D., *Lattice theory, data type and semantics*, in: Randall Rustin, editor, eds., Formal Semantics of Algorithmic Languages (Prentice Hall, 1972) 65–106.

[18] Weihrauch, K., "Computable Analysis", Springer-Verlag, 2000.