NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS

MARK ARMSTRONG, JEFFERY ZUCKER

Department of Computing & Software, McMaster University Hamilton, ON L8S 4K1, Canada

ABSTRACT. Several results from classical computability theory (computability over discrete structures such as the natural numbers and strings over finite alphabets, due to Turing, Church, Kleene and others) have been shown to hold for generalisations of computability theory over total abstract algebras, using a computation model of a high level imperative (*While*) language.

We present a number of results relating to computation on topological partial algebras using an *abstract* model of computation, **While**, based on high level imperative languages. We investigate the validity of several results from the classical theory in the context of topological algebras on the reals: closure of semicomputable sets under finite union, the equivalence of semicomputable and projectively (semi)computable sets, and Post's Theorem.

This research has significance in the field of scientific computation, which is underpinned by computability on the real numbers. By the Continuity Principle, computability of functions implies their continuity. Since equality, order, and other total boolean-valued functions on the reals are clearly discontinuous, we resolve this incompatibility by redefining such functions to be partial, leading us to consider topological partial algebras.

1. INTRODUCTION

1.1. Generalising computability theory.

In classical computability theory, many formalisms have been presented and been proven to be equivalent, including the formalism of Turing machines, λ - calculus, and the μ -recursive functions, presented by Alan Turing [Tur36], Alonzo Church [Chu36] and Stephen C. Kleene [Kle36] during the 1930's. These all capture the informal notion of computation by a finite, deterministic algorithm on \mathbb{N} or on Σ^* (the set of strings from a finite alphabet Σ).

We investigate generalisations of the classical computability theory to other abstract structures, especially the domain of real numbers \mathbb{R} . An important reason for this is that scientific computation is done largely on the reals.

We are very grateful to Professor Ludwig Bröcker for his input on the properties of basic semi-algebraic sets. This research was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

2 NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS

An important difference between \mathbb{R} and \mathbb{N} is that real numbers can in general only be constructed or described as infinite objects; for instance, as infinite sequences of rational numbers. Thus when working with \mathbb{R} , at least with concrete computation models (described below), we must work with the idea of finite approximations. Further, the topology of the reals gives us the concept of "nearness", and hence closeness of approximations. Hence the topology of the reals is a crucial concept in computation over the reals.

A model of computation is a mathematical model of some general method (algorithm) for computing functions, or deciding membership of a set. We distinguish two main kinds of such models: *abstract* and *concrete* [TZ04, XFZ15].

In abstract models of computation, the data are taken as primitive, so that the programs and algorithms do not depend on representations. Examples of abstract models are high level imperative programming languages, flow charts and register machines over any algebra [TZ00, dB80, AO91].

In concrete models, data are given by representations, and so the programs and algorithms depend crucially on the choice of representation, e.g. the representation of reals by (indices of) effective Cauchy sequences of rationals [SHT99, Wei00].

An important part of our work in this paper is to consider whether certain well-known results from classical computability theory still hold in the generalised theory, e.g. the closure of semicomputable sets under union. In addition, we investigate the properties of semicomputable subsets of the real plane, e.g. the equivalence or inequivalence with respect to some notion of semicomputability for variations of a high-level imperative language for *topological algebras* over the reals.

1.2. Related work.

There is a rich history of work in generalised computability theory, with many models having been proposed. Good overviews of several of them are provided in [Wei00, TZ00].

We will focus in this paper on models based on the simple imperative language **While**, which have been investigated thoroughly by the second author [TZ00, TZ15]. In [Fu07, Fu14], an extension of the **While** model (with non-determinism and approximability) was shown to be equivalent to several other models of computation over the reals, including Grzegorczyk-Lacombe (GL) computability [Grz55, Grz57, Lac55, PER89] and tracking computability [SHT99, SHT03, TZ04, TZ05], under some reasonable assumptions.

1.3. Overview.

In Section 2 we review many-sorted algebras, relations and projections, topological partial algebras (in particular the algebra \mathcal{R} on \mathbb{R} with the ring structure of the reals), and the abstract models *While*, *While*^{OR} and *While*^{$\exists N$} over \mathcal{R} , as well as "starred" versions of those languages (i.e. with arrays).

In Section 3 we give definitions and lemmas for the $While(\mathcal{R})$ language, in preparation for a Structure Theorem for $While(\mathcal{R})$ in Section 4. Using that Structure Theorem, we prove that the class of $While(\mathcal{R})$ semicomputable sets is not closed under union. In Section 5 we present results regarding the (in)equivalence of models of computation on \mathcal{R} based on the **While** language.

In Section 6 we summarise our conclusions, and give some ideas for future work.

In Appendix A we consider an adaptation of our (in)equivalence results to the 1-dimensional case over \mathbb{R} .

In Appendix B we outline a proof of the equivalence of $While(\mathcal{R})$ with its starred version.

In Appendix C we check that another result from the classical theory, Post's Theorem, also holds in the case of \mathcal{R} .

1.4. Previous work.

This paper developed out of the first author's Master's thesis. It serves as an extension of the work contained in [XFZ15].

Specifically, the structure theorems presented there for 1-dimensional computation were previously extended to 2-dimensional computation in [Fu14], with an incomplete structure theorem for *While*; we now present a complete version.

Additionally, in [XFZ15] it was shown that the model $While^{\exists N}$ is computationally equivalent to its projective version; we extend this result by showing the (in)equivalence of various models and their projective versions.

2. SIGNATURES; ALGEBRAS; THE While LANGUAGE

We will study the computation of functions and relations by high level imperative programming languages based on the 'while' construct, applied to a many-sorted signature Σ . We give semantics for this language relative to a topological partial Σ -algebra, and define the notions of *computability, semicomputability* and *projective semicomputability* for this language. Much of the material is taken from [TZ00, TZ15].

We begin by reviewing basic concepts of many-sorted signatures and algebras. Next we define the syntax and semantics of the abstract computation model *While*. Then we present several extensions to this language.

2.1. Basic algebraic definitions.

A many-sorted signature Σ is a pair $(Sort(\Sigma), Func(\Sigma))$, where $Sort(\Sigma)$ is a finite set of basic types or sorts $s_1, ..., and Func(\Sigma)$ is a finite set of basic function symbols, $F: s_1 \times ... \times s_m \to s, (m \ge 0)$ (the case m = 0 gives a constant symbol; we then write $F: \to s$).

A product type of Σ has the form $s_1 \times \cdots \times s_m$, where m > 0 and s_1, \ldots, s_m are Σ -sorts. We write u, v, \ldots for product types. A function type has the form $u \to s$, where u is a product type, or simply $\to s$ (for constant functions). For a signature Σ , a Σ -algebra A has, for each Σ -sort s, a non-empty set A_s , called the *carrier* of sort s, and for each function symbol $F: s_1 \times ... \times s_m \to s$, a function $F^A: A_{s_1} \times ... \times A_{s_m} \to A_s$.¹

We write $\Sigma(A)$ for the signature of an algebra A.

An algebra A is said to be *total* if F^A is total for all $F \in Func(\Sigma)$; otherwise, it is *partial*.

Example 2.1. The signature and (total) algebra of the booleans are as follows:

signature	$\Sigma(\mathcal{B})$	algebra carrier	\mathcal{B}
sorts	bool	carrier	$\mathbb B$
functions	$t, ff: \rightarrow bool$	functions	$\mathbf{t}^{B}, \mathbf{f}^{B} \colon \to \mathbb{B}$
	$or,and\colonbool^2 obool$		$or^{B},and^{B}\colon\mathbb{B}^{2} o\mathbb{B}$
	$not\colonbool\tobool$		$not^{B} \colon \mathbb{B} o \mathbb{B}$
	$if-then-else\colonbool^3\tobool$		$if\text{-}then\text{-}else^{B}\colon \mathbb{B}^3\to \mathbb{B}$

The (total) algebra of the naturals is as follows:

 $\begin{array}{ll} \text{algebra} & \mathcal{N} \\ \text{import} & \mathcal{B} \\ \text{carriers} & \mathbb{N} \\ \text{functions} & 0^{\mathsf{N}} \colon \to \mathbb{N} \\ & \text{suc}^{\mathsf{N}} \colon \mathbb{N} \to \mathbb{N} \\ & \text{eq}^{\mathsf{N}}, \text{less}^{\mathsf{N}} \colon \mathbb{N}^2 \to \mathbb{B} \\ & \text{if-then-else}^{\mathsf{N}} \colon \mathbb{B} \times \mathbb{N}^2 \to \mathbb{N} \end{array}$

where the carrier sets \mathbb{B} and \mathbb{N} , the functions and constants \mathbf{t}^{B} , \mathbf{f}^{B} , $\mathsf{and}^{\mathsf{B}}$, or^{B} , $\mathsf{not}^{\mathsf{B}}$, $\mathbf{0}^{\mathsf{N}}$, S^{N} , eq^{N} and $\mathsf{less}^{\mathsf{N}}$ have their usual meanings.

We will use the infix notations \lor and \land in place of 'or' and 'and also write '¬' in place of not, and drop the superscripts 'B' and 'N' where unambiguous.

Definition 2.2 (Standard signatures and algebras). A signature is called *standard* if it includes the sorts and functions of $\Sigma(\mathcal{B})$.

Correspondingly, an algebra is called *standard* if it is an expansion of \mathcal{B} (i.e. it contains the carrier \mathbb{B} with the standard boolean operators) and additionally, any equality operators, for sorts on which they are defined, are (partial) identities on these sorts.

Definition 2.3 (\mathbb{N} -standard signatures and algebras). A signature is called \mathbb{N} -standard if, in addition to being standard, it includes the sorts and functions of $\Sigma(\mathcal{N})$.

Correspondingly, an algebra is called \mathbb{N} -standard if, in addition to being standard, it is an expansion of \mathcal{N} (i.e. it contains the carrier \mathbb{N} with the standard arithmetic operators).

¹We use \rightharpoonup in place of \rightarrow to signify that a function is partial.

Remark 2.4. All signatures and algebras in this paper will be assumed to be \mathbb{N} -standard (and hence also standard). So Σ refers to any \mathbb{N} -standard signature and A to any \mathbb{N} -standard algebra.

2.2. Relations and projections.

Notation 2.5. For a Σ -product type $u = s_1 \times ... \times s_m$, we write

$$A_u =_{df} A_{s_1} \times \ldots \times A_{s_m}$$

A relation R on A of type u (written R: u) is a subset of A_u .

The *complement* of a relation R: u is the relation

$$R^c = A_u \backslash R = \{ a \in A_u \mid a \notin R \}$$

Definition 2.6 (**Projection**). Let R be a relation of type $u = s_1 \times ... \times s_m$ where m > 0. Let $\vec{i} = i_1, ..., i_r$ be a list of numbers such that $1 \le i_1 < ... < i_r \le m$, and let $\vec{j} = j_1, ..., j_{m-r}$ be the list $\{1, ..., m\} \setminus \vec{i}$. Then the projection of R off \vec{i} is the relation $S: s_{j_1} \times ... \times s_{j_{m-r}}$ where $\forall x_{j_1}: s_{j_1}, ..., x_{j_{m-r}}: s_{j_{m-r}}$,

 $S(x_{j_1}, ..., x_{j_{m-r}}) \iff \exists x_{i_1} : s_{i_1}, ..., x_{i_r} : s_{i_r}, R(x_1, ..., x_m).$

2.3. Topological partial algebras.

Recall the definition of continuity of partial functions:

Definition 2.7 (Continuity). Given two topological spaces X and Y, a partial function $f: X \to Y$ is *continuous* iff the preimage under f of any open subset of Y is open in X, or equivalently

- (1) dom(f) is open, and
- (2) for every open $V \subseteq Y$, $f^{-1}[V] =_{df} \{x \in X \mid x \in dom(f) \text{ and } f(x) \in V\}$ is open (in X).

Definition 2.8 (Topological partial algebra). A topological partial algebra is a partial Σ algebra with topologies on the carriers such that each of the basic Σ -functions is continuous. The carriers \mathbb{B} and \mathbb{N} have the discrete topology.

Discussion 2.9 (Continuity of computable functions). The significance of the continuity of the basic functions of a topological algebra A is that it implies continuity of all *While* computable function on A [TZ99, TZ00].

This is in accordance with the *Continuity Principle*, which can be expressed as

$computability \implies continuity$

This principle is a classical design decision for models of computation over the reals; see for example [PER89, Wei00, TZ99].

One motivation for this design decision is *Hadamard's principle* [Had52], which, as reformulated by Courant and Hilbert [CH53, Had64], states that for a scientific problem to be well posed, the solution must (apart from existing and being unique) *depend continuously* on the data.

2.4. The algebra \mathcal{R} of reals.

In the following sections, we work mostly with the following topological algebra²:

 $\begin{array}{ll} \text{algebra} & \mathcal{R} \\ \text{import} & \mathcal{B}, \mathcal{N} \\ \text{carriers} & \mathbb{R} \\ \text{functions} & 0^{\mathsf{R}}, 1^{\mathsf{R}} \colon \rightarrow \mathbb{R} \\ & \mathsf{neg}^{\mathsf{R}} \colon \mathbb{R} \rightarrow \mathbb{R} \\ & \mathsf{plus}^{\mathsf{R}}, \mathsf{times}^{\mathsf{R}} \colon \mathbb{R}^2 \rightarrow \mathbb{R} \\ & \mathsf{eq}^{\mathsf{R}}, \mathsf{less}^{\mathsf{R}} \colon \mathbb{R}^2 \rightarrow \mathbb{B} \\ & \mathsf{if-then-else}^{\mathsf{R}} \colon \mathbb{B} \times \mathbb{R}^2 \rightarrow \mathbb{R} \\ & \mathsf{cand}, \mathsf{cor} \colon \mathbb{B}^2 \rightarrow \mathbb{B} \end{array}$

where the functions and constants 0^{R} , 1^{R} , $\mathsf{plus}^{\mathsf{R}}$ and $\mathsf{times}^{\mathsf{R}}$ have their usual definitions, the function eq^{R} and $\mathsf{less}^{\mathsf{R}}$ are defined as in Remark 2.11 below and the "conditional" boolean operators cand and cor can defined using if-then-else^B as in Discussion 2.12.

We will often write -, + and * in place of 'neg^R', 'plus^R' and 'times^R', = and < in place of 'eq^R' and 'less^R' (and 'eq^N' and 'less^N'), $\stackrel{c}{\wedge}$ and $\stackrel{c}{\vee}$ in place of cand and cor, and drop the superscripts \cdot^{R} , \cdot^{B} and \cdot^{N} where unambiguous.

The signature $\Sigma(\mathcal{R})$ can be inferred from the above, with real as the sort of \mathbb{R} .

Notation 2.10. We write \uparrow to denote "undefinedness". We use the symbol ' \simeq ' to denote "Kleene equality", where the two sides are either both defined and equal, or both undefined [Kle52, §63].

Remark 2.11 (Equality on the reals). \mathcal{R} is a partial algebra, with the basic booleanvalued partial functions eq^{R} and $\mathsf{less}^{\mathsf{R}}$, which for $x, y \in \mathbb{R}$ are defined as:

$$\mathsf{eq}^{\mathsf{R}}(x,y) \simeq \begin{cases} \uparrow & \text{if } x = y \\ \mathbf{f} & \mathrm{o/w} \end{cases}$$

and

$$\mathsf{less}^{\mathsf{R}}(x,y) \simeq \begin{cases} \mathsf{t} & \text{if } x < y \\ \mathsf{f} & \text{if } x > y \\ \uparrow & \text{o/w.} \end{cases}$$

Note that by contrast, the basic functions eq^N and $less^N$ on \mathbb{N} are total.

²In [FZ15] ' \mathcal{R} ' was used for the algebra of reals which also included the multiplicative inverse operation on the reals. ' \mathcal{R}_0 ' referred to the algebra without the multiplicative inverse.

Discussion 2.12 (Conditional boolean operators). For $b_1, b_2 \in \mathbb{B}$, the semantics of the "conditional" operators $\stackrel{c}{\wedge}$ and $\stackrel{c}{\vee}$ of \mathcal{R} can be defined by:

$$b_1 \wedge b_2 \simeq \text{if } b_1 \text{ then } b_2 \text{ else ff}$$

and

$$b_1 \lor b_2 \simeq \mathsf{if} \ b_1 \mathsf{then} \mathsf{tt} \mathsf{else} \ b_2$$

i.e., the operators $\stackrel{c}{\wedge}$ and $\stackrel{c}{\vee}$ are "evaluated from the left".

By contrast, the semantics of operators \wedge and \vee are taken to be "strict", i.e. $b_1 \wedge b_2$ is undefined whenever either b_1 or b_2 is (and similarly for \vee).

While it is not strictly necessary to include the conditional operators, they seem a natural inclusion when dealing with partial functions.

Note that we will discuss the semantics of the *infinite* conditional disjunction in Discussion 3.3.

Discussion 2.13 (Motivation for use of partial functions). We present two approaches motivating the above partial functions. The first is a discussion of the continuity of comparison operators on \mathbb{R} . The second is a *Gedankenexperiment* involving concrete models of computation.

- (1) The total versions of the comparison operators eq^R and less^R on R are not continuous. (By contrast any boolean-valued operator on N is trivially continuous, because N has the discrete topology). Continuity of these operators is important because of the Continuity Principle and our definition of topological partial algebras (Definition 2.8 and Discussion 2.9), which requires all basic operators to be continuous.
- (2) Consider now the task of determining whether, in some concrete model, two representations denote the same real number or not. As an example, let us use, as representations of these real numbers, effective Cauchy sequences of rationals $(r_0, r_1, r_2, ...)$. We assume for our convenience³ that the sequences are "fast", i.e.,

$$\forall n, \forall m \ge n, |r_n - r_m| < 2^{-n}.$$

Now suppose given two reals x and y with such representations $(r_0, r_1, r_2, ...)$ and $(s_0, s_1, s_2, ...)$ respectively. Suppose also that for n = 0, 1, 2, 3, ... the inputs r_n and s_n are observed (from some device) at n time units. Then the first real is less than the second iff for some $n, r_n + 2 \cdot 2^{-n} < s_n$, and this can be determined in a finite amount of time. Correspondingly, the two reals are equal iff for all $n, |r_n - s_n| < 2 \cdot 2^{-n}$, but this cannot be determined in a finite amount of time. So from this example it is natural for comparison operators on reals x and y to diverge in cases when x = y.

³When using effective Cauchy sequences as a representation, we assume given a computable modulus of convergence; given this, we can (by effectively taking subsequences) easily restrict our attention to sequences which are fast.

Remark 2.14. Throughout this paper we focus on functions on \mathbb{R}^2 . The well-established methods of proof for \mathbb{R} do not generally "lift" to \mathbb{R}^2 (see e.g. Appendix A), but the methods we will use for \mathbb{R}^2 easily "lift" to \mathbb{R}^n for n > 2.

2.5. The algebra \mathcal{R}^* .

The algebra \mathcal{R}^* is formed from \mathcal{R} by adding the carriers \mathbb{R}^* , \mathbb{N}^* and \mathbb{B}^* (of sorts real^{*}, nat^{*} and bool^{*} respectively) consisting of all *finite sequences* or *arrays* of reals, naturals and booleans (respectively), together with certain standard constants and operations for the empty array, updating of arrays, etc.

The significance of arrays for computation is that they provide finite but unbounded memory. Note that despite the convenience of the starred sorts, \mathcal{R}^* is computationally equivalent to \mathcal{R} (a proof of this fact is outlined in Appendix B). As such, we omit the precise definition of \mathcal{R}^* , which can be found in [TZ00, TZ15].

2.6. Syntax of terms and the While programming language.

As has been mentioned, we will study the abstract models of high level imperative programming languages based on the 'while' construct. We begin with the syntax.

Note that Ξ denotes syntactic identity between two expressions.

- Σ -variables: For each Σ -sort s, there are variables $\mathbf{x}^s, \mathbf{y}^s, \dots$ of sort s. $Var_s(\Sigma)$ is the set of variables of sort s, and $Var(\Sigma)$ is the set of all Σ -variables.
- Σ -terms: $Tm(\Sigma)$ is the set of Σ -terms: t, ... and $Tm_s(\Sigma)$ is the set of Σ -terms of sort s: $t^s, ...$. We define this using a modified BNF:

$$t^s ::= \mathbf{x}^s \mid F(t_1^{s_1}, ..., t_m^{s_m})$$

where F is a Σ -function symbol of type $s_1 \times ... \times s_m \to s \ (m \ge 0)$.

• **Statements**: **Stmt**(Σ) is the set of Σ -statements S, ... generated by:

 $S ::= \mathsf{skip} \mid \mathbf{x} := t \mid S_1; S_2 \mid \text{ if } b \text{ then } S_1 \text{ else } S_2 \text{ fi} \mid \mathsf{while } b \text{ do } S_0 \text{ od}$

where $\mathbf{x} := t$ denotes simultaneous assignment, i.e., for some m > 0, $\mathbf{x} \equiv (\mathbf{x}_1, ..., \mathbf{x}_m)$ and $t \equiv (t_1, ..., t_m)$ are variable and term tuples of the same product type, with the condition that $\mathbf{x}_i \not\equiv \mathbf{x}_j$ for $i \neq j$; and b is a Σ -boolean⁴.

• **Procedures**: $Proc(\Sigma)$ is the set of Σ -procedures P, ... of the form:

$P \equiv \operatorname{proc} D$ begin S end

where the statement S is the body and D is a variable declaration of the form

$D \equiv in a: u \text{ out } b: v aux c: w$

where **a**, **b** and **c** are tuples of *input*, *output* and *auxiliary variables* respectively. We stipulate further:

- (i) a, b and c each consist of distinct variables, and they are pairwise disjoint,
- (ii) every variable occurring in the body S must be declared in D (among a, b or c).

If a: u and b: v, then P has type $u \to v$, written $P: u \to v$.

⁴That is, a Σ -term of sort bool.

2.7. Semantics of terms and the While language.

We now give the semantics of terms, statements and procedures. To begin, we need to define the notions of *state* and *variant* a of state.

A state over A is a family $\langle \sigma_s | s \in Sort(\Sigma) \rangle$ of functions $\sigma_s \colon Var_s(\Sigma) \to A_s$. State(A) is the set of states over A, with elements σ, \dots .

We write $\sigma(\mathbf{x})$ for $\sigma_s(\mathbf{x})$ when $\mathbf{x} \in Var_s(\Sigma)$. We also write, for tuples $\mathbf{x} \equiv (\mathbf{x}_1, ..., \mathbf{x}_m) : u$, $\sigma[\mathbf{x}]$ in place of $(\sigma(\mathbf{x}_1), ..., \sigma(\mathbf{x}_m))$.

Let σ be a state over A, and for some Σ -product type u, let $\mathbf{x} \equiv (\mathbf{x}_1, ..., \mathbf{x}_n)$: u and $a = (a_1, ..., a_n) \in A_u$ (for $n \ge 1$). We define the variant $\sigma\{\mathbf{x}/a\}$ to be the state over A formed from σ by replacing its value at \mathbf{x}_i by a_i for i = 1, ..., n. That is, for all variables \mathbf{y} :

$$\sigma\{\mathbf{x}/a\}(\mathbf{y}) = \begin{cases} \sigma(\mathbf{y}) & \text{if } \mathbf{y} \neq \mathbf{x}_i \text{ for } i = 1, ..., n \\ a_i & \text{if } \mathbf{y} \equiv \mathbf{x}_i \end{cases}$$

We can now define the semantics of terms, statements and procedures:

• Σ -terms: The meaning of a Σ -term t is given by the function

$$\llbracket t \rrbracket^A \colon \mathbf{State}(A) \rightharpoonup A_s$$

where $[t]^A \sigma$ is the value of t in A at state σ .

The definition of
$$\llbracket t \rrbracket^A \sigma$$
 is by structural induction on Σ -terms t :
 $- \llbracket \mathbf{x} \rrbracket^A \sigma = \sigma(\mathbf{x})$
 $- \llbracket F(t_1, ..., t_m) \rrbracket^A \sigma \simeq \begin{cases} F(\llbracket t_1 \rrbracket^A \sigma, ..., \llbracket t_m \rrbracket^A \sigma) & \text{if } \llbracket t_i \rrbracket^A \sigma \downarrow \text{ for all } i = 1, ..., m \\ \uparrow & o/w \end{cases}$

• **Statements**: The meaning of a **While** statement S w.r.t. A, written $[\![S]\!]^A$, is a partial state transformation on the algebra A:

$$\llbracket S \rrbracket^A \colon State(A) \rightharpoonup State(A),$$

Its definition is standard [TZ99, TZ00] and lengthy, and so we omit it. Briefly, it is based on defining the computation sequence of S starting in a state σ , or rather the *n*th component of this sequence, by a primary induction on n, and a secondary induction on the size of S.

• *Procedures*: The meaning of a *While* procedure

$$P \equiv$$
 proc in a: u out b: v aux c: w begin S end,

of type $u \to v$, written $\llbracket P \rrbracket^A : A_u \rightharpoonup A^v$, is defined as follows⁵. For $a \in A_u$, let σ be any state on A such that $\sigma[\mathbf{a}] = a$. Then⁶

$$\llbracket P \rrbracket^A(a) \simeq \begin{cases} \sigma'[\mathbf{b}] & \text{if } \llbracket S \rrbracket^A \sigma \downarrow \sigma' \\ \uparrow & \text{if } \llbracket S \rrbracket^A \sigma \uparrow . \end{cases}$$

2.8. While computability and semicomputability.

A function $f: A_u \to A_s$ is said to be *computable* (on A) by a **While** procedure $P: u \to s$ if $f = P^A$.

While(A) is the class of functions While computable on A.

The halting set of a procedure $P: u \to v$ on A is the set

$$Halt^{A}(P) =_{df} \{a \in A_u \mid P^{A}(a) \downarrow\}.$$

A set $R \subseteq A_u$ is **While** semicomputable on A if it is the halting set on A of some **While** procedure.

2.9. Extending While to While^{OR} and While^{$\exists N$}.

In preparation for the theorems in Sections 4 and 5, we give the semantics of *strong* disjunction and *strong* existential quantification to introduce the **While**^{OR} and **While**^{$\exists N$} extensions to the **While** language.

The motivation for these extensions is that with our model of **While** computation on \mathcal{R} , the partial operations leave us unable to implement *interleaving* or *merging*. The problem is that in the interleaving of two processes, one process may converge and the other diverge locally (because of the partial operations). The resulting process will then diverge, whereas we might want it to converge. Thus, as we will see in Section 4, the union of two semicomputable sets is not necessarily semicomputable. Concrete models, which work with representations of the reals instead of taking them as primitive, do not have this deficiency, as local divergence is not an issue. The extensions **While**^{OR} and **While**^{IN} compensate for this deficiency in **While**.

The **While**^{OR} language is created from **While** by introducing the strong disjunction operator ' ∇ ', where $b_1 \nabla b_2$ converges to **t** if either b_1 or b_2 do so, even if the other diverges.

⁵We overload the symbols \uparrow , \downarrow and \simeq , discussed in Notation 2.10 where they deal with the *definedness* of terms, for use with statements and procedures. Here instead of *definedness* they refer to the *convergence* or *divergence* of computations.

⁶The definition can be shown to be independent of the exact choice of σ (by the Functionality Lemma [TZ00, Lemma 3.4]).

The $While^{\exists N}$ language is created from While by introducing a strong existential quantification construct over the naturals in the context of an assignment:

$$\mathbf{x}^{\mathsf{B}} := \exists \mathbf{n} \ P(t, \mathbf{n})$$

where n: nat and P is a boolean-valued procedure. Its semantics are defined by

$$\llbracket \exists \mathbf{n} \ P(t,\mathbf{n}) \rrbracket^A \sigma \simeq \begin{cases} \mathbf{t} & \text{if } P(\llbracket t \rrbracket^A \sigma, \ m) \downarrow \mathbf{t} \text{ for some } m \\ \uparrow & \text{o/w.} \end{cases}$$

To simplify the exposition, we include the strong disjunction operator ∇ in the **While**^{$\exists N$} language.

If instead of strong existential quantification, we constructed $While^{\exists N}$ using existential quantification "evaluated from the left", this would (as can easily be shown) give a conservative extension of While, in that any function implemented in it could be implemented in While (assuming here we do not include the strong disjunction operator)⁷.

By means of these constructs, interleaving of processes may be simulated. The *While*^{OR} language allows for the interleaving of an arbitrary but finite number of processes, and the *While*^{$\exists N$} language for *infinitely* many processes.

3. Semantic disjointedness; Engeler's Lemma

We now present some important background relating to boolean terms. We begin by introducing notation used throughout this section and section 4:

Notation 3.1.

- We will often write **x** to mean a *u*-tuple of variables; i.e. $\mathbf{x} \equiv (\mathbf{x}_1, ..., \mathbf{x}_m) : u$.
- For a tuple of variables $\mathbf{x} \equiv (\mathbf{x}_1, ..., \mathbf{x}_m), m \ge 1$, let **Bool**(\mathcal{R})(\mathbf{x}) be the set of $\Sigma(\mathcal{R})$ -booleans containing variables in \mathbf{x} only.
- Similarly define $Bool(\mathcal{R}^{OR})(\mathbf{x})$, where \mathcal{R}^{OR} is the extension of the algebra \mathcal{R} including strong disjunction (∇) .

3.1. Engeler's Lemma.

This is of vital importance in proving our Structure Theorem for $While(\mathcal{R})$ semicomputable sets.

First we need the concepts of *semantic disjointedness* of a sequence of booleans.

Definition 3.2 (Semantic Disjointedness). A sequence $(b_0, b_1, b_2, ...)$ of boolean terms is *semantically disjoint* over A if for any state σ over A and any n,

 $\llbracket b_n \rrbracket^A \sigma \downarrow \mathbf{t} \implies \forall i \neq n, \llbracket b_i \rrbracket^A \sigma \downarrow \mathbf{f}.$

⁷cf. the two definitions of infinite disjunction given in Discussion 3.3.

Discussion 3.3 (Semantics of infinite disjunction). Let (b_k) be a sequence of Σ booleans. There are (at least) two different reasonable semantic definitions for the infinite disjunction

$$\bigvee_{k=0}^{\infty} b_k$$

for 3-valued logics ("reasonable" in the sense of having computational significance):

(1) Infinite conditional disjunction ("evaluation from the left"), written $\bigvee_{k=0}^{c} b_k$, with two possible results, **t** and \uparrow :

$$\llbracket \bigvee_{k=0}^{\infty} b_k \rrbracket^A \sigma \simeq \begin{cases} \mathbf{t} & \text{if } \exists k, \llbracket b_k \rrbracket^A \sigma \downarrow \mathbf{t} \text{ and } \forall i < k, \llbracket b_i \rrbracket^A \sigma \downarrow \mathbf{f} \\ \uparrow & \text{otherwise.} \end{cases}$$

This definition is easily seen to be **While** computable (in the sequence of codes $\lceil b_k \rceil$).

(2) Infinite strong disjunction ("strong Kleene evaluation"), written $\sum_{k=0}^{\infty} b_k$, again with two possible results, **t** and \uparrow :

$$\llbracket \bigvee_{k=0}^{\infty} b_k \rrbracket^A \sigma \simeq \begin{cases} \mathbf{t} & \text{if } \exists k, \llbracket b_k \rrbracket^A \sigma \downarrow \mathbf{t} \\ \uparrow & \text{otherwise.} \end{cases}$$

This definition is not (in general) **While** computable. If it were, **While** would be at least as powerful as **While**^{\exists N}, which we will show is not the case (Theorems 4, 5).

Remark 3.4. Note that if an effective sequence of booleans (b_k) is *semantically disjoint* over A, then for any σ ,

$$\llbracket \bigvee_{k=0}^{\infty} b_k \rrbracket^A \sigma \simeq \llbracket \bigvee_{k=0}^{\infty} b_k \rrbracket^A \sigma$$

i.e. $[\![\sum\limits_{k=0}^\infty b_k]\!]^A \sigma$ can be "evaluated from the left".

We will only consider infinite disjunction in the context of semantically disjoint sequences of booleans, and so for our purposes the choice of semantic definition is irrelevant.

Lemma 3.5 (Engeler's Lemma for While). Given a partial Σ -algebra A and a Σ -product type $u = s_1 \times ... \times s_m$, if a relation $R \subseteq A_u$ is While semicomputable over A then R can be expressed as the infinite disjunction of a semantically disjoint effective sequence of Σ -booleans over A; i.e., for all $x: A_u$

$$x \in R \iff \bigvee_{k=0}^{\infty} \llbracket b_k \rrbracket \sigma \{ \mathbf{x} / x \}$$

for some semantically disjoint effective sequence of booleans $(b_0, b_1, ...)$, where σ is any state and each b_k has no free variables other than those in \mathbf{x} : u. **Remark 3.6.** Engeler's Lemma can be proved by an analysis of the computation trees for *While* programs [TZ00]. It was originally stated in [Eng68] without the semantic disjoint-edness property; this property was subsequently noted in [XFZ15, §4].

3.2. Canonical form for $Bool(\mathcal{R})(\mathbf{x})$.

In the proof of our Structure Theorem in §4 for $While(\mathcal{R})$ semicomputable sets, we require a canonical form for booleans containing real variables only.

Lemma 3.7 (Canonical form for $Bool(\mathcal{R})(\mathbf{x})$). For a tuple \mathbf{x} of real variables, any term in $Bool(\mathcal{R})(\mathbf{x})$ is effectively semantically equivalent to a boolean combination of equations and inequalities of the form:

$$p(\mathbf{x}) = 0 \quad and \quad q(\mathbf{x}) > 0$$

where p and q are polynomials in x of degree $> 0.^8$

The proof is by structural induction on the booleans.

Note that the canonical form of booleans coincides with the notion of *semi-algebraic set*, to which we now turn.

3.3. Basic and semi-algebraic sets.

We introduce the concepts of *basic* and *semi-algebraic* sets, which are fundamental to our results. We consider these sets on \mathbb{R}^2 , though they can clearly be generalised to \mathbb{R}^n for any $n \ge 1$.

Definition 3.8 (Basic set). A *basic set* is a subset of \mathbb{R}^2 that can be expressed in the form $\{x \in \mathbb{R}^2 \mid p_1(x) > 0 \land \dots \land p_k(x) > 0\}$ (k > 0)

where $p_1, ..., p_k$ are polynomials.

Remark 3.9. In this paper, polynomials are always taken as having rational [or, equivalently, integer] coefficients.

Note that all basic sets are open⁹.

Definition 3.10 (Semi-algebraic set). A *semi-algebraic set* is a subset of \mathbb{R}^2 that can be expressed in the form

 $\bigcup_{i=1}^{n} \{x \in \mathbb{R}^2 \mid p_{i,1}(x) > 0 \land \dots \land p_{i,k_i}(x) > 0 \land q_{i,1}(x) = 0 \land \dots \land q_{i,l_i}(x) = 0\} \quad (k_i, l_i > 0)$ where each $p_{i,j}$ and $q_{i,j}$ is a polynomial.

Remark 3.11. The class of basic sets is closed under binary intersection.

⁸Note that this "canonical form" is not unique.

⁹We use "basic sets" to mean "basic open semialgebraic sets"

Remark 3.12. Given a polynomial $p(\mathbf{x})$ on \mathbb{R}^2 , there are disjoint basic sets B^+ , B^- and a semi-algebraic set D, such that

- p > 0 on B^+
- p < 0 on B^-
- p = 0 on D

and $B^+ \cup B^- \cup D = \mathbb{R}^2$

3.4. Positive and negative sets.

Definition 3.13 (Positive, negative and divergent sets of booleans). Let $\mathbf{x} \equiv (\mathbf{x}_1, \mathbf{x}_2)$. For any $b \in Bool(\mathcal{R})(\mathbf{x})$, let:

$$PS(b) =_{df} \{ x \in \mathbb{R}^2 \mid b[x] = \mathbf{t} \}$$

$$NS(b) =_{df} \{ x \in \mathbb{R}^2 \mid b[x] = \mathbf{f} \}$$

$$DS(b) =_{df} \{ x \in \mathbb{R}^2 \mid b[x] \uparrow \}.$$

These will be used in the proof of the Partition Lemma in the next section.

4. $While(\mathcal{R})$ semicomputable sets: Structure Theorem and failure of closure under union

We present a Partition Lemma for booleans in $Bool(\mathcal{R})(\mathbf{x})$, where $\mathbf{x} \equiv (\mathbf{x}_1, \mathbf{x}_2)$, which we then use to give a Structure Theorem for $While(\mathcal{R})$ semicomputability over \mathbb{R}^2 . By means of this we will give an example of two $While(\mathcal{R})$ semicomputable sets whose union is not $While(\mathcal{R})$ semicomputable.

Convention 4.1. For the remainder of this section, let $\mathbf{x} \equiv (\mathbf{x}_1, \mathbf{x}_2)$.

4.1. Partition Lemma for booleans in $Bool(\mathcal{R})(\mathbf{x})$.

Lemma 4.2 (Partition Lemma for booleans in $Bool(\mathcal{R})(\mathbf{x})$). Consider any boolean $b \in Bool(\mathcal{R})(\mathbf{x})$. The positive and negative sets¹⁰ for b in \mathbb{R}^2 can be expressed as:

$$PS(b) = \bigcup_{i=1}^{k} B_i^+$$
$$NS(b) = \bigcup_{j=1}^{l} B_j^-.$$

where B_i^+ , B_j^- are basic sets, and

$$\begin{array}{l} B_{i}^{+} \cap B_{j}^{-} = \emptyset \ for \ i = 1, ..., k \ and \ j = 1, ..., l \\ B_{i_{1}}^{+} \cap B_{i_{2}}^{+} = \emptyset \ for \ i_{1} \neq i_{2} \\ B_{j_{1}}^{-} \cap B_{j_{2}}^{-} = \emptyset \ for \ j_{1} \neq j_{2} \end{array}$$

 $^{^{10}}$ cf. Definition 3.13. For our purpose, the form of the divergent set of b, DS(b), is unimportant.

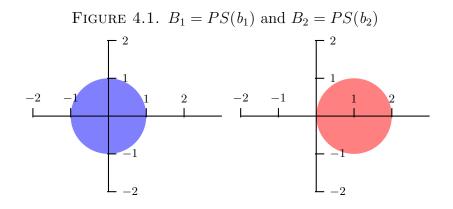
Before giving the proof, we consider some examples of positive and negative sets in \mathbb{R}^2 .

Examples 4.3 (Positive and negative sets in \mathbb{R}^2). These examples are of interest because our counterexample to the closure of semicomputable sets under union will build on them.

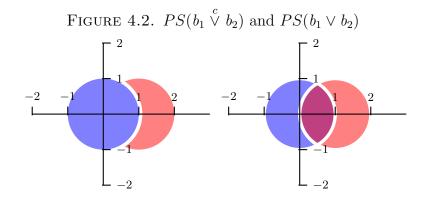
Consider the polynomials $p_1 \equiv -\mathbf{x}_1^2 - \mathbf{x}_2^2 + 1$ and $p_2 \equiv -(\mathbf{x}_1 - 1)^2 - \mathbf{x}_2^2 + 1$, and the booleans $b_1 \equiv p_1(\mathbf{x}_1, \mathbf{x}_2) > 0$ and $b_2 \equiv p_2(\mathbf{x}_1, \mathbf{x}_2) > 0$. Define:

$$B_1 = PS(b_1) = \{ (x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0 \}$$

$$B_2 = PS(b_2) = \{ (x_1, x_2) \in \mathbb{R}^2 \mid p_2(x_1, x_2) > 0 \}$$



 B_1 and B_2 are basic sets, and can be easily seen to be $While(\mathcal{R})$ semicomputable. They are pictured in Figure 4.1.



The sets $PS(b_1 \stackrel{c}{\lor} b_2)$ and $PS(b_1 \lor b_2)$, pictured in Figure 4.2, are also easily seen to be **While**(\mathcal{R}) semicomputable. These sets can be represented as the union of two and three

16 NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS disjoint basic sets respectively¹¹:

$$PS(b_1 \stackrel{c}{\vee} b_2) = \{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0\} \\ \cup \{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0, \ p_2(x_1, x_2) < 0\}$$

$$PS(b_1 \lor b_2) = \{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0, \ p_2(x_1, x_2) > 0\}$$
$$\cup \{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0, \ p_2(x_1, x_2) < 0\}$$
$$\cup \{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) < 0, \ p_2(x_1, x_2) > 0\}$$

Proof of the Partition Lemma. This is by structural induction on \mathbb{R} -booleans with variables in $\mathbf{x} \equiv (\mathbf{x}_1, \mathbf{x}_2)$, which we assume are in canonical form.

- Base case: $b \equiv p(\mathbf{x}) = 0$ or $p(\mathbf{x}) > 0$. Immediate from Remark 3.12. In each case there is a single basic set for each positive and negative set.
- *Induction step*: In what follows, suppose:

$$PS(b_1) = \bigcup_{i=1}^{k_1} B_{1i}^+$$
$$NS(b_1) = \bigcup_{i=1}^{l_1} B_{1i}^-$$
$$PS(b_2) = \bigcup_{j=1}^{k_2} B_{2j}^+$$
$$NS(b_2) = \bigcup_{j=1}^{l_2} B_{2j}^-$$

Now we consider the various cases based on the major operator of b:

(i) $b \equiv \neg b_1$. Just exchange the positive and negative sets of b_1 ; since all three properties hold for both, they still hold after switching.

 $[\]overline{}^{11}$ The proof of the Partition Lemma serves as an algorithm for constructing such representations.

(ii) $b \equiv b_1 \lor b_2$. Then let

$$PS_{1} = \bigcup_{i=1}^{k_{1}} \bigcup_{j=1}^{k_{2}} (B_{1i}^{+} \cap B_{2j}^{+})$$
$$PS_{2} = \bigcup_{i=1}^{k_{1}} \bigcup_{j=1}^{l_{2}} (B_{1i}^{+} \cap B_{2j}^{-})$$
$$PS_{3} = \bigcup_{i=1}^{l_{1}} \bigcup_{j=1}^{k_{2}} (B_{1i}^{-} \cap B_{2j}^{+})$$

Then:

$$PS(b) = PS_1 \cup PS_2 \cup PS_3$$
$$NS(b) = \bigcup_{i=1}^{l_1} \bigcup_{j=1}^{l_2} (B_{1i}^- \cap B_{2j}^-)$$

The sets PS_1 , PS_2 and PS_3 are disjoint, because for any i and j, $B_{1i}^+ \cap B_{1j}^- = \emptyset$ and $B_{2i}^+ \cap B_{2j}^- = \emptyset$. Further, the sets $B_{1i}^a \cap B_{2j}^b$ $(a, b \in \{+, -\})$ are all mutually disjoint, because $B_{ni_1}^+ \cap B_{ni_2}^+ = \emptyset$ for $i_1 \neq i_2$ and $n \in \{1, 2\}$, and $B_{nj_1}^- \cap B_{nj_2}^- = \emptyset$ for $j_1 \neq j_2$ and $n \in \{1, 2\}$. So PS(b) and NS(b) are finite unions of disjoint basic sets (by Remark 3.11).

(iii) $b \equiv b_1 \wedge b_2$. Then

$$PS(b) = \bigcup_{i=1}^{k_1} \bigcup_{j=1}^{k_2} (B_{1i}^+ \cap B_{2j}^+)$$
$$NS(b) = (\bigcup_{i=1}^{l_1} \bigcup_{j=1}^{l_2} (B_{1i}^- \cap B_{2j}^-)) \cup (\bigcup_{i=1}^{k_1} \bigcup_{j=1}^{l_2} (B_{1i}^+ \cap B_{2j}^-)) \cup (\bigcup_{i=1}^{l_1} \bigcup_{j=1}^{k_2} (B_{1i}^- \cap B_{2j}^+))$$

Similar to case (ii).

(iv) $b \equiv b_1 \stackrel{c}{\lor} b_2$. Then

$$PS(b) = \bigcup_{i=1}^{k_1} B_{1i}^+ \cup (\bigcup_{i=1}^{l_1} \bigcup_{j=1}^{k_2} (B_{1i}^- \cap B_{2j}^+))$$
$$NS(b) = \bigcup_{i=1}^{l_1} \bigcup_{j=1}^{l_2} (B_{1i}^- \cap B_{2j}^-)$$

Again, similar to case (ii).

(v) $b \equiv b_1 \stackrel{c}{\wedge} b_2$. Then

$$PS(b) = \bigcup_{i=1}^{k_1} \bigcup_{j=1}^{k_2} (B_{1i}^+ \cap B_{2j}^+)$$
$$NS(b) = \bigcup_{i=1}^{k_1} B_{1i}^- \cup (\bigcup_{i=1}^{k_1} \bigcup_{j=1}^{l_2} (B_{1i}^+ \cap B_{2j}^-))$$

Again, similar to (ii).

Remark 4.4. The Partition Lemma for booleans in $Bool(\mathcal{R})(\mathbf{x})$ does not hold for booleans in $Bool(\mathcal{R}^{\mathsf{OR}})(\mathbf{x})$ because, given any two booleans $b_1, b_2 \in Bool(\mathcal{R}^{\mathsf{OR}})(\mathbf{x})$, the positive set of $b_1 \nabla b_2$ cannot necessarily be reduced to a disjoint union of basic sets, as we will see in §4.3.

4.2. Structure Theorem for $While(\mathcal{R})$ semicomputability.

Theorem 1 (Structure Theorem for *While*(\mathcal{R})). For subsets of \mathbb{R}^2 ,

 $While(\mathcal{R}) \ s/comp \iff union \ of \ disjoint \ eff. \ seq. \ of \ basic \ sets.$

Proof. For the ' \Longrightarrow ' direction: If $R \subseteq \mathbb{R}^2$ is **While**(\mathcal{R}) semicomputable, then by Engeler's Lemma (Lemma 3.5), for all $x \in \mathbb{R}^2$,

$$x \in R \iff \bigvee_{k=0}^{\infty} b_k[x]$$

for some semantically disjoint effective sequence (b_k) of Σ -booleans in **Bool**(\mathcal{R})(\mathbf{x}).¹² By the Partition Lemma, each b_k defines a finite union of disjoint basic sets. Also since (b_k) is semantically disjoint, the positive sets for different b_k 's are disjoint.

Hence (b_k) is a disjoint effective sequence of basic sets as desired.

¹²Recall Remark 3.4.

For the ' \Leftarrow ' direction: Given an effective encoding of basic sets (we write $\lceil B \rceil$ for the code of B), there is a **While**(\mathcal{R}) computable function

$$in$$
: nat $imes$ real² $ightarrow$ bool

such that for any basic set B,

$$\boldsymbol{in}(\ulcornerB\urcorner, x_1, x_2) = \begin{cases} \boldsymbol{tt} & \text{if } (x_1, x_2) \in B\\ \boldsymbol{ff} & \text{if } (x_1, x_2) \notin \overline{B}\\ \uparrow & \text{o/w, i.e. } (x_1, x_2) \text{ is on the boundary of } B \end{cases}$$

where \overline{B} is the closure of B.

This is clear from the definition of basic sets.

Then the disjoint effective sequence (B_i) of basic sets gives us a total recursive function $f: \mathbb{N} \to \mathbb{N}$ such that f(n) is the code of the *n*th basic set. Hence the countable union of (B_i) is the halting set of the **While**(\mathcal{R}) procedure

proc in x_1, x_2 : real; aux i nat;			
begin			
i := 0;			
while $not(\boldsymbol{in}(f(\mathtt{i}), \mathtt{x}_1, \mathtt{x}_2))$			
do i := i + 1 od			
end			

Remark 4.5. An incomplete version of our structure theorem for $While(\mathcal{R})$ semicomputability was given in [Fu14, §4.6]; for subsets of \mathbb{R}^2 :

 $While(\mathcal{R})$ s/comp \implies union of disjoint eff. seq. of *finite unions* of basic sets

 $While(\mathcal{R})$ s/comp \Leftarrow union of disjoint eff. seq. of basic sets

4.3. Failure of closure of $While(\mathcal{R})$ semicomputable sets under union. For *total* standard algebras, we have the following proposition [TZ00, §5.2], [TZ15, §6.1]:

Proposition 4.6 (Closure of *While* semicomputable sets under union for total standard algebras). For any total standard algebra A, the class of *While*(A) semicomputable sets is closed under finite unions.

We now use the Structure Theorem for $While(\mathcal{R})$ (Theorem 1) to give a counterexample to the closure of semicomputable sets under finite union in our *partial* algebra on the reals.

This follows simply from:

Example 4.7 (A union of two basic sets which is not basic). Consider the overlapping basic sets (cf. Examples 4.3):

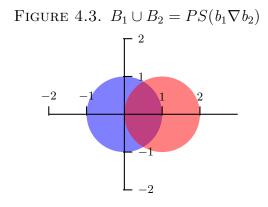
$$B_1 = \{ (x_1, x_2) \mid -x_1^2 - x_2^2 + 1 > 0 \},\$$

$$B_2 = \{ (x_1, x_2) \mid -(x_1 - 1)^2 - x_2^2 + 1 > 0 \}$$

Their union is clearly semi-algebraic, but not basic.

This follows from a more general result [ABR96]: if the boundaries of two semi-algebraic subsets of \mathbb{R}^2 intersect transversally at some point, then their union is never basic¹³.

Theorem 2 (Failure of closure of $While(\mathcal{R})$ semicomputable sets under union). The class of $While(\mathcal{R})$ semicomputable sets on \mathbb{R}^2 is not closed under finite unions.



Proof. Recall the sets B_1 and B_2 from Example 4.7. Consider $B_1 \cup B_2 = PS(b_1 \nabla b_2)$, pictured in Figure 4.3. It is a union of two semicomputable sets (pictured in Figure 4.1). If it is semicomputable, then by the Structure Theorem for **While**(\mathcal{R}) it is a union of a disjoint effective sequence of basic sets. However, since it is open and connected, it must in fact be equal to a single basic set. This contradicts the conclusion of Example 4.7.

Note that with respect to $While^{\mathsf{OR}}(\mathcal{R})$ and $While^{\exists \mathsf{N}}(\mathcal{R})$, the set $PS(b_1 \nabla b_2)$ is trivially semicomputable (cf. Remark 4.4).

5. Classes of subsets of \mathcal{R} semicomputable sets by models based on the *While* language

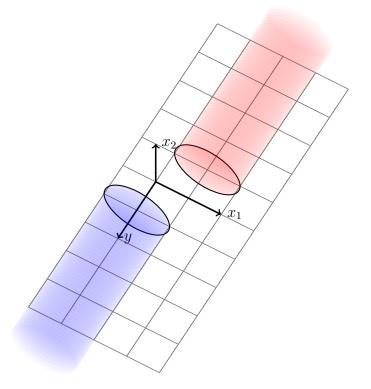
In this section we consider the equivalence or inequivalence of the classes of $While(\mathcal{R})$, $While^{OR}(\mathcal{R})$ and $While^{\exists N}(\mathcal{R})$ semicomputable sets, as well as the projectively semicomputable sets for these languages.

 $^{^{13}}$ We thank Professor Bröcker (Münster) for pointing this out (personal communication).

5.1. A set which is projectively $While(\mathcal{R})$ semicomputable but not $While(\mathcal{R})$ semicomputable.

We will show, by means of an example, that the concept of *projective* $While(\mathcal{R})$ semicomputability is strictly broader than $While(\mathcal{R})$ semicomputability on \mathbb{R}^2 .

FIGURE 5.1. Domain of $f_0(x_1, x_2, y)$.



Example 5.1. Consider the three-dimensional function $f_0 \colon \mathbb{R}^3 \to \mathbb{B}$, where:

$$f_0(x_1, x_2, y) \simeq \begin{cases} \mathbf{t} & \text{if } y > 1 \ \land \ x_1^2 + x_2^2 < 1 \\ \mathbf{t} & \text{if } y < -1 \ \land \ (x_1 - 1)^2 + x_2^2 < 1 \\ \uparrow & \text{o/w} \end{cases}$$

the domain of which is pictured in Figure 5.1.

The domain of f_0 is easily seen to be $While(\mathcal{R})$ semicomputable, and so its projection off the third argument:

 $\{(x_1, x_2) \mid \exists y \in \mathbb{R}, (y > 1 \land x_1^2 + x_2^2 < 1) \nabla (y < -1 \land (x_1 - 1)^2 + x_2^2 < 1)\}$ is projectively **While**(\mathcal{R}) semicomputable.

We have met this set previously in Figure 4.3, and we have seen that it is not $While(\mathcal{R})$ semicomputable (in the proof of Theorem 2), as it is not a union of disjoint basic sets.

From this example, we have:

Theorem 3. For subsets of \mathbb{R}^2 ,

 $While(\mathcal{R}) \ s/comp \ \rightleftharpoons \ proj-While(\mathcal{R}) \ s/comp$

5.2. Inequivalence of $While(\mathcal{R})$, $While^{OR}(\mathcal{R})$ and $While^{\exists N}(\mathcal{R})$ semicomputability.

In [Fu14, §4.6], structure theorems for $While^{\acute{OR}}(\mathcal{R})$ and $While^{\acute{IN}}(\mathcal{R})$ semicomputability over \mathbb{R}^2 were given. Along with the structure theorem for $While(\mathcal{R})$ given in §4.2, this gives us the following:

Semicomputability Structure Theorems for \mathbb{R}^2 . For subsets of \mathbb{R}^2 ,

 $While(\mathcal{R}) \ s/comp \iff union \ of \ disj. \ eff. \ seq. \ of \ basic \ sets$ $While^{OR}(\mathcal{R}) \ s/comp \iff union \ of \ disj. \ eff. \ seq. \ of \ finite \ unions \ of \ basic \ sets$ $While^{\exists N}(\mathcal{R}) \ s/comp \iff union \ of \ eff. \ seq. \ of \ basic \ sets.$

The above three structure theorems will be used (in Theorems 4 and 5 below) to show the inequivalence between $While(\mathcal{R})$, $While^{\mathsf{OR}}(\mathcal{R})$ and $While^{\exists\mathsf{N}}(\mathcal{R})$ semicomputability on \mathbb{R}^2 . This will be done by providing two examples: a subset of \mathbb{R}^2 which is $While^{\mathsf{OR}}(\mathcal{R})$ but not $While(\mathcal{R})$ semicomputable, and one which is $While^{\exists\mathsf{N}}(\mathcal{R})$ but not $While^{\mathsf{OR}}(\mathcal{R})$ semicomputable.

For the first example, we return to our working example, $B_1 \cup B_2$, pictured in Figure 4.3. As discussed, this set is not $While(\mathcal{R})$ semicomputable. However, from its definition and from the $While^{OR}(\mathcal{R})$ structure theorem, it is clearly $While^{OR}(\mathcal{R})$ semicomputable. So we have:

Theorem 4. For subsets of \mathbb{R}^2 , $While(\mathcal{R}) \ s/comp \implies While^{\mathsf{OR}}(\mathcal{R}) \ s/comp$

For the second example, consider a sequence of polynomials

$$p_i \equiv (\mathbf{x}_1 - 2i)^2 + \mathbf{x}_2^2 < 1$$

and booleans

$$b_i \equiv p_i(\mathbf{x}_1, \mathbf{x}_2) > 0$$

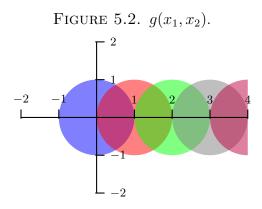
(i = 0, 1, 2, ...). Let

$$B_i = PS(b_i) = \{ (x_1, x_2) \in \mathbb{R}^2 \mid p_i(x_1, x_2) > 0 \}.$$

and let

(5.2)
$$B = \bigcup_{i} B_{i} = \{(x_{1}, x_{2}) \mid p_{i}(x_{1}, x_{2}) > 0 \text{ for some } i = 0, 1, 2, ...\}.$$

B is partially pictured in Figure 5.2, for $-1 \le x_1 \le 4$.



 (B_i) is an effective sequence of basic sets, and so their union B is $While^{\exists N}(\mathcal{R})$ semicomputable, by the $While^{\exists N}(\mathcal{R})$ structure theorem.

Now consider whether *B* is *While*^{OR}(\mathcal{R}) semicomputable. By the *While*^{OR}(\mathcal{R}) structure theorem, that would mean *B* is a union of a *disjoint sequence* of finite unions of basic sets. On the other hand, because *B* is connected, it must be a (single) finite union of basic sets, and (hence) a single *semi-algebraic set*. But that cannot be the case, for in (5.2), by substituting 0.9 for \mathbf{x}_2 , we get the 1-dimensional "slice" of *B*:

 $\{x \in \mathbb{R} \mid p_i(x, 0.9) > 0 \text{ for some } i = 0, 1, 2, ...\}\}.$

If *B* were semi-algebraic, then this set would also be semi-algebraic. However, this set is a union of infinitely many disjoint intervals (the intervals on which the horizontal line at $\mathbf{x}_2 = 0.9$ intersects the "tops" of the discs), and therefore cannot be semi-algebraic, since a semi-algebraic subset of \mathbb{R} can have at most finitely many components.

So B is not semi-algebraic, and therefore not $While^{OR}(\mathcal{R})$ semicomputable. Hence we have:

Theorem 5. For subsets of \mathbb{R}^2 , $While^{OR}(\mathcal{R}) \ s/comp \implies While^{\exists N}(\mathcal{R}) \ s/comp.$

5.3. Equivalence of projective $While(\mathcal{R})$ and $While^{\exists N}(\mathcal{R})$ semicomputability.

The following was proved in [XFZ15, §5.6].

Lemma 5.3. For subsets of \mathbb{R}^2 , proj-While^{$\exists N$}(\mathcal{R}) s/comp \iff While^{$\exists N$}(\mathcal{R}) s/comp.

Essentially, this involves replacing a projection onto a real plane, i.e. existential quantification over \mathbb{R} , by existential quantification over a countable dense subset $\mathbb{Q} \subset \mathbb{R}$, by *continuity* considerations, and (hence, by coding rationals as naturals) by existential quantification over \mathbb{N} .

We want to show further:

Lemma 5.4. For subsets of \mathbb{R}^2 ,

proj-
$$While(\mathcal{R}) \ s/comp \iff While^{\exists \mathbb{N}}(\mathcal{R}) \ s/comp.$$

Proof. The ' \Longrightarrow ' direction is obvious.

For the ' \Leftarrow ' direction: Consider any $While^{\exists N}(\mathcal{R})$ semicomputable set in \mathbb{R}^2 . This is the halting set of some $While^{\exists N}$ program P over \mathbb{R}^2 .

We will construct a $While(\mathcal{R})$ program P_0 over \mathbb{R}^3 such that the projection of the halting set of P_0 off \mathbb{R} is equal to the halting set of P.

We construct P_0 from P by replacing each statement of the form:

$$\mathbf{x}^{\mathsf{B}} := \exists \mathbf{n} \ Q(t, \mathbf{n})$$

by the two statements:

$$\mathbf{x}^{\mathsf{B}} := Q(t, \mathsf{item}[\mathsf{floor}(z), \mathbf{i}]); \\ \mathbf{i} := \mathbf{i} + 1$$

where z: real is the new argument for P_0 , i : nat is a new auxiliary variable which is initialized to 0 at the start of the program, item is a function for fetching the *i*th item from a list of naturals encoded as a single natural, and floor: $\mathbb{R} \to \mathbb{N}$ is defined as

$$\mathsf{floor}(z) \simeq \begin{cases} \text{the greatest } n \in \mathbb{N} \text{ s.t. } n < z & \text{if } z \in \mathbb{R} \backslash \mathbb{N} \\ \uparrow & \text{o/w} \end{cases}$$

which is easily seens to be $While(\mathcal{R})$ semicomputable.

Then suppose that for some input values $x_1, x_2 \in \mathbb{R}^2$, $P(x_1, x_2)$ halts. Then since P halted in finitely many steps, there exists a finite list of natural numbers $i_1, ..., i_n$ which are existentially quantified corresponding to the ' $\mathbf{x}^{\mathsf{B}} := \exists \mathbf{n}Q(t, \mathbf{n})$ ' nodes in the halting branch of the computation tree for P.¹⁴ This gives a list of naturals which may be encoded as a single natural i. Then if i < z < i + 1, $P_0(x_1, x_2, z)$ halts¹⁵. So the set $While^{\exists \mathsf{N}}(\mathcal{R})$ semicomputed by P is also a projection $\{(x_1, x_2) \mid \exists P_0(x_1, x_2, z)\}$ of the $While(\mathcal{R})$ semicomputable set P_0 .

¹⁴See [TZ00, TZ15] for information about computation trees.

¹⁵Note that the order of the naturals used in the existential quantification steps may have no relation to the order of the $x^{B} := \exists n \ P(t, n)$ lines in the code, due to loops and branches.

5.4. Classes of sets semicomputable by models based on the *While* language.

We now compare the classes of subsets of \mathbb{R}^2 semicomputable by the *While*, *While*^{OR} and *While*^{$\exists N$} languages, and their projective versions.

We begin by combining the results discussed in the previous subsection:

Theorem 6. For subsets of \mathbb{R}^2 ,

$$proj-While(\mathcal{R}) \ s/comp \iff proj-While^{\mathsf{OR}}(\mathcal{R}) \ s/comp$$
$$\iff proj-While^{\exists\mathsf{N}}(\mathcal{R}) \ s/comp$$
$$\iff While^{\exists\mathsf{N}}(\mathcal{R}) \ s/comp$$

Proof. This follows from

We have thus established the existence of three distinct classes of subsets of \mathbb{R}^2 , as shown in the following diagram:

$$While(\mathcal{R}) \text{ s/comp}$$

$$(Theorem 4)$$

$$While^{\mathsf{OR}}(\mathcal{R}) \text{ s/comp}$$

$$(Theorem 5)$$

$$While^{\exists \mathsf{N}}(\mathcal{R}) \text{ s/comp}$$

$$f$$

$$proj-While(\mathcal{R}) \text{ s/comp}$$

$$f$$

$$proj-While^{\mathsf{OR}}(\mathcal{R}) \text{ s/comp}$$

$$f$$

$$(Theorem 6)$$

We further have the equivalence of each model in the above diagram with its respective starred version (as shown in Appendix B).

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion.

In this paper, we investigated the possible generalisation of two results from classical computability theory to the context of topological partial algebras on the reals: closure of semicomputable sets under finite union, and the equivalence of semicomputable sets to projectively (semi)computable sets. Both results were shown not to hold over \mathbb{R}^2 (Theorems 2 and 3 respectively).

In the process we also developed a Structure Theorem for $While(\mathcal{R})$ semicomputability over \mathbb{R}^2 (Theorem 1), and distinguished the classes of sets semicomputed by $While(\mathcal{R})$, $While^{\mathsf{OR}}(\mathcal{R})$ and $While^{\exists \mathsf{N}}(\mathcal{R})$ programs and their projective versions (again over \mathbb{R}^2) (§5.4).

In Appendix A, we give an adaptation of our (in)equivalence results to the 1-dimensional case over \mathbb{R} .

In Appendix B, we outline a proof of the equivalence of $While(\mathcal{R})$ with its starred version.

In Appendix C, we show that another result from classical computability theory, Post's Theorem, holds in the case of **While** computation on \mathcal{R} .

6.2. Future work.

We have compared various classes of subsets of \mathbb{R}^2 with respect to semicomputability by abstract models based on the **While** language (§5.4). Similarly, we would like to investigate concrete models of computability (§1.1), and compare them amongst themselves and with abstract models with respect to computable functions and semicomputable sets. Equivalences have been shown between various concrete and abstract models of computability¹⁶ over the reals [TZ04, TZ05, Fu14]. However, important questions remain regarding equivalences between *digital* (abstract and conrete) and *analog* models, such as suitable extensions of Shannon's GPAC [Sha41]. Some interesting results have been obtained in this direction [GC03, BCGH07, BGP17a, BGP17b, Poç17], but much remains to be done.

References

 [ABR96] C. Andradas, L. Bröcker, and J. Ruiz. Constructible Sets in Real Geometry. Springer, 1996.
 [AO91] K.R. Apt and E.-R. Olderog. Verification of Sequential and Concurrent Programs. Springer-Verlag, 1991.

 16 See §1.2

NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS 27

- [Arm15] Mark Armstrong. Notions of semicomputability in topological algebras over the reals. MSc Thesis, Department of Computing & Software, McMaster University, 2015. Archived in DSpace at http://hdl.handle.net/11375/18334.
- [BCGH07] O. Bournez, M.L. Campagnolo, D.S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions. *Journal of Complexity*, 23:317–335, 2007.
- [BGP17a] O. Bournez, D. Graça, and A. Pouly. On the functions generated by the general purpose analog computer. *Information and Computation*, 257:34–57, 2017.
- [BGP17b] O. Bournez, D. Graça, and A. Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations. *Journal of the Association for Computing Machinery*, 64(6):38:1– 38:76, 2017. DOI: 10.1145/3127496.
- [CH53] R. Courant and D. Hilbert. Methods of Mathematical Physics, Vol. II. Interscience, 1953. Translated and revised from the German edition [1937].
- [Chu36] A. Church. An unsolvable problem of elementary number theory. American Journal of Mathematics, 58:345–363, 1936.
- [dB80] J.W. de Bakker. *Mathematical Theory of Program Correctness*. Prentice Hall, 1980.
- [DF91] D.S. Dummit and R.M. Foote. Abstract Algebra. John Wiley and Sons, 1991.
- [Eng68] E. Engeler. Formal Languages: Automata and Structures. Markham, 1968.
- [Fu07] Ming Quan Fu. Models of computability of partial functions on the reals. MSc Thesis, Department of Computing & Software, McMaster University, 2007. Technical Report CAS-08-01-JZ, January 2008.
- [Fu14] Ming Quan Fu. Characterizations of Semicomputable Sets, and Computable Partial Functions, on the Real Plane. PhD Thesis, Department of Computing & Software, McMaster University, 2014. Archived in DSpace at http://hdl.handle.net/11375/16066.
- [FZ15] M.Q. Fu and J.I. Zucker. Models of computation for partial functions on the reals. Journal of Logical and Algebraic Methods in Programming, 2015.
- [GC03] D.S. Graça and J.F. Costa. Analog computers and recursive functions over the reals. Journal of Complexity, 19:644–664, 2003.
- [Grz55] A. Grzegorczyk. Computable functions. Fundamenta Mathematicae, 42:168–202, 1955.
- [Grz57] A. Grzegorczyk. On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.
- [Had52] Jacques Hadamard. Lectures on Cauchy's Problem in Linear Partial Differential Equations. Dover, 1952. Translated from the French edition [1922].
- [Had64] J. Hadamard. La Théorie des Équations aux Dérivées Partielles. Éditions Scientifiques, 1964.
- [Kle36] S.C. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112:727–742, 1936.
- [Kle52] S.C. Kleene. Introduction to Metamathematics. North Holland, 1952.
- [Lac55] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles, I, II, III. C.R. Acad. Sci. Paris, 1955. 240:2470–2480, 241:13–14,151–153.
- [PER89] M.B. Pour-El and J.I. Richards. Computability in Analysis and Physics. Springer-Verlag, 1989.
- [Poç17] D. Poças. Analog Computability in Differential Equations. PhD Thesis, Department of Mathematics & Statistics, McMaster University, 2017. In progress.
- [Sha41] C. Shannon. Mathematical theory of the differential analyser. Journal of Mathematics and Physics, 20:337–354, 1941.
- [SHT99] V. Stoltenberg-Hansen and J.V. Tucker. Concrete models of computation for topological algebras. Theoretical Computer Science, 219:347–378, 1999.

28 NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS

- [SHT03] V. Stoltenberg-Hansen and J.V. Tucker. Computable and continuous homomorphisms on metric partial algebras. *Bulletin of Symbolic Logic*, 9:299–334, 2003.
- [Tur36] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42:230–265, 1936. With correction, ibid., 43, 544–546, 1937. Reprinted in The Undecidable, M. Davis, ed., Raven Press, 1965.
- [TZ99] J.V. Tucker and J.I. Zucker. Computation by 'while' programs on topological partial algebras. Theoretical Computer Science, 219:379–420, 1999.
- [TZ00] J.V. Tucker and J.I. Zucker. Computable functions and semicomputable sets on many-sorted algebras. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 5, pages 317–523. Oxford University Press, 2000.
- [TZ04] J.V. Tucker and J.I. Zucker. Abstract versus concrete computation on metric partial algebras. ACM Transactions on Computational Logic, 5:611–668, 2004.
- [TZ05] J.V. Tucker and J.I. Zucker. Computable total functions, algebraic specifications and dynamical systems. Journal of Logic and Algebraic Programming, 62:71–108, 2005.
- [TZ15] J.V. Tucker and J.I. Zucker. Generalizing computability theory to abstract algebras. In G. Sommaruga and T. Strahm, editors, *Turing's Revolution. The Impact of his Ideas about Computability*, pages 127–160. Birkhauser/Springer Basel, 2015.
- [Wei00] K. Weihrauch. Computable Analysis: An Introduction. Springer, 2000.
- [XFZ15] Bo Xie, Ming Quan Fu, and Jeffery Zucker. Characterizations of semicomputable sets of real numbers. Journal of Logic and Algebraic Programming, 84:124–154, 2015.

Appendix A. Inequivalence results for 1 dimension

The Semicomputability Structure Theorems (§5.1) were formulated for the case n = 2 (computation on \mathbb{R}^2). As pointed out in the introduction, these theorems, as well as the inequivalences (Theorems 4, 5) generalize readily to \mathbb{R}^n for all $n \ge 2$.

For the case n = 1, these Structure Theorems also hold. However there are two problems with their formulation:

 In 1 dimension, such structure theorems are formulated most naturally and perspicuously, not in terms of basic sets, but in terms of rational or algebraic intervals¹⁷ in R:

Semicomputability Structure Theorems for \mathbb{R} [XFZ15]. For subsets of \mathbb{R} ,

 $While(\mathcal{R}) \ s/comp \implies union \ of \ eff. \ seq. \ of \ rational \ intervals$

 $While(\mathcal{R}) \ s/comp \iff union \ of \ disjoint \ eff. \ seq. \ of \ rational \ intervals$

 $While^{\mathsf{OR}}(\mathcal{R}) \ s/comp \iff union \ of \ disjoint \ eff. \ seq. \ of \ algebraic \ intervals$

 $While^{\exists N}(\mathcal{R}) \ s/comp \iff union \ of \ eff. \ seq. \ of \ algebraic \ intervals.$

The relationship between these results and the formulations in terms of basic sets is unclear. It is not even obvious how to test, given an algebraic interval, whether it is also a basic set (or even a finite union of basic sets) or not.

 $^{^{17}}$ i.e. open intervals with rational or algebraic end-points respectively

(2) Furthermore, the inequivalence results (Theorems 4, 5) although holding in one dimension, require quite different proofs, using different counterexamples, in the language of algebraic intervals instead of basic sets: cf. Theorem 7, to which we now turn.

Theorem 7. For 1-dimensional computation over \mathbb{R} ,

 $While(\mathcal{R}) \ s/comp \ \rightleftharpoons \ While^{\mathsf{OR}}(\mathcal{R}) \ s/comp.$

To prove Theorem 7, we need some facts about algebraic intervals and algebraic numbers.

Proposition A.1. Let α be any algebraic number. Then there exists a polynomial p of minimum degree such that $p(\alpha) = 0$ and for all polynomials $q, q(\alpha) = 0 \implies p$ is a factor of q.

In other words, the set of polynomials over \mathbb{Z} forms a principle ideal domain¹⁸.

Proposition A.2. Suppose U is a finite union of basic sets, say

$$U = \{ x \in \mathbb{R} \mid \bigvee_{i=1}^{k} \bigwedge_{j=1}^{l_{i}} (p_{ij}(x) > 0) \},\$$

so that U has the form $\bigcup_{i=1}^{n} I_i$, where the I_i 's are disjoint algebraic intervals.

If α is an endpoint of any I_i , then α is a root of some p_{ij} .

Proof. By induction on the construction of $\bigvee_{i=1}^k \bigwedge_{j=1}^l (p_{ij}(x) > 0)$.

Proof of Theorem 7.

Consider $I = (-\sqrt{2}, \infty) = \{x \mid x^2 < 2 \lor x > 0\}.$

By the Semicomputability Structure Theorem for $While^{OR}(\mathcal{R})$ over \mathbb{R} , I is $While^{OR}(\mathcal{R})$ semicomputable.

Now suppose *I* is *While*(\mathcal{R}) semicomputable. Then by the Semicomputability Structure Theorem for *While*(\mathcal{R}) (Theorem 1) over \mathbb{R}^2 , which also applies to 1-dimensional computation over \mathcal{R} , *I* must be a finite union of basic sets, i.e. of the form $\{x \mid \bigvee_{i=1}^k \bigwedge_{j=1}^{l_i} (p_{ij}(x) > 0\}$. Then by Proposition A.2, $-\sqrt{2}$ is a root of some polynomial p_{ij} , call that polynomial p.

The minimum degree polynomial for $-\sqrt{2}$ is $x^2 - 2$. Hence by Proposition A.1, $x^2 - 2$ is a factor of p, so $\sqrt{2}$ is also a root of p. But then $p(\sqrt{2}) \neq 0$, so $\sqrt{2} \notin I$.

Hence I is not $While(\mathcal{R})$ semicomputable.

We may now show, by another example, that

Theorem 8. For 1-dimensional computation over \mathbb{R} ,

 $While^{\mathsf{OR}}(\mathcal{R}) \ s/comp \ \rightleftharpoons \ While^{\exists \mathsf{N}}(\mathcal{R}) \ s/comp.$

Proof. Consider $I = (0, \pi)$.

Since I is a single interval, if it were $While^{OR}(\mathcal{R})$ semicomputable, it would have to be an algebraic interval, but it is not.

¹⁸See e.g. [DF91].

However, it is **While**^{\exists N}(\mathcal{R}) semicomputable, since it is the union of the (non-disjoint) effective sequence of intervals (I_n) where $I_n = (0, r_n)$ and (r_n) is an increasing sequence of rational numbers converging to π .

Remark A.3. It is interesting to note that the counterexample used in the proof of Theorem 8 generalises easily to the case of \mathbb{R}^2 by taking the product of the intervals (I_n) with the unit interval.

However, the counterexamples used earlier in the paper for \mathbb{R}^2 do not reduce to \mathbb{R} .

Appendix B. The equivalence of $While(\mathcal{R})$ and $While^*(\mathcal{R})$

We wish to justify our claims that the $While(\mathcal{R})$ and $While^*(\mathcal{R})$ are equivalent in terms of computing power.

A similar result was shown in [TZ00, §4] for a *total* algebra \mathcal{R}_t on the reals¹⁹, by showing that:

- (1) \mathcal{R}_t has the term evaluation property (defined below);
- (2) for any N-standard total algebra A with the term evaluation property, a universal While(A) procedure may be constructed for $While^*(A)$;
- (3) hence, for any N-standard total algebra A with the term evaluation property, $While(A) = While^*(A)$.

We may use the same technique to show that $While(\mathcal{R}) = While^*(\mathcal{R})$ (and similarly for $While^{\mathsf{OR}}$ and $While^{\exists \mathsf{N}}$ and their starred versions). Steps (2) and (3) can be easily inferred from the respective proofs for \mathcal{R}_t in [TZ00, §4], as most of the proofs of those facts involve primitive recursive operations on the syntax of the *While* language, and so the partiality of \mathcal{R} is irrelevant. In these proofs, the only step that involves semantics is the use of term evaluation to traverse a "computation tree" for the universal *While*(A) procedure for *While**(A) programs. In that step, however, if a term which is evaluated diverges, the *While**(A) program being simulated by the universal procedure would diverge as well, and so the universal procedure behaves as expected.

So we proceed with a proof of Step (1), i.e. that \mathcal{R} has the term evaluation property (cf. [TZ00, Example 4.5]).

In this proof, we must work with encodings of the syntactic expressions used in the **While** language. We assume given a family of effective numerical codings for each of the classes of syntactic expressions over Σ . We write $\lceil E \rceil$ for the code of an expression E. We assume the we can go primitive recursively from codes of expressions to codes of their immediate subexpressions and vice versa; thus, for example, $\lceil t_1 \rceil$ and $\lceil t_2 \rceil$ are primitive recursive in $\lceil t_1 + t_2 \rceil$, and conversely. In short, we can primitive recursively simulate all operations involved in processing the syntax of the programming language.

¹⁹ \mathcal{R}_t is equivalent to \mathcal{R} except for the definitions of eq^R and $less^R$, which are taken to have their standard, total meaning. It was simply called \mathcal{R} in [TZ00, §4].

For the remainder of this Appendix, let A be any \mathbb{N} -standard (possibly partial) algebra, let u and v be product types of A, let x be a u-tuple of variables, let $Tm_x(\Sigma)$ be the set of all Σ -terms with variables among x only, and for all sorts s or Σ , let $Tm_{x,s}(\Sigma)$ be the class of such terms of sort s.

We define the term evaluation function on A relative to \mathbf{x}

$$TE_{x,s}^A : Tm_{x,s}(\Sigma) \times State(A) \rightarrow A_s$$

by

 $TE_{\mathbf{x}s}^{A}(t,\sigma) \simeq [t]^{A}\sigma.$

This term evaluation function on A relative to \mathbf{x} is then *represented* by the function

$$te^{A}_{\mathbf{x},s}$$
: $\ \ \mathbf{Tm}_{\mathbf{x},s}(\varSigma) \urcorner \times A_{u} \rightharpoonup A_{s}$

defined by

$$te^A_{\mathbf{x},s}(\ulcorner t\urcorner,a) \simeq \llbracket t \rrbracket^A \sigma,$$

where σ is any state on A such that $\sigma[\mathbf{x}] = a.^{20}$

Definition B.1. An algebra A has the term evaluation property (TEP) if for all x and s, $te_{\mathbf{x},s}^{A}$ is **While**(A) computable.

Lemma B.2. \mathcal{R} has the TEP.

Proof (outline).

The definition of $te_{\mathbf{x},s}^{\mathcal{R}}$ can be given by a series of primitive recursive clauses, e.g. in the case of boolean terms:

$$\begin{aligned} \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner t_{s} \operatorname{comp}^{\mathsf{B}} r_{s}\urcorner, a) &\simeq \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},s}^{\mathcal{R}}(\ulcorner t_{s}\urcorner, a) \operatorname{comp}^{\mathsf{B}} \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},s}^{\mathcal{R}}(\ulcorner r_{s}\urcorner, a) \\ \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner \mathsf{not}^{\mathsf{B}}(b_{1})\urcorner, a) &\simeq \operatorname{not}^{\mathsf{B}}(\boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner b_{1}\urcorner, a)) \\ \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner b_{1} \operatorname{op}^{\mathsf{B}} b_{2}\urcorner, a) &\simeq \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner b_{1}\urcorner, a) \operatorname{op}^{\mathsf{B}} \boldsymbol{t}\boldsymbol{e}_{\boldsymbol{\mathrm{x}},\boldsymbol{\mathrm{bool}}}^{\mathcal{R}}(\ulcorner b_{2}\urcorner, a) \end{aligned}$$

where sort s is either nat or real, b_1 and b_2 are terms of sort bool, comp is a comparison operator on s (one of eq^N, less^N, eq^R or less^R), and op is a binary boolean operator.

Similarly for the cases $s \equiv \text{real}$ and $s \equiv \text{nat}$.

The equivalence of $While(\mathcal{R})$ and $While^*(\mathcal{R})$ now follows from the preceding discussion.

Appendix C. Post's Theorem for the partial algebra \mathcal{R}

For total standard algebras, we have the following theorem [TZ00, §5.2], [TZ15, §6.1]:

Theorem (Post's theorem for *While* semicomputability on total standard alge**bras**). For any relation R on a total standard algebra A,

R is While(A) comp $\iff R$ and R^c are While(A) s/comp.

 $^{^{20}}$ This is well defined by the Functionality Lemma [TZ00, Lemma 3.4]

32 NOTIONS OF SEMICOMPUTABILITY IN TOPOLOGICAL ALGEBRAS OVER THE REALS

For partial algebras, Post's Theorem does not always hold [Arm15]. However it holds trivially on \mathcal{R} , since semicomputable sets are open by the Structure Theorem for **While**(\mathcal{R}), and the only clopen subsets of \mathbb{R}^n ($n \geq 1$) are \mathbb{R}^n and \emptyset .