

CHARACTERIZATION OF
SEMICOMPUTABLE SETS, AND
COMPUTABLE PARTIAL FUNCTIONS, ON
THE REAL PLANE

By

MING QUAN FU, MSc.

A Thesis

Submitted to the School of Graduate Studies

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computing and Software

McMaster University

© Copyright by Ming Quan Fu, September 2014

DOCTOR OF PHILOSOPHY (2014)

McMaster University

(Computing and Software)

Hamilton, Ontario

TITLE: Characterization of semicomputable sets, and computable partial functions, on the real plane

AUTHOR: Ming Quan Fu, MSc.(McMaster University)

SUPERVISOR: Prof. Jeffery I. Zucker

Abstract

This thesis is composed of two related projects:

- (1) **Characterizations of semicomputable subsets of \mathbb{R}^2 as unions of effective infinite sequences of “elementary” open sets.**

This developed from Bo Xie’s MSc thesis (2004) on semicomputable subsets of \mathbb{R} , where the elementary sets are simply open (rational or algebraic) intervals. This is generalized to two dimensions by taking elementary sets to be basic open semialgebraic sets. Three structure theorems are derived, based on computability by various forms of the ‘while’ programming language. By means of the *cell decomposition theorem*, variants of these structure theorem are derived, with “*basic* open semialgebraic sets” replaced by “*connected* open semialgebraic sets”.

- (2) **Equivalence of computability models for partial functions on \mathbb{R}^2 .**

This developed from the research in my MSc thesis (2007) for partial functions f on \mathbb{R} , in which four models of computability (including Grzegorzcyk-Lacombe (GL)

computability, tracking computability, and multi-polynomial approximability) were shown to be equivalent, under two *global assumptions*:

(i) the domain of f is the union of an effective exhaustion of finite unions of “elementary” open sets,

(ii) f is effectively locally uniformly continuous w.r.t this exhaustion.

This thesis extends this study to functions on \mathbb{R}^2 . For functions on \mathbb{R} , the elementary sets were simply rational open intervals. For functions on \mathbb{R}^2 (as in this thesis), the appropriate elementary sets turn out to be bounded, connected basic open semialgebraic subsets of the plane.

The most interesting of the equivalence proofs is in the direction “GL comp. \Rightarrow multipolynomial approx”. Here the function f , defined on an elementary set, is first *effectively extended* to a GL-computable function on a rectangle, and then approximated by a sequence of polynomials, using an effective version of the Weierstrass theorem in 2 dimensions. The cell decomposition theorem is again used here, to justify the algorithm extending the domain of f to a rectangle.

It is conjectured that the two global conditions are satisfied by all *elementary functions* of two real variables.

Acknowledgements

I would first like to express my sincere thanks and appreciation from the bottom of my heart to my dear supervisor, Prof. Jeffery Zucker, for his insight, thoughtful guidance throughout my PhD study. I have known Prof. Zucker since 2005. He has been my supervisor during all my years of my graduate study (2005–07 for my MSc and 2010–14 for my PhD). During these years, he has given me much academic guidance and help, and I have learned a great deal from him that will be very helpful to me all my life. And I would also like to express my thanks to my supervisor’s wife, Shirley, for her constant encouragement during the period of my PhD research.

I am very grateful to the members of my PhD Committee: Prof. Jacques Carette and Prof. Ryszard Janicki, and the external examiner Dr. Jens Blanck (Swansea University), for their careful review of the draft my thesis, and very valuable comments.

Thanks also to Profs. Lou van den Dries (University of Illinois at Urbana-Champaign) and Ludwig Bröcker (University of Münster) for their invaluable assistance with numerous emails on the subject of cell decomposition.

Thanks are due to my dear parents, my mother Xiu Rong Wang, and my late father Guang Liang Fu. I agree with the idea that parents are always their children's first and important teachers. They gave me my life, and a healthy rational personality, and inspired me to read enthusiastically and cultivate my interests in history and science; and were selflessly dedicated to my education. They always gave me understanding and support through all my problems. And I will always love them with all my heart.

Thanks to my three older sisters, Ming Yue Fu, Ming Zhu Fu and Ming Xing Fu, for their companionship during my childhood years, giving me so many beautiful memories. They have always been my best friends in our many years in four countries (China, USA, Japan and Canada) with constant encouragement and support over the years. Special thanks to Ming Zhu Fu, for giving me so much support during the difficult times in my life.

Thanks to my wife, Bin Fan, whose love brightens my life. To quote Oliver Wendell Holmes, Sr.: "Love is the master key that opens the gates of happiness."

The domain of science is broad, and has a long way to go. In ten thousands years, people will probably find the science of our time quite rudimentary, but as the proverb says, "A journey of a thousand miles begins with a single step". The domain of science is amazing and splendid, and if I can contribute to it at all during my life, I'll consider myself lucky. As the poet Robert Frost said, "The woods are lovely, dark and deep, But I have promises to keep, And miles to go before I sleep."

Contents

Abstract	iii
Acknowledgements	vi
1 Introduction	1
1.1 Background	1
1.1.1 Classical computability theory	1
1.1.2 Generalized computability theory: Motivation	2
1.2 Characterizations of semicomputable subsets of \mathbb{R}^2	6
1.2.1 Background: Subsets of \mathbb{R}	6
1.2.2 Extension to subsets of \mathbb{R}^2	7
1.3 Models of computation for partial functions on \mathbb{R}^2	8
1.3.1 Background: Total and partial functions on \mathbb{R}	8
1.3.2 Extension to partial functions on \mathbb{R}^2	11
1.4 Some remarks on the use of the cell decomposition in this thesis . . .	13

1.5	Overview of the thesis	14
2	Signatures; The topological partial algebras \mathcal{R}_0 and \mathcal{R}; <i>While</i> Computability	17
2.1	Basic Concepts: Signatures and algebras.	18
2.2	Topological algebras	19
2.3	The algebras \mathcal{R}_0 and \mathcal{R} of reals	20
2.4	The algebra \mathcal{R}^*	24
2.5	<i>While</i> programming language	25
2.6	States; Semantics of the <i>While</i> language	27
2.7	<i>While</i> computability	29
3	Infinite disjunction; <i>While</i>^{OR} and <i>While</i>^{∃N}; Semantic disjointness; Engeler's Lemma	31
3.1	Introduction	32
3.2	Expanding <i>While</i> to <i>While</i> ^{OR} and <i>While</i> ^{∃N}	33
3.3	Numerical coding of syntax	38
3.4	Semantics of infinite disjunctions	39
3.5	Engeler's Lemma for <i>While</i> ^(OR) Semantic disjointness	42
3.6	Engeler's Lemma for <i>While</i> ^{∃N}	43
4	Partition Lemma; Structure theorems for semicomputable sets over	

\mathcal{R}_0 on the real plane	44
4.1 Computational equivalence of terms; Semantics of atomic booleans; Canonical form for Σ^{OR} booleans	45
4.2 Semialgebraic and basic sets	54
4.3 Partition Lemma for Σ^{OR} booleans on \mathbb{R}^2	55
4.4 Characterizations of semicomputable sets in \mathbb{R}^2	64
4.5 Unions of effective sequences of basic sets are semicomputable	66
4.6 Structure theorems for semicomputable sets over \mathcal{R}_0	68
4.7 Cell Decomposition and Finiteness Theorems	69
5 Exhaustion, Grzegorzczuk-Lacombe (GL) computability on \mathbb{R}^2, Multipolynomial approximability on \mathbb{R}^2, Equivalence Lemma	72
5.1 Elementary set; Exhaustions; local approximability and continuity	73
5.2 GL-computability	80
5.3 Multipolynomial approximability	87
5.4 Equivalence between GL-computability and multipolynomial approximability on \mathbb{R}^2	88
6 Tracking computability; <i>While</i> and <i>WhileCC</i> approximability; Equivalence Theorem on \mathbb{R}^2; Conclusion and future work	115
6.1 Tracking computability.	116

6.2	<i>While</i> programming with countable choice	119
6.3	<i>While</i> [*] and <i>WhileCC</i> [*] computability and approximability	120
6.4	Comparison with concrete computability; Equivalence Theorem	122
6.5	Conclusion	124
6.6	Some ideas and conjectures about future work	127

Chapter 1

Introduction

1.1 Background

1.1.1 Classical computability theory

We give a brief introduction on classical computation theory.

This was developed by Church, Turing and Kleene in 1930's. Alan Turing used the formalism of Turing machines, Alonzo Church that of λ -calculus, and Stephen Kleene that of μ -recursion functions.

These three formalisms were all intended to 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).

They have all been shown to be equivalent, in the sense that they all generate the same set of computable functions on \mathbb{N} (or Σ^*).

Some important theorems in classical computability theory are:

- (1) Universal Function Theorem (Turing),
- (2) Recursion Theorem (Kleene),
- (3) Post's Theorem.

1.1.2 Generalized computability theory: Motivation

A number of generalizations of classical computability theory to abstract structures have been devised, especially to the domain of real numbers \mathbb{R} .

The first question is: Why?

One reason is clear: Scientific computation is done largely on the reals; so it is clearly desirable to try to apply the techniques of classical computability theory to the set \mathbb{R} of real numbers.

One important difference between \mathbb{R} and \mathbb{N} is that the elements of \mathbb{R} , i.e, real points, are (unlike elements of \mathbb{N} or Σ^*) “infinite objects”; for example, a real number is, in general, specified by an infinite sequence of approximating rationals. This means that, in computing with reals, we must work with “finite approximations” of reals.

Further, real numbers have a *metric* or *topological structure*, giving rise to concepts of “nearness”, and hence of closeness of approximations. Thus the topology of the reals is a crucial concept in computation on the reals, as will be seen.

Different models of computation are used to define functions and sets on data structures such as the reals. More precisely, a *model of computation* is a mathematical

model of some general method of computing functions, or deciding membership of sets

There are two main kinds of such models: abstract and concrete [TZ00,TZ04,TZ05].

Roughly speaking, in an *abstract model* of computation on the reals (let us say), the data are taken as primitives (so to say), i.e., just as “points”, independently of any representation. Two many-sorted algebras over the reals are constructed: a “field algebra” \mathcal{R} and a “ring algebra” \mathcal{R}_0 . Computations are performed there.

Examples of abstract models are: (1) a high level programming language (e.g. the ‘**While**’ language and extensions) over \mathcal{R} , and (2) recursion schemes given in (say) a typed lambda calculus. We will be working with the **While** model(s).

In a *concrete model*, by contrast, the data are given by *representations*. For example, reals may have finite representations by (indices of) effective Cauchy sequences, and the computations will, in general, depend on these representations. Examples of such models are: tracking computability, Grzegorzczuk-Lacombe computability and Weihrauch’s Type 2 computability.

These models will be described in more detail below.

This thesis is concerned with computation of partial functions on the reals. Briefly, it extends previous work on computable functions on the *real line* \mathbb{R} to computable functions on the *Euclidean plane* \mathbb{R}^2 .

There are two main topics. The first extends previous work (contained in Bo Xie’s

MSc Thesis [Xie04, XFZ13]) on characterizations of semicomputable sets on \mathbb{R} (based on **While** type languages) [Xie04, XFZ13] to such characterizations on \mathbb{R}^2 .

The second extends previous work (contained in my MSc Thesis [Fu07, FZ14]) on equivalence proofs for various models of computation for partial functions on \mathbb{R} [Fu07, FZ14] to the case of functions on \mathbb{R}^2 .

In Sections 1.2 and 1.3 we will discuss each of these topics in turn. Here we give a brief survey of each.

Topic 1 deals with the question:

Give a topological characterization of semicomputable subsets of the plane,

where a *semicomputable set* is the domain of a computable partial function.

In [XFZ13] this problem was posed for semicomputable subsets of the real line \mathbb{R} , using three different versions of the **While** language, leading to three corresponding solutions, which can all be summarized as

a set is semicomputable iff it is an effective countable union of “elementary sets”.

Here an “elementary set” here is simply an interval with either rational or algebraic end-points. (An exact formulation will be given in Section 1.2.)

In trying to generalize this to two dimensions, the main problem was to find the appropriate concept of “elementary” subset of the plane. This was found to be: *basic open semialgebraic set*, as detailed in Section 1.2.

Turning to topic 2: here we have a number of models of computation (abstract

and concrete) for partial functions f on the plane, and aims to prove their equivalence under certain topological *global assumptions*, to be determined.

There are actually two precursors to this research project:

- (i) In [TZ05] these equivalences were proved for *total* functions on \mathbb{R} . The global assumption needed there was: *effective locally uniform continuity*.
- (ii) In [FZ14] we considered *partial* functions f on \mathbb{R} . The global assumptions needed here to prove equivalence were, apart from continuity, a *domain exhaustion* assumption:

the domain U of f is a union of an *effective exhaustion*, i.e., an expanding sequence of finite unions of “elementary sets”.

Here the “elementary sets” were simply rational open intervals.

In trying to generalize these results to two dimensions, with essentially the same global assumptions, the main problem was, again, to find an appropriate concept of “elementary” subset of the plane. This was found to be: *bounded connected open semialgebraic set*, as detailed in Section 1.3. Note that this is similar, but not identical, to the definition of “elementary set” in topic 1.

Note also that each of these topics is concerned with the (non-trivial) generalization of topological properties of computable partial functions¹ $f : \mathbb{R}^m \multimap \mathbb{R}$ from the case $m = 1$ to $m > 1$. For each topic, we explicitly describe the case $m = 2$ only.

¹‘ \multimap ’ denotes a partial function.

This decision was taken to avoid an unmanageable welter of super- and subscripts.

This generalization to $m = 2$ was the big step, and it would be relatively straightforward to generalize this study further from $m = 2$ to $m > 2$.

1.2 Characterizations of semicomputable subsets of \mathbb{R}^2

1.2.1 Background: Subsets of \mathbb{R}

This is a development of work in the M.Sc. thesis of Bo Xie [Xie04, XFZ13] on characterizations of semialgebraic subsets of \mathbb{R} .

Here one works with a partial many-sorted algebra \mathcal{R} over \mathbb{R} , and semicomputability with respect to certain extensions of the *While* programming language over \mathcal{R} . These are *While*^{OR} and *While*^{∃N}, which extend *While* by (respectively) the operators of strong (boolean) disjunction, and existential quantification over \mathbb{N} .

The main tools used there were

- (1) *Engeler's Lemma* (explained in Chapter 3) derived from an analysis of computation trees, and
- (2) *the Partition Lemma*, characterizing the semantics of boolean expressions over \mathbb{R} in terms of a partition of \mathbb{R} into positive, negative and divergent sets, composed of finitely many algebraic open intervals and points.

1.2.2 Extension to subsets of \mathbb{R}^2

In extending the results in § 1.2.1 to subsets of \mathbb{R}^2 , Engeler’s Lemma is easily seen to generalize. However the generalization of the Partition Lemma to two dimensions – even its formulation – is more challenging. This was accomplished by working with \mathcal{R}_0 instead of \mathcal{R} , and replacing the intervals in \mathbb{R} by *basic open semialgebraic sets*, i.e. sets of the form

$$\{x \in \mathbb{R}^2 \mid p_1(x) > 0, \dots, p_k(x) > 0\} \quad (k > 0) \quad (1.1)$$

where p_1, \dots, p_k are polynomials over \mathbb{Z} . In the sequel (Sections 4 and 5) we will use “basic sets” to mean “basic open semialgebraic sets”.

In this way we derive two Structure Theorems for **While**^{OR} and **While**^{∃N} semi-computability over \mathcal{R}_0 :

Theorem 1. A subset U of \mathbb{R}^2 is **While**^{OR} semicomputable, iff U is a countable union of a disjoint effective sequence of finite unions of basic sets.

Theorem 2. A subset U of \mathbb{R}^2 is **While**^{∃N} semicomputable, iff U is a countable union of an effective sequence of basic sets.

– and a “partial structure theorem” for **While** semicomputability:

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

- (a) **While** semicomputable \implies union of *disjoint* effective sequence of finite

union of basic sets;

(b) union of *disjoint* effective sequence of basic sets \implies **While** semicomputable.

The use of the algebra \mathcal{R}_0 instead of \mathcal{R} in topic 1 is explained in Discussion 6.5.1.

1.3 Models of computation for partial functions on

\mathbb{R}^2

1.3.1 Background: Total and partial functions on \mathbb{R}

This topic was developed from [TZ05, Fu07, FZ14]. In [TZ05] five models of computation for total functions on the reals were investigated, and all five were shown to be equivalent for functions $f : \mathbb{R}^m \rightarrow \mathbb{R}$ that are (a) *total* or, more generally, defined on a product of intervals, and (b) *effectively locally uniformly continuous*.

The equivalence was proved for the following models:

- (i) Grzegorzcyk-Lacombe (GL) computability,
- (ii) tracking computability,
- (iii) effective locally uniform (\mathbb{Q} -)polynomial approximability,
- (iv) **WhileCC** approximability on a partial topological algebra \mathcal{R} on \mathbb{R} ,
- (v) local uniform **While** approximability on a total topological algebra \mathcal{R}_t on \mathbb{R} .

The first two of these are concrete, the last two are abstract, and (iii) is a ‘hybrid’ model.

First, a brief explanation of these models. (More detailed descriptions will be given below.) Model (i) and (ii) are well known concrete models [PER89][TZ05]. (ii) uses a “tracking function” on \mathbb{N} according to a standard enumeration α of the rationals, and (hence) an enumeration $\bar{\alpha}$ of the computable reals. (iv) and (v) are abstract models of computability, based on versions of the **While** programming language. **WhileCC** is a nondeterministic extension of **While** which incorporates *countable choice*, *i.e.*, nondeterministic choice of a natural number satisfying a given predicate. \mathcal{R} and \mathcal{R}_t are both topological algebras on \mathbb{R} : \mathcal{R} is a partial algebra, which includes partial equality, order and inverse operations on the reals as basic functions, and \mathcal{R}_t is a total algebra, without these partial operations (but with the inverse of naturals).

In [TZ05] all five models of computability were shown to be equivalent, for functions $f : \mathbb{R}^m \rightarrow \mathbb{R}$ ($m > 0$) that are (a) *total* or, more generally, defined on a closed interval (or product of intervals, in the case $m > 1$) and (b) *effectively locally uniformly continuous*.

In [Fu07, FZ14], this Equivalence Theorems was generalized to the case of *partial* $f : \mathbb{R} \rightarrow \mathbb{R}$. Because of complications associated with partiality, two *global assumptions* were made on f :

- (a) the domain U of f is a union of an *effective open exhaustion*, *i.e.*, a sequence of *stages* (U_0, U_1, U_2, \dots) , where² for $\ell = 0, 1, 2, \dots$, $\overline{U_\ell} \subseteq U_{\ell+1}$ and U_ℓ is

² \overline{A} denotes the topological closure of a set A .

a finite union of rational open intervals $I_1^\ell, \dots, I_{k_\ell}^\ell$ with disjoint closures, the *components* of the stage U_ℓ ; and

(b) f is effectively locally uniformly continuous with respect to this exhaustion.

So the “totality” assumption of [TZ05] was replaced by a more general “domain exhaustion” assumption.) It should be noted that these two assumptions hold for all the well-known unary functions of elementary real analysis, for example, $f(x) = \cot(\pi x)$, which has domain $U = \mathbb{R} \setminus \mathbb{N}$.

In fact, it was conjectured in [FZ14] that these assumptions seem to hold for all unary *elementary functions* on \mathbb{R} , where an elementary function on \mathbb{R} is defined as any function denoted by an expression built up from the variable x and constants for computable reals, by repeated application of the four field operations, n -th roots, the exponential and trigonometric functions and their inverses.

Since $\mathbf{dom}(f)$ is (in general) no longer *connected* as a subspace of \mathbb{R} , some of the arguments used in [TZ05] to prove the equivalences listed above are invalid, or at least complicated. We list two significant issues:

- (1) The proof of equivalence of **While**(\mathcal{R}_t) approximability with the other four models listed above fails. Hence this model is left out of further consideration.
- (2) *Polynomial approximability* must be replaced by *multipolynomial approximability* in which each multipolynomial approximant q_ℓ is the union of a tuple of

polynomials $(p_1^\ell, \dots, p_{k_\ell}^\ell)$, where we take $\mathbf{dom}(p_i^\ell) = \overline{I_i^\ell}$, the closure of the i -th component of the stage U_ℓ ($i = 1, \dots, k_\ell$).

1.3.2 Extension to partial functions on \mathbb{R}^2

The most interesting topic in this thesis, we feel, is the generalization of the Equivalence Theorem for models (i) , . . . , (iv) above to the case of partial functions $f : \mathbb{R}^m \rightarrow \mathbb{R}$ for $m > 1$. (As stated above, we only give an explicit exposition for the case $m = 2$.)

To repeat the models investigated:

- (i) Grzegorzcyk-Lacombe (GL) computability,
- (ii) tracking computability,
- (iii) effective locally uniform multipolynomial approximability,
- (iv) **WhileCC** approximability on a partial topological algebra \mathcal{R} .

Their equivalence is proved (as in [FZ14]) by means of three Equivalence Lemmas: Equivalence Lemma 1, 2 and 3.

These are proved under (versions of) the two *global assumptions* used for $m = 1$ (see Section 1.2.2): *domain exhaustion* and *continuity*.

The most interesting of these equivalences is Equivalence Lemma 1, since it is here that the difficulties in generalizing these results to $m > 1$ are most apparent. There are two main problems here.

The first problem is that the direction $(iii) \Rightarrow (i)$, (multipolynomial approximability \Rightarrow GL computability) was proved in [FZ14] by connecting the components of the multipolynomial approximants by *linear interpolation*, so as to construct the GL-computable approximants to f , and then using a lemma about closure of GL-computable functions under uniform limits.

For $m > 1$, it is not clear how one can apply interpolation for the analogous step. The solution here is to extend the GL-closure lemma (5.2.7) to apply to approximants f_n defined not on U , but on U_n .

The second problem in generalizing from $m = 1$ to $m = 2$ is: even the definition of “effective exhaustion” becomes unclear. In one dimension a stage U_ℓ is a finite union of finite open rational intervals with disjoint closures. How is this to be generalized?

The solution is to define a stage U_ℓ as a finite union of *elementary sets* E_i^ℓ , where an elementary set is a bounded, connected basic set (see § 1.2.2) above), where (in (1.1) p_1, \dots, p_n are now polynomials over the set \mathbb{R}_c of computable reals.

It is still necessary to extend the function f continuously from the closure of each elementary set to a containing rectangle K . This is done by our *Extension Theorem*, which was quite challenging (Theorem 4).

The reason for this extension is to be able to apply a Weierstrass -type theorem for polynomial approximations on K . In [FZ14] this step involved a straightforward adaptation of the effective Weierstrass theorem given in [PER89, Ch.0, Sec 7]. Here

(with intervals replaced by rectangles) the proof of this theorem provided another challenge, which was solved by the construction of an appropriate 2-dimensional pulse function (Theorem 5). Details of the proof of Equivalence Lemma 1 are given in Chapter 5.

1.4 Some remarks on the use of the cell decomposition in this thesis

The cell decomposition theorem, applied to semialgebraic sets on \mathbb{R}^2 , was used for the purpose of obtaining a number of “finiteness theorems” in two distinct locations in the thesis:

- (1) In topic 1 (Chapter 4), for the purpose of formulating alternative versions of the structure theorems, formed by replacing *basic sets* by *connected open semialgebraic sets* (Section 4.7); and
- (2) in topic 2 (Chapter 5), in connection with the proof of the Extension Theorem (see Discussion 5.4.3).

Interestingly, the family of semialgebraic sets used is based on polynomials over the integers in topic 1, and polynomials over the computable reals in topic 2.

The subject of CAD (cylindrical algebra decomposition)[BPM06] is, essentially, the practical application of the theory of cell decomposition for semialgebraic sets.

I am grateful to Prof. Jacques Carette for originally suggesting the investigation of

CAD in connection with the topics of my thesis, and to Dr Martin Bays for introducing me to cell decomposition, and recommending Lou van den Dries’s book [vdD98].

Finally, I am very grateful to Profs. Lou van den Dries and Ludwig Bröcker for their invaluable assistance with numerous emails on this subject.

1.5 Overview of the thesis

In Chapter 2 we review abstract many-sorted signatures and algebras, topological partial algebras and, in particular, the topological partial algebra \mathcal{R}_0 and \mathcal{R} over \mathbb{R} (with respectively the ring and field structures of the reals), and the **While** programming language over \mathcal{R} .

In our first main topic, working with the ring algebra \mathcal{R}_0 , we give characterizations of **While**^{OR} and **While**^{∃N} semicomputable subsets of \mathbb{R}^2 , This is covered in Chapters 3 and 4. Our second main topic, the equivalence of various models of computability for partial functions on \mathbb{R}^2 , is covered in Chapters 5 and 6. In more detail:

In Chapter 3, the semantics of infinite disjunctions is discussed, the **While**^{OR} and **While**^{∃N} language are defined, and the concept of semantic disjointness, and Engeler’s Lemma for the **While**^{OR} and **While**^{∃N} languages, are presented.

Chapter 4 presents the Partition Lemma for boolean expressions over \mathbb{R}^2 , and hence the two Structure Theorems for **While**^{OR}(\mathcal{R}_0) and **While**^{∃N}(\mathcal{R}_0) semicomputable subsets of \mathbb{R}^2 (and a “partial structure theorem” for **While** semicomputable

sets).

We introduce the concepts of *semialgebraic* and *basic* subsets of \mathbb{R}^2 , which are fundamental for this purpose. In addition, we discuss “finiteness theorems” based on *cell decomposition*, and use these to present alternative versions of the Structure Theorems.

In Chapters 5 and 6, we work with the full “field algebra” \mathcal{R} , and consider the four models of computation and prove their equivalence under the global assumptions. In Chapter 5 we first define the concepts of *elementary set* and *effective exhaustion*, and then state our global assumptions (domain exhaustion and continuity) on functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Next we define two of the four models: *GL-computability* and *effective locally uniform multipolynomial approximability*, and prove their equivalence under the global assumptions (Equivalence Lemma 1).

In Chapter 6 we define the language **WhileCC** and the two remaining models: *$\bar{\alpha}$ -tracking computability* and **WhileCC** *approximability*. The remaining two equivalence Lemmas:

“Tracking computability \iff **WhileCC** approximability ”

and

“GL-computability \iff tracking computability ”

are stated. They are not proved in detail, since their proofs are fairly straightforward generalizations of the corresponding proofs in [FZ14] for unary functions.

The Equivalence Theorem, linking all four models, follows.

Chapter 6 ends with a discussion on the ease of generalizing the results in this thesis to \mathbb{R}^m for $m > 2$, and ideas for future research, including (1) connecting our computation models on \mathbb{R}^2 with Weihrauch's [Wei00], and (2) investigating the conjecture that our two global assumptions hold for all elementary functions on \mathbb{R}^2 .

Chapter 2

Signatures; The topological partial algebras \mathcal{R}_0 and \mathcal{R} ; *While* Computability

In this thesis, some abstract models¹ of computability on \mathbb{R}^2 are given. To prepare for this, we discuss abstract many-sorted signatures and algebras (Section 2.1), and more specifically, topological partial algebras (Section 2.2), and in particular the topological partial algebras \mathcal{R}_0 and \mathcal{R} of reals (Section 2.3). We then describe the *While* programming language, and the concept of *While* computable functions on \mathbb{R}^2 .

The material in this chapter is in the nature of a review. It is taken largely from [TZ00, XFZ13, FZ14], and included so as to make the thesis self-contained.

¹See discussion in Section 1.1.

2.1 Basic Concepts: Signatures and algebras.

Definition 2.1.1 (Many-sorted signatures). A *many-sorted signature* Σ is a pair $\langle \mathbf{Sort}(\Sigma), \mathbf{Func}(\Sigma) \rangle$ where

(a) $\mathbf{Sort}(\Sigma)$ is a finite set of *sorts*, written s, s', \dots

(b) $\mathbf{Func}(\Sigma)$ is a finite set of (primitive or basic) *function symbols* F with

$$F : s_1 \times \cdots \times s_m \rightarrow s \quad (m \geq 0).$$

Each symbol F has a *type* $s_1 \times \cdots \times s_m \rightarrow s$ (written $F : s_1 \times \cdots \times s_m \rightarrow s$), where $m \geq 0$ is the *arity* of F . The case $m = 0$ corresponds to *constant symbols*; we then write $F : \rightarrow s$.

Definition 2.1.2 (Product types over Σ). A Σ -*product type* has the form $s_1 \times \cdots \times s_m$ ($m \geq 0$), where s_1, \dots, s_m are Σ -sorts. Product types are written u, v, \dots

Definition 2.1.3 (Σ -algebras). A Σ -*algebra* A has, for each sort s of Σ , a non-empty set A_s , called the *carrier of sort* s , and for each Σ -function symbol $F : u \rightarrow s$, a (*not necessary total*) function² $F^A : A^u \rightarrow A_s$, where

$$A^u =_{df} A_{s_1} \times \cdots \times A_{s_m}.$$

²We use ' \rightarrow ' to denote partial functions.

The algebra A is *total* if F^A is total for each Σ -function symbol F . Otherwise it is *partial*.

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

2.2 Topological algebras

Definition 2.2.1 (Continuity). Given two topological spaces X and Y , a partial function $f : X \rightarrow Y$ is *continuous* if for every open $V \subseteq Y$,

$$f^{-1}[V] =_{df} \{x \in X \mid x \in \mathbf{dom}(f) \text{ and } f(x) \in V\}$$

is open in X .

Definition 2.2.2 (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} (if present) have the discrete topology.

Remark 2.2.3 (Continuity of computable functions; the continuity principle). 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

$$\text{“} \mathbf{computability} \implies \mathbf{continuity} \text{.”}$$

This principle is discussed in [TZ99, TZ04].

2.3 The algebras \mathcal{R}_0 and \mathcal{R} of reals

In this thesis, we will work with the following algebra:

algebra	\mathcal{R}
carriers	$\mathbb{R}, \mathbb{B}, \mathbb{N}$
functions	$0^{\mathbb{R}}, 1^{\mathbb{R}} : \rightarrow \mathbb{R},$ $\text{plus}^{\mathbb{R}}, \text{times}^{\mathbb{R}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ $\text{inv}^{\mathbb{R}} : \mathbb{R} \rightarrow \mathbb{R},$ $0^{\mathbb{N}} : \rightarrow \mathbb{N},$ $\text{suc}^{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}$ $\text{tt}, \text{ff} : \rightarrow \mathbb{B},$ $\text{or}, \text{and} : \mathbb{B}^2 \rightarrow \mathbb{B},$ $\text{cor}, \text{cand} : \mathbb{B}^2 \rightarrow \mathbb{B},$ $\text{not} : \mathbb{B} \rightarrow \mathbb{B},$ $\text{eq}^{\mathbb{N}}, \text{less}^{\mathbb{N}} : \mathbb{N}^2 \rightarrow \mathbb{B}$ $\text{eq}^{\mathbb{R}}, \text{less}^{\mathbb{R}} : \mathbb{R}^2 \rightarrow \mathbb{B}$

The signature Σ of \mathcal{R} , with sorts `real`, `bool`, and `nat`, can be inferred from the above.

The topic 1 we actually work with the “ring algebra” \mathcal{R}_0 , with signature Σ_0 , which is the retract of \mathcal{R} formed by removing the inverse operator $\text{inv}^{\mathbb{R}}$.

Remarks 2.3.1.

- (1) \mathcal{R} contains three carriers: \mathbb{R} , \mathbb{N} and \mathbb{B} , of sorts `real`, `nat` and `bool` respectively.

(2) \mathcal{R} contains (as retracts) the field of reals, the naturals with 0 and successor, and the booleans with their standard operations, including equality and order on \mathbb{R} and \mathbb{N} .

(3) \mathcal{R} contains two sets of boolean operators: (1) the strict operators ‘or’ and ‘and’; and (2) the “conditional” operators ‘cor’ and ‘cand’ (denoted by ‘||’ and ‘&&’ in C-like languages), “evaluated from the left”, and non-strict in the 2nd argument.

(4) \mathcal{R} is a partial algebra, with the following basic partial functions: $\text{inv}^{\mathbb{R}}$, $\text{eq}^{\mathbb{R}}$ and $\text{less}^{\mathbb{R}}$, where for $x, y \in \mathbb{R}$:

$$\text{inv}^{\mathbb{R}}(x) \simeq \begin{cases} 1/x & \text{if } x \neq 0 \\ \uparrow & \text{otherwise.} \end{cases}$$

$$\text{eq}^{\mathbb{R}}(x, y) \simeq \begin{cases} \uparrow & \text{if } x = y \\ \text{ff} & \text{otherwise.} \end{cases}$$

and

$$\text{less}^{\mathbb{R}}(x, y) \simeq \begin{cases} \text{tt} & \text{if } x < y \\ \text{ff} & \text{if } x > y \\ \uparrow & \text{if } x = y. \end{cases}$$

By contrast, the boolean functions on \mathbb{N} : $\text{eq}^{\mathbb{N}}$ and $\text{less}^{\mathbb{N}}$, are total.

The reason for these semantic definitions will be given in Discussion 2.3.4.

Remark 2.3.2 (Terminology).

- (a) The symbol ‘ \simeq ’ denotes Kleene equality, where the two sides are either both defined and equal, or both undefined.
- (b) Above, the symbol ‘ $=$ ’ on the r.h.s means (of course) equality at the semantic level. Below we will also often write ‘ $t_1 = t_2$ ’ for both $\text{eq}^{\mathbb{R}}(t_1, t_2)$ and $\text{eq}^{\mathbb{N}}(t_1, t_2)$. Which of the three meaning is intended in any particular case will generally be clear from the context.

Remarks 2.3.3 (Standard and N-standard algebras).

- (a) The algebras \mathcal{N} and \mathcal{R} are *standard*, in the sense that they contains the carrier \mathbb{B} with the standard boolean operations. (In other words, they are expansions³ of the algebra \mathcal{B} .) Standardness of \mathcal{R} is necessary for a decent theory of computation on \mathbb{R} [TZ00, TZ05].
- (b) \mathcal{R} is also *N-standard*, in the sense that it contains the carrier \mathbb{N} with the standard arithmetic operations. (In other words, it is an expansion of \mathcal{N} .) N-standardness of \mathcal{R} is not really necessary for our main result, since the integers, and hence the naturals, can be implemented in the reals [TZ00, Prop. 6.17]. However, it is a very useful assumption.

³The concept of “expansion” is defined in [TZ00, Def.2.6].

Discussion 2.3.4 (Motivation for definition of partial functions).

We want to motivate the definitions of partial functions in general, and more specifically, the functions eq_p and less_p in \mathcal{R} . We present our motivation in two ways: the first based on continuity considerations, and the second based on a “thought experiment” concerning (concrete) computation of the basic functions under discussion.

(a) The total versions of eq_p and less_p are not continuous, as can easily be checked. (By contrast, the total functions eq_{nat} , less_{nat} on \mathcal{N} are continuous, because of the discrete topology on \mathcal{N} .) Continuity of basic functions such as eq_p and less_p in accordance with our definition of topological algebra, is important in connection with the Continuity Principle (see Remark 3.3.3).

(b) Consider now a thought experiment involving the computation of an atomic formula ‘ $\mathbf{x} = \mathbf{y}$ ’, where \mathbf{x} and \mathbf{y} are real variables. Suppose, at a particular state⁴, we want to determine whether $\mathbf{x} = \mathbf{y}$ is true at σ . Suppose also (we are now combining “abstract” and “concrete” modes of description⁵) that the values of \mathbf{x} and \mathbf{y} at σ are “given by” Cauchy sequences of rationals (r_0, r_1, r_2, \dots) and (s_0, s_1, s_2, \dots) , which (for convenience) we assume to be “fast”, *i.e.*,

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

and similarly for (s_n) . Suppose also that for $n = 1, 2, 3, \dots$ the inputs r_n and s_n are observed (from some device) at n time units. Now $\mathbf{x} < \mathbf{y}$ is true at σ iff for some

⁴States are defined in Section 2.6

⁵Recall the discussion in §1.1

n , $r_n + 2 \cdot 2^{-n} < s_n$, and this can be determined within a finite amount of time. Correspondingly, $\mathbf{x} = \mathbf{y}$ is true iff for all n , $|r_n - s_n| \leq 2 \cdot 2^{-n}$, but this cannot be determined within any finite amount of time. These considerations explain the form of the partial definitions of equality and order on the reals.

2.4 The algebra \mathcal{R}^*

The algebra \mathcal{R}^* is formed from \mathcal{R} by adding the carrier \mathbb{R}^* (of sort `real*`) consisting of all *finite sequences* or *arrays* of reals, together with certain standard constants and operations for the empty array, for updating arrays, etc.

The significance of arrays for computation is that they provide finite but unbounded memory. The reason for introducing the starred sort `real*` is the lack of effective coding of finite sequences from \mathbb{R} (unlike the case with \mathbb{N} and \mathbb{B}).

Since the use of \mathcal{R}^* is not essential in the thesis, and it plays only a heuristic role, we omit a precise definition, which can be found in [TZ99, TZ00, TZ05].

Although the use of \mathcal{R}^* is convenient for computational purposes, it is not strictly stronger than \mathcal{R} , as we will see (Section 6.3).

We will be using the topological partial algebras \mathcal{R}_0 , \mathcal{R} and \mathcal{R}^* in the rest of the thesis.

2.5 *While* programming language

Our abstract models of computation on \mathbb{R}^2 are based on the *While* programming language and certain extensions, applied to \mathcal{R} [TZ99, TZ00, TZ04, TZ05].

Let A be a standard algebra with signature Σ .

We begin with the syntax of Σ -terms. Note that we use ‘ \equiv ’ to denote syntactic identity between two expressions.

Definition 2.5.1 (Σ -variables). For each Σ -sort s , there are variables $\mathbf{x}^s, \mathbf{y}^s, \dots$ of sort s . $\mathbf{Var}_s(\Sigma)$ is the set of variables of sort s , and $\mathbf{Var}(\Sigma)$ is the set of all Σ -variables, $\mathbf{x}, \mathbf{y}, \dots$

Definition 2.5.2 (Σ -terms). $\mathbf{Tm}(\Sigma)$ is the set of Σ -terms t, \dots , and $\mathbf{Tm}_s(\Sigma)$ the set of Σ -terms of sort s (denoted t^s), defined (in modified BNF) by

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

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

We often drop the sort superscript s , and also write $t : s$ to indicate that $t \in \mathbf{Tm}_s(\Sigma)$. More generally, we write $t : u$ to indicate that t is a tuple of Σ -terms of product type u . We write \mathbf{Term}_s for $\mathbf{Term}_s(\Sigma)$, etc. We also write b, b', \dots for boolean Σ -terms, i.e. Σ -terms of sort \mathbf{bool} .

Next, we consider program statements and procedures.

Definition 2.5.3 (Statements). $\mathbf{Stmt}(\Sigma)$ is the class of Σ -statements S, \dots generated by:

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

where $\mathbf{x} := t$ denotes simultaneous assignment, i.e for some $m > 0$, $\mathbf{x} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_m)$ and $t \equiv (t_1, \dots, t_m)$ are variable and term tuples of the same product type, with the condition that $\mathbf{x}_i \neq \mathbf{x}_j$ for $i \neq j$, and b is a boolean term.

Definition 2.5.4 (Procedures). $\mathbf{Proc}(\Sigma)$ is the class of Σ -procedures $P \dots$ of the form:

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

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

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

where \mathbf{a} , \mathbf{b} and \mathbf{c} are tuples of *input variables*, *output variables* and *auxiliary variables* respectively. We stipulate:

- (i) \mathbf{a} , \mathbf{b} and \mathbf{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 \mathbf{a} , \mathbf{b} , \mathbf{c}).

If $\mathbf{a} : u$ and $\mathbf{b} : v$, then P has type $u \rightarrow v$, written $P : u \rightarrow v$.

We will write \mathbf{Tm} for $\mathbf{Tm}(\Sigma)$, \mathbf{While} for $\mathbf{While}(\Sigma)$ etc.

2.6 States; Semantics of the *While* language

Definition 2.6.1 (State).

- (a) A *state* on A is a family $\langle \sigma_s \mid s \in \mathbf{Sort}(\Sigma) \rangle$ of functions $\sigma_s : \mathbf{Var}_s \rightarrow A_s$.
- (b) $\mathbf{State}(A)$ is the set of states on A , with elements σ, \dots .

Notation 2.6.2. For $\mathbf{x} \in \mathbf{Var}_s$, we write $\sigma(\mathbf{x})$ for $\sigma_s(\mathbf{x})$. Also, for a tuple $\mathbf{x} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_m)$, we write $\sigma[\mathbf{x}]$ for $(\sigma(\mathbf{x}_1), \dots, \sigma(\mathbf{x}_m))$.

Definition 2.6.3 (Variant of a state). Let σ be a state over A , and for some Σ -product type u , let $\mathbf{x} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n) : u$ and $a = (a_1, \dots, a_n) \in A^u$ (for $n \geq 1$). We define $\sigma\{\mathbf{x}/a\}$ to be the state over A formed from σ by replacing its value at \mathbf{x}_i by a_i for $i = 1, \dots, n$. That is, for all variables y :

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

For $t \in \mathbf{Term}_s$, we will define the function

$$\llbracket t \rrbracket^A : \mathbf{State}(A) \rightarrow A_s$$

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

Notation 2.6.4

- (a) We write $\llbracket t \rrbracket^A \sigma \downarrow$ to mean that evaluation of $\llbracket t \rrbracket^A \sigma$ halts, or converges, and

$\llbracket t \rrbracket^A \sigma \downarrow a$ to mean that evaluation of $\llbracket t \rrbracket^A \sigma$ converges to a value a .

(b) We write $\llbracket t \rrbracket^A \sigma \uparrow$ to mean that evaluation of $\llbracket t \rrbracket^A \sigma$ diverges, i.e. does not halt.

Notation 2.6.5 (Kleene equality). We write e.g

$$\llbracket t_1 \rrbracket^A \sigma \simeq \llbracket t_2 \rrbracket^A \sigma$$

to mean that the two sides of the equality either both converge to the same value, or both diverge (cf. [Kle52, §63]).

Definition 2.6.6 (Semantics of terms). The definition of $\llbracket t \rrbracket^A \sigma$ is by structural induction on Σ -terms t :

$$\begin{aligned} \llbracket \mathbf{x} \rrbracket^A \sigma &= \sigma(\mathbf{x}) \\ \llbracket F(t_1, \dots, t_m) \rrbracket^A \sigma &\simeq \begin{cases} F^A(\llbracket t_1 \rrbracket^A \sigma, \dots, \llbracket t_m \rrbracket^A \sigma) & \text{if } \llbracket t_i \rrbracket^A \sigma \downarrow \text{ for } 1 \leq i \leq m \\ \uparrow & \text{otherwise} \end{cases} \end{aligned}$$

Note that if $c : \rightarrow s$, i.e. c is a constant symbol of sort s , then $\llbracket c \rrbracket^A \sigma = c^A \in A_s$

Note also that the semantics of the language (so far) is strict, based on the strict definitions of F^A for function symbols F . This will change with the definitions of “strong” boolean operator below (Section 3.2).

Definition 2.6.7 (Semantic equivalence of terms). Two Σ -terms t_1 and t_2 of the same sort s are (*semantically*) *equivalent over* A , written $t_1 \approx t_2$, iff

$$\forall \sigma \in \mathbf{State}(A) \ (\llbracket t_1 \rrbracket^A \sigma \simeq \llbracket t_2 \rrbracket^A \sigma).$$

Definition 2.6.8 (Weak semantic equivalence of booleans). Two Σ -booleans b_1 and b_2 are *weakly (semantically) equivalent over* A , written $b_1 \sim b_2$, iff

$$\forall \sigma \in \mathbf{State}(A) \left(\llbracket b_1 \rrbracket^A \sigma \downarrow \mathbf{tt} \iff \llbracket b_2 \rrbracket^A \sigma \downarrow \mathbf{tt} \right).$$

The meaning $\llbracket S \rrbracket^A$ of a **While**(Σ) statement S is a partial state transformer on an algebra A :

$$\llbracket S \rrbracket^A : \mathbf{State}(A) \rightarrow \mathbf{State}(A).$$

Its definition is standard [TZ00, §§ 3.4-3.6] and lengthy, and so we omit it here. Briefly, it is based on the definition of the *computation sequence* of S starting in a given 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 .

The semantics of procedures is defined from the semantics of statements in a standard way [TZ99, TZ00], and we omit details. So if

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

is a procedure of type $u \rightarrow v$, then its meaning is a partial function

$$P^A : A^u \rightarrow A^v$$

2.7 *While* computability

Definition 2.7.1 (*While* computable function). Let A be a standard algebra.

(a) A function $f: A^u \rightarrow A_s$ is said to be *computable* (on A) by a **While** procedure

$$P : u \rightarrow s \text{ if } f = P^A.$$

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

Definition 2.7.2 (Halting set). The *halting set* of a procedure $P : u \rightarrow v$ on A is the set

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

Definition 2.7.3 (While semicomputable set). A set $R \subseteq A^u$ is **While semicomputable** on A if it is the halting set on A of some **While** procedure.

From now on, we restrict our attention to the algebras \mathcal{R}_0 and \mathcal{R} .

Chapter 3

Infinite disjunction; $While^{OR}$ and $While^{\exists N}$; Semantic disjointness; Engeler's Lemma

Chapters 3 and 4 cover the first topic of this thesis. In Chapter 4, we will prove certain structure theorems for $While(\mathcal{R})$ -semicomputable sets of reals in the real plane. In this chapter, in preparation for these theorems, we discuss the semantics of strong disjunction ('OR'), infinite disjunctions, the corresponding extensions $While^{OR}$ and $While^{\exists N}$ of the $While$ language, the concept of semantic disjointedness, and Engeler's Lemma, which is an important theoretical tool for the research described in this thesis. It states (roughly) that a semicomputable set can be expressed as the disjunction of an effective infinite sequence of booleans over the appropriate signature.

We write $\Sigma^{(OR)}$ for the signature Σ or Σ^{OR} , and $While^{(OR)}$ for the $While$ or

$While^{OR}$ language.

The work in this chapter was essentially covered in [XFZ13], for computation on \mathbb{R} . It is repeated here applied to computation on \mathbb{R}^2 , for the sake of completeness.

3.1 Introduction

As we mentioned in Chapter 1 [TZ04, §1][XFZ13, §1], there are two main kinds of models of computation: abstract and concrete. Abstract models of computation based on the **While** language, with partial basic operations on \mathbb{R} , suffer from a limitation, namely the inability to implement *interleaving* or *dovetailing*. The problem is that when interleaving two processes, one process may converge and the other diverge locally (because of the partial basic operations). The resulting process will then diverge, whereas we would want it to converge. Thus we cannot even prove that the union of two semicomputable sets is semicomputable! (Concrete models do not have this limitation.)

To correct this deficiency, we construct two enhancements of the **While** language: $While^{OR}$ and $While^{\exists N}$.

In the $While^{OR}$ language, we introduce a strong (Kleene) disjunction operation ' ∇ ', where $b_1 \nabla b_2$ converges to true if either component converges to true, even if the other one diverges. By means of this, interleaving of finitely many processes can be simulated at the abstract level.

The $While^{\exists N}$ language includes a strong 'Exist' construct over the naturals:

$$x^B := \text{Exist } z : P(t, z)$$

where $z : \text{nat}$ and P is a boolean-valued procedure. By means of this, interleaving of *infinitely* many processes can be simulated at the abstract level.

We will study the structure of semicomputable subsets of \mathbb{R}^2 , where a set is said to be (for example) **While** semicomputable if it is the halting set of a **While** procedure on \mathbb{R}^2 .

3.2 Expanding $While$ to $While^{OR}$ and $While^{\exists N}$

Let Σ be a standard signature. Note that it contains the "conditional" boolean

operators 'cor' ($\overset{c}{\vee}$) and 'cand' ($\overset{c}{\wedge}$), as well as the strict boolean operators (\vee, \wedge) (see Remarks 2.3.1 (3)).

We now give their (3-valued) truth table semantics:

$$b_1 \overset{c}{\vee} b_2$$

$b_1 \backslash b_2$	tt	ff	\uparrow
tt	tt	tt	tt
ff	tt	ff	\uparrow
\uparrow	\uparrow	\uparrow	\uparrow

$$b_1 \overset{c}{\wedge} b_2$$

$b_1 \backslash b_2$	tt	ff	\uparrow
tt	tt	ff	\uparrow
ff	ff	ff	ff
\uparrow	\uparrow	\uparrow	\uparrow

Note that these two operators are *not* strict, unlike the boolean operators ‘ \vee ’ and ‘ \wedge ’.

Next we give the semantics of the two strong Kleene operators ‘OR’ (‘ ∇ ’) and ‘AND’ (‘ Δ ’)

$$b_1 \nabla b_2:$$

$b_1 \backslash b_2$	tt	ff	\uparrow
tt	tt	tt	tt
ff	tt	ff	\uparrow
\uparrow	tt	\uparrow	\uparrow

$$b_1 \Delta b_2:$$

$b_1 \backslash b_2$	tt	ff	\uparrow
tt	tt	ff	\uparrow
ff	ff	ff	ff
\uparrow	\uparrow	ff	\uparrow

Note that these two operators are also not strict.

The ‘OR’ operator allows us to simulate interleaving (or “dovetailing”) at an abstract level, since it allows us to decide a disjunction of two boolean terms $b_1 \nabla b_2$ to be true if either of these converges to **tt** (even if the other one diverges).

Note that the ‘AND’ operator can be defined as dual to ‘OR’:

$$b_1 \Delta b_2 \quad \equiv_{df} \quad \neg(\neg b_1 \nabla \neg b_2).$$

Note that negation here (and elsewhere) is defined strictly, i.e. $\neg b$ evaluates to \uparrow wherever b does.

Let Σ^{OR} be the expansion of Σ formed by adding ‘OR’. We then define:

$$\mathbf{Term}^{OR}(\Sigma) = \mathbf{Term}(\Sigma^{OR})$$

$$\mathbf{Bool}^{OR}(\Sigma) = \mathbf{Bool}(\Sigma^{OR})$$

$$\mathbf{While}^{OR}(\Sigma) = \mathbf{While}(\Sigma^{OR})$$

We can also extend the \mathbf{While} language by adding a new boolean term

$$\text{Exist } z : P(t, z),$$

where the procedure P has type $u \times \text{nat} \rightarrow \text{bool}$, and z is a “new” variable of sort nat . This will occur only in the context:

$$x^B := \text{Exist } z : P(t, z).$$

We define its semantics as:

$$\llbracket \text{Exist } z : P(t, z) \rrbracket^{A\sigma} \simeq \begin{cases} \mathbf{tt} & \text{if } P^A(\llbracket t \rrbracket^{A\sigma}, n) \downarrow \mathbf{tt} \text{ for some } n \\ \uparrow & \text{otherwise.} \end{cases} \quad (3.1)$$

This corresponds to the following operational semantics: interleave the computations for

$$P^A(t, 0), P^A(t, 1), P^A(t, 2), \dots$$

and return \mathbf{tt} if and only if any of these procedures terminates and returns \mathbf{tt} ; otherwise keep on going.

This operation allows us to simulate *infinite interleaving* at the abstract level. Note that this is different from “evaluating from the left”, which can be implemented by a simple loop:

```

find := false;
z:=0;
repeat
  find := P(t,z)
  z:=z+1;
until find = true.

```

which will *diverge* in case, e.g.,

$$P^A(t, 1) \downarrow \mathbf{ff}, \quad P(t, 2) \uparrow, \quad P(t, 3) \downarrow \mathbf{tt},$$

whereas $\text{Exist } z : P(t, z)$ will converge to \mathbf{tt} .

The usefulness of these new program constructs will become apparent in Section 3.

Using the ‘Exist’ construct, we can “weakly simulate” OR, *i.e.*, define a procedure P such that¹

$$\text{Exist } z : P(b_1, b_2, z) \sim b_1 \nabla b_2.$$

¹See Definition 2.6.8.

In fact, we can define $P(b_1, b_2, z) \equiv$

```

proc
in  $b_1, b_2 : \text{bool}$ 
     $z : \text{nat}$ 
out  $b : \text{bool}$ 
begin
 $b :=$  if  $z=1$  then  $b_1$  else
        if  $z=2$  then  $b_2$  else
            false
        fi
    fi
end.

```

Note that $\text{Exist } z: P(b_1, b_2, z)$ is only weakly semantically equivalent to $b_1 \vee b_2$; in fact no construct of the form $\text{Exist } z: P(b_1, b_2, z)$ can be strongly equivalent to $b_1 \vee b_2$, since when b_1 and b_2 both have the value ff , then $b_1 \vee b_2$ has the value ff , but $\text{Exist } z: P(b_1, b_2, z)$ can only have values \# and \uparrow , by (3.1).

We can nevertheless think of ‘OR’ as a “finite” version of ‘Exist’, and so we adjoin the ‘OR’ construct together with ‘Exist’ to form the language $While^{\exists N}(\Sigma)$.

Remark 3.2.1 (Continuity of $While^{OR}$ and $While^{\exists N}$ computable functions).

As stated before all $While$ computable functions on a topological partial algebra are continuous. The same applies to $While^{OR}$ and $While^{\exists N}$ computable functions. We

omit proofs. Again, this is important because of the Continuity Principle².

Remark 3.2.2. The ‘Exist’ construct can be implemented from the ‘choose’ construct (or the “countable choice” operator” [TZ04]), by

$$x^B := \text{Exist } z : P(t, z) \iff n := \text{choose } z : P(t, z) ; x^B := P(t, n)$$

However, unlike the ‘choose’ construct [TZ04] which is nondeterministic, the ‘Exist’ construct is “weakly” or “globally” deterministic, i.e., deterministic at the abstract level, although there is nondeterminism in the actual choice of z in a concrete implementation.

We have the obvious

Lemma 3.2.3.

- (1) A $While$ computable function is $While^{OR}$ computable.
- (2) A $While^{OR}$ computable function is $While^{\exists N}$ computable.

3.3 Numerical coding of syntax

We assume given a family of effective numerical codings of the classes of Σ -expressions, with $\lceil E \rceil$ denoting the code of the expression E . This coding is assumed to satisfy:

²See Remark 2.2.3.

- $\lceil E \rceil$ increases strictly with the complexity of E , and in particular, the code of an expression is larger than those of its subexpressions;
- sets of codes of the various syntactic classes, and their respective subclasses, such as $\{\lceil t \rceil \mid t \in \mathbf{Term}\}$, $\{\lceil t \rceil \mid t \in \mathbf{Term}_s\}$ and $\{\lceil S \rceil \mid S \in \mathbf{Stmt}\}$, etc., are primitive recursive;
- we can go primitive recursively from codes of expressions to codes of their immediate subexpressions and vice versa; thus, for example, $\lceil S_1 \rceil$ and $\lceil S_2 \rceil$ are primitive recursive in $\lceil S_1; S_2 \rceil$, and conversely.

In short, we can primitive recursively simulate all operations involved in processing the syntax of the programming language.

Numerical coding (or Gödel numbering) of programming syntax for a very similar language, satisfying all the above conditions, can be found in [DSW94, §4.1].

3.4 Semantics of infinite disjunctions

We will see that the halting set of a $While$, $While^{OR}$ or $While^{\exists N}$ procedure can be expressed as the countable disjunction of an effective infinite sequence of booleans. We must therefore first consider carefully some possible semantics for infinite disjunctions in 3-valued logic.

Discussion 3.4.1 (Two semantics for infinite disjunctions). Let (b_k) be a sequence of $\Sigma^{(OR)}$ -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 $\bigvee_{k=0}^{\infty} b_k$, with two possible outputs, \mathbf{tt} and \uparrow :

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

This definition is **While** computable (in the sequence of codes $\ulcorner b_k \urcorner$), with the following procedure:

Evaluate b_k ($k = 0, 1, \dots$) one by one. There are 3 possibilities:

- for some k , b_k converges to \mathbf{tt} , and all earlier b_j converges to \mathbf{ff} , or
- for some k , evaluation of b_k *diverges* and all earlier b_j converge to \mathbf{ff} (“local divergence”), or
- all the b_k converge to \mathbf{ff} (“global divergence”).

In the first case, evaluation of the infinite disjunction converges to \mathbf{tt} .

In the latter two cases, it diverges.

(2) *Infinite strong disjunction* (“strong Kleene evaluation”), written $\bigvee_{k=0}^{\infty} b_k$, again

with two possible outputs, \mathbf{tt} and \uparrow :

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

This definition is not (in general) $While^{(OR)}$ computable (in the sequence of codes $\lceil b_k \rceil$), but it is $While^{\exists N}$ computable, by the semantic definition of $\text{Exist } z : P(t, z)$. This is the definition we will mainly use in this paper, e.g. in the formulation of Engeler's Lemma (Lemma 3.5.4 below).

Intuitively, both definitions (1) and (2) are “concretely computable” (cf. Section 1.2).

They generalize (respectively) the finite disjunctions ‘cor’ (\bigvee^c) and ‘OR’ (\bigvee).

Notation 3.4.2. For any boolean term b with $\mathbf{Var}(b) \subseteq \mathbf{x} : u$, and $a \in A^u$, we write $b[a]$ to mean $\llbracket b \rrbracket^A \sigma \downarrow \mathbf{tt}$ for some $\sigma \in \mathbf{State}(A)$ such that $\sigma[\mathbf{x}] = a$. Note that this is well-defined, by the Functionality Lemma for terms [TZ00, §3.2].

Definition 3.4.3 (Relation defined by boolean). A $\Sigma^{(OR)}$ -boolean term b with $\mathbf{Var}(b) \subseteq \mathbf{x} : u$, is said to define a relation $R \subseteq A^u$ (w.r.t \mathbf{x}) iff for all $a \in A^u$

$$a \in R \iff b[a].$$

3.5 Engeler's Lemma for $While^{(OR)}$ Semantic disjointness

The following Lemma is of vital importance in our work. [Eng68]

Lemma 3.5.1 (Engeler's Lemma for $While^{(OR)}$). If a relation $R \subseteq A^u$ is $While^{(OR)}$ semicomputable over a standard partial Σ -algebra A , then R can be defined as the (strong) disjunction of an effective sequence of $\Sigma^{(OR)}$ -booleans over A .

In [XFZ13, §3], this Lemma was proved by the use of computation trees for $While^{(OR)}$ computations.

We also need the following concept.

Definition 3.5.2 (Semantic disjointness). A sequence (b_0, b_1, b_2, \dots) of boolean terms is *semantically disjoint* over A if for any state σ on A and any n ,

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

The following two lemmas are proved in [XFZ13, § 4].

Lemma 3.5.3 (Semantic Disjointness Lemma). The sequence of computable boolean terms generated from a $While^{OR}$ computation tree S by the construction using computation trees in the proof of Engeler's Lemma³ is semantically disjoint.

Lemma 3.5.4 (Semantic disjointness evaluation). If an effective sequence of

³[XFZ14, Lemma 4.3.1].

booleans (b_k) is semantically disjoint over A , then⁴

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

i.e. for any σ , $[[\bigvee_{k=0}^{\infty} b_k]]^A \sigma$ can be “evaluated from the left”.

3.6 Engeler's Lemma for $While^{\exists N}$

In [XFZ13], in order to develop a form of Engeler's lemma for $While^{\exists N}$, **computation trees** (or hypertrees) for $While^{\exists N}$ computation were developed. These are countably infinitely branching and, strictly speaking, are not trees, but directed acyclic graph. Nevertheless they are suitable for proving a version of Engeler's lemma for $While^{\exists N}$ computation.

Lemma 3.6.1 (Engeler's Lemma for $While^{\exists N}$). If a relation R is $While^{\exists N}$ semicomputable over a standard partial Σ -algebra A , then R can be defined as the (strong) disjunction of an effective sequence of Σ -booleans over A .

Remark 3.6.1. Here (unlike Engeler's Lemma for $While^{OR}$ computation) the sequence of booleans constructed in the proof of Engeler's Lemma for $While^{\exists N}$ does not satisfy semantic disjointness, because of the structure of the $While^{\exists N}$ computation hypertrees.

⁴Recall the notation ‘ \approx ’ for semantic equivalence (Definition 2.6.7).

Chapter 4

Partition Lemma; Structure theorems for semicomputable sets over \mathcal{R}_0 on the real plane

We now present our Structure Theorems characterizing the $\mathbf{While}^{\text{OR}}$ and $\mathbf{While}^{\exists\mathbb{N}}$ \mathcal{R}_0 -semicomputable subsets of \mathbb{R}^2 . We will discuss the limitations of the \mathbf{While} language in this regard and show how the $\mathbf{While}^{\text{OR}}$ and the $\mathbf{While}^{\exists\mathbb{N}}$ languages correct these deficiencies, generalizing the corresponding results for semicomputable subsets of \mathbb{R} [XFZ13].

From now on, we consider only the algebra $\mathcal{A} = \mathcal{R}_0$, and write Σ , Σ^{OR} and $\Sigma^{\exists\mathbb{N}}$ for $\Sigma(\mathcal{R})$ and $\Sigma^{\text{OR}}(\mathcal{R})$ and $\Sigma^{\exists\mathbb{N}}(\mathcal{R}_0)$ respectively.

The results here extend to two dimensions those of [XFZ13], from which the introductory material (Section 4.1) is largely taken.

In Section 4.1, we introduce a modified semantics for booleans over \mathcal{R}_0 , and define a canonical form for such booleans.

In Section 4.2, we define the notions of *semialgebraic* and *basic* set in \mathbb{R}^2 , which are fundamental to what follows.

In Section 4.3, we present the Partition Lemma for booleans on \mathbb{R}^2 , which is used to prove the Structure Theorems in Section 4.4 - 4.6.

In Section 4.7, we discuss “finiteness theorems” based on *cell decomposition*, and use these to present *alternative* versions of the Structure Theorems.

4.1 Computational equivalence of terms; Semantics of atomic booleans; Canonical form for Σ^{OR} booleans

Recall that we are working over the *ring algebra* \mathcal{R}_0 over the reals, with signature Σ_0 .

Note that according to the syntax of terms over $\Sigma(\mathcal{R}_0)$ (Section 2.3), all atomic terms have one of the forms, $\text{eq}^{\text{R}}(t_1, t_2)$, $\text{less}^{\text{R}}(t_1, t_2)$, $\text{eq}^{\text{N}}(t_1, t_2)$, $\text{less}^{\text{N}}(t_1, t_2)$, where t_1 and t_2 have types **real** in the first two cases and **nat** in the last two cases.

Below we will be interested mainly in the *first two* forms. We will call these (boolean) *atoms*, and write them as $(t_1 = t_2)$ and $(t_1 < t_2)$ respectively. (Thus ‘=’ and ‘<’ will generally refer to equality and order over the reals, unless otherwise specified)

4. Partition Lemma; Structure theorems for semicomputable sets over \mathcal{R}_0 46 on the real plane

For convenience, we write \mathbf{x} for a tuple of real variables (x_1, \dots, x_m) , and assume t_1 and t_2 contain only variables in \mathbf{x} . Further, for a state σ on \mathcal{R}_0 , we write $\sigma[\mathbf{x}] = a$, where $a = (a_1, \dots, a_m) \in \mathbb{R}^m$, and $\sigma(x_i) = a_i$ for $i = 1, \dots, m$.

The proof of the Canonical Form Lemma 4.1.12 below (and hence the Partition Lemma 4.3.3) requires a careful analysis of the semantics of atomic booleans of the form

- (1) $t_1 = t_2$ and (2) $t_1 < t_2$.

We will see below (Lemma 4.1.6), these atoms can be simplified, respectively, to the forms (1) $p(\mathbf{x}) = 0$ and (2) $p(\mathbf{x}) > 0$, for some integer polynomials $p(\mathbf{x})$. Now according to the semantics of '=' and '<' (Remark 2.3.1), together with the semantic rules for terms (Definition 2.6.6), the semantic evaluation of these two atoms at a state σ , where $\sigma[\mathbf{x}] = a$, is given by

$$\llbracket p(\mathbf{x}) = 0 \rrbracket \sigma \simeq \begin{cases} \uparrow & \text{if } p(a) = 0 \\ \text{ff} & \text{if } p(a) \neq 0 \end{cases} \quad (4.1a)$$

$$\llbracket p(\mathbf{x}) > 0 \rrbracket \sigma \simeq \begin{cases} \text{tt} & \text{if } p(a) > 0 \\ \text{ff} & \text{if } p(a) < 0 \\ \uparrow & \text{if } p(a) = 0. \end{cases} \quad (4.1b)$$

Hence at a zero a of $p(\mathbf{x})$, the booleans $p(\mathbf{x}) = 0$ and $p(\mathbf{x}) > 0$ both diverge.

Now suppose $p(\mathbf{x})$ has degree 0, i.e. $p(\mathbf{x}) \equiv c$ for some (integer) constant c .

Consider the two cases:

- (1) c is non-zero. Then p has no zeros, and (as we would want) at all states $p(\mathbf{x}) = 0$ evaluates to **ff**, and $p(\mathbf{x}) > 0$ evaluates to **tt** if c is positive, and **ff** if c is negative.
- (2) c is zero. Now *every* real point is a root of p , but by (4.1) the atoms $p(\mathbf{x}) = 0^{\mathbb{R}}$ and $p(\mathbf{x}) > 0^{\mathbb{R}}$, which simplify to $0^{\mathbb{R}} = 0^{\mathbb{R}}$ and $0^{\mathbb{R}} > 0^{\mathbb{R}}$ respectively, diverge at all states! But this is quite counter-intuitive.

Similarly, we would (presumably) want atoms $t_1 = t_2$ to evaluate to **tt**, and not diverge, if (e.g.) $t_1 \equiv t_2 \equiv 3$, or $t_1 \equiv 2 * \mathbf{x} + 2$ and $t_2 \equiv 1 + \mathbf{x} + \mathbf{x} + 1$, or more generally, where the equality $t_1 = t_2$ follows from the ring axioms.

Hence we must modify the semantics given by (4.1).

We proceed as follows. First some remarks on the set $\mathbb{Z}[\mathbf{x}]$ of polynomials with integer coefficients, and real variable $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_k)$.

Remark 4.1.1 (Standard form for polynomials). Any polynomial can be written in a standard form by (1) assuming a standard listing $\mathbf{x}_1, \mathbf{x}_2, \dots$ of the real variables, and (2) ordering the monomials $\mathbf{x}_1^{e_1} \mathbf{x}_2^{e_2} \dots \mathbf{x}_m^{e_m}$, firstly by decreasing weight ($= e_1 + e_2 + \dots + e_n$), and secondly, anti-lexicographically in (e_1, e_2, \dots, e_n) . We can then define a numerical coding of polynomials in standard form.

Note that our polynomial expressions in standard form are written with integer coefficients, although the signature Σ does not have a data type `int`. However our “polynomial notation” does not involve integers essentially. For example, the polynomial expression ‘ $2\mathbf{x}^2 - 3\mathbf{x} + 4$ ’ stands for the Σ -term $\mathbf{x} * \mathbf{x} + \mathbf{x} * \mathbf{x} + (-\mathbf{x}) + (-\mathbf{x}) +$

$(-x) + 1 + 1 + 1 + 1$ (suitably parenthesized) of type **real**.

Let \mathbf{E} be the equational calculus in the language $(0, 1, +, -, *)$, with the axioms for commutative rings. By “real term” we mean a Σ_0 -term of type **real**.

Definition 4.1.2 (Computational equivalence). Two real terms t_1, t_2 are *computationally equivalent* (written $t_1 \cong t_2$) iff $\mathbf{E} \vdash t_1 = t_2$.

Lemma 4.1.3. Any real term t can be re-written uniquely as a polynomial in standard form; more precisely, there is a unique polynomial $P[t]$ in standard form such that $t \cong P[t]$.

Lemma 4.1.4. For any real terms t_1, t_2 , the following three assertions are equivalent:

- (1) $t_1 \cong t_2$
- (2) $P[t_1] \equiv P[t_2]$
- (3) $P[t_1 - t_2] \equiv 0$ (the zero polynomial).

Proof. Clear. □

Note that by the equivalence (1) \iff (2) above, computational equivalence between real terms is decidable.

Lemma 4.1.5. Any atom of the form (1) $t_1 = t_2$ or (2) $t_1 < t_2$ can be rewritten in the form (1) $p(x) = 0$ or (2) $p(x) > 0$ respectively, where $p(x) \in \mathbb{Z}[x]$.

Proof. This follows from the observation that $t_1 = t_2 \iff t_1 - t_2 = 0 \iff P[t_1 - t_2] = 0$,

and similarly for case (2) $t_1 < t_2$. □

Recall the definition (2.6.7) of semantic equivalence:

$$t_1 \approx t_2 \iff \forall \sigma (\llbracket t_1 \rrbracket^A \sigma = \llbracket t_2 \rrbracket^A \sigma)$$

i.e. for all σ , the two expression $\llbracket t_1 \rrbracket^A \sigma$ and $\llbracket t_2 \rrbracket^A \sigma$ both converge to the same value, or both diverge.

The following Lemma expresses the *soundness* and *completeness* of computational equivalence w.r.t. semantic equivalence.

Lemma 4.1.6. For any two real terms t_1, t_2

$$t_1 \cong t_2 \iff t_1 \approx t_2$$

Proof. (\implies) This is clear.

(\impliedby) This follows from the fact that if a polynomial over \mathbb{R}^m ($m \geq 1$) has value 0 everywhere, then it must be the zero polynomial, by [XFZ13, Cor.2.3.2]. □

Definition 4.1.7 (Modified semantics of boolean atoms). For real terms t_1, t_2 , we define:

$$\llbracket t_1 = t_2 \rrbracket \sigma \simeq \begin{cases} \mathbf{tt} & \text{if } t_1 \cong t_2 \\ \uparrow & \text{if } \llbracket t_1 \rrbracket \sigma = \llbracket t_2 \rrbracket \sigma \text{ but } t_1 \not\cong t_2 \\ \mathbf{ff} & \text{if } \llbracket t_1 \rrbracket \sigma \neq \llbracket t_2 \rrbracket \sigma. \end{cases} \quad (4.2a)$$

$$\llbracket t_1 < t_2 \rrbracket \sigma \simeq \begin{cases} \mathbf{tt} & \text{if } \llbracket t_1 \rrbracket \sigma < \llbracket t_2 \rrbracket \sigma \\ \mathbf{ff} & \text{if } \llbracket t_1 \rrbracket \sigma > \llbracket t_2 \rrbracket \sigma \text{ or } t_1 \cong t_2 \\ \uparrow & \text{if } \llbracket t_1 \rrbracket \sigma = \llbracket t_2 \rrbracket \sigma \text{ but } t_1 \not\cong t_2. \end{cases} \quad (4.2b)$$

Note here that $\llbracket \dots \rrbracket$ means $\llbracket \dots \rrbracket^{\mathcal{R}_0}$.

These definitions will be used in the proof of the Canonical Form Lemma (4.1.12).

Discussion 4.1.8 (Justification for modified semantics).

As in Discussion 2.3.3, we consider this issue in two ways: the first based on continuity considerations, and the second, again, based on a thought experiment involving concrete computations.

(a) Recall the discussion (2.3.3(a)) on the motivation for defining equality and order on the reals as partial functions eq_p and less_p . Continuity of **While**^{OR} computable functions is a central concern here. We may then well ask: do the above modified semantic definitions (4.2.6) not “spoil” this continuity result? The answer is no: with the above definitions, it still holds that **While** (or **While**^{OR}, or **While**^{∃N}) computable functions are continuous. The proof depends on the fact that the condition for the atomic formula ‘ $t_1 = t_2$ ’ to have an output of \mathbf{t} instead of \uparrow (i.e. that $t_1 \cong t_2$) is *independent* of the state. Hence the continuity proof given in [TZ99, §6] and [TZ00, §7.6] for **While** computable functions on topological algebras can be easily adapted to the semantics based on the present definition. We omit details.

(b) Another (“concrete”) approach to justifying this definition lies in continuing with our thought experiment in Discussion 2.3.3(b). So consider again an atomic formula of the form $t_1 = t_2$, and see what is involved in trying to decide whether it is true or not. First, take the case considered in Discussion 2.3.3(b)) where $t_1 \equiv \mathbf{x}$, and

$t_2 \equiv y$. Suppose, again, that these are presented to us, at a given state σ , as fast Cauchy sequences (r_n) and (s_n) of rationals respectively. Then, as shown in part (b) of Discussion 2.3.3, we can only gain “negative” information in finite time. In other words, if $x = y$ is true at σ , then we cannot determine this in finite time, and so the computation diverges. Suppose, however, that (for example) $t_1 \equiv 1 + x$ and $t_2 \equiv x + 1$. Then it is clear *a priori* that these terms are equal, regardless of the state, and without any need to consult the Cauchy sequence for x at that state.¹ After all, it is the same variable, and hence the same Cauchy sequence, on both sides of the equation! Hence in this case we let the atom $t_1 = t_2$ evaluate to \mathbf{t} at all states.

Unless otherwise stated, the definitions and lemmas in this subsection refer to the Σ^{OR} -language, with the Σ -language as a special case. We generally write b, b', \dots for Σ^{OR} -booleans.

Definition 4.1.9. A *boolean combination* of a set of atomic booleans is a boolean expression built up from the boolean atoms $(t_1 < t_2)$ and $(t_1 = t_2)$ (with $t_1, t_2 : \text{real}$) by ‘ \vee ’, ‘ \wedge ’, ‘ $\overset{c}{\vee}$ ’, ‘ $\overset{c}{\wedge}$ ’, ‘ ∇ ’, ‘ Δ ’ and ‘ \neg ’.

We need a technical lemma (Note that $\Sigma_0 = \Sigma(\mathcal{R}_0)$, where the algebra \mathcal{R}_0 is defined in Section 2.3).

¹We are assuming here that the ring axioms, such as commutativity of $+$, hold over \mathcal{R}_0 (which is not always true in practical computing).

Lemma 4.1.10. Every Σ_0 -term $t : \text{nat}$ has one of two forms:

$$\textit{either} \quad \text{suc}(\text{suc}(\dots(\text{suc } 0^{\mathbb{N}})\dots)) \quad (4.3a)$$

$$\textit{or} \quad \text{suc}(\text{suc}(\dots(\text{suc } v)\dots)) \quad (4.3b)$$

where there are n times ‘suc’ (for some $n \geq 0$) and in case (4.3b), v is a **nat** variable.

Proof. An easy structural induction on $t : \text{nat}$.

For the base case, t must be either 0 or a **nat** variable. For the inductive step, t must have the form $\text{suc } t'$, for $t' : \text{nat}$, and the result follows from the induction hypothesis.

□

Corollary 4.1.11. Every Σ_0 -term $t : \text{nat}$ without a **nat** variable is a *numeral*, i.e. of the form:

$$\bar{n} \equiv \text{suc}(\text{suc}(\dots(\text{suc } 0^{\mathbb{N}})\dots)) \quad (\bar{n} \text{ times suc}). \quad \square$$

Lemma 4.1.12 (Canonical form for booleans over \mathcal{R}_0). A Σ_0^{OR} -boolean with variables among $\mathbf{x} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_m)$ of sort **real** only, is effectively semantically equivalent to a boolean combination of equations and inequalities of the form:

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

where p and q are polynomials in \mathbf{x} of degree > 0 .

Proof. By structural induction on the boolean b .

Base cases:

- $b \equiv (t_1 = t_2)$ or $(t_1 < t_2)$ for terms t_1, t_2 : **real**.

By Lemma 4.1.4, these are respectively semantically equivalent to $P[t_1 - t_2] = 0$ and $P[t_2 - t_1] > 0$.

- $b \equiv (t_1 = t_2)$ or $(t_1 < t_2)$ for terms t_1, t_2 : **nat**.

By Corollary 4.1.10, t_1 and t_2 must be numerals, \bar{n}_1 and \bar{n}_2 respectively, for some $n_1, n_2 \in \mathbb{N}$. Hence in this case b has the form $(\bar{n}_1 = \bar{n}_2)$ or $(\bar{n}_1 < \bar{n}_2)$ for some $n_1, n_2 \in \mathbb{N}$, which reduces to **true** or **false** in all cases.

Induction step:

Suppose both b_1 and b_2 are effectively strongly equivalent to boolean combinations of equations and inequalities: $p(\mathbf{x}) = 0$ and $q(\mathbf{x}) > 0$.

Then, clearly the same holds for $\neg b_1$, $(b_1 \vee b_2)$, $(b_1 \wedge b_2)$, $(b_1 \overset{c}{\vee} b_2)$, $(b_1 \overset{c}{\wedge} b_2)$, $(b_1 \nabla b_2)$ and $(b_1 \Delta b_2)$. □

Remark 4.1.13.

(a) Although the statement of Lemma 4.1.12 does not actually specify a unique “canonical form” of a give boolean, it can easily be made unique by suitable conventions for listing variables, etc.

(b) The proof of the lemma is *effective* or *constructive*, in the sense that it can be made to effectively give a (or “the”) canonical form $CF(b)$ of a given boolean

b , so that the map $\ulcorner b \urcorner \mapsto \ulcorner CF(b) \urcorner$ is recursive.

4.2 Semialgebraic and basic sets

We introduce some concepts which will be very important in our subsequent work.

We consider subsets of \mathbb{R}^2 .

Definition 4.2.1 (Semi-algebraic set). A *semi-algebraic set* is a finite union of sets of the form

$$\{x \in \mathbb{R}^2 \mid p_1(x) > 0, \dots, p_k(x) > 0, q_1(x) = 0, \dots, q_\ell(x) = 0\} \quad (k, \ell \geq 0) \quad (4.4)$$

where $p_1, \dots, p_k, q_1, \dots, q_\ell$ are polynomials over \mathbb{Z} .

Definition 4.2.2 (Basic set). A *basic set* is a particular kind of semi-algebraic set of the form

$$\{x \in \mathbb{R}^2 \mid p_1(x) > 0, \dots, p_k(x) > 0\} \quad (k > 0) \quad (4.5)$$

where p_1, \dots, p_k are polynomials over \mathbb{Z} .

Note that basic sets are open.

Notation 4.2.3. We will use B, B', B_1, \dots to range over basic sets.

Lemma 4.2.4. 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 on B^+ , $p > 0$, on B^- , $p < 0$, and on D , $p = 0$, and $B^+ \cup B^- \cup D = \mathbb{R}^2$.

Proof. Clear. □

This will be used in the Partition Lemma (Lemma 4.3.3).

4.3 Partition Lemma for Σ^{OR} booleans on \mathbb{R}^2

Notation 4.3.1. For a pair of variables $\mathbf{x} \equiv (x_1, x_2) : \text{real}^2$, let $\mathbf{Bool}(\mathbf{x})$ be the set of Σ^{OR} -booleans with no free variables other than \mathbf{x} .

For the rest of this section, we consider only booleans in $\mathbf{Bool}(\mathbf{x})$.

Definition 4.3.2. For any $b \in \mathbf{Bool}(\mathbf{x})$, we define

$\text{PS}(b)$ (the positive set of b) $=_{df} \{x \in \mathbb{R}^2 \mid b[x] = \mathbf{tt}\}$

$\text{NS}(b)$ (the negative set of b) $=_{df} \{x \in \mathbb{R}^2 \mid b[x] = \mathbf{ff}\}$

$\text{DS}(b)$ (the divergence set of b) $=_{df} \{x \in \mathbb{R}^2 \mid b[x] \uparrow\}$

Recall Notation 3.4.2. The significance of the terminology *positive set*, *negative set* and *divergent set* is given by Lemmas 4.2.4 and 4.3.3.

Lemma 4.3.3 (Partition Lemma for booleans on \mathbb{R}^2). Every boolean $b \in \mathbf{Bool}(\mathbf{x})$ over \mathcal{R}_0 has semantics effectively represented by a partition of \mathbb{R}^2 of the form:

$$\begin{aligned} \text{PS}(b) &= \bigcup_{i=1}^k B_i^+ \\ \text{NS}(b) &= \bigcup_{i=1}^l B_i^- \\ \text{DS}(b) &= \bigcup_{i=1}^m D_i \end{aligned}$$

where B_i^+ , B_j^- are basic sets, $B_i^+ \cap B_j^- = \phi$, and D_i has the form:

$$\{x \in \mathbb{R}^2 \mid p_1(x) > 0, \dots, p_r(x) > 0, q_1(x) = 0, \dots, q_s(x) = 0\} \quad (4.6)$$

where $p_1, \dots, p_r, q_1, \dots, q_s$ are polynomials over \mathbb{Z} , with $r \geq 0$, and $s > 0$. Then (since $s > 0$) D_i is a finite set of curves and/or points. Also $\text{DS}(b)$ is the *boundary* of $\text{PS}(b)$, and of $\text{NS}(b)$. And

$$\bigcup_{i=1}^k B_i^+ \cup \bigcup_{i=1}^l B_i^- \cup \bigcup_{i=1}^m D_i = \mathbb{R}^2$$

The proof will be by structural induction on the canonical form of booleans.

To clarify the proof, we first examine a simple case. Suppose $b_1 \equiv p(x) > 0$, $b_2 \equiv p_2(x) > 0$, where (say) $p_1(x) = -x_1^2 - x_2^2 + 4$, and $p_2(x) = -(x_1 - 3)^2 - x_2^2 + 4 > 0$.

Then (writing $x = (x_1, x_2)$)

$$\text{PS}(b_1) = \{x \mid p_1(x) > 0\}$$

$$\text{NS}(b_1) = \{x \mid p_1(x) < 0\} \cup \{(x) \mid -p_1(x) > 0\}$$

$$\text{DS}(b_1) = \{x \mid p_1(x) = 0\}$$

$$\text{PS}(b_2) = \{x \mid p_2(x) > 0\}$$

$$\text{NS}(b_2) = \{x \mid p_2(x) < 0\} \cup \{(x) \mid -p_2(x) > 0\}$$

$$DS(b_2) = \{x \mid p_2(x) = 0\}$$

Now, consider the partition of $b_1 \overset{c}{\vee} b_2$, $b_1 \vee b_2$ and $b_1 \nabla b_2$.

(1) $PS(b_1 \overset{c}{\vee} b_2)$ can be shown as (see Figure 4.1):

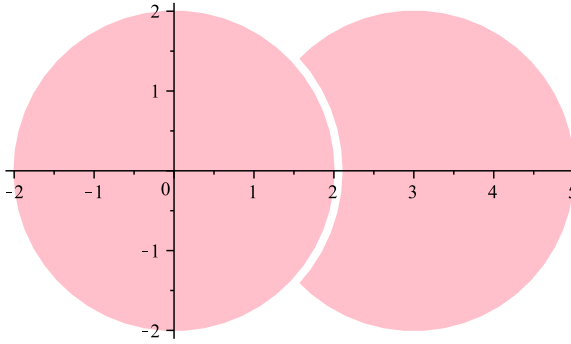


Figure 4.1

and in fact,

$$\begin{aligned} PS(b_1 \overset{c}{\vee} b_2) &= PS(b_1) \cup (PS(b_2) \cap NS(b_1)) \\ &= \{x \mid p_1(x) > 0\} \cup \{x \mid p_2(x) > 0, p_1(x) < 0\} \end{aligned}$$

$$\begin{aligned} NS(b_1 \overset{c}{\vee} b_2) &= NS(b_1) \cap NS(b_2) \\ &= \{x \mid p_1(x) < 0, p_2(x) < 0\} \end{aligned}$$

$$DS(b_1 \overset{c}{\vee} b_2) = DS(b_1) \cup (DS(b_2) \cap NS(b_1))$$

$$= \{x \mid p_1(x) = 0\} \cup \{x \mid p_2(x) = 0, p_1(x) < 0\}$$

(2) $PS(b_1 \vee b_2)$ can be shown as (see Figure 4.2):

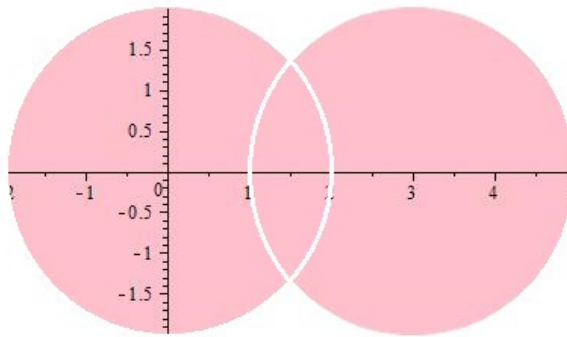


Figure 4.2

In fact,

$$PS(b_1 \vee b_2) = (PS(b_1) \cap PS(b_1)) \cup (PS(b_1) \cap NS(b_2)) \cup (NS(b_1) \cap PS(b_2))$$

$$= \{x \mid p_2(x) > 0, p_1(x) > 0\} \cup \{x \mid p_1(x) > 0, p_2(x) < 0\}$$

$$\cup \{x \mid p_1(x) < 0, p_2(x) > 0\}$$

$$NS(b_1 \vee b_2) = NS(b_1) \cup NS(b_2)$$

$$= \{x \mid p_1(x) < 0, p_2(x) < 0\}$$

$$\begin{aligned} DS(b_1 \vee b_2) &= DS(b_1) \cup DS(b_2) \\ &= \{x \mid p_1(x) = 0\} \cup \{x \mid p_2(x) = 0\} \end{aligned}$$

(3) $PS(b_1 \nabla b_2)$ can be shown as (see Figure 4.3):

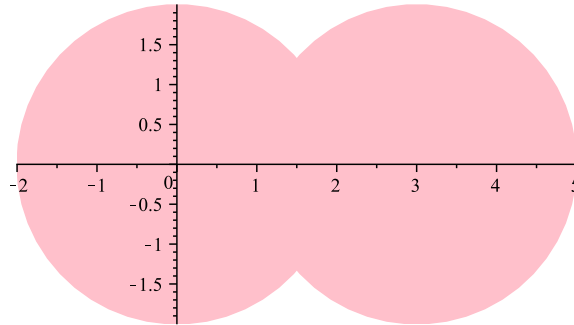


Figure 4.3

in fact

$$\begin{aligned} PS(b_1 \nabla b_2) &= PS(b_1) \cup PS(b_2) \\ &= \{x \mid p_1(x) > 0\} \cup \{x \mid p_2(x) > 0\} \end{aligned}$$

$$\begin{aligned} NS(b_1 \nabla b_2) &= NS(b_1) \cap NS(b_2) \\ &= \{x \mid p_1(x) < 0, p_2(x) < 0\} \end{aligned}$$

$$DS(b_1 \nabla b_2) = (DS(b_1) \cap DS(b_2)) \cup (DS(b_1) \cap NS(b_2)) \cup (NS(b_1) \cap DS(b_2))$$

$$= \{x \mid p_1(x) = 0, p_2(x) = 0\} \cup \{x \mid p_1(x) = 0, p_2(x) < 0\}$$

$$\cup \{x \mid p_1(x) < 0, p_2(x) = 0\}$$

We proceed to the proof of the Partition Lemma.

Proof. Partition Lemma 4.3.3.

We first transform the given boolean to canonical form, by the algorithm contained in the proof of the Canonical Form Lemma (see Remark 4.1.13(b)).

We proceed by structural induction on the canonical form of b .

Base case: $b \equiv p(\mathbf{x}) = 0$ or $p(\mathbf{x}) > 0$. Use Lemma 4.2.4.

Note that in the case that $p(\mathbf{x})$ has degree 0, i.e., it is a constant integer c , the atomic boolean $p(\mathbf{x}) > 0$ has the form $c > 0$, and so reduces to **true** or **false**, depending on the value of c . Similarly with the case of an atomic boolean $p(\mathbf{x}) = 0$.

Induction step: Briefly, this follows from the fact that the class of finite unions of basic sets is closed under (binary) union and intersection. In more detail, we consider the various cases:

- $b \equiv \neg b_1$. Then just interchange the positive and negative sets of b_1 .

Now suppose:

$$\text{PS}(b_1) = \bigcup_{i=1}^{k_1} B_{1i}^+$$

$$\text{NS}(b_1) = \bigcup_{i=1}^{l_1} B_{1i}^-$$

$$\text{DS}(b_1) = \bigcup_{i=1}^{m_1} D_{1i}$$

$$\text{PS}(b_2) = \bigcup_{j=1}^{k_2} B_{2j}^+$$

$$\text{NS}(b_2) = \bigcup_{j=1}^{l_2} B_{2j}^-$$

$$\text{DS}(b_1) = \bigcup_{j=1}^{m_2} D_{2j}$$

- $b \equiv b_1 \vee b_2$. Then

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

$$\text{NS}(b) = \bigcup_{i=1}^{l_1} \bigcup_{j=1}^{l_2} (B_{1i}^- \cap B_{2j}^-)$$

$$\text{DS}(b) = \bigcup_{i=1}^{m_1} \bigcup_{j=1}^{m_2} (D_{1i} \cup D_{2j})$$

- $b \equiv b_1 \wedge b_2$. Then

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

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

$$\text{DS}(b) = \bigcup_{i=1}^{m_1} \bigcup_{j=1}^{m_2} (D_{1i} \cup D_{2j}).$$

- $b \equiv b_1 \overset{c}{\vee} b_2$. Then

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

$$\text{NS}(b) = \bigcup_{i=1}^{l_1} \bigcup_{j=1}^{l_2} (B_{1i}^- \cap B_{2j}^-)$$

$$\text{DS}(b) = \bigcup_{i=1}^{m_1} \bigcup_{i'=1}^{l_1} \bigcup_{j=1}^{m_2} (D_{1i} \cup (B_{1i'}^- \cap D_{2j}))$$

- $b \equiv b_1 \overset{c}{\wedge} b_2$. Then

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

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

$$DS(b) = \bigcup_{i=1}^{m_1} \bigcup_{i'=1}^{k_1} \bigcup_{j=1}^{m_2} (D_{1i} \cup (B_{1i'}^+ \cap D_{2j}))$$

• $b \equiv b_1 \nabla b_2$. Then

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

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

$$DS(b) = \bigcup_{i=1}^{m_1} \bigcup_{j=1}^{m_2} (D_{1i} \cap D_{2j}) \cup \left(\bigcup_{i=1}^{m_1} \bigcup_{j=1}^{l_2} D_{1i} \cap B_{2j}^- \right) \cup \left(\bigcup_{i=1}^{l_1} \bigcup_{j=1}^{m_2} B_{1i}^- \cap D_{2j} \right).$$

• $b \equiv b_1 \Delta b_2$. Then

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

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

$$DS(b) = \left(\bigcup_{i=1}^{m_1} \bigcup_{j=1}^{m_2} D_{1i} \cap D_{2j} \right) \cup \left(\bigcup_{i=1}^{m_1} \bigcup_{j=1}^{k_2} D_{1i} \cap B_{2j}^+ \right) \cup \left(\bigcup_{i=1}^{k_1} \bigcup_{j=1}^{m_2} B_{1i}^+ \cap D_{2j} \right). \quad \square$$

Remark 4.3.4. Again, the proof of the Partition Lemma is effective, i.e. it gives an effective map

$$b \mapsto \langle PS(b), NS(b), DS(b) \rangle$$

from booleans b in canonical form to partitions (i.e. recursive in codes). Composing this map with the map given by the Canonical Form Lemma ($b \mapsto CF(b)$) then gives an effective map from booleans to their partitions.

Remark 4.3.5. The basic sets B_i^+ , B_j^- given by Lemma 4.3.3 need not be connected. However, by the Cell Decomposition Theorem [vdD98], each has finitely many connected components (which need not be basic! – see section 4.7, Proposition 1).

Lemma 4.3.6. There is a *While*^{OR} computable function

$$in_B : \mathbb{N} \times \text{real}^2 \rightarrow \text{bool}$$

such that for any basic set B ,

$$in_B(\ulcorner B \urcorner, x_1, x_2) \simeq \begin{cases} \text{tt} & \text{if } (x_1, x_2) \in B \\ \text{ff} & \text{if } (x_1, x_2) \notin \bar{B} \\ \uparrow & \text{otherwise, i.e. } (x_1, x_2) \in \bar{B} \setminus B = \partial B \end{cases} \quad (4.7)$$

Proof. Suppose B is defined by:

$$\{(x_1, x_2) \in \mathbb{R}^2 \mid p_1(x_1, x_2) > 0, \dots, p_k(x_1, x_2) > 0\}. \quad (k > 0)$$

Then we can define (4.6), i.e. “ $(x_1, x_2) \in B$ ”, by

$$p_1(x_1, x_2) > 0 \Delta \dots \Delta p_k(x_1, x_2) > 0 \quad (4.8)$$

,

(where Δ associates to the left, let us say).

Then by the semantics of ‘ Δ ’ (Section 3.2), we note that

(1) if for any i , $(p_i(x_1, x_2) > 0)$ evaluates to false, then, (4.8) evaluates to false;

(2) if for all i , $(p_i(x_1, x_2) > 0)$ evaluates to true, then, (4.8) evaluates to true.

(3) otherwise (4.8) diverges.

□

Lemma 4.3.7. There is a **While**^{OR} computable function

$$in_S : \mathbb{N} \times \text{real}^2 \rightarrow \text{bool}$$

such that:

$$in_S(\ulcorner S \urcorner, x_1, x_2) \simeq \begin{cases} \text{tt} & \text{if } (x_1, x_2) \in S \\ \text{ff} & \text{if } (x_1, x_2) \notin \bar{S} \\ \uparrow & \text{otherwise, i.e. } (x_1, x_2) \in \bar{S} \setminus S \end{cases}$$

where S is a finite union of basic (open) sets.

Proof. By Lemma 4.3.5 and the semantics of ‘ Δ ’.

□

4.4 Characterizations of semicomputable sets in \mathbb{R}^2

In this section, we prove the “ \implies ” direction of the Structure Theorems.

Lemma 4.4.1. If a set $R \subseteq \mathbb{R}^2$ is **While**^{OR} semicomputable over \mathcal{R}_0 , then R can be expressed as the countable union of a disjoint effective sequence of finite unions of basic sets.

Proof. If $R \subseteq \mathbb{R}^2$ is **While**^{OR} semicomputable, then by Engeler’s Lemma for **While**^{OR} (Lemma 3.5.1), for all $x \in \mathbb{R}^2$,

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

for an effective sequence (b_k) of Σ^{OR} -booleans in $\mathbf{Bool}(\mathbf{x})$. By the Partition Lemma (Lemma 4.3.3) each b_k defines a finite union of effective basic sets.

By the Semantic Disjointedness Lemma (Lemma 3.5.3), the sequence (b_k) is semantically disjoint over \mathcal{R}_0 , and hence the positive sets² for different b_k 's are disjoint. □

Lemma 4.4.2. If a set $R \subseteq \mathbb{R}$ is $\mathbf{While}^{\exists\mathbb{N}}$ semicomputable over \mathcal{R}_0 , then R can be expressed as the countable union of an effective sequence of basic sets.

Proof. By Engeler's Lemma for $\mathbf{While}^{\exists\mathbb{N}}$ (Lemma 3.6.1), a $\mathbf{While}^{\exists\mathbb{N}}$ semicomputable set over \mathcal{R}_0 can be expressed as a countable disjunction of Σ^{OR} -booleans, to which the Partition Lemma (Lemma 4.3.3) again applies. □

Remark 4.4.3. Note we lose the disjointedness of the basic sets sequences because the failure of semantic disjointedness of the boolean sequence of the hypertrees for $\mathbf{While}^{\exists\mathbb{N}}$ computation (cf. Remark 3.6.1).

²Recall Definition 4.3.2.

4.5 Unions of effective sequences of basic sets are semicomputable

In this subsection, we prove the reverse “ \Leftarrow ” direction of the Structure Theorems.

Lemma 4.5.1. The countable union of a disjoint effective sequence of finite unions of basic sets on \mathbb{R}^2 is **While**^{OR} semicomputable over \mathcal{R}_0 .

Proof. Finite unions of basic sets gives us a total recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $f(n)$ is the code of the n^{th} basic set. So the countable union of a disjoint effective sequence of finite union basic sets is equal to the halting set of the procedure

<pre> proc in x₁ x₂ : real; aux i : nat; begin i := 0; while not(<i>in</i>_B(P_f(i), x₁, x₂)) do i := i + 1 od end </pre>	(4.9)
--	-------

where P_f is the **While**(\mathcal{N}) (and hence **While**(\mathcal{R}_0)) procedure which computes f on \mathbb{R}^2 .

By Lemma 4.3.6, in_B is **While**^{OR} computable, and so the above procedure is **While**^{OR} computable. □

Next, if we drop the condition of disjointness on the sequence of basic sets, we need to strengthen the corresponding programming language.

Lemma 4.5.2. The countable union of an effective sequence of basic sets is **While**^{∃N} semicomputable over \mathcal{R}_0 .

Proof. An effective sequence of basic sets is given by a total **While** computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $f(n)$ returns the code of the n^{th} basic set.

So the countable union of an effective sequence of basic sets equal to the halting set of the **While**^{∃N} procedure:

```

proc
in  x1, x2: real;
out b : bool;
begin
  b := Exist z : P(x1, x2, z)
end
```

where the procedure $P(x_1, x_2, z)$ is defined as

$$in_B(P_f(z), x_1, x_2)$$

and $P_f : \text{nat} \rightarrow \text{nat}$ is the **While**(\mathcal{N}) (and hence **While**(\mathcal{R}_0)) procedure which computes f on \mathbb{R}^2 .

By Lemma 4.3.6, in_B is **While**^{OR} (and hence **While**^{∃N}) computable, and so the above procedure is **While**^{∃N} computable. □

4.6 Structure theorems for semicomputable sets over \mathcal{R}_0

We finally present our two Structure Theorems for **While**, **While**^{OR} and **While**^{∃N} semicomputable sets over \mathcal{R}_0 .

Theorem 1. A subset U of \mathbb{R}^2 is **While**^{OR} semicomputable over \mathcal{R}_0 iff U is a countable union of a disjoint effective sequence of finite unions of basic sets.

Proof. By Lemmas 4.5.1 and 4.4.1. □

Theorem 2. A subset U of \mathbb{R}^2 is **While**^{∃N} semicomputable over \mathcal{R}_0 iff U is a countable union of an effective sequence of basic sets.

Proof. By Lemmas 4.5.2 and 4.4.2. □

Unfortunately, we have only a partial result for **While** semicomputability:

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

- (a) **While** semicomputable over $\mathcal{R}_0 \implies$ union of *disjoint* effective sequence of finite
union of basic sets.

(b) union of *disjoint* effective sequence of basic sets \implies **While** semicomp. over \mathcal{R}_0 .

Proof. (a) follows from the “ \implies ” direction of Theorem 1.

(b) We modify the proof of Lemma 4.5.1 as follows. In procedure (4.8), reinterpret ‘ in_B ’ (cf. Lemma 4.3.6) by replacing ‘ Δ ’ by ‘ \wedge ’ in (4.7). This works because of the disjointedness of the sequence of basic sets, reduces the language from **While**^{OR} to **While**. □

Remark 4.6.1 (Use of \mathcal{R}_0 for the Structure Theorems). The Structure Theorems for semicomputable subsets of \mathbb{R}^2 , as presented here, are not really generalizations of the corresponding theorems stated in [XFZ13] for \mathcal{R} , since the latter used **While** (etc.) semicomputability w.r.t. the field algebra \mathcal{R} , whereas we use the ring algebra \mathcal{R}_0 . The reason for this is that the theory of Section 4.1, notably Lemmas 4.1.3 - 4.1.6 and Definition 4.1.7 (modified semantics) applies, as it stands, only to real terms in the language of \mathcal{R}_0 .

We believe that the Structure Theorems do, in fact, hold also with the field algebra \mathcal{R} . However we were unable to investigate this properly, due to lack of time.

4.7 Cell Decomposition and Finiteness Theorems

The technique of cell decomposition in \mathbb{R}^n , as described e.g in [vdD98], can be used to prove some *finiteness theorems*, for example (working, as usual, with open sets in

\mathbb{R}^2):

Proposition 1. An open semialgebraic set S in \mathbb{R}^2 has *finitely* many connected components, each semialgebraic.

Proposition 2. An open semialgebraic set S in \mathbb{R}^2 is a union of *finitely* many basic sets.

Proposition 1 is proved in [vdD98], and Proposition 2 in [Del82].

Remark 4.7.1 (Effective Versions of Finiteness Theorems). The proofs of Propositions 1 and 2 can be effectivized. For Proposition 1: if S has connected components C_1, \dots, C_n then we can effectively find, from a numerical code for S (as a boolean combination of polynomial equations and inequalities) a code for the tuple $\langle n, C_1, \dots, C_n \rangle$. Similarly for Proposition 2 ³.

This will be important for the variant formulations of the Structure Theorems given below (Remark 4.7.3).

Remark 4.7.2 Note that the components of S given by Proposition 1 need not be basic. Moreover, the basic sets given by Proposition 2 need not be connected.

On the other hand, we have the following positive result ⁴.

Proposition 3. If the components of a basic set are bounded, and they have disjoint closures, then they are also basic.

³Personal communication from Prof. Lou van dan Dries.

⁴Personal communication from Prof. Ludwig Bröcker.

Counterexamples, to this proposition, in the absence of either of these two conditions, are given in [ABR96, p.26, Ex.4.8].

Remark 4.7.3 (Variant formulations of the Structure Theorems). Using the effective forms of Proposition 1 and Proposition 2 (see Remark 4.7.1), we can prove *variants* of our Structure Theorems, formed by *replacing*, in Theorems 1 and 2, the phrase “**basic sets**” by “**connected open semialgebraic sets**”.

However (see Remark 4.7.1) we *cannot*, in Theorems 1 and 2, replace “**basic sets**” by “**connected basic sets**”.

Chapter 5

Exhaustion, Grzegorz-Lacombe (GL) computability on \mathbb{R}^2 , Multipolynomial approximability on \mathbb{R}^2 , Equivalence Lemma

In Chapter 5 and 6, covering our topic 2, we return to the full field algebra \mathcal{R} .

In this chapter we will consider two models of computation on \mathbb{R}^2 : the concrete model of *GL computability* and the “hybrid” model of *multipolynomial approximability*, for partial functions on \mathbb{R}^2 . We will make two assumptions about such functions: (1) their domain has an “effective exhaustion” (to be defined below) and (2) they are *effectively locally uniformly continuous* with respect to this exhaustion.

Under these assumptions we will prove the equivalence of these models (Equiv-

alence Lemma 1). In Chapter 6, we will combine this with two other Equivalence Lemmas to derive the Equivalence Theorem for four computability models on \mathbb{R}^2 .

We will use the concepts of *semialgebraic* and *basic* sets, as in Chapter 4 (Section 4.2), for the purpose of defining “effective exhaustion”, in such a way to be able to prove the Extension Theorem (Theorem 4) used in the proof of this Equivalence Lemma.

The cell decomposition theorem is also relevant here, in connection with the proof of the Extension Theorem (Theorem 4).

It turns out that the appropriate definition of “polynomial” here is polynomial over the set \mathbb{R}_c of computable reals (not over \mathbb{Z} , as in Chapter 4). We therefore make the following terminological convention.

Convention 5.0.1. In this and the following chapter, “polynomial” will mean polynomial over \mathbb{R}_c . *Semialgebraic* and *basic sets* will be defined accordingly.

5.1 Elementary set; Exhaustions; local approximability and continuity

Semialgebraic and *basic* sets over \mathbb{R}^2 are defined as in Chapter 4 (Definitions 4.2.1 and 4.2.2) except that the polynomials f_i and g_j in (4.4) and (4.5) are taken to have coefficients in \mathbb{R}_c (see Convention 5.0.1).

Definition 5.1.1 (Elementary set). An *elementary set* is a non-empty, bounded, connected basic subset of \mathbb{R}^2 .

Remark 5.1.2. Semialgebraic sets, and in particular elementary sets E , have *finite descriptions* given by their defining equations (4.4, 4.5) and hence have effective numerical codings $\ulcorner E \urcorner$.

Definition 5.1.3 (Exhaustion on \mathbb{R}^2). Assume U is an open set of \mathbb{R}^2 , which is the union of a sequence of compact sets or “stages” (U_0, U_1, U_2, \dots) , where each stage U_ℓ is a finite union of *elementary sets*. More precisely:

$$(i) \quad U = \bigcup_{\ell=0}^{\infty} U_\ell,$$

$$(ii) \quad U_\ell = E_1^\ell \cup \dots \cup E_{p_\ell}^\ell \text{ for some } p_\ell \geq 1, \quad \ell = 0, 1, 2, \dots$$

where E_i^ℓ is an elementary set, and $\overline{E_i^\ell} \cap \overline{E_j^\ell} = \emptyset$ for $i \neq j$,

$$(iii) \quad \forall \ell, \forall i \in \{1, \dots, p_\ell\}, \exists j \in \{1, \dots, p_{\ell+1}\} : \overline{E_i^\ell} \subset E_j^{\ell+1}.$$

Then (U_ℓ) is called an (open) *exhaustion* of U , and for each ℓ , U_ℓ is a *stage* of the exhaustion with *components* $E_1^\ell, \dots, E_{p_\ell}^\ell$.

Remark 5.1.4. From clause (iii) follows:

$$\overline{U_\ell} \subset U_{\ell+1}$$

for all ℓ .

Remark 5.1.5. The motivation for clause (iii) will be made clear later (Remark 5.4.4, page 97).

Definition 5.1.6 (Effective exhaustion). An exhaustion (U_ℓ) of U is called *effective* if for all ℓ , the stages U_ℓ are computable, that is, the map

$$\ell \mapsto \langle \ulcorner E_1^\ell \urcorner, \dots, \ulcorner E_{p_\ell}^\ell \urcorner \rangle$$

is recursive.

We will work with functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ where $U = \mathbf{dom}(f)$ has an effective exhaustion.

Remark 5.1.7. The use of *elementary sets* as components of the stages U_ℓ of U is important for the Extension Theorem (Theorem 4), and also permits the inclusion of many functions commonly considered in calculus and elementary real analysis. (See Section 6.6, Problem 2 for a conjecture in this regard.)

In order to illustrate this concept of exhaustion on \mathbb{R}^n , we will give some examples for $n = 2$:

Examples 5.1.8 (Effective exhaustion).

- (1) $f(x_1, x_2) = \frac{1}{x_1 x_2}$. The functions looks as follows (Figure 5.1):

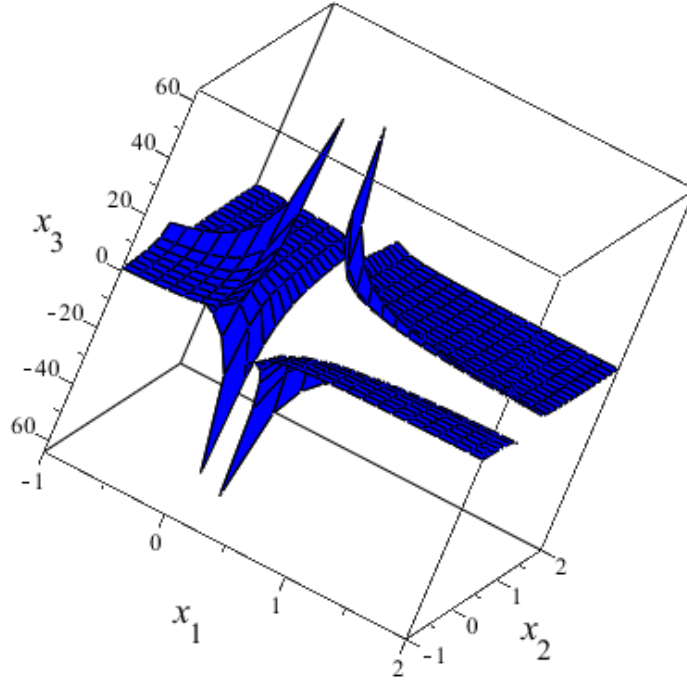


Figure 5.1: $f(x_1, x_2) = \frac{1}{x_1 x_2}$

Then (see Figure 5.2)

$$U = \mathbf{dom}(f) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 \neq 0, x_2 \neq 0\} = \bigcup_{\ell=0}^{\infty} U_{\ell}$$

where $U_{\ell} = \bigcup_{i=0}^3 E_i^{\ell}$, and

$$E_0^{\ell} = \{(x_1, x_2) \mid \frac{1}{\ell+1} < x_1 < \ell, \frac{1}{\ell+1} < x_2 < \ell\}$$

$$E_1^{\ell} = \{(x_1, x_2) \mid -\ell < x_1 < -\frac{1}{\ell+1}, \frac{1}{\ell+1} < x_2 < \ell\}$$

$$E_2^{\ell} = \{(x_1, x_2) \mid -\ell < x_1 < -\frac{1}{\ell+1}, -\ell < x_2 < -\frac{1}{\ell+1}\}$$

$$E_3^{\ell} = \{(x_1, x_2) \mid \frac{1}{\ell+1} < x_1 < \ell, -\ell < x_2 < -\frac{1}{\ell+1}\}$$

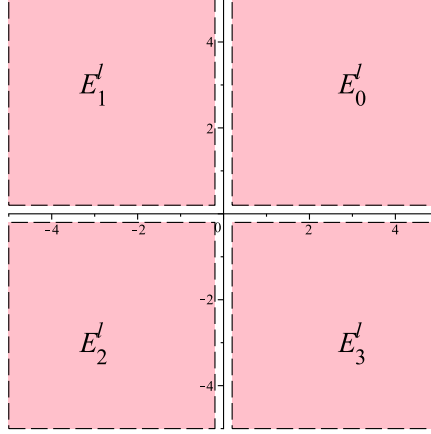


Figure 5.2: Exhaustion of domain of function $f(x_1, x_2) = \frac{1}{x_1 x_2}$

We will return to this example in Section 5.4 (multipolynomial approximations of GL-computable functions: see Example 5.4.10 page 111).

(2) $f(x, y) = \tan(\frac{\pi}{2}\sqrt{(x^2 + y^2)})$. Then (see Figure 5.3)

$$U = \mathbf{dom}(f) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \neq 2k + 1, k \in \mathbb{Z} \text{ and } k \geq 0\} = \bigcup_{\ell=0}^{\infty} U_{\ell}$$

where $U_{\ell} = \bigcup_{i=0}^{\ell} E_i^{\ell}$, and

$$E_0^{\ell} = \{(x_1, x_2) \mid x_1^2 + x_2^2 < 1 - \frac{1}{\ell + 2}\}$$

for $i > 0$,

$$E_i^{\ell} = \{(x_1, x_2) \mid x_1^2 + x_2^2 > (2 * i - 1) + \frac{1}{\ell + 2}, \quad x_1^2 + x_2^2 < (2 * i + 1) - \frac{1}{\ell + 2}\},$$

Note that the E_i^{ℓ} are not convex, or even simply connected – they are “2-dimensional rings”.

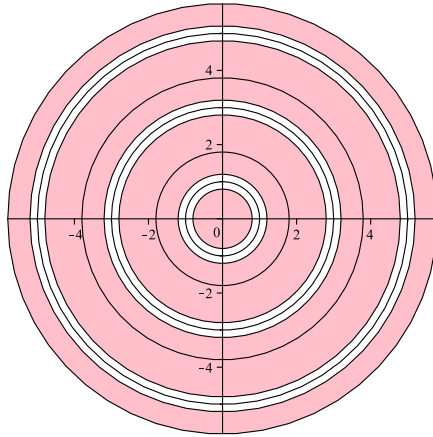


Figure 5.3: Exhaustion of domain of function $f(x, y) = \tan(\frac{\pi}{2}\sqrt{x^2 + y^2})$

Terminology 5.1.9

(i) By “point” we will mean element of \mathbb{R}^2 . We will generally denote points in \mathbb{R}^2 by x, y, \dots , with $x = (x_1, x_2)$ and $y = (y_1, y_2)$. For function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, we write $f(x) = f(x_1, x_2)$.

(ii) We write d for the Euclidean metric on the plane:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

(iii) By “rational point” we will mean a point $r = (r_1, r_2)$ where $r_1, r_2 \in \mathbb{Q}$.

We will investigate the computability of partial functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ where $\mathbf{dom}(f)$ is an open set U with an effective exhaustion (U_ℓ) .

Definition 5.1.10 (Local uniform continuity). f is *locally uniformly continuous* w.r.t (U_ℓ) iff $\forall \ell \forall \varepsilon > 0 \exists \delta > 0 \forall x, y \in U_\ell$

$$d(x, y) < \delta \implies |f(x) - f(y)| < \varepsilon.$$

Remark 5.1.11. We get an equivalent definition by stipulating that x and y are in the *same* component E_i^ℓ of U_ℓ , since (by compactness of $\overline{E_i^\ell}$) any two points in different components of U^ℓ will have a positive minimum distance.

Definition 5.1.10 can be effectivized:

Definition 5.1.12 (Effective local uniform continuity). f is *effectively locally uniformly continuous* w.r.t (U_ℓ) if there is a recursive function $M: \mathbb{N}^2 \Rightarrow \mathbb{N}$ such that for all k, ℓ and all $x, y \in U_\ell$

$$d(x, y) < 2^{-M(k, \ell)} \implies |f(x) - f(y)| < 2^{-k}.$$

Now let (f_n) be a sequence of functions $f_n: \mathbb{R}^2 \rightarrow \mathbb{R}$ with $\mathbf{dom}(f_n) = U_n$.

Definition 5.1.13 (Effectively locally uniformly continuous sequence). The sequence (f_n) is *effectively locally uniformly continuous* w.r.t (U_ℓ) if there is a recursive function $M: \mathbb{N}^3 \rightarrow \mathbb{N}$ such that for all k, ℓ, n , all $x, y \in U_\ell$, and all $n \geq \ell$,

$$d(x, y) < 2^{-M(k, \ell, n)} \implies |f_n(x) - f_n(y)| < 2^{-k}.$$

(Note that for $n \geq \ell$, $\mathbf{dom}(f_n) = U_n \supseteq U_\ell$.)

From now on, in Chapters 5 and 6, we investigate the computability properties of partial functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, and we make the following two global assumptions on f .

Assumptions 5.1.14 (Global assumptions on f). $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfies:

- (a) **Domain exhaustion:** The domain U of f is a union of an effective exhaustion (U_ℓ) , as in Definition 5.1.6;
- (b) **Continuity:** f is effectively locally uniformly continuous w.r.t (U_ℓ) (Definition 5.1.12).

These assumptions are satisfied by many functions commonly encountered in calculus and elementary real analysis. In Section 6.6 we make a conjecture connecting the applicability of these global assumptions to all *elementary functions* on \mathbb{R}^2 .

5.2 GL-computability

The following six definitions are adapted from [PER89, Ch. 0], where it was assumed that the domains of f and f_n are products of intervals: \mathbb{R}^m or $[0, 1]^m$ ($m > 0$).

Definition 5.2.1 (Computable sequence of points). A sequence of real points $(x_n) = ((x_{1n}, x_{2n}))$ is *computable* iff there exists a computable double sequence of

rational numbers $(r_{nk}) = ((r_{1nk}, r_{2nk}))$ such that for all n, k :

$$d(x_n, r_{nk}) \leq 2^{-k}$$

Definition 5.2.2 (Sequential computability of function). f is *sequentially computable* on U iff f maps every computable sequence of points $((x_{1n}, x_{2n})) \in U$ into a computable sequence $(f(x_{1n}, x_{2n}))$ of reals.

Below we assume, in addition to the Global Assumption, for sequences (f_n) of functions,

$$\text{dom}(f_n) = U_n.$$

Definition 5.2.3 (Sequential computability of sequence of functions). The sequence (f_n) of functions is *sequentially computable* w.r.t (U_ℓ) iff for any computable sequence $((x_{1m}, x_{2m}))$ of points in U_m , the double sequence $y_{mn} = (f_n(x_{1m}, x_{2m}))$ ($n \geq m$) of reals is computable, i.e. there exists a computable triple sequence (r_{mnl}) of rationals, $(r_{mnl}) \rightarrow y_{mn}$ effectively in m, n, l . In other words, there exists a recursive function $M : \mathbb{N}^3 \rightarrow \mathbb{N}$, such that for all m, n, k ,

$$N \geq M(m, n, k) \implies |r_{mnl} - f_n(x_{1m}, x_{2m})| < 2^{-k}$$

.

(Recall that for $n > m$, $U_n \supset U_m$, so that $f_n(x_{1m}, x_{2m})$ is defined.)

The variable k correspond to the error 2^{-k} , and the function $M(m, n, k)$ gives a bound on N sufficient to attain this error.

Definition 5.2.4 (GL-computability on \mathbb{R}^2). f is GL-computable w.r.t. (U_ℓ) iff:

- (1) f is sequentially computable on U (Definition 5.2.2), and
- (2) f is effectively locally uniformly continuous w.r.t. (U_ℓ) (Definition 5.1.12).

Note that condition (2) in the definition is subsumed under Global Assumption 5.1.14(b).

Definition 5.2.5 (GL-computable sequence on \mathbb{R}^2) The sequence (f_n) is GL-computable w.r.t. (U_ℓ) iff

- (1) (f_n) is sequentially computable w.r.t (U_ℓ) (Definition 5.2.3), and
- (2) (f_n) is effectively locally uniformly continuous w.r.t (U_ℓ) (Definition 5.1.13).

Definition 5.2.6 (Effective local uniform convergence). The sequence (f_n) converges to f ($f_n \rightarrow f$) *effectively locally uniformly* w.r.t (U_ℓ) iff there is a recursive function $M : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for all k, ℓ , all $x \in U_\ell$, and $n \geq \ell$

$$n \geq M(k, \ell) \Rightarrow |f_n(x) - f(x)| < 2^{-k}$$

Lemma 5.2.7 (GL-Closure). Assume $\mathbf{dom}(f_\ell) = U_\ell$, (f_n) is a GL-computable sequence w.r.t (U_ℓ) , and $f_n \rightarrow f$ *effectively locally uniformly* w.r.t (U_ℓ) . Then f is GL-computable w.r.t (U_ℓ) .

Proof:

By the assumption, we know

$$(f_n) \text{ is effective locally uniformly continuous w.r.t } (U_\ell), \quad (5.1)$$

$$(f_n) \text{ is sequentially computable w.r.t } (U_\ell), \quad (5.2)$$

$$f_n \rightarrow f \text{ effectively locally uniformly w.r.t } (U_\ell). \quad (5.3)$$

We must show that f is GL-computable.

First, we can rewrite (5.3) as:

$$\forall n, \forall x \in U_n : |f_n(x) - f(x)| < 2^{-n} \quad (5.3')$$

by effectively taking a subsequence of (f_n) . Note that (5.1) and (5.2) still hold for this subsequence.

Now given a computable sequence $(x_m) = ((x_{1m}, x_{2m}))$ of points in U : we must show $(f(x_m))$ is a computable sequence.

By Definition 5.2.1, there exists a computable double sequence of rational points $(r_{mk}) = ((r_{1mk}, r_{2mk}))$ such that for all m , $r_{mk} \rightarrow x_m$.

We can assume (by taking an effective subsequence of (r_{mk})) that

$$\forall m, k : d(r_{mk}, x_m) < 2^{-k} \quad (5.4)$$

Note that for all m, ℓ , if $x_m \in E_i^\ell$, we can effectively find $\delta > 0$ s.t. $B(x_m, \delta) \subseteq E_i^\ell$.

Hence we can find $k, \ell = \ell_m$ such that

$$N(r_{mk}, 2^{-2k+1}) \subset U_{\ell_m}.$$

Then for all j ,

$$j \geq k \implies r_{mj} \in U_{\ell_m},$$

and so,

$$x_m \in U_{\ell_m}. \tag{5.5}$$

Now taking the subsequence $V_m = U_{\ell_m}$, note that (5.1) and (5.3) still hold for this subsequence, with U_{ℓ_m} replaced by V_m , and the subsequence $(f_n)_{n \geq m}$.

We have from (5.5) for all m :

$$x_m \in V_m \tag{5.6}$$

and for all m, k :

$$r_{mk} \in V_m \tag{5.7}$$

Let $y_{mn} = f_n(x_m)$ ($n \geq m$), and $y_m = f(x_m)$. By (5.7), (5.2) (and from Definition 5.2.3) the sequence

$$(y_{m,m}, y_{m,m+1}, \dots) \quad (m = 0, 1, 2, \dots)$$

is a computable sequence of reals. We must now show (y_m) is a computable sequence of reals.

By (5.3'):

$$\forall m, n \geq m : |f_n(x_m) - f(x_m)| < 2^{-n} \quad (5.8)$$

i.e.

$$\forall m, n \geq m : |y_{mn} - y_m| < 2^{-n}. \quad (5.9)$$

Also by (5.2) and Definition 5.2.3, there exists a computable triple sequence of rationals (s_{mnN}) : $(n \geq m)$ such that

$$s_{mnN} \rightarrow y_{mn} \text{ effectively in } m, n, N$$

,

i.e. there exists a recursive function $M : \mathbb{N}^3 \rightarrow \mathbb{N}$ such that

$$\forall k, m, n \geq m : N \geq M(m, n, k) \implies |s_{mnN} - y_{mn}| < 2^{-k} \quad (5.10)$$

Putting $N = M(m, n, k)$, by (5.10) we get

$$\forall k, m, n \geq m : |s_{m,n,M(m,n,k)} - y_{mn}| < 2^{-k} \quad (5.11)$$

Put $t_{m,k} = s_{m,k,M(m,k,k)}$ (i.e. put $n = k$). Then (t_{mk}) is computable double sequence of rationals.

Then for $k \geq m$, by (5.11),

$$|t_{mk} - y_{mk}| < 2^{-k} \quad (5.12)$$

and (again for $(k \geq m)$): by (5.9) and (5.12),

$$|t_{mk} - y_m| \leq |t_{mk} - y_{mk}| + |y_{mk} - y_m| < 2^{-k} + 2^{-k} = 2^{-(k+1)}.$$

Hence (y_m) is a computable sequence of reals. That is, for any $x_m \in U$, $y_m = f(x_m)$ is computable. So (by Definition 5.2.2) f is sequentially computable on U .

Further, by the global assumption, we know that f is effectively locally uniformly continuous w.r.t (U_ℓ) .

Hence, finally, f is GL-computable w.r.t. (U_ℓ) . \square

5.3 Multipolynomial approximability

Definition 5.3.1 (Multipolynomial). Given a finite sequence of (\mathbb{R}_c) -polynomials (p_1, p_2, \dots, p_k) and a sequence of elementary sets (E_1, E_2, \dots, E_k) with disjoint closures, we define an (\mathbb{R}_c) -multipolynomial $q(x)$ (where $x = (x_1, x_2)$) with domain $\bigcup_{i=1}^k \overline{E_i}$ as follows:

$$q(x) = \begin{cases} p_1(x) & \text{if } x \in \overline{E_1} \\ p_2(x) & \text{if } x \in \overline{E_2} \\ \dots & \\ p_k(x) & \text{if } x \in \overline{E_k} \\ \uparrow & \text{otherwise.} \end{cases}$$

We denote this multipolynomial as

$$q = [p_1 \upharpoonright \overline{E_1}, \dots, p_k \upharpoonright \overline{E_k}].$$

Definition 5.3.2 (Effective sequence of multipolynomials). Given an effective exhaustion (U_ℓ) of U , with $U_\ell = E_1^\ell \cup \dots \cup E_{k_\ell}^\ell \subseteq \mathbb{R}^2$, where the $\overline{E_i^\ell}$ are disjoint, and an effective sequence of polynomials (i.e. effective in their numerical codes) $(p_1^\ell, \dots, p_{k_\ell}^\ell)$ ($\ell=0, 1, 2, \dots$), the sequence (q_ℓ) , where $q_\ell = [p_1^\ell \upharpoonright \overline{E_1^\ell}, \dots, p_{k_\ell}^\ell \upharpoonright \overline{E_{k_\ell}^\ell}]$, is called an *effective sequence* of multipolynomials.

Definition 5.3.3 (Effective local uniform multipolynomial approximability).

Given $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, and an effective exhaustion (U_ℓ) of $U = \mathbf{dom}(f)$, where $U_\ell = E_1^\ell \cup \dots \cup E_{k_\ell}^\ell$, we say that the effective sequence of multipolynomials (q_ℓ) , where

$$q_\ell = [p_1^\ell \upharpoonright \overline{E_1^\ell}, \dots, p_{k_\ell}^\ell \upharpoonright \overline{E_{k_\ell}^\ell}],$$

converges to f ($q_\ell \rightarrow f$) *effectively locally uniformly* w.r.t (U_ℓ) if there is a recursive function $M : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for all k, ℓ, n , and all $x \in U_\ell$:

$$n \geq M(k, \ell) \Rightarrow |q_n(x) - f(x)| < 2^{-k}.$$

We also say that f is *effectively locally multipolynomially approximable* by (q_ℓ) w.r.t. (U_ℓ) .

5.4 Equivalence between GL-computability and multipolynomial approximability on \mathbb{R}^2

In this section we prove, under the Global Assumptions, Equivalence Lemma 1:

$$\text{“GL-computability} \iff \text{multipolynomial approximability”} \quad (5.13)$$

In preparation for the “ \implies ” direction, we prove an “Extension Theorem”, which is crucial for the application of the Weierstrass approximation theorem.

Definition 5.4.1 (Containing rectangle). A *containing rectangle* K of an elementary set E is a closed rectangle K s.t. $\overline{E} \subseteq K$ (see Figure 5.4).

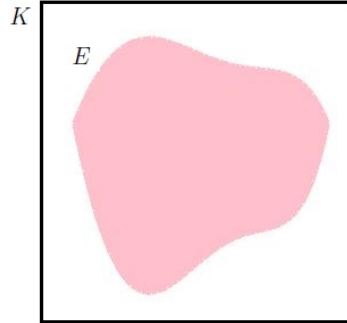


Figure 5.4: Containing rectangle

Theorem 4 (Extension Theorem). Let $E \subseteq \mathbb{R}^2$ be an elementary set. Let K be a containing rectangle of E . Let f be a GL-computable function on \overline{E} . Then f can be extended to a GL-computable \widehat{f} on K .

Proof. We first give a rough idea of the proof (see Figure 5.5):

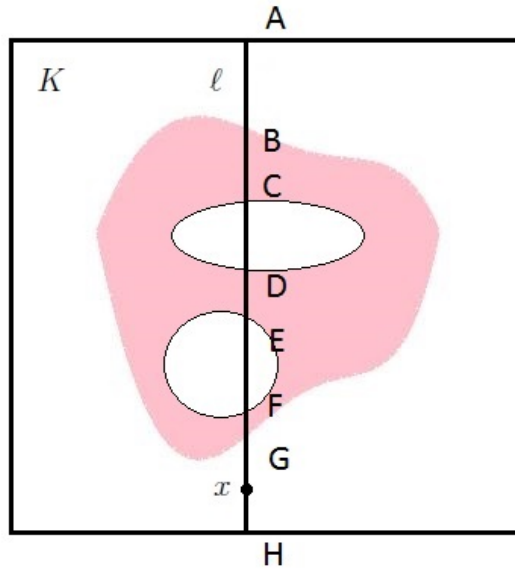


Figure 5.5: Extend f to \hat{f}

We must define $\hat{f}(x)$ for any point $x \in K \setminus \bar{E}$.

Note first that $x = (x_1, x_2)$ is on a vertical line ℓ (Figure 5.5). This line ℓ can be divided into a finite number of disjoint open segments ($AB, BC, CD, DE, EF, FG, GH$ in Figure 5.5), each one being either completely inside, or completely outside E .

Depending on which segment x is in, we define $\hat{f}(x)$ by either *linear interpolation* (for x in CD or EF) or *constant extrapolation* (for x in AB or GH), e.g. for $x \in AB$, define $\hat{f}(x) = f(B)$. In this way we can (apparently) extend f on \bar{E} to \hat{f} on K , maintaining continuity and GL-computability.

There is however a problem with this method,

Consider the following situation (Figure 5.6)

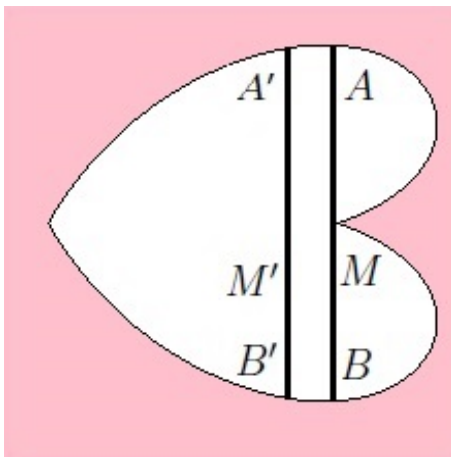


Figure 5.6: A hole with LLM M

with M a “local left maximum” on ∂E^1 and M' arbitrarily near M . Then \widehat{f} (defined by linear interpolation on $A'B'$) may be discontinuous at M . For suppose $f(M) = 1$, but $f(A) = f(A') = f(B) = f(B') = 0$, then $\widehat{f}(M') = 0$, no matter how close M' is to M .

We now describe how to deal with such problems.

First, we define a *local left maximum* (LLM) as a point $(a_1, a_2) \in \partial E$ such that for some $\delta > 0$, and all $(x_1, x_2) \in \partial E$:

$$(a) \quad |x_2 - a_2| < \delta \implies x_1 \geq a_1, \text{ and also} \tag{5.14a}$$

$$(b) \quad \textit{either} \quad a_2 < x_2 < a_2 + \delta \implies x_1 > a_1 \tag{5.14b}$$

$$\textit{or} \quad a_2 - \delta < x_2 < a_2 \implies x_1 > a_1$$

Clause (b) says that (a_1, a_2) must be a *strict* local left maximum *either* from above *or* from below (or both). The reason for this clause is that in its absence, any point on

¹Where ∂E is the boundary of E ($= \overline{E} \setminus E$ since E is open).

a vertical edge would count as an LLM. This is explained in more detail in Remark 5.4.2 (page 95). Local right maxima (LRM's) are defined similarly.

Back to the problem shown in Figure 5.6: The solution to this problem is to extend f to \hat{f} on K in two steps.

Step 1:

At each LLM M on ∂E , draw a *level line* ℓ extending from M to the *left*, either to ∂E (Figure 5.7) or to ∂K . (Figure 5.8):

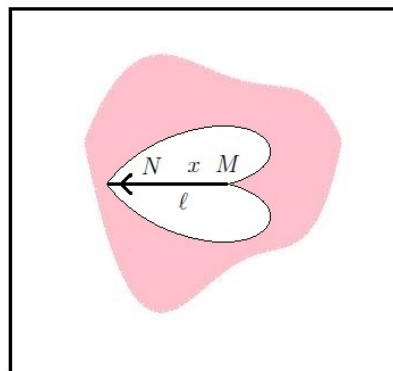


Figure 5.7: A line from an LLM point M of ∂E to the left, ending in ∂E

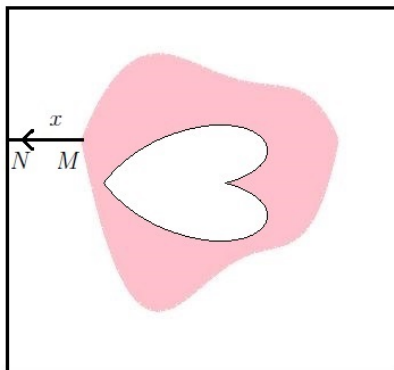


Figure 5.8: A line from one LLM point M of ∂E to the left, ending in ∂K

Define $\tilde{f}(x)$ for x in the segment MN either by linear interpolation of f on M and N (as in Figure 5.7) or by constant extrapolation of f from M to N . (as in Figure 5.8).

Do this for each LLM of ∂E . Similarly, for each LRM (local right maximum) M , define $\tilde{f}(x)$ on the level line ℓ extending from M to the right, and define \tilde{f} on ℓ as with the LLM's.

We should note, in this connection, that there are only finitely many LLM's and LRM's on ∂E (see Discussion 5.4.3).

Let ℓ_1, \dots, ℓ_k be all the level lines obtained from local maximum in this way, and let

$$L = \overline{E} \cup \bigcup_{i=1}^k \ell_i,$$

By the above construction, we have extended f to a function \tilde{f} on L , which is

clearly continuous and GL-computable on L .

For example (see Figure 5.9):

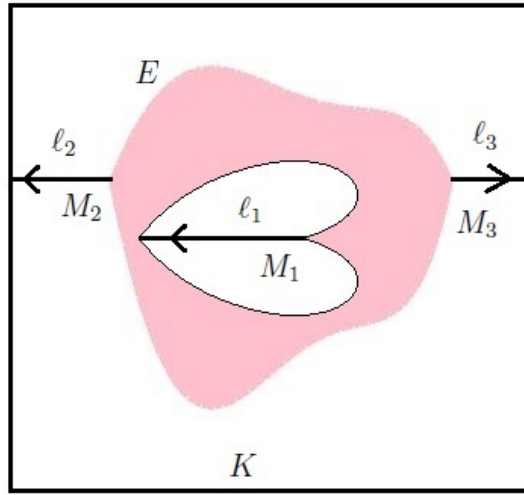


Figure 5.9: Extending f on E to \tilde{f} on L

Here ℓ_1 and ℓ_2 are level lines from the LLM's M_1 and M_2 respectively, and ℓ_3 is a level line from the LRM M_3 . \tilde{f} is defined by linear interpolation on ℓ_1 and by constant extrapolation on ℓ_2 and ℓ_3 .

Step 2:

We must extend \tilde{f} to a continuous GL-computable function \hat{f} on K .

So let x be a point in $K \setminus L$, and consider the *vertical* line ℓ through x . This line ℓ is partitioned into a finite number of segments, formed by each crossing of ℓ with *either* ∂E *or* a level line.

This is best shown by an example (Figure 5.10).

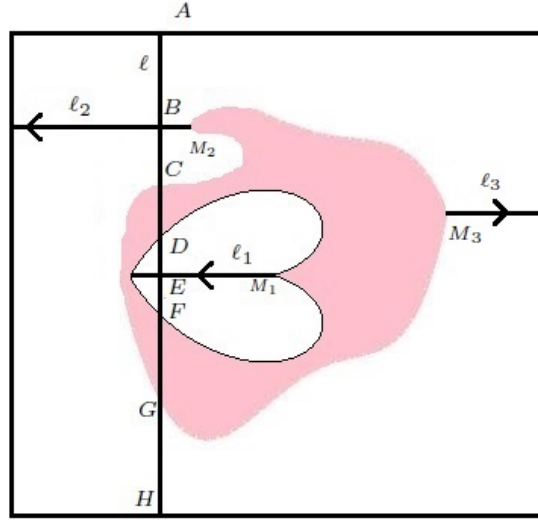


Figure 5.10: Extending f on E to \hat{f} on K

There are two level lines ℓ_1 and ℓ_2 , extending from LLM's M_1 and M_2 respectively, and a level line ℓ_3 extending from LRM's M_3 .

The vertical line ℓ is divided into the segments AB , BC , CD , DE , EF , FG , GH .

Then \hat{f} is defined on the segments BC , DE and EF by *linear interpolation*. E.g. for $x \in DE$, $\hat{f}(x)$ is defined by linearly interpolating the values of \tilde{f} at D and E .

And \hat{f} is defined on the segments AB and GH by *constant extrapolation*. E.g. for $x \in AB$, $\hat{f}(x) = \tilde{f}(x)$.

In this way, the problem with discontinuity has been avoided, since (returning to our example in Figure 5.6) there is now a level line ℓ_M extending left from the LLM

at M (Figure 5.11):

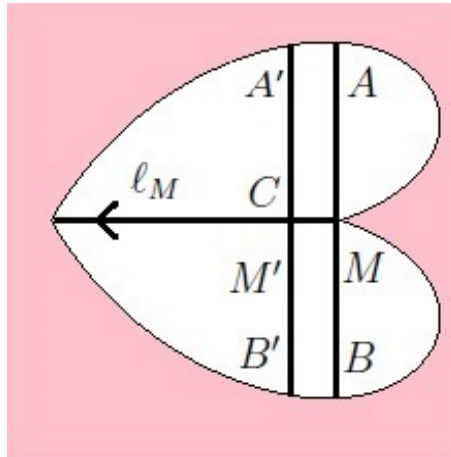


Figure 5.11: Maintaining the continuity at M'

Now $\hat{f}(M')$ is obtained by linearly interpolating \tilde{f} between B' and C , (not between B' and A') giving

$$\hat{f}(M') \approx \tilde{f}(C) \approx f(M) = 1$$

as desired.

In this way, we obtain an extension \hat{f} of f to K , which is continuous. It is also GL-computable on K , by an extension of the patching theorem [PER89, p. 32] to 2 dimensions. □

Remark 5.4.2. To explain the need for clause (b) in the definition (5.14) of LLM's:

In its absence, every point on a vertical segment, e.g. AB in Figure 5.12, would be an LLM! With clause (b) only the two points, A and B , are LLM's, with a level

line from each of them, as we would wish:

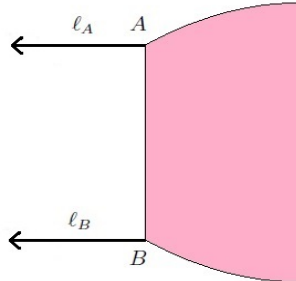


Figure 5.12: With a level line from each of them (LLM's points A and B)

Similarly, in Figure 5.13:

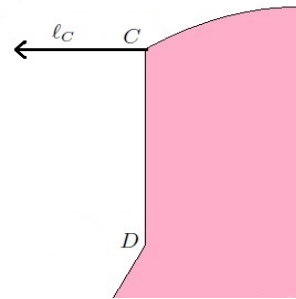


Figure 5.13: With a level line from LLM's point C

Only point C is an LLM, with level line ℓ_c , as desired.

Discussion 5.4.3 (On the proof of the Extension Theorem). In order to ensure that the above proof is quite correct, we must check certain steps to ensure that we are not being misled by our geometric intuition. We note the following.

Let E be an elementary set, i.e. a bounded, connected basic subset of \mathbb{R}^2 . Let $B = \partial E$. Then

- (1) B is composed of finitely many algebraic curves (i.e curves given by polynomial equations in x_1 and x_2).
- (2) E has only finitely many holes.
- (3) Any straight line ℓ crosses B only finitely many times. (If ℓ and B coincide for part of their lengths, that counts as a single crossing).
- (4) B has only finitely many LLM's and LRM's (local left and right maxima).
- (5) Each LLM and LRM is a computable point.
- (6) It is decidable whether any given computable point on B is an LLM (or LRM).
- (7) All LLM's and LRM's on B can be effectively located (from (4), (5) and (6)).

The results above can be proved from the cell decomposition theorem [vdD98, Ch.3] applied to E and B .

This applies particularly to the “finiteness” results (2),(3),(4),² which justify (e.g) the division of the vertical line ℓ in Figure 5.10 into finitely many segments, each either completely inside, or completely outside, E .

Discussion 5.4.4 (Definition of GL-computable functions on compact domains). In the Extension Theorem above, we assumed that the functions were GL-computable on compact sets C (i.e. \overline{E} , or \overline{E} with level lines, or K). Strictly

²We thank Prof. Lou van den Dries for clarifying this.

speaking, we have not defined GL-computability of a function f for such domains C .

We can do so by modifying Definition 5.2.4 (page 81) as follows: In (1), replace ‘ U ’ by ‘ C ’, and replace (2) by (2’): f is effectively uniformly continuous on C .

Then our assertion that f (as given) is GL-computable on \overline{E} assumes that \overline{E} is a subset of U (the union of the given exhaustion (U_ℓ)). But this follows from clause (iii) of our definition (5.1.3) (page 71) of exhaustion, (see Remark 5.1.5, page 71), since $E = E_i^\ell$ for some ℓ, i , and so

$$\overline{E} = \overline{E_i^\ell} \subset E_j^{\ell+1} \quad (\text{for some } j) \subset U$$

Remark 5.4.5 (Other Proofs of the Extension Theorem). The Tietze Extension Theorem [Kel55] states that if X is a normal topological space and A is any closed subset of X and $f : A \rightarrow \mathbb{R}$ is continuous, then f can be extended to a continuous function $\widehat{f} : X \rightarrow \mathbb{R}$.

The proof in [Kel55] is, as it stands, quiet non-constructive³. A constructive version is given by Bishop and Bridges [BB85] for the case that X is a metric space, and A is a locally compact subset of X .

In principle we could use their construction for our Extension Theorem (Theorem 4). Clearly, however our construction, for the special case that X is a rectangle in \mathbb{R}^2 and A is the closure of a bounded, connected basic set, is much simpler than the

³At least it appears so.

construction given by the proof in [BB85].

The following Theorem is an adaption to two dimensions of the classical, effective form of the Weierstrass Theorem for unary functions defined on an interval [PER89, Ch.0, Sec.7].

Theorem 5 (Effective Weierstrass Theorem for \mathbb{R}^2). Let D be the rectangle $[-1/4, 1/4]^2 \in \mathbb{R}^2$, and let f be a function on D which is GL-computable. Then there exists an effective sequence of polynomials p_n which converges effectively and uniformly to f on D .

First we give an outline of the proof. It follows Weierstrass's proof ([Rud76, Thm 7.26], [PER89, Thm. p.45]) adapted to two dimensions. Briefly, it proceeds by assuming first that (by use of the Extension Theorem) the function f is defined on the unit square $I = [-1, 1]^2$. We next construct a sequence $(P_m(x_1, x_2))$ (eqn (5.15) below) of polynomial "pulse functions" which are "large" (≈ 1) within a small distance d of $(0,0)$ and "small" (≈ 0) beyond that on I .

By convoluting the P_m with f (eqn (5.24)) we obtain a sequence $(p_m(x_1, x_2))$ of polynomials which approximate f uniformly on I , to any desired degree of accuracy (eqn (5.34) below).

We turn to the precise proof. To facilitate comparison with the 1-dimensional case, we follow the notation of [PER89] where possible.

Proof of the effective Weierstrass Theorem for \mathbb{R}^2 .

First, we can extend f from D to $I = [-1, 1] \times [-1, 1]$ by a simple constant extrapolation from ∂D to I (cf. [PER89, Thm.3, p.33]). Let d be a large integer (greater than 4) to be specified later.

We define:

$$I = \{(x_1, x_2) \mid |x_1|, |x_2| \leq 1\},$$

$$J = \{(x_1, x_2) \in I \mid |x_1|, |x_2| \geq \frac{1}{d}\},$$

$$K = \{(x_1, x_2) \in I \mid |x_1|, |x_2| \leq \frac{1}{2d}\}.$$

the sets I, J, D, K are as follows (Figure 5.14):

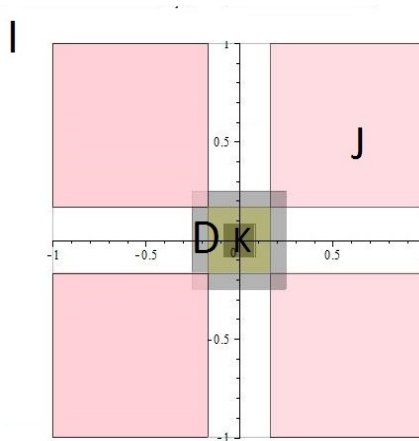


Figure 5.14: I, J, D and K

We define a polynomial “pulse function” $P_m(x_1, x_2)$ by:

$$P_m(x_1, x_2) = [(1 - x_1^2)(1 - x_2^2)]^m. \tag{5.15}$$

Then on J ,

$$P_m(x_1, x_2) \leq \left\{ \left[1 - \left(\frac{1}{d} \right)^2 \right] \left[1 - \left(\frac{1}{d} \right)^2 \right] \right\}^m = \left(1 - \frac{1}{d^2} \right)^{2m} \quad (5.16)$$

and on K :

$$P_m(x_1, x_2) \geq \left\{ \left[1 - \left(\frac{1}{2d} \right)^2 \right] \left[1 - \left(\frac{1}{2d} \right)^2 \right] \right\}^m = \left(1 - \frac{1}{4d^2} \right)^{2m} \quad (5.17)$$

We want to compare the ratio of (5.16) and (5.17). Putting $u=1/4d^2$, we rewrite (5.16) and (5.17) as follows:

On I :

$$P_m(x_1, x_2) \leq (1 - 4u)^{2m} \quad (5.18)$$

and on K :

$$P_m(x_1, x_2) \geq (1 - u)^{2m} \quad (5.19)$$

We will compare the ratio of (5.18) and (5.19), note that for $k = 1, 2, \dots$

$$(1 + u)^k \geq 1 + ku \quad (5.20)$$

$$(1 - u)^k \geq 1 - ku \tag{5.21}$$

by induction on k . Hence (by (5.21) with $k=4$)

$$\begin{aligned} \frac{1 - u}{1 - 4u} &\geq \frac{1}{(1 - u)^3} \\ &\geq (1 + u)^3 \text{ since } ((1 + u)^3(1 - u)^3 = (1 - u^2)^3 \\ &\geq 1 + 3u \quad (\text{by (5.20) with } k=3) \end{aligned} \tag{5.22}$$

Hence,

$$\begin{aligned} \frac{\text{r.h.s. of (5.17)}}{\text{r.h.s. of (5.16)}} &= \left(\frac{1 - u}{1 - 4u}\right)^{2m} \\ &\geq (1 + 3u)^{2m} && \text{by (5.22)} \\ &\geq 1 + 6mu && \text{by (5.20)} \\ &> 6mu = \frac{3m}{2d^2} \end{aligned} \tag{5.23}$$

Let

$$p_m(x_1, x_2) = \frac{1}{C_m} \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 \tag{5.24}$$

where C_m is the “normalizing constant”,

$$C_m = \int_{-1}^1 \int_{-1}^1 P_m(u_1, u_2) du_1 du_2$$

then also (putting $u_1 = t_1 - x_1$, $u_2 = t_2 - x_2$):

$$C_m = \int_{x_2-1}^{x_2+1} \int_{x_1-1}^{x_1+1} P_m(t_1 - x_1, t_2 - x_2) dt_1 dt_2$$

Hence

$$f(x_1, x_2) = \frac{1}{C_m} \int_{x_2-1}^{x_2+1} \int_{x_1-1}^{x_1+1} P_m(t_1 - x_1, t_2 - x_2) f(x_1, x_2) dt_1 dt_2 \quad (5.25)$$

So,

$$p_m(x_1, x_2) - f(x_1, x_2) = \frac{1}{C_m} \left[\int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 - \int_{x_2-1}^{x_2+1} \int_{x_1-1}^{x_1+1} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 \right]$$

We write p_m as follows:

$$p_m(x_1, x_2) - f(x_1, x_2) = (A) + (B) + (C) \quad (5.26)$$

where,

$$(A) = p_m(x_1, x_2) - (B1) = \frac{1}{C_m} \left[\int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 - \int_{x_2-\frac{1}{d}}^{x_2+\frac{1}{d}} \int_{x_1-\frac{1}{d}}^{x_1+\frac{1}{d}} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 \right],$$

$$(B) = (B1) - (B2)$$

$$= \frac{1}{C_m} \left[\int_{x_2 - \frac{1}{d}}^{x_2 + \frac{1}{d}} \int_{x_1 - \frac{1}{d}}^{x_1 + \frac{1}{d}} P_m(t_1 - x_1, t_2 - x_2) [f(t_1, t_2) - f(x_1, x_2)] dt_1 dt_2 \right], \quad (5.27)$$

and

$$(B1) = \frac{1}{C_m} \int_{x_2 - \frac{1}{d}}^{x_2 + \frac{1}{d}} \int_{x_1 - \frac{1}{d}}^{x_1 + \frac{1}{d}} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2$$

$$(B2) = \frac{1}{C_m} \int_{x_2 - \frac{1}{d}}^{x_2 + \frac{1}{d}} \int_{x_1 - \frac{1}{d}}^{x_1 + \frac{1}{d}} P_m(t_1 - x_1, t_2 - x_2) f(x_1, x_2) dt_1 dt_2$$

$$(C) = (B2) - f(x_1, x_2) = \frac{1}{C_m} \left[\int_{x_1 - \frac{1}{d}}^{x_1 + \frac{1}{d}} \int_{x_2 - \frac{1}{d}}^{x_2 + \frac{1}{d}} P_m(t_1 - x_1, t_2 - x_2) f(x_1, x_2) dt_1 dt_2 - \int_{x_2 - 1}^{x_2 + 1} \int_{x_1 - 1}^{x_1 + 1} P_m(t_1 - x_1, t_2 - x_2) f(t_1, t_2) dt_1 dt_2 \right].$$

Here, $(t_1 - x_1, t_2 - x_2) \in I$, and I contains $[-1/2, 1/2] \times [-1/2, 1/2]$. Also,

$$[x_1 - 1/d, x_1 + 1/d] \times [x_2 - 1/d, x_2 + 1/d] \subseteq [-1/2, 1/2] \times [-1/2, 1/2]$$

. In both (A) and (C), note that

$$(t_1 - x_1, t_2 - x_2) \in J,$$

and

$$C_m = \int_{-1}^1 \left(\int_{-1}^1 P_m(t_1, t_2) dt_1 \right) dt_2$$

.

Now, we consider the ratio:

$$\frac{\int \int_J P_m(x_1, x_2) dx_1 dx_2}{\int \int_I P_m(x_1, x_2) dx_1 dx_2}.$$

Since $\int \int_I P_m(x_1, x_2) dx_1 dx_2 \geq \int \int_K P_m(x_1, x_2) dx_1 dx_2$

then,

$$\frac{\int \int_J P_m(x_1, x_2) dx_1 dx_2}{\int \int_I P_m(x_1, x_2) dx_1 dx_2} \leq \frac{\int \int_J P_m(x_1, x_2) dx_1 dx_2}{\int \int_K P_m(x_1, x_2) dx_1 dx_2} \quad (5.28)$$

By (5.23) above, the ratio

$$\frac{\sup \text{ of } P_m(x_1, x_2) \text{ on } J}{\inf \text{ of } P_m(x_1, x_2) \text{ on } K} \leq \frac{2d^2}{3m}$$

And the ratio

$$\frac{\text{area of } J}{\text{area of } K} = 4(1 - (\frac{1}{d})^2) / \frac{1}{4d^2} \leq 16d^2.$$

Note that

$$\int \int_J P_m(x_1, x_2) dx_1 dx_2 \leq \sup \text{ of } P_m(x_1, x_2) \text{ on } J \times \text{area of } J$$

and

$$\int \int_K P_m(x_1, x_2) dx_1 dx_2 \geq \inf \text{ of } P_m(x_1, x_2) \text{ on } K \times \text{area of } K.$$

Then we get:

$$\frac{\int \int_J P_m(x_1, x_2) dx_1 dx_2}{\int \int_K P_m(x_1, x_2) dx_1 dx_2} \leq \frac{32d^4}{3m}. \quad (5.29)$$

By (5.28) and (5.29), we have:

$$\frac{\int \int_J P_m(x_1, x_2) dx_1 dx_2}{\int \int_I P_m(x_1, x_2) dx_1 dx_2} \leq \frac{32d^4}{3m} \quad (5.30)$$

Now, we can consider (A) and (C) above.

Hence, letting S be an (effective) upper bound of $f(x)$ for $x \in D$:

$$|A| \leq S \frac{32d^4}{3m}$$

$$|C| \leq S \frac{32d^4}{3m}$$

For (B), because f is effectively uniformly continuous, there is a recursive function $d(N)$ such that

$$\mathbf{d}(x, y) \leq \frac{1}{d(N)} \Rightarrow |f(x_1, x_2) - f(y_1, y_2)| \leq \frac{1}{3} 2^{-N},$$

where $x = (x_1, x_2), y = (y_1, y_2)$.

Hence, (cf. equations (5.27)) for $(t_1, t_2) \in [x_1 - \frac{1}{d}, x_1 + \frac{1}{d}] \times [x_2 - \frac{1}{d}, x_2 + \frac{1}{d}]$,

$$|f(t_1, t_2) - f(x_1, x_2)| \leq \frac{1}{3}2^{-N}$$

,

where $d = d(N)$, i.e. $d(t, x) \leq \frac{1}{d(N)}$. Then, we get,

$$|B| \leq \frac{1}{3}2^{-N} \tag{5.31}$$

Now, we can define the recursive function:

$$m(N) = 32Sd(N)^4 2^N$$

so that

$$S \frac{32d(N)^4}{3m(N)} = \frac{1}{3}2^{-N}$$

then for $m \geq m(N)$,

$$|A| \leq \frac{1}{3}2^{-N} \tag{5.32}$$

and

$$|C| \leq \frac{1}{3}2^{-N}. \tag{5.33}$$

Hence, by (5.26), (5.31), (5.32) and (5.33), for all $(x_1, x_2) \in [-1/4, 1/4] \times [-1/4, 1/4]$

$$|p_m(x_1, x_2) - f(x_1, x_2)| \leq 2^{-N} \tag{5.34}$$

□

Remark 5.4.6 (Other Proofs of the Effective Weierstrass Theorem).

As stated above, our proof of Theorem 5 and the construction of an effective approximating polynomial sequence to a function defined on a *closed rectangle*, is an adaptation to 2 dimensions of the construction given in [PER89, Ch. 0, p.45, Sec. 7], for a function f defined on a *closed interval*, which essentially follows Weierstrass's original proof, involving convoluting f with a sequence of pulse functions. The Weierstrass Theorem was generalized by M.N. Stone to the case of a function defined on a compact topological space X , with the algebra of polynomial functions replaced by more general subalgebras of the algebra $C(X)$ of continuous functions on X [Kel55, Rud76].

Constructive versions of the Stone-Weierstrass theorem have also been given in Bishop and Bridges [BB85, p.105], and Banaschewski and Mulveyb [BM97] for X a compact space. In principle, we could use their construction for Theorem 5 directly on the closure of our elementary set E , without the need for an Extension Theorem (Theorem 4). However, our effective version clearly gives a much simpler method for computing the effective polynomial sequence (cf. Remark 5.4.3), as illustrated below

(Example 5.4.10).

Corollary 5.4.7. Let $K = [a_1, b_1] \times [a_2, b_2]$ be a rectangle in \mathbb{R}^2 , and let $f : K \rightarrow \mathbb{R}$ be GL-computable. Then there exists an effective sequence of polynomials p_n which converges effectively uniformly to f on K .

Proof. First, let $D = [-1/4, 1/4] \times [-1/4, 1/4]$. We use a linear transformation

$\varphi : K \rightarrow D$ with $\varphi(x_1, x_2) = (x'_1, x'_2)$ where

$$2x'_1 + \frac{1}{2} = \frac{x_1 - a_1}{b_1 - a_1}$$

$$2x'_2 + \frac{1}{2} = \frac{x_2 - a_2}{b_2 - a_2}$$

Define $f' : D \rightarrow \mathbb{R}$ by $f'(x'_1, x'_2) = f(x_1, x_2)$.

By the composition Lemma [PER89, p.28, Theorem 1], we know that $f' = f \circ \varphi^{-1}$ is GL-computable.

By the Weierstrass Theorem 5, there exists an effective sequence of polynomials p'_n which converges effectively uniformly to f' on D .

Hence, using the linear transformation again, and letting $p_n = p'_n \circ \varphi$, we have an effective sequence of polynomials p_n which converges effectively uniformly to f on K . □

By combining the above with the Extension Theorem, we obtain

Corollary 5.4.8. Let E be an elementary set in \mathbb{R}^2 , and let f be a function on \overline{E} which is GL-computable. Then there exists an effective sequence of polynomials p_n which converges effectively uniformly to f on \overline{E} .

Proof.

We first extend the GL-computable function f on \overline{E} to a GL-computable function \hat{f} on a rectangle K containing \overline{E} by the Extension Theorem (Theorem 4). Then by Corollary 5.4.7, we get an effective sequence of polynomials which converges effectively uniformly to \hat{f} on K . Since $f = \hat{f} \upharpoonright \overline{E}$, this gives an effective sequence of polynomials which converges effectively uniformly to f on \overline{E} . \square

We can now prove the “ \implies ” direction of the Equivalence Lemma ((5.13) in Section 5.4):

Lemma 5.4.9 Suppose f is GL-computable w.r.t. (U_ℓ) . Then f is effectively locally uniformly multipolynomially approximable w.r.t. (U_ℓ) .

Proof: Suppose, for each ℓ , U_ℓ has components $E_1^\ell, \dots, E_{k_\ell}^\ell$. For each ℓ and $i = 1, \dots, k_\ell$, apply Corollary 5.4.8, to get a polynomial p_i^ℓ which approximates $f \upharpoonright \overline{E_i^\ell}$ uniformly on $\overline{E_i^\ell}$ by $\leq 2^{-\ell}$, *i.e.*

$$\forall x \in \overline{E_i^\ell} (|p_i^\ell(x) - f(x)| \leq 2^{-\ell}).$$

The above procedure is effective in ℓ and i .

Hence, defining the multipolynomial

$$q_\ell = [p_1^\ell \upharpoonright \overline{E_1^\ell}, \dots, p_{k_\ell}^\ell \upharpoonright \overline{E_{k_\ell}^\ell}],$$

we have an effective locally uniform multipolynomial approximation (q_ℓ) of f . \square

Recall our global assumptions (5.1.14) on $f : \mathbb{R}^2 \rightarrow \mathbb{R}$.

Example 5.4.10 (Multipolynomial approximations).

Consider (cf. Example 5.1.8 (1), page 74) the function f with domain U , where

$$f(x_1, x_2) = \frac{1}{x_1 x_2},$$

$$U = \mathbf{dom}(f) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 \neq 0, x_2 \neq 0\} = \bigcup_{\ell=0}^{\infty} U_\ell$$

where $U_\ell = \bigcup_{i=1}^4 E_i^\ell$, and let

$$E_1^\ell = \{(x_1, x_2) \mid \frac{1}{\ell+10} < x_1 < \ell + 2, \quad \frac{1}{\ell+10} < x_2 < \ell + 2 \}$$

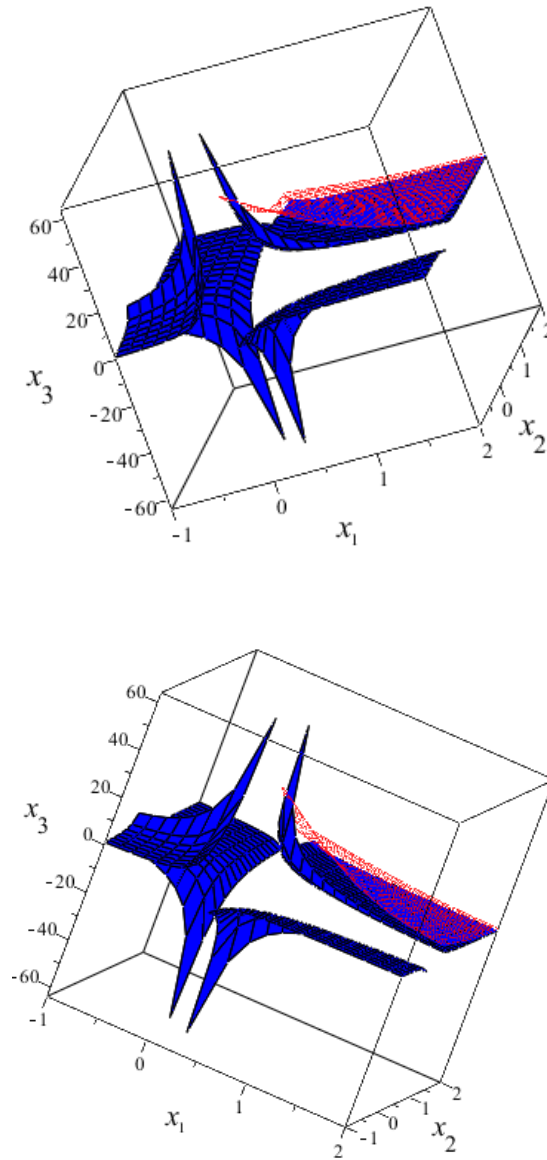
$$E_2^\ell = \{(x_1, x_2) \mid -\ell - 10 < x_1 < -\frac{1}{\ell+2}, \quad \frac{1}{\ell+10} < x_2 < \ell + 2 \}$$

$$E_3^\ell = \{(x_1, x_2) \mid -\ell - 10 < x_1 < -\frac{1}{\ell+2}, \quad -\ell - 10 < x_2 < -\frac{1}{\ell+2} \}$$

$$E_4^\ell = \{(x_1, x_2) \mid \frac{1}{\ell+10} < x_1 < \ell + 2, \quad -\ell - 10 < x_2 < -\frac{1}{\ell+2} \}$$

In Figure 5.15, we show the multipolynomial constructed according to equation 5.24 (page 102), taking $m = 50$. For the sake of clarity, only the polynomial on the 1st quadrant of stage U_0 , i.e, $E_1^0 = [\frac{1}{10}, 2]^2$, is shown.

The views have been generated using Maple 15.01.





$x_3 = \frac{1}{x_1 x_2}$	
$x_3 = p_{50}(x_1, x_2)$	

Figure 5.15: Two views of multipolynomial approximation $p_m(x_1, x_2)$ of $f(x_1, x_2) = \frac{1}{x_1 x_2}$, from eqn (5.24) with $m = 50$, on the 1st quadrant $E_1^0 = \left[\frac{1}{10}, 2\right]^2$

We can now prove the equivalence (5.13).

Equivalence Lemma 1 (Multipolynomial approx. and GL comp.).

The following are equivalent:

(i) f is effectively locally uniformly multipolynomially approximable w.r.t. (U_ℓ) ,

(ii) f is GL-computable w.r.t. (U_ℓ) .

Proof. (ii) \Rightarrow (i) is just Lemma 5.4.9.

(i) \Rightarrow (ii) follows simply from Lemma 5.2.7, by noting that the sequence of multipolynomials themselves forms a GL-computable sequence w.r.t (U_ℓ) .

□

Chapter 6

Tracking computability; *While* and *WhileCC* approximability; Equivalence Theorem on \mathbb{R}^2 ; Conclusion and future work

In this chapter, we present, in Section 6.1, our second concrete model of computability: $(\bar{\alpha})$ -tracking computability (or just $\bar{\alpha}$ -computability), and state the second Equivalence Lemma between $(\bar{\alpha})$ -tracking computability and GL-computability on \mathbb{R}^2 . Next, considering various abstract models over the field algebra \mathcal{R} , we describe the **While** programming language's extensions, such as **WhileCC** (**While** with “countable choice”) (Section 6.3), and hence the concepts of **WhileCC*** approximability for functions on \mathbb{R}^2 (Section 6.4). We then state (in Section 6.5) the third Equivalence

Lemma on the equivalence of *WhileCC* approximability with $\bar{\alpha}$ -computability. From this follows Theorem 1, on the equivalence, under the global assumptions (5.1.17), of our four models of computation on \mathbb{R}^2 .

Much of the work in this chapter is a straightforward extension to \mathbb{R}^2 of the theory developed in [FZ14] for computation on \mathbb{R} .

6.1 Tracking computability.

Let α be a standard enumeration of \mathbb{Q} , *i.e.* a bijection of \mathbb{N} with \mathbb{Q} . Given two functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $\varphi : \mathbb{N}^2 \rightarrow \mathbb{N}$, we say that φ is an α -tracking function for f if the following diagram commutes:

$$\begin{array}{ccc}
 \mathbb{R}^2 & & \mathbb{R} \\
 \cup & & \cup \\
 \mathbb{Q}^2 & \xrightarrow{f \upharpoonright \mathbb{Q}^2} & \mathbb{Q} \\
 \uparrow \alpha^2 & & \uparrow \alpha \\
 \mathbb{N}^2 & \xrightarrow{\varphi} & \mathbb{N}
 \end{array}$$

in the sense that for all $k_1, k_2 \in \mathbb{N}$,

(i) $f(\alpha(k_1), \alpha(k_2)) \downarrow \implies \varphi(k_1, k_2) \downarrow \wedge f(\alpha(k_1), \alpha(k_2)) = \alpha(\varphi(k_1, k_2))$, and

(ii) $f(\alpha(k_1), \alpha(k_2)) \uparrow \implies \varphi(k_1, k_2) \uparrow$

For the purpose of computation on \mathbb{R} , the enumeration α of \mathbb{Q} and the use of α -tracking functions are clearly inadequate, since, for example, a computable function

on the reals could very well map rationals to irrationals, making a commuting diagram as above impossible. We must extend α to an enumeration $\bar{\alpha}$ of the computational closure of \mathbb{Q} , as we now explain.

Definition 6.1.1 (Computational closure of \mathbb{Q}). We define the α -computational closure of \mathbb{Q} , i.e., the set \mathbb{R}_c of (α) -computable reals, where

$$\mathbb{Q} \subseteq \mathbb{R}_c \subseteq \mathbb{R},$$

with an enumeration¹

$$\bar{\alpha} : \Omega \rightarrow \mathbb{R}_c.$$

The set $\Omega \subset \mathbb{N}$ consists of codes for \mathbb{R}_c , i.e. pairs of numbers $c = \langle e, m \rangle$ where

- (i) e is an index for a total recursive function $\{e\}: \mathbb{N} \rightarrow \mathbb{N}$ defining a Cauchy sequence

$$\alpha(\{e\}(0)), \alpha(\{e\}(1)), \alpha(\{e\}(2)), \dots, \tag{6.1}$$

of elements of \mathbb{Q} , and

- (ii) m is an index for a computable modulus of convergence for this sequence:

¹ ‘ \twoheadrightarrow ’ denotes a surjection

$$\forall k, l \geq \{m\}(n) : |(\alpha(\{e\}(k)) - \alpha(\{e\}(l)))| < 2^{-n}. \quad (6.2)$$

For any such code $c = \langle e, m \rangle \in \Omega$, $\bar{\alpha}(c)$ is defined as the limit in \mathbb{R} of the Cauchy sequence (6.1), and \mathbb{R}_c is the range of $\bar{\alpha}$:

$$\begin{array}{ccccc} \mathbb{Q} & \subset & \mathbb{R}_c & \subset & \mathbb{R} \\ \uparrow & & \uparrow & & \\ \alpha & & \bar{\alpha} & & \\ \mathbb{N} & & \Omega & & \end{array}$$

We define $\bar{\alpha}$ -tracking functions just like α -tracking functions, with ‘ α ’ replaced by ‘ $\bar{\alpha}$ ’

Definition 6.1.2 ($\bar{\alpha}$ -computability). The function $f : \mathbb{R} \rightarrow \mathbb{R}$ is $\bar{\alpha}$ -computable if it has a recursive $\bar{\alpha}$ -tracking function.

Remark 6.1.3 (Fast Cauchy sequence). As explained in [TZ04], we get an equivalent theory if we assume (by effectively taking subsequences) that the sequences (6.1) are *fast* Cauchy sequences, *i.e.*, the modulus of convergence is always the identity function on \mathbb{N} , so that (6.2) becomes

$$\forall n, \forall k > n : |(\alpha(\{e\}(k)) - \alpha(\{e\}(n)))| < 2^{-n}$$

and so we can work with “*e*-codes” instead of “*c*-codes” as elements of Ω .

We come to the equivalence lemma 6.1.4 for our two concrete models of computability on \mathbb{R}^2 .

Recall again our global assumptions (5.1.14) on $f : \mathbb{R}^2 \rightarrow \mathbb{R}$

Equivalence Lemma 2 (GL and $\bar{\alpha}$ -computability). The following are equivalent:

- (i) f is GL-computable w.r.t (U_ℓ),
- (ii) f is $\bar{\alpha}$ -computable.

A detailed proof for partial functions f on \mathbb{R} was given in [FZ14]. That proof can be lifted to \mathbb{R}^2 easily, and we omit it here.

6.2 *While* programming with countable choice

We extend the *While* language over \mathcal{R} to a language *WhileCC* (“CC” for countable choice) by adding a new assignment statement:

$$x := \text{choose } z : P(z, \dots)$$

[TZ04, TZ05] where x and the ‘choose’ variable z have sort nat , and $P(z, \dots)$ is a *semi-computable predicate* of z (and other variables), i.e., the halting set of a *WhileCC* procedure with z among its input variables.

Then ‘choose $z : P(z, \dots)$ ’ selects some value k such that $P(\bar{k}, \dots)$ is true if any such k exists (and is undefined otherwise). In the abstract semantics [TZ04], the

meaning of ‘choose $z : P(z, \dots)$ ’ is the set of *all* such k ’s (hence “countable choice”).

Any concrete model will select a particular k , according to the implementation.

The abstract semantics for ***WhileCC*** associates with a ***WhileCC*** procedure $P : \text{real}^2 \rightarrow \text{real}$ a (many-valued) function:

$$P^{\mathcal{R}} : \mathbb{R}^2 \rightarrow \mathcal{P}_{\omega}^+(\mathbb{R}^{\uparrow}),$$

where $\mathcal{P}_{\omega}^+(X)$ is the set of all countable *non-empty* subsets of X , and $\mathbb{R}^{\uparrow} = \mathbb{R} \cup \{\uparrow\}$,

where ‘ \uparrow ’ represents a divergent computation.

6.3 ***While*^{*} and *WhileCC*^{*} computability and approximability**

A ***While*^{*}**(Σ) procedure is a ***While***(Σ^*) procedure with the restriction that the array variables (i.e. variables of sort real^*) are used only as auxiliary variables, not for input or output.

The ***While*^{*}** language is clearly more convenient than ***While*** for writing programs over \mathcal{R} . However, it is not (in theory) stronger than ***While*** for defining functions on \mathcal{R} ; in fact (writing ***While*^{*}**(\mathcal{R}) for the set of functions ***While*^{*}** definable on \mathcal{R}):

$$\mathbf{While}^*(\mathcal{R}) = \mathbf{While}(\mathcal{R})$$

by [TZ00, §4.9], adapting the proof there to partial algebras.

Similarly we can define the language

$$\mathbf{WhileCC}^*(\Sigma) = \mathbf{WhileCC}(\Sigma^*)$$

and again show that

$$\mathbf{WhileCC}^*(\mathcal{R}) = \mathbf{WhileCC}(\mathcal{R}).$$

Analogously, we can also define the concepts of *WhileCC** approximability on \mathbb{R}^2 .

From now on, we will write $\mathbf{WhileCC}^{(*)}$ to refer to the languages either with or without arrays.

Next we consider *WhileCC* approximable computability or *WhileCC* approximability. Let

$$P : \text{nat} \times \text{real}^2 \rightarrow \text{real}$$

be a *WhileCC* procedure. Again we write

$$P_n^{\mathcal{R}} =_{df} P^{\mathcal{R}}(n, \cdot) : \mathbb{R}^2 \rightarrow \mathcal{P}_{\omega}^+(\mathbb{R}^{\uparrow})$$

Definition 6.3.1 (*WhileCC* approximability to a single-valued function). A function $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ is **approximable** by a *WhileCC* procedure P on \mathcal{R} if for all $n \in \mathbb{N}$ and all $x \in \mathbb{R}^2$:

$$(i) x \in \mathbf{dom}(f) \Rightarrow \uparrow \notin P_n^{\mathcal{R}}(x) \subseteq \mathbf{B}(f(x), 2^{-n}),$$

where $\mathbf{B}(a, \delta)$ is the open ‘ball’ or neighborhood with center a and radius δ , and

$$(ii) x \notin \mathbf{dom}(f) \Rightarrow P_n^{\mathcal{R}}(x) = \{\uparrow\}.$$

6.4 Comparison with concrete computability; Equivalence Theorem

We come to the equivalence lemma for abstract ($\mathbf{WhileCC}^{(*)}(\mathcal{R})$) approximability and concrete ($\bar{\alpha}$ -tracking) computability.

This can also be viewed as a *completeness* result for abstract ($\mathbf{WhileCC}^{(*)}(\mathcal{R})$) vs concrete (tracking) computability. It was proved in [TZ04] for complete separable metric spaces.

Suppose $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Recall the global assumption for f (5.1.17).

Equivalence Lemma 3 (Abstract and concrete computability). The following are equivalent:

- (i) f is $\bar{\alpha}$ -computable
- (ii) f is $\mathbf{WhileCC}^{(*)}(\mathcal{R})$ approximable.

This was proved in [TZ04] for *complete separable* metric spaces.

Note that the proof of this equivalence lemma also requires the global (domain exhaustion and continuity) assumptions for f , even though the definitions of $\bar{\alpha}$ and

WhileCC^(*) approximability do not mention them explicitly.

We can now present the theorem which connects all the models considered in this thesis.

Equivalence Theorem. Given a partial function $f: \mathbb{R}^2 \multimap \mathbb{R}$, and an effective exhaustion (U_ℓ) (cf. Definition 5.1.6) of $U = \mathbf{dom}(f)$, suppose f is effectively locally uniformly continuous w.r.t (U_ℓ) . Then the following are equivalent:

- (i) f is GL-computable w.r.t (U_ℓ) ,
- (ii) f is $\bar{\alpha}$ -computable,
- (iii) f is effectively locally \mathbb{R}_c -multipolynomially approximable w.r.t (U_ℓ) ,
- (iv) f is *WhileCC*^(*)(\mathcal{R}) approximable.

Proof. This follows from the three equivalence lemmas in Section 5.4, 6.1 and 6.5.
 □

Remark 6.4.1. Of the three equivalence lemmas used in proving this Theorem, Equivalence Lemmas 2 ((i) \Leftrightarrow (ii)) and 3 ((ii) \Leftrightarrow (iv)) were proved in [FZ14] for computation on \mathbb{R} , and the extension to \mathbb{R}^2 is fairly routine. The really interesting result here is Equivalence Lemma 1 ((i) \Leftrightarrow (iii)), proved in Chapter 5, which requires a conceptually and technically non-trivial extension from computation on \mathbb{R} to computation on \mathbb{R}^2 .

6.5 Conclusion

In this thesis, we studied two topics: one about the characterization of semicomputable sets on \mathbb{R}^2 , and the other about characterizations of computable partial functions on the real plane.

In the first topic, we presented two Structure Theorems for ***While*^{OR}** and ***While*^{∃N}** semicomputable sets over the ring algebra \mathcal{R}_0 in \mathbb{R}^2 :

Theorem 1. A subset U of \mathbb{R}^2 is ***While*^{OR}** semicomputable, iff U is a countable union of a disjoint effective sequence of finite unions of basic sets.

Theorem 2. A subset U of \mathbb{R}^2 is ***While*^{∃N}** semicomputable, iff U is a countable union of an effective sequence of basic sets.

– and a “partial structure theorem” for ***While*(\mathcal{R})** semicomputable sets in \mathbb{R}^2 :

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

(a) ***While*** semicomputable \implies union of *disjoint* effective sequence of finite

union of basic sets.

(b) union of *disjoint* effective sequence of basic sets \implies ***While*** semicomputable.

Here basic sets are defined as sets of the form

$$\{x \in \mathbb{R}^2 \mid p_1(x) > 0, \dots, p_k(x) > 0\} \quad (k > 0) \tag{6.3}$$

where p_1, \dots, p_k are polynomials over \mathbb{Z} .

By the use of *finiteness theorems* based on cell decomposition theory, alternative forms of the above theorems were obtained, in which the phrase “basic sets” is replaced throughout by “connected open semialgebraic sets”.

In the second topic, four models of computability of partial functions on \mathbb{R}^2 were described:

(i) *GL-computability*;

(ii) *Effective local uniform multipolynomial approximability*.

(iii) $\bar{\alpha}$ -*computability*;

iv) ***WhileCC***^(*) *approximable computability*;

Their equivalence was proved for functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfying two global assumptions: (1) ***dom***(f) is the union of an effective sequence (U_ℓ) of elementary sets; and (2) f is effectively locally uniformly continuous w.r.t. (U_ℓ) .

Here elementary sets are defined (roughly) as bounded connected basic sets. Basic sets are defined as in (6.3) except that $p_i(x)$ ($i = 1, \dots, k$) are now polynomials over the computable reals.

Discussion 6.5.1 (Generalization of results to dimensions > 2).

The main theme of this research, for both topics, lies in the generalization of results previously obtained in computation theory over \mathbb{R} , to analogous results over \mathbb{R}^m , for $m > 1$.

The interesting, and challenging, aspect here lay in finding an appropriate generalization of the concept of “elementary set” from open linear intervals (when $m = 1$) to ... what? (when $m > 1$).

The “correct” generalization was found to lie in the concept of *basic open semialgebraic* subset of \mathbb{R}^2 . Interestingly, this was the case in both topics, with relatively minor variations: in topic 1, an elementary set, which is used in the characterization of semicomputable subsets of the plane, is simply a *basic open semialgebraic* subset of \mathbb{R}^2 , based on polynomials over \mathbb{Z} (see Definitions 4.2.1/2), while in topic 2, where it is used in the definition of effective exhaustions, it is again a basic open semialgebraic set, which is also *connected* and *bounded*, and based on polynomials over the computable reals \mathbb{R}_c .

It seems clear that the big conceptual leap, in both topics, occurs in going from $m = 1$ to $m = 2$.

What about generalizations to $m > 2$? For topic 1, this is absolutely routine, as is clear from an inspection of the relevant definitions and proofs.

When it comes to topic 2, this is also clear for the most part. However, things are not quite so simple in the case of Equivalence Lemma 1 (Chapter 5), in the direction

“GL-computable \implies effective multipolynomial approximability”.

Let us consider the two “big theorems” used here: the effective Weierstrass The-

orem and the Extension Theorem (Theorems 5 and 6 respectively). The effective Weierstrass Theorem will, it seems, generalize without too much difficulty to $m > 2$. However, generalizing the Extension Theorem to $m = 3$ already presents a challenge, although we believe it to be true.

6.6 Some ideas and conjectures about future work

In topic 2, we have the equivalence of four models under the global assumptions (5.1.14), taking the domain of f to be a countable union of stages U_ℓ , where each U_ℓ is a finite union of elementary sets. Two problems arising from this research are:

Problem 1: Relationship of our models with Weihrauch's TTE

In [Wei00] it is shown that (a) the domain of every TTE (type two effective) function is a G_δ set, i.e. a countable intersection of open sets, and conversely (b) every *effective* G_δ subset of \mathbb{R} (i.e. a set of the form $\bigcap_{i=1}^{\infty} \bigcup_{j=1}^{\infty} I_{ij}$ for an effective double sequence of rational interval (I_{ij})) is a domain of some TTE computable function.

Then we can ask:

Given $f : \mathbb{R} \rightarrow \mathbb{R}$ and replacing our global assumptions by:

***dom*(f)** is an effective G_δ set

with some suitable corresponding continuity assumption, to what extent do the equiv-

alence theorems still hold for f ?

Note that the precise definition of GL-computability is also problematic here.

A positive solution to this problem would also show an equivalence between TTE computability and the other models considered here.

Problem 2: Global assumptions and elementary functions

We return to the two global assumptions (5.1.14) of *domain exhaustion* and *continuity* for functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, which underline the equivalence proofs of our four models of computability. It was remarked earlier (Section 5.1) that these assumptions are satisfied by many functions commonly encountered in calculus and elementary real analysis.

To make this more precise, we define an *elementary function* on \mathbb{R}^2 to be any function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ denoted by an expression built up from the variables x_1 and x_2 and constants for computable reals, by (repeated) application of the four field operations, n -th roots, the exponential and trigonometric functions and their inverses.

This is a very interesting class of functions, investigated by, among others, G.H. Hardy [Har05]².

Note that the functions $f(x) = \sqrt[n]{x}$ (for n even) have domain $\mathbf{dom}(f) = [0, \infty)$, which is not open, and hence could not possibly be a union of an open exhaustion. We therefore extend f to the whole of \mathbb{R} by defining $f(x) = 0$ for $x < 0$. This makes

²for functions of 1 variable. Hardy also included function $y = f(x)$ implicitly defined by polynomial equations in x and y .

f total, effectively uniformly continuous, and computable by any of our four models, on \mathbb{R} .

Note however that we cannot “totalize” other elementary functions such as $\tan x$, $\log x$, or $1/x$ in this way, so as to preserve continuity and computability.

Then (following the analogous conjecture for functions on \mathbb{R} in [FZ14]), we propose the following

Conjecture. Every elementary function on \mathbb{R}^2 satisfies the two global assumptions.

It is not hard to show that the *basic* elementary functions (field operations, n -th roots, exp and trig functions and their inverses) all satisfy the two global assumptions. Hence, in order to prove this conjecture, it would be sufficient to prove that the property of satisfying these global assumptions is preserved under composition of functions. This we have been unable to do.

Bibliography

- [ABR96] Carlos Andradas, Ludwig Bröcker, and Jesús Ruiz. *Constructible sets in real geometry*. Springer, 1996.
- [BB85] Errett Bishop and Douglas Bridges. *Constructive Analysis*. Springer, 1985.
- [BM97] Bernhard Banaschewski and Christopher J. Mulveyb. A constructive proof of the Stone-Weierstrass theorem. *Journal of Pure and Applied Algebra* 116 (1997) 25-40, 1997.
- [BPM06] Saugata Basu, Richard Pollack, and M.Roy. *Algorithms in real algebraic geometry*. Springer, 2006.
- [CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics, Vol. II*. Interscience, 1953. Translated and revised from the German edition [1937].
- [Del82] Charles N. Delzell. A finiteness theorem for open semi-algebraic sets with application to Hilbert's 17th problem. *Contemporary Mathematics*, 8:79–97, 1982.

- [DSW94] Martin D. Davis, Ron Sigal, and Elaine J. Weyuker. *Computability, Complexity, and Languages (Second Edition)*. Academic Press, 1994.
- [Eng68] E. Engeler. *Formal Languages: Automata and Structures*. Markham, 1968.
- [Fu07] Ming Quan Fu. Models of computability of partial functions on the reals. M.Sc. Thesis, Department of Computing & Software, McMaster University, 2007. Technical Report CAS 01-08-JZ, Jan. 2008, McMaster University.
- [FZ14] Ming Quan Fu and J.I. Zucker. Models of computation for partial functions on the reals. *Submitted to J. Logic and Algebraic Programming*, <http://www.cas.mcmaster.ca/zucker/Pubs/FZ/tx.pdf>, 2014.
- [Grz55] A. Grzegorzcyk. Computable functions. *Fundamenta Mathematicae*, 42:168–202, 1955.
- [Grz57] A. Grzegorzcyk. On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.
- [Had52] J. Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Dover, 1952. Translated from the French edition [1922].
- [Har05] G. H. Hardy. *The integration of functions of a single variable*. Cambridge Tracts in Mathematics and Mathematical physics, no. 2. Cambridge University press, 1905.

-
- [Kel55] J.L. Kelley. *General Topology*. Van Nostrand, 1955.
- [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] Marian B. Pour-El and Jonathan I. Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1989.
- [Rog67] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.
- [Rud76] Walter Rudin. *Principles of Mathematical analysis*. New York, 1976.
- [Rud91] Walter Rudin. *Functional Analysis*. McGraw-Hill, 1991.
- [SHT99] V. Stoltenberg-Hansen and J.V. Tucker. Computable rings and fields. In E. Griffor, editor, *Handbook of Computability Theory*. North Holland, 1999.
- [TZ99] J.V. Tucker and J.I. Zucker. Computable functions 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. *Oxford University Press*, 5:317–523, 2000.

- [TZ04] J.V. Tucker and J.I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational logical*, 5:611–688, 2004.
- [TZ05] J.V. Tucker and J.I. Zucker. Computable total functions on metric algebras, algebraic specifications and dynamical systems. *Journal of Logic and Algebraic Programming*, 62:71–108, 2005.
- [TZ06] J.V. Tucker and J.I. Zucker. Abstract versus concrete computability: The case of countable algebras. In V. Stoltenberg-Hansen and J. Väänänen, editors, *Logic Colloquium '03, Proc. Annual European Summer Meeting, Association for Symbolic Logic, Helsinki, August 2003*, volume 24 of *Lecture Notes in Logic*, pages 377–408. Association for Symbolic Logic, 2006.
- [TZ11] J.V. Tucker and J.I. Zucker. Continuity of operators on continuous and discrete time streams. *Theoretical Computer Science*, 412:3378–3403, 2011.
- [vdD98] Lou van den Dries. *Tame Topology and O-minimal Structure*. Cambridge University Press, 1998.
- [Wei00] Klaus Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

- [XFZ13] Bo Xie, Ming Quan Fu, and Jeffery Zucker. Characterizations of semicomputable sets of real numbers. *Journal of Logic and Algebraic Programming*, 2013. DOI: 10.1016/j.jlap.2013.11.001.
- [Xie04] Bo Xie. Characterizations of semicomputable sets of real numbers. M.Sc. Thesis, Department of Computing & Software, McMaster University, 2004. Technical Report CAS 04-06-JZ, April 2006, McMaster University.