University of London
Imperial College of Science, Technology and Medicine
Department of Computing

PCF extended with real numbers:
a domain-theoretic approach to higher-order exact real number computation

Martín Hötzel Escardó

Dr. Michael B. Smyth
Supervisor

Prof. Achim Jung
External Examiner

Prof. Edmund Robinson
Internal Examiner

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the Diploma of the Imperial College

Submitted November 1996
Defended February 1997

Declaration

The results of Sections 5.2 and 5.3 of Part II consist of joint work with Thomas Streicher [ES97]. The results of Part V consist of joint work with Abbas Edalat [EE96a]. All other results reported in this thesis are due to the author, except for background results, which are clearly stated as such. Some of them have already appeared as [Esc96a, Esc96b], and preliminary ideas have appeared in the extended abstract [Esc94].

Note This version of the thesis, produced on Monday $1^{\underline{\text{st}}}$ December 1997, is the same I submitted after the defence, except that I have added the references [ES, EE, Esc, EEK].

À Daniele

Abstract

We develop a theory of higher-order exact real number computation based on Scott domain theory. Our main object of investigation is a higher-order functional programming language, Real PCF, which is an extension of PCF with a data type for real numbers and constants for primitive real functions. Real PCF has both operational and denotational semantics, related by a computational adequacy property.

In the standard interpretation of Real PCF, types are interpreted as continuous Scott domains. We refer to the domains in the universe of discourse of Real PCF induced by the standard interpretation of types as the *real numbers type hierarchy*. Sequences are functions defined on natural numbers, and predicates are truth-valued functions. Thus, in the real numbers types hierarchy we have real numbers, functions between real numbers, predicates defined on real numbers, sequences of real numbers, sequences of sequences of real numbers, sequences of functions, functionals mapping sequences to numbers (such as limiting operators), functionals mapping functions to numbers (such as integration and supremum operators), functionals mapping predicates to truth-values (such as existential and universal quantification operators), and so on.

As it is well-known, the notion of computability on a domain depends on the choice of an effective presentation. We say that an effective presentation of the real numbers type hierarchy is *sound* if all Real PCF definable elements and functions are computable with respect to it. The idea is that Real PCF has an effective operational semantics, and therefore the definable elements and functions should be regarded as concretely computable. We then show that there is a unique sound effective presentation of the real numbers type hierarchy, up to equivalence with respect to the induced notion of computability. We can thus say that there is an *absolute notion of computability* for the real numbers type hierarchy.

All computable elements and all computable first-order functions in the real numbers type hierarchy are Real PCF definable. However, as it is the case for PCF, some higher-order computable functions, including an existential quantifier, fail to be definable. If a constant for the existential quantifier (or, equivalently, a computable supremum operator) is added, the computational adequacy property remains true, and Real PCF becomes a *computationally complete programming language*, in the sense that all computable functions of all orders become definable.

We introduce induction principles and recursion schemes for the real numbers domain, which are formally similar to the so-called Peano axioms for nat-

ural numbers. These principles and schemes abstractly characterize the real numbers domain up to isomorphism, in the same way as the so-called Peano axioms for natural numbers characterize the natural numbers. On the practical side, they allow us to derive recursive definitions of real functions, which immediately give rise to correct Real PCF programs (by an application of computational adequacy). Also, these principles form the core of the proof of absoluteness of the standard effective presentation of the real numbers type hierarchy, and of the proof of computational completeness of Real PCF.

Finally, results on integration in Real PCF consisting of joint work with Abbas Edalat are included.

Acknowledgements

Before coming to Imperial, Rocha Costa introduced me to domain theory, Dalcidio Claudio introduced me to interval analysis, and Benedito Acióly introduced me to the connections between the two. In a group organized by Rocha Costa, including Álvaro Moreira, Vanderlei Rodrigues and myself, we studied the book [Pau87]. All this forms a vast part of the background material needed to develop this work. I also had many discussions with Paulo Azeredo and Daltro Nunes. I am grateful to all of them. Rocha Costa deserves especial credit, for having organized the very fruitful theory group at the Instituto de Informática da Universidade Federal do Rio Grande do Sul, of which I had the opportunity of being a member.

I acknowledge Benedito Acióly and Wilson de Oliveira (a former student of Mike Smyth, who I met personally only a few months ago) for having suggested that I should apply for a position as a research student at Imperial. Juarez Muylaert, who was at Imperial at that time, helped me to make contact with Mike, via e-mail.

I benefited from many discussions about real number computability with Klaus Weihrauch, most of them by e-mail, which started when I was in Brazil, before coming to Imperial. I also had some interesting discussions with his student Vasco Brattka.

The sponsorship by the Brazilian agency CNPq made this work possible. An ARC project "A Computational Approach to Measure and Integration Theory", organized by Abbas Edalat, Achim Jung and Klaus Keimel, allowed me to visit the Technische Hochschule Darsmtadt on three occasions, which were very productive times. Achim Jung gave me valuable feedback during the first visit, as did Frank Pfenning (who was also visiting Darmstadt), Martin Hofmann, Klaus Keimel and Thomas Streicher during the second visit. In particular, I established the computational completeness result for Real PCF after I spoke with Thomas about my partial success (first-order case) and he subsequently explained his technique [Str94] to me. Also, the collaboration with him, partially reported in Section 5.2, started during that visit. Dana Scott gave me detailed feedback, in the form of questions, criticism and directions, in three workshops (two of them sponsored by the ARC project) and one conference.

The fixed-point definition of the integration operator given in Section 14.2 is based on a suggestion by Peter Freyd.

Michael Huth carefully read the paper [Esc96a] (after it was accepted for publication). He spotted some mistakes and imprecisions, which are corrected in this thesis. He also indicated how to improve some points.

I had interesting discussions with Pietro di Gianantonio, during a workshop in Hagen and during a visit by himself to Imperial.

During these three years at Imperial, I have had many discussions with Samson Abramsky, David Clark, Roy Crole, Abbas Edalat, Lindsay Errington, Ian Hodkinson, François Lamarche, Guy McCusker, Juarez Muylaert, Peter Potts, Paul Taylor, Mike Smyth, Chis Townsend, Steve Vickers, and Julian Webster.

Samson played the rôle of "second supervisor" during my transfer from MPhil to PhD, when he gave me valuable feedback. We also had many interesting discussions during a recent conference, in particular while we spent a whole day waiting to come back to the UK.

Abbas deserves especial credit. He has closely followed my work since its early stages, and we have had uncountably many discussions about it. As a result, an intense collaboration arose. A portion of it is reported in Part V of this thesis, which is the full version of the extended abstract [EE96a]. Also, Abbas has introduced me to many people, including some of the above.

Last but not least, I would like to acknowledge Mike Smyth, my supervisor. I have had weekly meetings with him and his students, and sometimes with guests, during which I benefited from his vast knowledge and original ideas. He has always managed to relate my ideas to his and other people's work, in unexpected ways. On the rare occasions when he didn't feel confident about a particular point, he sent me to speak to the appropriate people, including some of the above. This has resulted in fruitful informal extra supervision.

Both Abbas and Mike have carefully proof-read this thesis. Of course, all remaining mistakes are due to myself.

Most commutative diagrams in this thesis were produced with Paul's commutative diagrams package. Lindsay and Peter have kept the machines running. They also helped me to recover the file containing the paper [Esc96b], which I managed to delete just before submitting it.

Contents

Chapter 1

Introduction

We develop a theory of higher-order exact real number computation based on Scott domain theory [AJ94]. Our main object of investigation is a higher-order functional programming language, Real PCF, which is an extension of PCF [Plo77] with a data type for real numbers and constants for primitive real functions.

Traditionally, in computing science one represents real numbers by floating-point approximations. If we assume that these approximations are "exact" then we can prove correctness of numerical programs by analytical methods. Such an idealization is the idea behind the so-called BSS model [BSS89]. However, such "correct" programs do not produce correct results in practice, due to the presence of round-off errors. Striking examples are described in [MM96]. Moreover, such programs are inappropriate for problems whose solution is sensitive to small variations on the input.

As a consequence, exact real number computation has been advocated as an alternative solution (see e.g [Boe87, BC90, BCRO86, Gru, MM96, Pot96, Viu90, Vui88] on the practical side and [Bra96, Gia93b, Gia96b, Gia96a, Gia93a, Grz57, KV65, Ko91, ML70, Myh53, PeR83, Ric54, Tur37, Wei87, Wei95, Wie80] on the foundational side). In exact real number computation, as opposed to floating point-computation, the produced outputs are guaranteed to be correct, and, moreover, the results can be effectively computed to within any desired degree of accuracy. These approaches, as well as ours, differ from the BSS model in that they consider only effective computations in the sense of logic and recursion theory [Kle52, Rog67, Tur37], whereas the BSS model admits non-effective operations such as equality tests on real numbers. The BSS model is best conceived as a useful idealization of floating-point computation, as we have indicated above, and as Blum, Shub and Smale indicate in their seminal paper [BSS89].

However, work on exact real number computation has focused on *representations* of real numbers and has neglected the issue of *data types* for real numbers. Two exceptions are [BC90] and [Gia93a]. In particular, programming languages for exact real number computation with an explicit distinction between operational semantics [Gun92], which is representation-dependent, and denotational semantics [Gun92], which is representation-independent, have hardly been investigated. An exception is [Gia93a]. Such programming languages allow for correctness proofs based on analytical methods, in a representation-independent fashion, as discussed below.

The real numbers type of Real PCF is interpreted as the interval domain introduced by D.S. Scott [Sco72b], which is related to the interval space introduced by R.E. Moore [Moo66]. The elements of the interval domain are considered as "generalized real numbers" or "partial real numbers". Such an extension of PCF was one of the problems left open by G.D. Plotkin in his seminal paper [Plo77]. Pietro di Gianantonio [Gia93a] also presents an extension of PCF with a ground type interpreted as a domain of real numbers, with the purpose of obtaining an abstract data type for real numbers. His extension is based on an algebraic domain of real numbers. In this thesis we push his ideas further; in particular, in addition to work with a continuous domain of real numbers which faithfully represents the real line, we give an operational semantics to our extension of PCF with real numbers.

There have been a number of applications of domain theory in constructing computational models for classical spaces, including locally compact Hausdorff spaces [Eda95e] and metric spaces [EH96]. These models have resulted in new techniques in computation with real numbers. In particular the computational measure and integration theory [Eda95e, Eda95b, Eda96b, EN96] has had various applications, including exact computation of integrals, in fractals [Eda96a], statistical physics [Eda95a], stochastic processes [Eda95d] and neural networks [Eda95c, Pot95].

An important feature of our approach to exact real number computation is that the programmer does not have access to representations within the programming language and can think of real numbers as abstract entities in the usual mathematical sense. Of course, the Real PCF machinery has access only to concrete representations. The correct interaction between the abstract level and the concrete level is usually referred to as *computational adequacy*. At the denotational level, a Real PCF program is just a mathematical expression denoting a number or a function. The task of the programmer is to find a mathematical expression in this language denoting the entity that he or she has in mind. This entity is usually given by unrestricted mathematical means. In this case the programmer has to find an equivalent Real PCF expression. Computational adequacy ensures that the entity denoted by the program will be actually computed by the Real PCF machinery. This is why the programmer is not concerned with representations in correctness proofs. Of course, in order to obtain efficient programs, the programmer has to reason about representations. The point is that efficiency considerations and correctness proofs can be tackled separately.

We introduce induction principles and recursion schemes for the real numbers domain, which are formally similar to the so-called Peano axioms for natural numbers [Sto66]. These principles and schemes allow us to prove correctness of programs and derive correct programs respectively.

In addition to computational adequacy, a main result is *computational completeness*, which roughly means that Real PCF has enough primitive operations to express all computable functions. This result is proved by means of the above induction principles and recursion schemes.

1.1 Main results and underlying ideas

We now proceed to a slightly more detailed and technical exposition of our main results and underlying ideas.

Computational Adequacy The operational semantics of Real PCF implements computations on real numbers via computations on rational intervals. We show that the operational semantics is computationally adequate with respect to the denotational semantics, in the sense that every piece of information about the value assigned by the denotational semantics to a program is eventually produced in a finite number of steps by the operational semantics. The intuitive idea of "piece of information" is formalized via the use of continuous domains [AJ94, Sco72a], and the the proof of computational adequacy relies on the basic machinery of continuous domains, including the so-called "way-below relation" and "$\epsilon - \delta$ characterization of continuity". The domain of partial numbers is a (non-algebraic) continuous domain. Its subspace of maximal points (single-point intervals) is homeomorphic to the Euclidean real line, so that real numbers are special cases of partial real numbers. Notice that no algebraic domain can have the real line as its subspace of maximal points.

Continuous words We show that partial real numbers can be considered as "continuous words", and we use this fact to obtain the operational semantics of Real PCF. Partial real numbers can be considered as continuous words in the sense that they can be given the structure of a monoid, in such a way that it has a finitely generated submonoid isomorphic to the monoid of words over any finite alphabet. Moreover, as it is the case for words, the prefix preorder of the monoid of continuous words coincides with its information order. This coincidence is the basis for the successful interaction between the operational and denotational semantics of Real PCF.

Concatenation of continuous words has intuitive geometrical and computational interpretations. Geometrically, a concatenation of continuous words corresponds to a rescaling of an interval followed by a translation (an affine transformation). Computationally, a concatenation of continuous words corresponds to refinement of partial information; in a concatenation xy, the partial real number y refines the information given by x, by selecting a subinterval of x.

The notion of length of words generalizes to partial real numbers. The length of a partial real number is an extended non-negative real number, being infinity iff the partial number is maximal, and zero iff the partial number is bottom. Roughly speaking, the length of a partial number x considered as a partial realization of an unknown real number y gives the number of digits of a digit expansion of y that x is able to give correctly. The concatenation operation "adds lengths", in the sense that the length function is a monoid homomorphism from the monoid of partial real numbers to the additive monoid of non-negative extended real numbers.

The usual basic operations cons, head and tail used to explicitly or recursively define functions on words generalize to partial real numbers. Geometrically, the operation cons is an affine transformation (in analytic terms, a linear map) on the unit interval, tail reverses the effect of cons, and head decides in which half of the real line its argument lies, telling us nothing in ambiguous cases.

Concatenation of partial numbers can be infinitely iterated. This fact gives a notion of meaningful infinite computation. The concatenation of finite initial segments of infinite computations gives more and more information about the final result, in such a way that every piece of information about the (ideal) result is eventually produced in a finite amount of steps. In practice, the terms of a computation can be taken as the partial numbers with distinct rational end-points.

The interpretation of partial real numbers as continuous words is related to a well-known approach to real number computation. Usual binary expansions are not appropriate representations for real number computation; for instance, multiplication by three is not computable if we read the expansions from left to right [Wie80]. But binary expansions of numbers in the signed unit interval $[-1, 1]$ allowing a digit -1 turn out to be effective [BCRO86, Gia93a, Gru, Wei87, Wei95, Wie80]. In the domain of partial numbers contained in the signed unit interval, infinite concatenations of the partial numbers $[-1, 0]$, $[-\frac{1}{2}, \frac{1}{2}]$ and $[0, 1]$ correspond to binary expansions of numbers in the signed unit interval using the digits -1, 0 and 1 respectively.

To be accurate, the interpretation of partial numbers as continuous words holds only for the domain of partial real numbers contained in the unit interval (or any other compact interval). The domain of partial numbers contained in the unit interval is referred to as the *partial unit interval*, and the domain of all partial real numbers is referred to as the *partial real line*. The above results are extended from the partial unit interval to the partial real line via an action of the partial unit interval on the partial real line.

Higher-order real number computation In the standard interpretation of Real PCF, types are interpreted as bounded complete continuous domains [AJ94], which form a cartesian closed category [LS86] suitable for higher-order real number computation. We refer to the domains in the universe of discourse of Real PCF induced by the standard interpretation of types as the *real numbers type hierarchy*. Sequences are functions defined on natural numbers, and predicates are truth-valued functions. Thus, in the real numbers types hierarchy we have real numbers, functions between real numbers, predicates defined on real numbers, sequences of real numbers, sequences of sequences of real numbers, sequences of functions, functionals mapping sequences to numbers (such as limiting operators), functionals mapping functions to numbers (such as integration and supremum operators), functionals mapping predicates to truth-values (such as existential and universal quantification operators), and so on.

Higher-order exact real number computability As it is well-known, the notion of computability on a domain depends on the choice of an effective presentation [KP79]. We say that an effective presentation of the real numbers type hierarchy is *sound* if all Real PCF definable elements and functions are computable with respect to it. The idea is that Real PCF has an effective operational semantics, and therefore the definable elements and functions should be regarded as concretely computable. An important result is that there is a unique sound effective presentation of the real numbers type hierarchy, up to equivalence with respect to the induced notion of computability. We can thus say that there is an *absolute notion of computability* for the real numbers type hierarchy.

Computational completeness of Real PCF We show that all computable elements and all computable first-order functions in the real numbers type hierarchy are Real PCF definable. However, as it is the case for PCF, some higher-order computable functions, including an existential quantifier, fail to be definable. If a constant for the existential quantifier (or, equivalently, a computable supremum operator) is added, the computational adequacy property remains true, and Real PCF becomes a *computationally complete programming language*, in the sense that all computable functions of all orders become definable [Esc96b], as it is also the case for PCF. Informally, this means that Real PCF has enough primitive operations.

Recursive real functions So far, except for the recent work by Brattka [Bra96], there has been no attempt to characterize computability on the real line via recursiveness. The computational completeness result for Real PCF immediately gives rise to an inductively defined collection of higher-order functions on real numbers (namely the Real PCF definable ones) which contains exactly the computable ones. Thus, we can reason about computability in a representation-independent way, by reducing computability questions to recursiveness questions.

Induction and recursion on the real line We introduce induction principles and recursion schemes for the real line, based on the theory of domain equations [AJ94, Plo80, SP82] and previous work on the uniform real line [Esc94]. These principles abstractly characterize the partial real line up to isomorphism in the category of bounded complete domains, in the same way as the so-called Peano axioms for natural numbers characterize the natural numbers up to isomorphism in the category of sets. On the practical side, they allow us to derive recursive definitions of real functions, which immediately give rise to correct Real PCF programs (by an application of computational adequacy). Also, these principles form the core of the proof of absoluteness of the standard effective presentation of the real numbers type hierarchy, and of the proof of computational completeness of Real PCF.

Integration in Real PCF We report results on integration in Real PCF consisting of joint work with Abbas Edalat [EE96a]. We begin by generalizing Riemann integration of real valued functions of real variables to partial real valued functions of partial real variables. We then show that several basic properties of standard Riemann integration generalize to interval Riemann integration. In particular, we prove a generalization of the so-called Fubini's rule, which reduces multiple integrals to iterated simple integrals. We then consider two approaches to integration in Real PCF. First, we show how to introduce integration as a primitive operation, with a simple operational semantics which is proved to be computationally adequate. Second, we introduce the supremum operator mentioned above as a primitive operation, again with a simple operational semantics which is proved to be computationally adequate, and then we show that the integration operator can be recursively defined in Real PCF extended with the supremum operator.

1.2 Additional contributions

In order to either justify some of the decisions taken in the course of our investigation or else obtain our results on real number computation, we were led to obtain some subsidiary results in domain theory, which are interesting in their own right. These results are reported in Part II, and are about, or directly related to

- domain equations [AJ94, Plo80, Sco72a, SP82],
- effectively given domains [EC76, Plo80, Smy77], and
- coherently complete domains [Plo78].

1.3 Background

The prerequisites for this work are basic topology [Bou66, Kel55, Smy92b, Vic89], basic recursion theory [Phi92, Rog67], very basic category theory [Cro93, ML71, McL92, Poi92], domain theory [AJ94, GHK$^+$80, Plo80], effective domain theory [EC76, Plo80, Smy77], and operational and denotational semantics of PCF [Gun92, Plo77]. Since domain theory, effective domain theory, and PCF are heavily used in this thesis, we include background chapters on these subjects. Some notions of the other subjects are recalled when they are needed.

1.4 Organization

This thesis is organized in five parts,

I Background

II Domain theory revisited

III The partial real line

IV Computation on the partial real line

V Integration on the partial real line,

in addition to a part containing concluding remarks. Each part contains a description of its contents and organization, and so does each of its chapters.

The reader who is mainly interested in real number computation can safely skip Part II at a first reading, as we clearly indicate when some chapter in Part II is needed to proceed further.

An automatically generated index of definitions is included; it contains the ***emphasized*** defined terms and some mathematical symbols.

Part I

Background

This part can be used as a reference. In Chapter 2 we introduce domain theory. In Chapter 3 we introduce effective domain theory. In Chapter 4 we introduce the programming language PCF.

Chapter 2

Domain theory

We regard domain theory as a branch of topology which is conveniently presented via order theoretic concepts, as it is done in the seminal papers [Sco72a, Sco76] by Dana Scott. This chapter, which can be used as a reference, introduces basic concepts, terminology, and results.

Our main references to domain theory are [AJ94, Plo80]. Additional references to the topological aspects of domain theory are [GHK$^+$80, Smy92b, Vic89].

2.1 Directed complete posets and the Scott topology

A ***preordered set*** is a set together with a reflexive and transitive binary relation $\sqsubseteq$. Preordered sets naturally arise in topology as follows. For any space X, the binary relation $\sqsubseteq$ defined by

$$\begin{aligned} x \sqsubseteq y \quad &\text{iff} \quad \text{every neighbourhood of } x \text{ is a neighbourhood of } y \\ &\text{iff} \quad x \text{ belongs to the closure of } \{y\}, \end{aligned}$$

is a preorder on the points of X, called the ***specialization order*** of X. It is clear from the definition that any continuous map between topological spaces preserves the specialization order. If X is Hausdorff, then this preorder is trivial because it is the identity relation (also called the ***discrete order***). A main feature of domain theory is that it considers spaces with a rich specialization order.

Convention 2.1 Unless otherwise stated, every mention of order in a topological space implicitly refers to its specialization order. □

A ***poset*** (***partially ordered set***) is a preordered set $(P, \sqsubseteq)$ with $\sqsubseteq$ antisymmetric. Recalling that a space is T_0 if no two distinct points share the same system of neighbourhoods, antisymmetry of the specialization order is equivalent to the T_0 axiom.

Let P be a poset, $x \in P$, and $X \subseteq P$. Then we write

1. $\uparrow x = \{v \in P | x \sqsubseteq v\}$,

2. $\downarrow x = \{u \in P | u \sqsubseteq x\}$,

3. $\uparrow X = \bigcup_{x \in X} \uparrow x$,

4. $\downarrow X = \bigcup_{x \in X} \downarrow x$.

Convention 2.2 We use the words ***below*** and ***above*** in partial ordered sets to refer to order in the non-strict sense. For example, if $x \sqsubseteq y$ we say that x is below y and that y is above x. If we want to imply that x and y are distinct, we say ***strictly below*** and ***strictly above***. □

A subset A of a poset P is ***directed*** if every finite subset of A has an upper bound in A. Since the empty set is included in this definition, a directed set is non-empty. A ***dcpo*** (***directed complete poset***) is a poset with least upper bounds of directed subsets. The order relation of a dcpo is called its ***information order***.

Convention 2.3 We write $\bigsqcup A$ to denote the least upper bound of a subset A which has a least upper bound, and we write $\bigsqcup^{\uparrow} A$ to denote the least upper bound of a set A which is assumed to be directed. □

A function between dcpos is ***order-continuous*** if it preserves least upper bounds of directed subsets. Such a function is necessarily monotone.

The ***Scott topology*** of a dcpo D has as open sets the sets O such that

1. O is an upper set, and

2. if O contains the least upper bound of a directed subset A of D then O already contains a member of A.

It follows that a set is closed in the Scott topology iff it is a lower set containing the least upper bounds of its directed subsets. The main feature of this topology is that order-continuity coincides with topological continuity. Moreover, if D is a dcpo endowed with its Scott topology, then its specialization order coincides with its information order.

Convention 2.4 Unless otherwise stated, when we implicitly or explicitly refer to a dcpo D as a topological space, we mean the set of elements of D endowed with the Scott topology of D. □

2.2 Pointed dcpos and least fixed-points

A dcpo is ***pointed*** if it contains a least element $\perp$, called ***bottom***. A function between pointed dcpos is ***strict*** if it preserves $\perp$.

Every dcpo D can be made into a pointed dcpo by the addition of a new least element $\perp$. The resulting dcpo is denoted by $D_\perp$ and is called the ***lifting*** of D. This construction makes sense even if D is already pointed.

A fixed-point of an endofunction $f : X \to X$ is an element x such that $f(x) = x$. Every continuous endomap $f : D \to D$ on a pointed dcpo D has a least fixed-point, given by ${\bigsqcup^{\uparrow}}_{n \in \omega} f^n(\perp)$. Least fixed points are interesting mainly in connection with function spaces, which are discussed below.

2.3 Domains

The ***way-below*** order of a dcpo D is the binary relation $\ll$ on D defined by

> $x \ll y$ in D iff for every directed set $A \subseteq D$, $y \sqsubseteq \bigsqcup^{\uparrow} A$ implies $x \sqsubseteq a$ for some $a \in A$.

A continuous map $f : D \to E$ preserves the way-below relation iff its a semiopen map, in the sense that $\uparrow f(O) \subseteq E$ is Scott open for every Scott open set $O \subseteq D$. The following properties are easily verified (for example, for the first one, it suffices to consider $A = \{y\}$), and some of them are entailed by the others:

1. $x \ll y$ implies $x \sqsubseteq y$,
2. if $\perp$ is a least element then $\perp \ll x$,
3. $u \sqsubseteq x \ll y \sqsubseteq v$ implies $u \ll v$,
4. $x \ll y \ll z$ implies $x \ll z$,
5. if $u \ll x$, $v \ll x$ and $u \sqcup v$ exists then $u \sqcup v \ll v$.

There is no reason why any pair (x, y) (except if x is a minimal element) should be related by the way-below relation for general dcpos (an example in which only such pairs are related by the way-below order is given by two complete chains with top elements identified). A dcpo is ***continuous*** if its way-below relation is frequent, in the sense that it satisfies the following ***axiom of approximation***:

$$x = \bigsqcup_{y \ll x}^{\uparrow} y.$$

That is, the set of elements way-below x is directed and has x as its least upper bound. The axiom of approximation implies that

> If $x \not\sqsubseteq y$ then there is some $u \ll x$ such that already $u \not\sqsubseteq y$.

We write

1. $\twoheaduparrow x = \{v \in P | x \ll v\}$,
2. $\twoheaddownarrow x = \{u \in P | u \ll x\}$,
3. $\twoheaduparrow X = \bigcup_{x \in X} \twoheaduparrow x$,
4. $\twoheaddownarrow X = \bigcup_{x \in X} \twoheaddownarrow x$.

With this notation, the axiom of approximation reads

> $x = \bigsqcup^{\uparrow} \twoheaddownarrow x$.

In a continuous dcpo, the way-below relation satisfies the following ***interpolation property***:

> If $x \ll y$ then there is some u such that $x \ll u \ll y$.

This implies that

> $x \ll y$ iff for every directed set A, we have that $x \ll a$ for some $a \in A$ whenever $y \sqsubseteq \bigsqcup^{\uparrow} A$.

A ***domain*** is a continuous dcpo. A ***basis*** of a domain D is a subset B such that

$$x = \bigsqcup^{\uparrow} \twoheaddownarrow x \cap B.$$

That is, x is the directed join of the members of B which are way-below x. By definition, a domain is a basis of itself. A domain is ***countably based*** if it has a countable basis (the terminology ***ω-continuous*** is also used).

For any basis B of a domain D, the sets $\twoheaduparrow b$ for $b \in B$ form a base of the Scott topology. Thus, if D and E are domains with bases B and C, then a function $f : D \to E$ is continuous at x iff

> $c \ll f(x)$ for $c \in C$ iff there is some $b \ll x$ in B such that $c \ll f(b)$.

This is often referred to as the ***$\epsilon - \delta$ characterization of continuity***. Also, f is continuous at x iff

$$f(x) = \bigsqcup^{\uparrow}_{b \ll x} f(b).$$

An element x is ***finite*** (or ***compact***, or ***isolated from below***) if it is way-below itself. This means that x is below some member of each directed set with least upper abound above x. Any basis of a continuous dcpo contains the set of compact elements (which can be empty). A dcpo is ***algebraic*** if the compact elements form a basis. The basis of finite elements of an algebraic domain D is denoted by K_D.

2.4 Lattice-like domains

A ***continuous lattice*** is a continuous dcpo which is also a complete lattice (has all joins and meets, including those of the empty set).

A subset of a poset is ***bounded*** (or ***consistent***) if is has an upper bound. A poset is ***bounded complete*** (or ***consistently complete***) if every bounded subset has a least upper bound. Since this definition includes the empty set, which is bounded iff the poset is non-empty, a bounded complete poset is pointed if it is non-empty. A ***bounded complete domain*** is a continuous dcpo which is also bounded complete. Every continuous lattice is a bounded complete domain, but not conversely. However, if a bounded complete domain fails to be a continuous lattice, it fails only by lacking a top element. Moreover, the bounded complete domains are the closed subspaces of continuous lattices. Notice that for this to be true we have to admit the empty bounded complete domain – as we did. We refer to bounded complete, countably based domains as ***continuous Scott domains***.

A ***subbasis*** of a bounded complete domain D is a subset S such that the joins of finite consistent subsets of S form a basis, called the basis ***induced*** by S.

A poset is ***coherently complete*** if every pairwise consistent subset has a least upper bound. Plotkin [Plo78] says that a poset is coherent in this case. We follow the terminology of [Str94], because coherence is used to denote many different notions in computing science and mathematics. Every coherently complete poset is bounded complete, but not conversely. A ***coherently complete domain*** is a continuous dcpo which is also coherently complete.

For any poset P we write

1. $x \uparrow y$ in P iff $\{x, y\}$ is bounded,

2. $x \# y$ iff not $x \uparrow y$.

In these cases we respectively say that x and y are ***consistent*** and that x and y lie ***apart***. This terminology is taken from [Plo77, Plo78]. Monotone functions preserve the consistency relation and reflect the apartness relation. A bounded complete domain is coherently complete iff whenever an element is apart from the least upper bound of a bounded set, it is already apart from some member of the set.

2.5 Retracts and universal domains

A ***section-retraction*** pair between objects X and Y of a category $\mathbf{X}$ consists of morphisms

$$X \underset{s}{\overset{r}{\leftrightarrows}} Y \quad \text{with} \quad r \circ s = id_X.$$

In this case $s \circ r$ is an idempotent on Y and X is called a ***retract*** of Y.

The categories of domains, continuous lattices, bounded complete domains, and coherently complete domains are closed under the formation of retracts in the ambient category of dcpos and continuous maps.

An object of a category is ***universal*** if it has every object of the category as a retract. This (well-established) terminology is obviously misleading, as the term "universal" is already used in another sense in category theory.

Let $\mathbb{B} = \{\mathbf{tt}, \mathbf{ff}\}$ be the discretely ordered ***set of truth values***, $\mathcal{B} = \mathbb{B}_\perp$ be the ***flat domain of truth values***. Then $\mathcal{B}^\omega$ is a universal domain in the category of countably based coherently complete domains and strict continuous maps (and hence is also universal in the larger category with all continuous maps) [Plo78]. (But notice that it is not known whether every coherently complete domain is the retract of some power of $\mathcal{B}$; Plotkin [Plo78] conjectures that the answer is negative.)

2.6 Function spaces

We assume that the reader is familiar with the concept of cartesian closed category [ML71, McL92, Poi92]. Roughly, a cartesian closed category is a category with finite products and well-behaved function spaces, called exponentials.

The category of dcpos and continuous maps is cartesian closed. The empty product is the one-point dcpo, and binary products are obtained by forming the set-theoretical product and endowing it with the componentwise order defined by

$$(x, y) \sqsubseteq (u, v) \text{ iff } x \sqsubseteq u \text{ and } y \sqsubseteq v.$$

Exponentials are given by the set of all continuous maps endowed with the pointwise order defined by

$$f \sqsubseteq g \text{ iff } f(x) \sqsubseteq g(x) \text{ for all } x.$$

The exponential $D \Rightarrow E$ of two dcpos is endowed with the application map $\mathrm{app} : (D \Rightarrow E) \times D \to E$ defined by

$$\mathrm{app}(f, x) = f(x),$$

which is continuous. Moreover, in this case, for any dcpo C there is a bijection between continuous maps $C \times D \to E$ and continuous maps $C \to (D \Rightarrow E)$ which sends a map $f : C \times D \to E$ to the map $\mathrm{curry}(f) : C \to (D \Rightarrow E)$ defined by

$$\mathrm{curry}(f)(x)(y) = f(x, y),$$

also called the ***transpose*** of f. In the other direction, the bijection sends a map $g : C \to (D \Rightarrow E)$ to the map $\mathrm{uncurry}(g) : C \times D \to E$ defined by

$$\mathrm{uncurry}(g)(x, y) = g(x)(y).$$

For any pointed dcpo E, the function $\mathrm{fix} : (E \Rightarrow E) \to E$ which sends a function $f \in (E \Rightarrow E)$ to its least fixed-point $\bigsqcup^{\uparrow}_{n} f^n(\bot) \in E$ is continuous, and it is called the ***least fixed-point combinator***. In applications, it is often the case that E is a function space $C \Rightarrow D$, and the least fixed-point operator is used to solve functional equations of the form $f = F(f)$ for $F : (C \Rightarrow D) \to (C \Rightarrow D)$ continuous. Such a functional equation is often called a ***recursive definition*** of f, and it is implicitly assumed that it defines f to be the least fixed-point of F.

The category of domains and continuous maps fails to be cartesian closed, because exponentials fail to exist in general. However, the full subcategories of continuous lattices, bounded complete domains, and coherently complete domains are cartesian closed, and, moreover, products and function spaces are calculated as in the category of dcpos and continuous maps. There are other interesting examples of cartesian closed categories of domains [Jun90, JS96a, JS96b], but they are outside the scope of this work.

If D and E are bounded complete domains with (countable) bases A and B respectively, then a (countable) subbasis of $D \Rightarrow E$ is the set of ***single-step functions*** $A \Rightarrow_s B = \{a \Mapsto b | a \in A, b \in B\}$, where

$$(a \Mapsto b)(x) = \begin{cases} b & \text{if } a \ll x, \\ \bot & \text{otherwise.} \end{cases}$$

The induced (countable) basis is denoted by $A \Rightarrow B$, and its members (finite joins of consistent single-step functions) are referred to as ***step functions***.

2.7 Ideals and round ideals

Let P be a poset. An ***ideal*** in P is a directed lower set in P. The set of all ideals of P ordered by inclusion is an algebraic domain, referred to as the ***ideal completion of*** P. Every algebraic domain is isomorphic to the ideal completion of its basis of finite elements. The isomorphism is given by the map which sends an element x to the ideal of finite elements below x.

Every continuous domain is a retract of an algebraic domain. In fact, if B is a basis of a domain D, then D is a retract of the ideal completion of B. The section sends an element $x \in D$ to the ideal $\twoheaddownarrow x \cap B$, and the retraction sends an ideal $I \subseteq B$ to its least upper bound $\bigsqcup^{\uparrow} I \in D$.

Smyth [Smy77] introduced round ideals in the context of R-structures (also called abstract bases in [AJ94]) as a generalization of Dedekind cuts, with the purpose of constructing continuous domains as certain completions of their bases, discussed below.

An ***abstract basis*** is a transitively ordered set $(B, \prec)$ such that for all finite $M \subseteq B$,

$M \prec x$ implies $M \prec y \prec x$ for some $y \in B$.

Here $M \prec x$ is a short-hand for $m \prec x$ for each $m \in M$. Lower sets and directed sets for transitively ordered sets are defined in the same way as for partially ordered sets. A ***round ideal*** of an abstract basis $(B, \prec)$ is a directed set A with $A = \downarrow A$, or, equivalently, such that

1. for all $x \prec y$ in B, if $y \in A$ then $x \in A$, and

2. for every $x \in A$ there is some $y \in A$ with $x \prec y$.

The ***round completion*** of an abstract basis consists of the set of round ideals ordered by inclusion. The round completion of any abstract basis is a domain; moreover, the image of the embedding $b \mapsto \downarrow b$ of B into the completion is a basis of the completion. Conversely, for any basis B of a domain D, the restriction of the way-below order of D to B gives rise to an abstract basis, with round completion isomorphic to D.

2.8 Injective spaces

In typical applications of domain theory, one starts by embedding given spaces into appropriate domains (see e.g. [Sco76, Plo78, Eda95e, Eda95b, Eda96a, Eda96b, EE96a, Esc96a, EH96]). Thus, given spaces X and Y embedded into domains D and E respectively, it is natural to ask whether the continuous maps $D \to E$ capture the continuous maps $X \to Y$, in the sense that every continuous map $f : X \to Y$ extends to a continuous map $h : D \to E$. Since every continuous map $f : X \to Y$ trivially coextends to a continuous map $g : X \to E$, it suffices to consider extensions of continuous maps $g : X \to E$ to continuous maps $h : D \to E$. This brings us to the subject of injective spaces.

Let $\mathbf{X}$ be a category and J be a class of monomorphisms of $\mathbf{X}$ closed under composition with isomorphisms. An object $E \in \mathbf{X}$ is ***injective*** with respect to J if every morphism $g : X \to E$ extends to a morphism $h : D \to E$, for any subobject inclusion $j : X \to D$ in J:

$$\begin{tikzcd} X \arrow[rr, hook, "j"] \arrow[dr, "g"'] & & D \arrow[dl, dotted, "h"] \\ & E & \end{tikzcd}$$

Here we consider the category of T_0 spaces.

When the class J consists of all subspace inclusions, we simply speak of ***injective spaces***, as in [Sco72a], and we have that:

The injective spaces are the continuous lattices.

Moreover, extensions can be found in a canonical way. If X is a subspace of a space Y, and E is any injective space, then every continuous map $g : X \to E$ has a *greatest* continuous extension $\hat{g} : D \to E$, with respect to the pointwise order induced by the specialization order of E, given by

$$\hat{g}(d) = \bigsqcup^{\uparrow}_{d \in V \in \Omega D} \; \prod_{x \in V \cap X} g(x).$$

Here ΩD is the lattice of open sets of D.

In several applications, the spaces embedded into domains are Hausdorff (e.g. the discrete space of natural numbers, the ω-power of the discrete space $\{0, 1\}$ of bits) and the embeddings are onto the subspace of maximal points of bounded complete domains (e.g., the flat domain of natural numbers, the domain of finite and infinite sequences of bits ordered by prefix). In our applications, we consider the embedding of the Euclidean real line onto the subspace of maximal points of the interval domain discussed in Chapter 8.

The closure of any subset of a domain contains its lower set. By Zorn's Lemma, every point of a domain is below some maximal point. Hence the closure of the subspace of maximal points is the whole domain. Therefore the set of maximal points of a domain is dense.

A space is ***densely injective*** if it is injective with respect to the class of dense subspace inclusions. Since there are fewer dense subspace inclusions than subspace inclusions, there are more densely injective spaces than injective spaces, and we have that:

The densely injective spaces are the bounded complete domains.

Notice that for this to be true we have to admit the empty bounded complete domain (cf. Section 2.4). Again, extensions can be found in a canonical way. If X is a dense subspace of a space Y, and E is any domain, then every continuous map $g : X \to E$ has a greatest continuous extension $\hat{g} : Y \to E$, given by the same rule as for injective spaces. The fact that X is dense is equivalent to the fact that the set $X \cap V$ is non-empty for every non-empty open subset V of Y. Therefore the meet cannot be the missing top element, which shows that the rule gives rise to a well-defined map.

The examples of embeddings given above are, in fact, embeddings into coherently complete domains. A subspace X of a space Y is called ***isochordal*** in [Plo78] if for any pair of disjoint open sets U and V in X there is a pair of disjoint open sets U' and V' in Y such that $U' \cap X = U$ and $V' \cap X = V$. For example, every dense subspace is isochordal. A space is ***isochordally injective*** if it is injective with respect to the class of isochordal subspace inclusions. There is no known domain-theoretic characterization of the isochordally injective spaces. However, there is a domain-theoretic characterization of the isochordalily injective second countable spaces:

> *In the category of* second countable T_0 *spaces, the isochordally injective spaces are the* countably based*, coherently complete domains.*

2.9 Bifree algebras

Let $\mathbf{X}$ be a category and $\mathbf{F} : \mathbf{X} \to \mathbf{X}$ be a functor. Recall that an **F*-algebra*** is a morphism $\alpha : \mathbf{F}X \to X$, and that an **F*-algebra homomorphism*** from an algebra $\alpha : \mathbf{F}X \to X$ to an algebra $\beta : \mathbf{F}Y \to Y$ is a morphism $h : X \to Y$ such that $h \circ \alpha = \beta \circ \mathbf{F}h$:

$$\begin{array}{ccc} \mathbf{F}X & \xrightarrow{\alpha} & X \\ {\scriptstyle \mathbf{F}h}\big\downarrow & & \big\downarrow{\scriptstyle h} \\ \mathbf{F}Y & \xrightarrow[\beta]{} & Y \end{array}$$

Dually, an **F*-coalgebra*** is a morphism $\alpha : X \to \mathbf{F}X$, and an **F*-coalgebra homomorphism*** from a coalgebra $\alpha : X \to \mathbf{F}X$ to a coalgebra $\beta : X \to \mathbf{F}X$ is a morphism $h : X \to Y$ such that $\mathbf{F}h \circ \alpha = \beta \circ h$:

$$\begin{array}{ccc} X & \xrightarrow{\alpha} & \mathbf{F}X \\ {\scriptstyle h}\big\downarrow & & \big\downarrow{\scriptstyle \mathbf{F}h} \\ Y & \xrightarrow[\beta]{} & \mathbf{F}Y \end{array}$$

Algebras (resp. coalgebras) compose and form a category.

If $\alpha : \mathbf{F}X \to X$ is an initial algebra then α is an isomorphism. Freyd [Fre90, Fre91, Fre92] attributes this fact to Lambek.

Definition 2.5 A ***bifree algebra*** is an initial algebra $\alpha : \mathbf{F}X \to X$ whose inverse is a final coalgebra (Freyd speaks of free algebras in this case). □

Sometimes a bifree algebra $\alpha : \mathbf{F}X \to X$ is referred to as the ***canonical solution*** to the ***domain equation***

$$X \cong \mathbf{F}X.$$

In this work we consider bifree algebras in the category **SDom** of bounded complete domains and strict continuous maps. A functor $\mathbf{F} : \mathbf{SDom} \to \mathbf{SDom}$ is ***locally continuous*** if for all D and E, the map

$$f \mapsto \mathbf{F}f : (D \Rightarrow E) \to (\mathbf{F}D \Rightarrow \mathbf{F}E)$$

is continuous. A basic result of Smyth and Plotkin [SP82] is that

> *Every locally continuous functor* $\mathbf{F} : \mathbf{SDom} \to \mathbf{SDom}$ *has a bifree algebra.*

This result is stated and proved in a much more general form which does not concern us in this work. Also, it is not relevant for the purposes of this work how the canonical solution is constructed, because we can deduce the results that we are interested in from the formal properties of bifree algebras. The interested reader can find the construction of the bifree algebra as a bilimit in [AJ94, Plo80, SP82].

Chapter 3

Effectively given domains

This chapter introduces effectively given domains. The material of Section 3.3 on effectively given algebraic domains is standard [EC76, Plo80]. However, the material on effectively given coherently complete domains is slightly non-standard, but it is based on well-known ideas by Plotkin [Plo78] and Smyth [Smy77].

3.1 Recursion theory preliminaries

Recall that a ***pairing function*** is a recursive bijection $(m, n) \mapsto \langle m, n\rangle : \mathbb{N}^2 \to \mathbb{N}$. For any pairing function there are unique recursive maps $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N}$ such that $l \mapsto (\pi_1(l), \pi_2(l))$ is its inverse. See Rogers [Rog67] for a construction of a pairing function. In this section we consider an unspecified pairing function denoted as above, with projections also denoted as above.

Definition 3.1 Let $A \subseteq \mathbb{N}$ be a set of the form $\{n_1, n_2, \ldots, n_k\}$ with $n_1 < n_2 < \cdots < n_k$. Then the natural number $\sum_{i=1}^{k} 2^{n_i}$ is called the ***canonical index*** of A. Notice that if A is empty then the canonical index of A is 0. The finite set with canonical index n is denoted by Δ_n. □

The idea is to use binary expansions. The condition $n_1 < n_2 < \cdots < n_k$ is not actually needed; it is enough that $n_i \neq n_j$ for $i \neq j$. Rogers writes D_n instead of Δ_n, but this is in conflict with the usual notation for domains. The particular form of the indexing of finite sets given in Definition 3.1 is not important. The only relevant properties are:

1. The map $n \mapsto \Delta_n$ is a bijection between natural numbers and finite sets of natural numbers.

2. There are recursive functions δ and ρ such that if Δ_n is non-empty then

   (a) $\delta(n) \in \Delta_n$,

   (b) $\Delta_{\rho(n)} = \Delta_n - \{\delta(n)\}$.

Definition 3.2 Let A and B be sets endowed with enumerations $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ of its elements respectively.

1. A subset X of A is ***recursive*** (respectively r.e.) if the set $\hat{X} = \{n \in \mathbb{N} | a_n \in X\}$ is recursive (respectively r.e.).

2. A function $f : A \to B$ is ***recursive*** if there is a recursive function $\hat{f} : \mathbb{N} \to \mathbb{N}$ such that $f(a_n) = b_{\hat{f}(n)}$.

3. A relation $R \subseteq A \times B$ is ***recursive*** (respectively r.e.) if the relation $\hat{R} \subseteq \mathbb{N} \times \mathbb{N}$ defined by $m\hat{R}n$ iff $a_m R b_n$ is recursive (respectively r.e.). This definition generalizes to n-ary relations on n enumerated sets in the obvious way.

4. A set $\mathcal{C}$ of finite subsets of A is recursive (respectively r.e.) if the set $\{n \mid \{b_k | k \in \Delta_n\} \in \mathcal{C}\}$ is recursive (respectively r.e.).

5. A relation $R \subseteq A \times \mathcal{P}_{\text{fin}} B$ is ***recursive*** (respectively r.e.) if the relation $\hat{R} \subseteq \mathbb{N} \times \mathbb{N}$ defined by $m\hat{R}n$ iff $a_m R\{b_k | k \in \Delta_n\}$ is recursive (respectively r.e.). □

3.2 Effectively given domains

This subsection contains preliminary definitions and motivations for the definitions of algebraic and continuous effectively given domains given in the following sections.

Definition 3.3 Let D and E be domains together with enumerated bases A and B respectively.

1. An *element* $x \in D$ is ***computable*** if the set $A_x \stackrel{\text{def}}{=} \mathord{\downarrow\!\!\downarrow} x \cap A$ is recursively enumerable.

2. A *continuous function* $f : D \to E$ is ***computable*** if the relation

$$\text{Graph}(f) \stackrel{\text{def}}{=} \{(a, b) \in A \times B | b \ll f(a)\}$$

is recursively enumerable. □

These definitions of computability are appropriate only if certain properties of the enumerations are assumed. First, computable functions should form a category under ordinary function composition; that is, the identity function should be computable, and the composition of two composable computable functions should be computable. For instance, the identity is computable iff the way-below order restricted to basis elements is recursively enumerable. Moreover, if $f : D \to E$ is computable and $x \in D$ is computable, $f(x)$ should be computable. Also, it would be reasonable to have that $x \in D$ is computable iff there is some r.e. directed subset of A with lub x, and this cannot be proved without further assumptions on the enumeration of A.

Second, a category of computable functions appropriate for higher-order computability has to be cartesian closed. We restrict our attention to the category of bounded complete domains as it is done in [EC76, Plo80, Smy77].

Convention 3.4 In this chapter, the term ***domain*** designates a bounded complete countably based domain. □

To say that a cartesian category of computable maps is closed amounts to saying that the evaluation map is computable and that a function defined on a cartesian product is computable iff its transpose is computable. Given domains D and E endowed with enumerated bases $(A, \{a_n\}_{n\in\mathbb{N}})$ and $(B, \{b_n\}_{n\in\mathbb{N}})$ respectively, it is natural to construct an enumerated basis $(A \Rightarrow B, \{(a \Rightarrow b)_n\}_{n\in\mathbb{N}})$ of the function space $D \Rightarrow E$ as follows:

1. $A \Rightarrow B$ is the basis induced by the subbasis consisting of single-step functions $a \Mapsto b$ for $a \in A$ and $b \in B$ (cf. 2.6).

2. $(a \Rightarrow b)_n = \begin{cases} \bigsqcup M & \text{if } M \stackrel{\text{def}}{=} \{a_l \Mapsto b_m \mid \langle l, m\rangle \in \Delta_n\} \text{ is consistent,} \\ \bot & \text{otherwise.} \end{cases}$

It would be reasonable to have that a continuous map $f : D \to E$ is computable as a function iff it is computable as an element of the function space $D \Rightarrow E$. Since consistency is not decidable for arbitrary enumerations [Smy77], this is not true in general. In particular, it follows that the category of domains with enumerated bases and computable functions is not cartesian closed.

3.3 Effectively given algebraic domains

This section is based on [EC76, Plo80]. We only present the (parts of the) proofs which are necessary for this work.

Definition 3.5 An ***effectively given algebraic domain*** is an algebraic domain D together with an enumeration $\{b_n\}_{n\in\mathbb{N}}$ of K_D such that the following relations are recursive:

1. $x{\uparrow}y$,

2. $x = y \sqcup z$,

for x, y, and z ranging over K_D (cf. Definition 3.2).

The notions of computability of elements and functions are given by Definition 3.3. □

Lemma 3.6 *$(D, \{b_n\}_{n\in\mathbb{N}})$ is an effectively given algebraic domain iff the following are recursive:*

1. *the set of finite consistent subsets of K_D,*

2. *the relation $x \sqsubseteq \bigsqcup Y$ for $x \in \mathrm{K}_D$ and $Y \subseteq_{\text{fin}} \mathrm{K}_D$.*

Lemma 3.7 *In order to effectively present an algebraic domain D it suffices to give an enumeration $\{s_n\}_{n\in\mathbb{N}}$ of a subbasis S of D contained in K_D, such that the following are recursive:*

1. *the set of finite consistent subsets of* S,

2. *the relation* $x \sqsubseteq \bigsqcup X$ *for* $x \in S$ *and* $X \subseteq_{\text{fin}} S$.

Proof Take

$$b_n \;=\; \begin{cases} \bigsqcup M & \text{if } M \stackrel{\text{def}}{=} \{s_k | k \in \Delta_n\} \text{ is consistent,} \\ \bot & \text{otherwise.} \end{cases} \qquad \square$$

Proposition 3.8 *An element* x *of an effectively given algebraic domain* $(D, \{b_n\}_{n \in \mathbb{N}})$ *is computable iff there is a primitive recursive function* $k \mapsto n_k$ *such that the set* $\{b_{n_k}\}_{k \in \mathbb{N}}$ *is consistent and has lub* x.

We say that a set M is ***way-consistent*** it is has an upper bound in the way-below order.

Lemma 3.9 *Let* D *and* E *be bounded complete algebraic domains, and* $\{a_i\}_{i \in I}$ *and* $\{b_i\}_{i \in I}$ *be finite families of elements of* D *and* E *respectively. Then the following statements are equivalent:*

1. $\{a_i \Mapsto b_i\}_{i \in I}$ *is a consistent subset of* $D \Rightarrow E$.

2. *For every* $J \subseteq I$, *if* $\{a_j\}_{j \in J}$ *is way-consistent then* $\{b_j\}_{j \in J}$ *is consistent.*

As a corollary, we have that

Lemma 3.10 *Let* D *and* E *be bounded complete domains respectively,* $\{a_i\}_{i \in I}$ *and* $\{b_i\}_{i \in I}$ *be finite families of compact elements of* D *and* E *respectively, and assume that the set* $\{a_i \Mapsto b_i\}_{i \in I}$ *is consistent. Then the following statements hold:*

1. $\bigsqcup_{i \in I}\{a_i \Mapsto b_i\} \sqsubseteq f$ *iff* $b_i \sqsubseteq f(x)$ *for all* $i \in I$ *and all compact* $x \in \uparrow a_i$.

2. $\bigsqcup_{i \in I}\{a_i \Mapsto b_i\} \mathrel{\#} f$ *iff* $b_i \mathrel{\#} f(x)$ *for some* $i \in I$ *and some compact* $x \in \uparrow a_i$.

Recall that, under the conditions of the above lemma, $\bigsqcup\{a_i \Mapsto b_i\}_{i \in I}$ is a compact element of $D \Rightarrow E$.

Proposition 3.11 *A map* f *between effectively given algebraic domains* D *and* E *is computable as a function* $D \to E$ *iff it is computable as an element of the function space* $D \Rightarrow E$. *An element* $x \in D$ *is computable iff the global element* $x : 1 \to D$ *is a computable function.*

Proposition 3.12 *The category of effectively given algebraic domains and computable functions is cartesian closed.*

In particular, the evaluation map is computable, and thus if $f : D \to E$ and $x \in D$ are computable so is $f(x)$.

Proof Let $(D, \{a_n\}_{n \in \mathbb{N}})$ and $(E, \{b_n\}_{n \in \mathbb{N}})$ be effectively given algebraic domains. We only consider function spaces. Define an enumeration of the subbasis of $D \Rightarrow E$ consisting of finite single-step functions by $s_{\langle m,n \rangle} = (a_m \Rightarrow b_n)$. Then the conditions of Lemma 3.7 apply, as Lemmas 3.9 and 3.6 show. If the induced enumeration of the the step functions of $D \Rightarrow E$ is denoted by $\{(a \Rightarrow b)_n\}_{n \in \mathbb{N}}$ then $(D \Rightarrow E, \{(a \Rightarrow b)_n\}_{n \in \mathbb{N}})$ is an effectively given algebraic domain. $\square$

Lemma 3.13 *For any effectively given algebraic domain D, the least fixed-point combinator of type $(D \Rightarrow D) \to D$ is computable.*

3.4 Effectively given coherently complete domains

As we have seen, ω-algebraic bounded complete domains have a relatively simple theory of effectivity. Unfortunately, this is not the case for ω-continuous bounded complete domains.

Smyth [Smy77] defines three notions of effectivity for ω-continuous bounded complete domains, namely effectively given *domain*, effectively given *M-domain*, and effectively given *A-domain*. The notions are equivalent, in the sense that they can be effectively translated to each other. Moreover, it can be shown that they give rise to equivalent categories of effective maps.

The notion of effectively given A-domain gives rise to a simple notion of effectively given coherently complete domain, as it is shown below. We slightly modify Smyth's definition in order to make effective presentations explicit, as Kanda and Park [KP79] showed that it is possible to effectively present some domains in essentially different ways (as it is made clear in Chapter 6):

Definition 3.14 An ***effectively given A-domain*** is a computable retract of an effectively given algebraic domain. More precisely, it is a list $D = (D, E, s, r)$ where D is a domain, E is an effectively given algebraic domain, $(s : D \to E, r : E \to D)$ is a section-retraction pair, and the idempotent $s \circ r : E \to E$ is computable. The list (E, s, r) is referred to as the ***effective presentation*** of D. □

The idea is to reduce effectivity in the continuous case to effectivity in the algebraic case, using the fact that every continuous domain is a retract of an algebraic domain.

Since $\mathcal{B}^\omega$ is a universal coherently complete domain with a simple theory of effectivity [Plo78], it is natural to specialize the above definition to the case when $E = \mathcal{B}^\omega$, obtaining a simple definition of effectively given coherently complete domain.

An element $p \in \mathcal{B}^\omega$ is finite iff $p^{-1}(\mathbf{tt})$ and $p^{-1}(\mathbf{ff})$ are finite subsets of ω. Plotkin [Plo78] considers the following effective presentation of $\mathcal{B}^\omega$:

$b_n = p$, where p is the unique finite element such that

$$n = \sum_{i \in p^{-1}(\mathbf{tt})} 2 \cdot 3^i + \sum_{i \in p^{-1}(\mathbf{ff})} 3^i.$$

The idea is to use ternary expansions. It turns out that

$p \in \mathcal{B}^\omega$ is computable iff $p^{-1}(\mathbf{tt})$ and $p^{-1}(\mathbf{ff})$ are r.e. subsets of ω,

and that

A function $g : \mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable iff the relation

$$b \sqsubseteq g(a)$$

is r.e. for finite $a, b \in \mathcal{B}^\omega$.

Definition 3.15 We call this effective presentation the ***standard effective presentation of*** $\mathcal{B}^\omega$, and from now on we implicitly assume the standard effective presentation of $\mathcal{B}^\omega$, unless otherwise stated. $\square$

This induces effective presentations on the product $\mathcal{B}^\omega \times \mathcal{B}^\omega$ and the function space $\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega$ as we have discussed above.

The following definition is certainly not original, but it is non-standard. Therefore we are forced to prove some results about it.

Definition 3.16

1. An ***effective presentation of a coherently complete domain*** D is a section-retraction pair
$$(s : D \to \mathcal{B}^\omega, r : \mathcal{B}^\omega \to D)$$
such that the induced idempotent $s \circ r : \mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable.

2. An ***effectively given coherently complete domain*** is a list (D, s, r) where D is a coherently complete domain and (s, r) is an effective presentation of D.

3. Let (D, s_D, r_D) and (E, s_E, r_E) be effectively given coherent domains.

   (a) An *element* $x \in D$ is ***computable*** if $s_D(x) \in \mathcal{B}^\omega$ is computable. This is equivalent to say that there is some computable $p \in \mathcal{B}^\omega$ such that $x = r_D(p)$.

   (b) A *continuous function* $f : D \to E$ is ***computable*** if the function $(r_D \Rightarrow s_E)(f) : \mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable:
$$\begin{array}{ccc} \mathcal{B}^\omega & \xrightarrow{(r_D \Rightarrow s_E)(f)} & \mathcal{B}^\omega \\ {\scriptstyle r_D}\downarrow & & \uparrow{\scriptstyle s_E} \\ D & \xrightarrow[f]{} & E \end{array}$$
   This is equivalent to either of the following conditions:
   - There is a computable $g : \mathcal{B}^\omega \to \mathcal{B}^\omega$ with $f = (s_D \Rightarrow r_E)(g)$.
   - There is a computable $g : \mathcal{B}^\omega \to \mathcal{B}^\omega$ such that the following diagram commutes:
$$\begin{array}{ccc} \mathcal{B}^\omega & \xrightarrow{g} & \mathcal{B}^\omega \\ {\scriptstyle r_D}\downarrow & & \downarrow{\scriptstyle r_E} \\ D & \xrightarrow[f]{} & E \end{array}$$

$\square$

Proposition 3.17 *Any effectively given coherently complete domain* (D, s, r) *has a basis of computable elements. In particular, any finite element of* D *is computable.*

Proof It is given by $r\,(\mathrm{K}_{\mathcal{B}^\omega})$. $\square$

We already know that coherently complete domains form a cartesian closed category. The domains $\mathcal{B}^\omega \times \mathcal{B}^\omega$ and $\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega$ are retracts of $\mathcal{B}^\omega$, simply because $\mathcal{B}^\omega$ is universal. In fact, $\mathcal{B}^\omega \times \mathcal{B}^\omega$ is isomorphic to $\mathcal{B}^\omega$. An isomorphism is given by $\mathrm{Pair} : \mathcal{B}^\omega \to \mathcal{B}^\omega \times \mathcal{B}^\omega$ defined by

$$\begin{aligned} \mathrm{Pair}^{-1}(p,q)(2n) &= p(n) \\ \mathrm{Pair}^{-1}(p,q)(2n+1) &= q(n). \end{aligned}$$

Then $(\mathcal{B}^\omega \times \mathcal{B}^\omega, \mathrm{Pair}^{-1}, \mathrm{Pair})$ is an effectively given coherently complete domain. We assume the section-retraction pair

$$(\mathrm{Pred} : (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega) \to \mathcal{B}^\omega, \mathrm{Fun} : \mathcal{B}^\omega \to (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega))$$

constructed in [Plo78]. But notice that the details of the construction are unimportant; the relevant fact is that there are such computable maps. Since Pred and Fun are computable, so is $\mathrm{Pred} \circ \mathrm{Fun} : \mathcal{B}^\omega \to \mathcal{B}^\omega$. Therefore $(\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega, \mathrm{Pred}, \mathrm{Fun})$ is an effectively given coherently complete domain.

Theorem 3.18 *Effectively given coherently complete domains and computable functions form a cartesian closed category.*

Proof Let C, D, E be effectively given coherently complete domains. The identity of D is computable, because $s_D \circ \mathrm{id}_D \circ r_D = s_D \circ r_D$ is computable by definition. Let $f : C \to D$ and $g : D \to E$ be computable. Then

$$s_E \circ g \circ f \circ r_C \;=\; s_E \circ g \circ (r_D \circ s_D) \circ f \circ r_C = (s_E \circ g \circ r_D) \circ (s_D \circ f \circ r_C).$$

Hence $s_E \circ g \circ f \circ r_C$ is computable, because by definition $s_E \circ g \circ r_D$ and $s_D \circ f \circ r_C$ are computable functions $\mathcal{B}^\omega \to \mathcal{B}^\omega$, and the composition of two computable functions $\mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable. Thus $g \circ f : C \to E$ is computable. Therefore effectively given coherently complete domains and computable maps form a category.

Clearly, a terminal domain can be effectively presented, and the unique map from any effectively given coherently complete domain to it is clearly computable. An effective presentation of $D \times E$ is given by

$$(\mathrm{Pair}^{-1} \circ (s_D \times s_E), (r_D \times r_E) \circ \mathrm{Pair}),$$

The projections are easily shown to be computable, and also $f : C \to D$ and $g : C \to E$ are computable iff $\langle f, g \rangle : C \to D \times E$ is computable. Hence the category has finite products.

An effective presentation of $D \Rightarrow E$ is given by

$$(\mathrm{Pred} \circ (r_D \Rightarrow s_E), (s_D \Rightarrow r_E) \circ \mathrm{Fun}).$$

The evaluation map $\mathrm{ev} : (D \Rightarrow E) \times D \to E$ is easily checked to be computable, and also a computable function $f : C \times D \to E$ is computable iff its transpose $\bar{f} : C \to (D \Rightarrow E)$ is computable. Hence the category has exponentials, and therefore it is cartesian closed. $\square$

Definition 3.19 We denote by **ECDom** the category of effectively given coherently complete domains and computable functions.

Given effective presentations of D and E, we implicitly assume the effective presentations of $D \times E$ and $D \Rightarrow E$ constructed in the above theorem, unless otherwise stated. $\square$

Proposition 3.20 *1. A continuous map is computable as a function iff it is a computable element of the corresponding function space.*

2. An element is computable iff the induced global element is a computable function.

In particular, if $f : D \to E$ and $x \in D$ are computable, so is $f(x) \in E$.

Proof A function $f : D \to E$ is computable iff the element $f \in D \Rightarrow E$ is computable, because $s_E \circ f \circ r_D$ is computable iff $(\mathrm{Pred} \circ (r_D \Rightarrow s_E))(f)$ is computable, as

$$\mathrm{Pred}((r_D \Rightarrow s_E)(f)) \quad = \quad \mathrm{Pred}(s_E \circ f \circ r_D),$$

and $\mathrm{Pred}(\phi)$ is computable iff $\phi = \mathrm{Fun}(\mathrm{Pred}(\phi))$ is computable. The second assertion is immediate. $\square$

Proposition 3.21 *The fixed-point combinator* $\mathrm{fix}_D : (D \Rightarrow D) \to D$ *is computable for every effectively given coherently complete domain* $D = (D, s, r)$.

In particular, the least fixed-point of a computable function is computable.

Proof By Theorem 3.18 we have that

$$D \Rightarrow D = (D \Rightarrow D, s', r'),$$

where

$$s' = \mathrm{Pred} \circ (r \Rightarrow s) \qquad \text{and} \qquad r' = (s \Rightarrow r) \circ \mathrm{Fun}.$$

Hence, by definition, fix_D is computable iff there is a computable $g : \mathcal{B}^\omega \to \mathcal{B}^\omega$ with

$$\mathrm{fix}_D = (s' \Rightarrow r)(g).$$

Let $f : D \to D$. Then

$$\begin{aligned}
(s' \Rightarrow r)(\mathrm{fix}_{\mathcal{B}^\omega} \circ \mathrm{Fun})(f) &= r \circ (\mathrm{fix}_{\mathcal{B}^\omega} \circ \mathrm{Fun}) \circ s'(f) \\
&= r \circ (\mathrm{fix}_{\mathcal{B}^\omega} \circ \mathrm{Fun}) \circ (\mathrm{Pred} \circ (r \Rightarrow s))(f) \\
&= r \circ \mathrm{fix}_{\mathcal{B}^\omega} \circ (r \Rightarrow s)(f) \\
&= r \circ \mathrm{fix}_{\mathcal{B}^\omega}(s \circ f \circ r) \\
&= r\left(\bigsqcup_{n\in\omega}^{\uparrow} (s \circ f \circ r)^n(\perp)\right) \\
&= r\left(\bigsqcup_{n\in\omega}^{\uparrow} s \circ f^n \circ r(\perp)\right) \\
&= r \circ s\left(\bigsqcup_{n\in\omega}^{\uparrow} f^n(r(\perp))\right) \\
&= \bigsqcup_{n\in\omega}^{\uparrow} f^n(\perp) \\
&= \mathrm{fix}_D(f).
\end{aligned}$$

Hence we can take $g = \mathrm{fix}_{\mathcal{B}^\omega} \circ \mathrm{Fun}$, because $\mathrm{fix}_{\mathcal{B}^\omega} \circ \mathrm{Fun}$ is computable. Therefore fix_D is computable. □

Let D be a bounded complete domain. The ***sequential conditional***

$$\mathrm{if} : \mathcal{B} \times D \times D \to D$$

is defined by

$$\mathrm{if}(p, x, y) = \begin{cases} x & \text{if } p = \mathbf{tt}, \\ y & \text{if } p = \mathbf{ff}, \\ \perp & \text{otherwise.} \end{cases}$$

Since bounded complete dcpos have binary meets, and since the binary meet operation is continuous in bounded complete domains, we also have the ***parallel conditional***

$$\mathrm{pif} : \mathcal{B} \times D \times D \to D$$

defined by

$$\mathrm{pif}(p, x, y) = \begin{cases} x & \text{if } p = \mathbf{tt}, \\ y & \text{if } p = \mathbf{ff}, \\ x \sqcap y & \text{otherwise.} \end{cases}$$

We sometimes write

$$\text{if } p \text{ then } x \text{ else } y$$

and

$$\text{pif } p \text{ then } x \text{ else } y$$

instead of $\mathrm{if}(p, x, y)$ and $\mathrm{pif}(p, x, y)$ respectively.

Proposition 3.22 *For any effectively given coherently complete domain D, the sequential and parallel conditionals if_D and pif_D are computable maps, regardless of the choice of effective presentation of the domain of truth-values.*

Proof As above, using the fact that $\mathrm{if}_{\mathcal{B}^\omega}$ and $\mathrm{pif}_{\mathcal{B}^\omega}$ are computable [Plo78]. □

Definition 3.23 **ESDom** is the category of effectively given coherently complete domains and strict computable maps. $\square$

We say that a functor $\mathbf{F} : \mathbf{ESDom} \to \mathbf{ESDom}$ is ***locally computable*** if for all D and E, the map

$$f \mapsto \mathbf{F}f : (D \Rightarrow E) \to (\mathbf{F}D \Rightarrow \mathbf{F}E)$$

is computable, uniformly in the effective presentations of D and E, which means that there is a single computable $\mathbf{F} : (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega) \to (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega)$. From [Plo78, Plo80, Smy77] we know that functors formed from basic constructions such as product and exponential by composition are locally computable, and that

Every locally computable functor has a computable bifree algebra.

Chapter 4

The programming language PCF

We introduce the basic notions of PCF needed in this thesis. See [Gun92] for an excellent textbook account.

This chapter is based on [Plo77], with minor adaptations and simplifications convenient to our needs, and can be safely skipped and used as a reference.

4.1 The languages $\mathcal{L}_{\mathrm{DA}}$, $\mathcal{L}_{\mathrm{PA}}$ and $\mathcal{L}_{\mathrm{PA}+\exists}$

Given a collection of symbols called ***ground types***, the set of ***types*** is the least set containing the ground types and containing the formal expression $(\sigma \to \tau)$ whenever it contains the expressions σ and τ. The greek letters σ and τ range over types. We let $(\sigma_1, \ldots, \sigma_n, \tau)$ stand for $(\sigma_1 \to (\sigma_2 \to \cdots (\sigma_n \to \tau) \cdots))$.

Given a collection $\mathcal{L}$ of formal constants, each having a fixed type, and a family of formal variables $\{\alpha_i^\sigma\}$ $(i \geq 0)$ for each type σ, the ***$\mathcal{L}$-terms*** are given by the following inductive rules:

1. Every variable α_i^σ is an $\mathcal{L}$-term of type σ.

2. Every constant of type σ is an $\mathcal{L}$-term of type σ.

3. If M and N are $\mathcal{L}$-terms of types $(\sigma \to \tau)$ and σ respectively then (MN) is an $\mathcal{L}$-term of type τ.

4. If M is an $\mathcal{L}$-term of type τ then $(\lambda\alpha_i^\sigma M)$ is an $\mathcal{L}$-term of type $\sigma \to \tau$.

When $\mathcal{L}$ is understood from the context it need not be used as a prefix. The letters L, M, and N range over terms. The letter c range over constants. We denote the fact that a term M has type σ by $M : \sigma$. Notice that every term has a unique type. Terms of the form (MN) are called ***combinations***. Terms of the form $(\lambda\alpha M)$ are called ***abstractions***. Parentheses around combinations and abstractions are sometimes omitted with the convention that juxtaposition associates to the left. We also omit parentheses in type expressions with the convention that $\to$ associates to the right.

The set of ***free variables*** of a term M is $\mathrm{FV}(M)$, inductively defined by

1. $\mathrm{FV}(\alpha_i^\sigma) = \{\alpha_i^\sigma\}$

2. $\mathrm{FV}(c) = \emptyset$

3. $\mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N)$

4. $\mathrm{FV}(\lambda\alpha_i^\sigma M) = \mathrm{FV}(M) - \{\alpha_i^\sigma\}$

A term M is ***closed*** if $\mathrm{FV}(M) = \emptyset$ and ***open*** otherwise. ***Programs*** are closed terms of ground type. The idea is that the ground types are the data types and programs produce data, via the operational semantics. The remaining terms are significant as subprograms. Sometimes we will informally refer to any PCF term as a program.

$[N/\alpha]M$ is the result of replacing all free occurrences of the variable α in M by N, making the appropriate changes in the bound variables of M so that no free variables of N become bound.

The ground types for the language $\mathcal{L}_{\mathrm{DA}}$ are $\mathbf{N}$ and $\mathbf{T}$, and its constants are

$\mathbf{tt} : \mathbf{T} \quad \mathbf{ff} : \mathbf{T}$

$\mathbf{if}_\sigma : (\mathbf{T}, \sigma, \sigma, \sigma)$ for σ ground

$\mathbf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$ for each σ

$\mathbf{k}_n : \mathbf{N}$ for each natural number n

$(+\mathbf{1}) : \mathbf{N} \to \mathbf{N}, \quad (-\mathbf{1}) : \mathbf{N} \to \mathbf{N}, \quad (= \mathbf{0}) : \mathbf{N} \to \mathbf{T}$

The language $\mathcal{L}_{\mathrm{PA}}$ is the language $\mathcal{L}_{\mathrm{DA}}$ extended by the constants

$\mathbf{pif}_\sigma : (\mathbf{T}, \sigma, \sigma, \sigma)$ for σ ground

The language $\mathcal{L}_{\mathrm{PA}+\exists}$ is the language $\mathcal{L}_{\mathrm{PA}}$ extended with the constant

$\exists : (\mathbf{N} \to \mathbf{T}) \to \mathbf{T}$

4.2 Denotational semantics

A ***collection of domains for PCF*** is a family $\{D_\sigma\}_\sigma$ of domains, one for each type σ, such that $D_{\sigma\to\tau} = D_\sigma \Rightarrow D_\tau$. It is ***standard*** if $D_{\mathbf{T}}$ is $\mathcal{B}$ (the flat domain of truth values), and $D_{\mathbf{N}}$ is $\mathcal{N}$ (the flat domain of natural numbers).

An ***interpretation*** of a language $\mathcal{L}$ is a collection $\{D_\sigma\}_\sigma$ of domains for PCF together with a mapping

$$c \quad \mapsto \quad \mathcal{A}[\![c]\!] : \mathcal{L} \to \bigcup_\sigma \{D_\sigma\}$$

which is type-respecting, in the sense that if $c : \sigma$ then $\mathcal{A}[\![c]\!] \in D_\sigma$.

An interpretation is ***standard*** if it interprets the constants $\mathbf{tt}$, $\mathbf{ff}$, $\mathbf{if}_\sigma$, $\mathbf{Y}_\sigma$, $\mathbf{k}_n$, $(+\mathbf{1})$, $(-\mathbf{1})$, $(= \mathbf{0})$, $\mathbf{pif}_\sigma$ and $\exists$ respectively as $\mathbf{tt}$, $\mathbf{ff}$, the sequential conditional, the fixed point operator, the natural number n, the successor function, the predecessor function, the test for zero, the parallel conditional and the continuous existential quantifier $\exists : (\mathcal{N} \Rightarrow \mathcal{B}) \to \mathcal{B}$ defined by

$$\exists(p) = \begin{cases} \mathbf{tt} & \text{if } p(n) = \mathbf{tt} \text{ for some } n \in \mathcal{N}, \\ \mathbf{ff} & \text{if } p(\bot) = \mathbf{ff}, \\ \bot & \text{otherwise.} \end{cases}$$

Notice that $p(\bot) = \mathbf{ff}$ iff $p(n) = \mathbf{ff}$ for all $n \in \mathcal{N}$, by monotonicity of p.

An interpretation $\langle \{D_\sigma\}_\sigma, \mathcal{A} \rangle$ induces a ***denotational semantics*** $\hat{\mathcal{A}}$ for $\mathcal{L}$.

First, the set Env of ***environments*** is the set of type-respecting functions from the set of variables to $\bigcup_\sigma \{D_\sigma\}$. It is ranged over by ρ. If $\alpha : \sigma$ and $x \in D_\sigma$ then $\rho[x/\alpha]$ is the environment which maps α to x and any other variable α' to $\rho(\alpha')$. The ***undefined environment*** $\bot$ maps each variable of type σ to the bottom element of the domain D_σ.

The denotational semantics

$$M \quad \mapsto \quad \left(\rho \mapsto \hat{\mathcal{A}}[\![M]\!](\rho)\right) : \text{Terms} \to (\text{Env} \to \bigcup_\sigma \{D_\sigma\})$$

is inductively defined by:

1. $\hat{\mathcal{A}}[\![\alpha]\!](\rho) = \rho(\alpha)$
2. $\hat{\mathcal{A}}[\![c]\!](\rho) = \mathcal{A}[\![c]\!]$
3. $\hat{\mathcal{A}}[\![MN]\!](\rho) = \hat{\mathcal{A}}[\![M]\!](\rho)\left(\hat{\mathcal{A}}[\![N]\!](\rho)\right)$
4. $\hat{\mathcal{A}}[\![\lambda\alpha M]\!](\rho)(x) = \hat{\mathcal{A}}[\![M]\!](\rho[x/\alpha])$ (with $x \in D_\sigma$ if $\alpha : \sigma$)

Informally,

1. A variable denotes what the environment assigns to it.
2. A constant denotes what the interpretation assigns to it.
3. If a term M denotes the function $f : D \to E$ and a term N denotes the value $x \in D$ then the combination MN denotes the value $f(x) \in E$.
4. If a term M denotes the value $y_x \in E$ in an environment which assigns the value $x \in D$ to the variable α, then the abstraction $\lambda\alpha M$ denotes the function $f : D \to E$ defined by $f(x) = y_x$.

If M is closed then its denotation does not depend on the environment, in the sense that $\hat{\mathcal{A}}[\![M]\!](\rho) = \hat{\mathcal{A}}[\![M]\!](\rho')$ for all ρ and ρ'.

In order to simplify notation, we let $[\![M]\!]$ stand for the denotation $\hat{\mathcal{A}}[\![M]\!](\bot)$ of a closed term M with respect to an implicit semantics $\hat{\mathcal{A}}$. Also, for any term M, we let $[\![M]\!](\rho)$ stand for $\hat{\mathcal{A}}[\![M]\!](\rho)$.

4.3 Operational semantics

The operational semantics of $\mathcal{L}_{\mathrm{DA}}$ is given by an ***immediate reduction relation***, defined by the following rules:

1. $(\lambda\alpha M)N \to [N/\alpha]M$
2. $\mathbf{Y}M \to M(\mathbf{Y}M)$
3. $(\mathbf{+1})\mathbf{k}_n \to \mathbf{k}_{n+1}, \quad (\mathbf{-1})\mathbf{k}_{n+1} \to \mathbf{k}_n$
4. $(= \mathbf{0})\mathbf{k}_0 \to \mathbf{tt}, \quad (= \mathbf{0})\mathbf{k}_{n+1} \to \mathbf{ff}$
5. $\mathbf{if}\,\mathbf{tt}MN \to M, \quad \mathbf{if}\,\mathbf{ff}MN \to N$
6. $\dfrac{M \to M'}{MN \to M'N}, \quad \dfrac{N \to N'}{MN \to MN'}$ if M is $\mathbf{if}$, $(\mathbf{+1})$, $(\mathbf{-1})$ or $(= \mathbf{0})$

We omit the reduction rules of the languages $\mathcal{L}_{\mathrm{PA}}$ and $\mathcal{L}_{\mathrm{PA}+\exists}$. The reduction relation preserves types, in the sense that if $M \to M'$ and M has type σ, so does M'.

Evaluation is given by a partial function Eval from programs to constants, defined by

$$\mathrm{Eval}(M) = c \text{ iff } M \to^* c$$

It is well-defined because if $M \to^* c$ and $M \to^* c'$ then $c = c'$.

The following theorem is often referred to as the ***adequacy property*** of PCF. It asserts that the operational and denotational semantics coincide.

Theorem 4.1 (Plotkin [Plo77], Theorem 3.1) *For any $\mathcal{L}_{\mathrm{DA}}$ program M and constant c,*

$$\mathrm{Eval}(M) = c \textit{ iff } [\![M]\!] = [\![c]\!].$$

Proof Lemma 4.6 below. □

The following definitions are introduced to formulate and prove Lemma 4.6.

Definition 4.2 The predicates Comp_σ are defined by induction on types by:

1. If $M : \sigma$ is a program then M has property Comp_σ iff $[\![M]\!] = [\![c]\!]$ implies $\mathrm{Eval}(M) = c$.
2. If $M : \sigma \to \tau$ is a closed term it has property $\mathrm{Comp}_{\sigma\to\tau}$ iff whenever $N : \sigma$ is a closed term with property Comp_σ then MN is a term with property Comp_τ.
3. If $M : \sigma$ is an open term with free variables $\alpha_1 : \sigma_1, \ldots, \alpha_n : \sigma_n$ then it has property Comp_σ iff $[N_1/\alpha_1]\cdots[N_2/\alpha_n]M$ has property Comp_σ whenever $N_1, \ldots, N_n$ are closed terms having properties Comp_{σ_1}, …, Comp_{σ_n} respectively.

A term of type σ is ***computable*** if it has property Comp_σ. □

If $M : \sigma \to \tau$ and $N : \sigma$ are closed computable terms, so is MN and also a term $M : (\sigma_1, \ldots, \sigma_n, \tau)$ is computable iff $\tilde{M} N_1 \ldots N_n$ is computable whenever the terms $N_1 : \sigma_1, \ldots, N_n : \sigma_n$ are closed computable terms and $\tilde{M}$ is a closed instantiation of M by computable terms.

In the following definition, α has to be chosen as some variable of appropriate type in each instance.

Definition 4.3 Define terms Ω_σ by

$$\Omega_\sigma = \mathbf{Y}_\sigma(\lambda\alpha\alpha)$$

for σ ground and

$$\Omega_{\sigma\to\tau} = \lambda\alpha\Omega_\tau,$$

and define terms $\mathbf{Y}_\sigma^{(n)}$ by induction on n by

$$\begin{aligned}
\mathbf{Y}_\sigma^{(0)} &= \lambda\alpha.\Omega_\sigma, \\
\mathbf{Y}_\sigma^{(n+1)} &= \lambda\alpha.\alpha(\mathbf{Y}_\sigma^{(n)}\alpha).\ \square
\end{aligned}$$

Then $[\![\mathbf{Y}_\sigma]\!] = \bigsqcup_n [\![\mathbf{Y}_\sigma^n]\!]$ for any standard interpretation.

Definition 4.4 The ***syntactic information order*** $\preccurlyeq$ is the least relation between terms such that

1. $\Omega_\sigma \preccurlyeq M : \sigma$ and $\mathbf{Y}_\sigma^{(n)} \preccurlyeq \mathbf{Y}_\sigma$,
2. $M \preccurlyeq M$, and
3. if $M \preccurlyeq M' : \sigma \to \tau$ and $N \preccurlyeq N' : \sigma$ then $\lambda\alpha N \preccurlyeq \lambda\alpha N'$ and also $MN \preccurlyeq M'N'$.

$\square$

Lemma 4.5 (Plotkin [Plo77], Lemma 3.2) *If $M \preccurlyeq N$ and $M \to M'$ then either $M' \preccurlyeq N$ or else for some N', $N \to N'$ and $M' \preccurlyeq N'$.*

Proof By structural induction on M and cases according to why the immediate reduction $M \to M'$ takes place. $\square$

We include the proof of the following lemma since we are going to extend it to Real PCF:

Lemma 4.6 (Plotkin [Plo77], Lemma 3.3) *Every $\mathcal{L}_{\mathrm{DA}}$-term is computable.*

Proof By structural induction on the formation rules of terms:

(1) Every variable is computable since any closed instantiation of it by a computable term is computable.

(2) Every constant other than the $\mathbf{Y}_\sigma$'s is computable. This is clear for constants of ground type. Out of $+\mathbf{1}$, $-\mathbf{1}$, $=\mathbf{0}$, and **if** we only consider $-\mathbf{1}$ as an example. It is enough to show $(-\mathbf{1})M$ computable when M is a closed computable term of type $\mathbf{N}$. Suppose $[\![(-\mathbf{1})M]\!] = [\![c]\!]$. Then $c = \mathbf{k}_m$ for some m and so $[\![M]\!] = m + 1$. Therefore as M is computable, $M \to^* \mathbf{k}_{m+1}$ and so $(-\mathbf{1})M \to^* \mathbf{k}_m = c$.

(3) If $M : \sigma \to \tau$ and $N : \sigma$ are computable, so is the combination MN. If MN is closed so are M and N and its computability follows from clause 2 of the definition of computability. If it is open, any closed instantiation L of it by computable terms has the form $\tilde{M}\tilde{N}$ where $\tilde{M}$ and $\tilde{N}$ are closed instantiations of M and N respectively and therefore themselves computable which in turn implies the computability of L and hence of MN.

(4) If $M : \tau$ is computable, so is the abstraction $\lambda\alpha M$. It is enough to show that the ground term $LN_1 \cdots N_n$ is computable when $N_1, \cdots, N_n$ are closed computable terms and L is a closed instantiation of $\lambda\alpha M$ by computable terms. Here L must have the form $\lambda\alpha\tilde{M}$ where $\tilde{M}$ is an instantiation of all free variables of M, except α, by closed computable terms. If $[\![LN_1 \cdots N_n]\!] = [\![c]\!]$, then we have $[\![[N_1/\alpha]\tilde{M}N_2 \cdots N_n]\!] = [\![LN_1 \cdots N_n]\!] = [\![c]\!]$. But $[N_1/\alpha]\tilde{M}$ is computable and so too therefore is $\tilde{M}N_2 \cdots N_n$. Therefore $LN_1 \cdots N_n \to [N_1/\alpha]\tilde{M}N_2 \cdots N_n \to^* c$, as required.

(5) Each $\mathbf{Y}_\sigma$ is computable. It is enough to prove $\mathbf{Y}_\sigma N_1 \cdots N_k$ is computable when $N_1, \cdots, N_k$ are closed computable terms and $\mathbf{Y}_\sigma N_1 \cdots N_k$ is ground. Suppose $[\![\mathbf{Y}N_1 \cdots N_k]\!] = [\![c]\!]$. Since $[\![\mathbf{Y}_\sigma]\!] = \bigsqcup_n [\![\mathbf{Y}^n_\sigma]\!]$, $[\![\mathbf{Y}^{(n)}N_1 \cdots N_k]\!] = [\![c]\!]$ for some n. Since $[\![\Omega_\sigma]\!] = \perp$ for σ ground, Ω_σ is computable for σ ground. From this and (1), (3), and (4) proved above it follows that every Ω_σ and $\mathbf{Y}^{(n)}_\sigma$ is computable. Therefore $\mathbf{Y}^{(n)}N_1 \cdots N_k \to^* c$ and so by Lemma 4.5, $\mathbf{Y}N_1 \cdots N_k \to^* c$, concluding the proof. □

This lemma extends to the languages $\mathcal{L}_{\mathrm{PA}}$ and $\mathcal{L}_{\mathrm{PA}+\exists}$. It suffices to show that **pif** and $\exists$ are computable terms, with respect to appropriate reduction rules.

Part II

Domain theory revisited

This part contains new results on domain theory, which are about, or directly related to

- bifree algebras,
- effectively given domains, and
- coherently complete domains.

In Chapter 5 we introduce the concept of inductive retraction as a generalization of the concept of bifree algebra. Inductive retractions are used in Chapter 10 of Part III in order to obtain structural recursion schemes for the partial real line.

In Chapter 6 we introduce some results about equivalence of effectively given domains. These results are used in Chapter 12 of Part IV to establish absoluteness of the notion of computability in the real numbers type hierarchy.

In Chapter 7 we characterize the coherently complete domains as the bounded complete domains having certain "joining maps". These joining maps are used in Chapter 12 to prove that every computable first-order function in the real numbers type hierarchy is Real PCF definable (without the existential quantifier).

The reader who is mainly interested in real number computation can safely proceed directly to Part III, as we clearly indicate when some chapter or portion of a chapter in the present part is needed to proceed further.

Chapter 5

Bifree algebras generalized

In this chapter we introduce a new technique for defining structural recursion schemes based on the theory of domain equations. This technique was first introduced in [Esc96b] by the author and then further developed in collaboration with Thomas Streicher [ES97]. The results obtained in collaboration are contained in Sections 5.2 and 5.3.

Recursion is a paradigmatic notion in theoretical computer science. Domain theory incorporates recursion both at the level of elements (via least fixed-points of continuous maps) and at the level of objects (via canonical solutions of domain equations). Although recursion at the element level can be explained from first principles, recursion at the object level leads to recursion at the element level [Gun92]. Moreover, recursive definitions of domains give rise to *structural* recursion at the element level [LS81, SP82], via bifree algebras (cf. Definition 2.5 in Section 2.9). For instance, both primitive recursion and general recursion (in the form of minimization) are induced by a recursive definition of the natural numbers [Smy92a].

Thus, we are led to consider recursion in the partial real line, at both levels. This is the subject of Chapter 10 in Part III. There is a fundamental problem, however, about domain equations for the partial real line, namely that the canonical solution of a domain equation involving usual functors is algebraic, but the partial real line is not algebraic.

A relevant observation is that there are two main aspects of a domain equation $D \cong \mathbf{F}D$. First, its canonical solution is uniquely determined by the functor $\mathbf{F}$. Second, its canonical solution gives rise to recursion schemes via bifree algebras. Our solution to the problem of structural recursion on the partial real line consists in giving up the first aspect and weakening the second aspect, in such a way as to still have structural recursion. In this chapter we develop and investigate such a weakening.

The basic idea is to consider not a distinguished domain D such that $D \cong \mathbf{F}D$, but instead a domain D such that D is a retract of $\mathbf{F}D$, in a special way to be defined in the technical development that follows this discussion. We refer to such a domain D as an ***$\mathbf{F}$-inductive retract***.

The section and retraction involved in the situation referred above are not a final coalgebra and an initial algebra respectively, because the existence part in the definition of finality and initiality are missing. However, the uniqueness part is present; there is at most one homomorphism from any coalgebra to the section, and there is at most one homomorphism from the retraction to any algebra.

The lack of the existence part is not a severe problem in practice, because it is often the case that in a recursive definition we know the entity which we intend to define. Hence it is enough to be sure that the entity is the only solution to the recursive definition. Therefore the uniqueness part is enough for such purposes.

5.1 Inductive retractions

The material of this section was originally introduced in [Esc96b] with the purpose of establishing both absoluteness of the notion of computability in the real numbers type hierarchy and computational completeness of Real PCF.

Convention 5.1 In the remaining of this chapter, $\mathbf{X}$ is any category and $\mathbf{F}$ is an endofunctor of $\mathbf{X}$. □

Definition 5.2 An $\mathbf{F}$-***inductive retraction*** is a pair of arrows $X \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{F}X$ such that

$$f = \alpha \circ \mathbf{F}f \circ \beta \text{ iff } f = \mathrm{id}_X. \qquad \square$$

The right-to-left implication shows that

$$\alpha \circ \beta = \mathrm{id}_X$$

and hence X is a retract of $\mathbf{F}X$. Also, notice that if $\langle \alpha, \beta \rangle$ is an $\mathbf{F}$-inductive retraction in $\mathbf{X}$, then $\langle \beta, \alpha \rangle$ is an $\mathbf{F}$-inductive retraction in $\mathbf{X}^{\mathrm{op}}$.

The following proposition shows that inductive retractions generalize bifree algebras:

Proposition 5.3 *Let $X \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{F}X$ be an $\mathbf{F}$-inductive isomorphism. If $\mathbf{F}$ has a bifree algebra then it is isomorphic to α.*

Proof Let $i : \mathbf{F}C \to C$ be a bifree algebra, $r : i \to \alpha$ be the unique algebra homomorphism and $s : \beta \to i^{-1}$ be the unique coalgebra homomorphism. This means that $r \circ i = \alpha \circ \mathbf{F}r$ and $i^{-1} \circ s = \mathbf{F}s \circ \beta$. Hence

$$\begin{aligned} r \circ s &= r \circ i \circ i^{-1} \circ s \\ &= \alpha \circ \mathbf{F}r \circ \mathbf{F}s \circ \beta \\ &= \alpha \circ \mathbf{F}(r \circ s) \circ \beta. \end{aligned}$$

By inductivity, $r \circ s = \mathrm{id}_X$. Since $s = i \circ \mathbf{F}s \circ \beta$, we have that

$$\begin{aligned} s \circ r \circ i &= i \circ \mathbf{F}s \circ \beta \circ \alpha \circ \mathbf{F}r \\ &= i \circ \mathbf{F}(s \circ r). \end{aligned}$$

Hence $s \circ r : i \to i$ and therefore $s \circ r = \mathrm{id}_C$. □

Convention 5.4 In the remaining of this chapter, $i : \mathbf{F}C \to C$ is a bifree algebra and $X \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{F}X$ is an $\mathbf{F}$-inductive retraction. $\square$

The first part of the proof of Proposition 5.3 shows that every inductive retract is a retract of the bifree algebra, in a canonical way:

Lemma 5.5 *If $r : i \to \alpha$ and $s : \beta \to i^{-1}$ are the unique (co)algebra homomorphisms then $X \underset{s}{\overset{r}{\leftrightarrows}} C$ is a retraction with $r \circ s = \mathrm{id}_X$.*

Lemma 5.6 *There are unique (co)algebra homomorphisms $\alpha \to \alpha$ and $\beta \to \beta$.*

Proof Let $f : \alpha \to \alpha$ be an algebra homomorphism. This means that $f \circ \alpha = \alpha \circ \mathbf{F}f$. Hence $f = \alpha \circ \mathbf{F}f \circ \beta$. By inductivity, $f = \mathrm{id}_C$. The second part follows by duality. $\square$

A technical lemma

The following lemma, which may appear to be rather technical, is applied to establish computational completeness of Real PCF in Chapter 12 of Part IV. The idea is to replace a canonical solution of a domain equation by a domain having the canonical solution as a retract.

Lemma 5.7 *Let $\mathbf{X}$ be a category, let $\mathbf{F} : \mathbf{X} \to \mathbf{X}$ be a functor, let*

$$C \underset{j}{\overset{i}{\leftrightarrows}} \mathbf{F}C$$

be a bifree algebra, let

$$X \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{F}X$$

be an $\mathbf{F}$-inductive section-retraction pair, let

$$C \underset{e}{\overset{p}{\leftrightarrows}} C'$$

be a section-retraction pair with $p \circ e = \mathrm{id}$, and define

$$C' \underset{j'}{\overset{i'}{\leftrightarrows}} \mathbf{F}C'$$

by

$$i' = e \circ i \circ \mathbf{F}p \qquad \textit{and} \qquad j' = \mathbf{F}e \circ j \circ p$$

Then any pair of maps

$$X \underset{s'}{\overset{r'}{\leftrightarrows}} C'$$

such that

$$r' = \alpha \circ Fr' \circ j' \qquad \textit{and} \qquad s' = i' \circ Fs' \circ \beta$$

is a section-retraction pair with $r' \circ s' = \mathrm{id}$.

Proof Let

$$X \underset{s}{\overset{r}{\leftrightarrows}} C$$

be the canonical section-retraction pair constructed in Proposition 5.5. Since

$$\begin{aligned}
r' \circ e \circ i &= \alpha \circ \mathbf{F}r' \circ j' \circ e \circ i \quad \text{by the assumption on } r' \\
&= \alpha \circ \mathbf{F}r' \circ \mathbf{F}e \circ j \circ p \circ e \circ i \quad \text{by definition of } j' \\
&= \alpha \circ \mathbf{F}(r' \circ e) \circ j \circ i \quad \text{because } p \circ e = \mathrm{id} \\
&= \alpha \circ \mathbf{F}(r' \circ e) \quad \text{because } j \text{ is the inverse of } i,
\end{aligned}$$

and since this means that $r' \circ e$ is an algebra homomorphism from i to α, it follows that $r' \circ e = r$. Similarly,

$$\begin{aligned}
j \circ p \circ s' &= j \circ p \circ i' \circ \mathbf{F}s' \circ \beta \quad \text{By the assumption on } s' \\
&= p \circ e \circ i \circ \mathbf{F}p \circ \mathbf{F}s' \circ \beta \quad \text{by definition of } i' \\
&= j \circ i \circ \mathbf{F}(p \circ s') \circ \beta \quad \text{because } p \circ e = \mathrm{id}. \\
&= \mathbf{F}(p \circ s') \circ \beta.
\end{aligned}$$

Since this means that $p \circ s'$ is a coalgebra homomorphism from β to j, it follows that $p \circ s' = s$. We also have that

$$\begin{aligned}
j' \circ i' &= \mathbf{F}e \circ j \circ p \circ e \circ i \circ \mathbf{F}p \quad \text{by definition of } i' \text{ and } j' \\
&= \mathbf{F}e \circ j \circ i \circ \mathbf{F}p \quad \text{because } p \circ e = \mathrm{id} \\
&= \mathbf{F}e \circ \mathbf{F}p \quad \text{because } j \text{ is the inverse of } i.
\end{aligned}$$

It follows that

$$\begin{aligned}
r' \circ s' &= \alpha \circ \mathbf{F}r' \circ j' \circ i' \circ \mathbf{F}s' \circ \beta \quad \text{by the assumption on } r' \text{ and } s' \\
&= \alpha \circ \mathbf{F}r' \circ \mathbf{F}e \circ \mathbf{F}p \circ \mathbf{F}s' \circ \mathbf{F}\beta \quad \text{because } j' \circ i' = \mathbf{F}e \circ \mathbf{F}p \\
&= \alpha \circ \mathbf{F}r \circ \mathbf{F}s \circ \beta \quad \text{because } r' \circ e = r \text{ and } p \circ s' = s \\
&= \alpha \circ \beta \quad \text{because } r \circ s = \mathrm{id} \\
&= \mathrm{id} \quad \text{by definition.} \; \square
\end{aligned}$$

5.2 Structural recursion

This section contains results developed in collaboration with Thomas Streicher.

Proposition 5.8 *Let $r : i \to \alpha$, $s : \beta \to i^{-1}$, $e = s \circ r : C \to C$, $h : i \to a$, and $k : b \to i^{-1}$.*

1. *For any algebra $a : \mathbf{F}A \to A$, there is a homomorphism $f : \alpha \to a$ iff $h = h \circ e$, and in this case $f = h \circ s$.*

2. *For any coalgebra $b : B \to \mathbf{F}B$, there is a homomorphism $g : b \to \beta$ iff $k = e \circ k$, and in this case $g = r \circ k$.*

Proof (1): If $f : \alpha \to a$ then $f \circ r = h$ because $r : i \to \alpha$. Therefore $f = h \circ s$ and $h = h \circ s \circ r$. Conversely, if $f \circ r : i \to a$ then $f : \alpha \to a$ because

$$\begin{aligned}
f \circ \alpha &= f \circ \alpha \circ \mathbf{F}(r \circ s) \\
&= f \circ \alpha \circ \mathbf{F}r \circ \mathbf{F}s \\
&= f \circ r \circ i \circ \mathbf{F}s \\
&= a \circ \mathbf{F}(f \circ r) \circ \mathbf{F}s \\
&= a \circ \mathbf{F}f \circ \mathbf{F}(r \circ s) \\
&= a \circ \mathbf{F}f.
\end{aligned}$$

If $h \circ s \circ r = h$ then this holds in particular for $f = h \circ s$. (2): Dual to (1). □

Roughly, condition (1) means that h respects the congruence on C induced by the idempotent $e = s \circ r$, and that f is the restriction of h to A via s. Dually, condition (2) means that the image of k is contained in image of e and that g is the corestriction of k to B via r.

Corollary 5.9

1. *For any algebra $a : \mathbf{F}A \to A$ there is at most one homomorphism $f : \alpha \to a$.*
2. *For any coalgebra $b : B \to \mathbf{F}B$ there is at most one homomorphism $g : b \to \beta$.*

Only for the last result of this section, we assume that our base category $\mathbf{X}$ is the category **SDom** of domains and strict continuous maps.

Proposition 5.10 *Let $\mathbf{F} : \mathbf{SDom} \to \mathbf{SDom}$ be locally continuous.*

1. *If there is a homomorphism $f : \alpha \to a$ for a given algebra $a : \mathbf{F}A \to A$ then it is the least $f' : X \to A$ such that $f' = a \circ \mathbf{F}f' \circ \beta$.*
2. *If there is a homomorphism $g : b \to \beta$ for a given coalgebra $b : B \to \mathbf{F}B$ then it is the least $g' : B \to X$ such that $g' = \alpha \circ \mathbf{F}g' \circ b$.*

Proof (1): The least solution of the above equation is $f' = \bigsqcup_n f_n$, where the sequence f_n is inductively defined by

$$\begin{aligned}
f_0 &= \bot \\
f_{n+1} &= a \circ \mathbf{F}f_n \circ \beta.
\end{aligned}$$

Define $\mathrm{id}_n : X \to X$ by

$$\begin{aligned}
\mathrm{id}_0 &= \bot \\
\mathrm{id}_{n+1} &= \alpha \circ \mathbf{F}\mathrm{id}_n \circ \beta.
\end{aligned}$$

By local continuity of $\mathbf{F}$,

$$\begin{aligned}
\bigsqcup_n \mathrm{id}_n &= \bigsqcup_n \mathrm{id}_{n+1} \\
&= \bigsqcup_n (\alpha \circ \mathbf{F}\mathrm{id}_n \circ \beta) \\
&= \alpha \circ \mathbf{F}\left(\bigsqcup_n \mathrm{id}_n\right) \circ \beta.
\end{aligned}$$

Hence $\bigsqcup_n \mathrm{id}_n = \mathrm{id}_X$ by inductivity. Since f is strict, we have that $f_0 = f \circ \mathrm{id}_0$. Assuming that $f_n = f \circ \mathrm{id}_n$ we deduce that

$$\begin{aligned} f_{n+1} &= a \circ \mathbf{F} f_n \circ \beta \\ &= a \circ \mathbf{F} f \circ \mathbf{F}\mathrm{id}_n \circ \beta \\ &= f \circ \alpha \circ \mathbf{F}\mathrm{id}_n \circ \beta \\ &= f \circ \mathrm{id}_{n+1}. \end{aligned}$$

Hence $f_n = f \circ \mathrm{id}_n$ for every n. Therefore

$$\begin{aligned} f' &= \bigsqcup f_n \\ &= \bigsqcup_n (f \circ \mathrm{id}_n) \\ &= f \circ \bigsqcup_n \mathrm{id}_n \\ &= f \circ \mathrm{id}_X \\ &= f. \end{aligned}$$

(2): Dual to (1). □

Thus, in order to to find a recursive definition of a function $f : X \to A$ we can try to find an algebra a such that $f : \alpha \to a$ is a homomorphism, and in order to find a recursive definition of a function $g : B \to X$ we can try to find a coalgebra b such that $g : b \to \beta$ is a homomorphism. If we succeed in finding such algebra a and coalgebra b, then we obtain a definition of f by ***structural recursion*** and a definition of g by ***structural corecursion***.

5.3 Biquotients of bifree algebras

This section also contains results developed in collaboration with Thomas Streicher.

We have seen that any $\mathbf{F}$-inductive retraction

$$\mathbf{X} \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{FX}$$

appears as a retract of the bifree $\mathbf{F}$-algebra $i : \mathbf{F}C \to C$ via $r : i \to \alpha$ and $s : \beta \to i^{-1}$ with $r \circ s = \mathrm{id}_X$. We now characterize for a bifree algebra $i : \mathbf{F}C \to C$ the idempotents $e : C \to C$ which admit a splitting $e = s \circ r$ of the kind just described.

Definition 5.11 Let $e : C \to C$ be an idempotent and define $C \underset{b}{\overset{a}{\leftrightarrows}} \mathbf{F}C$ by

$$a = e \circ i \qquad \text{and} \qquad b = i^{-1} \circ e.$$

We say that e is a ***biquotient*** of the bifree algebra $i : \mathbf{F}C \to C$ if the following conditions hold:

(i) $e : i \to a$

(ii) $e : b \to i^{-1}$

(iii) $h = a \circ \mathbf{F}h \circ b$ iff $h = e$. $\square$

Theorem 5.12

1. *If* $X \underset{\beta}{\overset{\alpha}{\leftrightarrows}} \mathbf{F}X$ *is an* $\mathbf{F}$*-inductive retraction,* $r : i \to \alpha$ *and* $s : \beta \to i^{-1}$ *then* $e \stackrel{\text{def}}{=} s \circ r$ *is a biquotient of* i*. Moreover,* α *and* β *can be recovered from* r *and* s *as*
$$\alpha = r \circ i \circ \mathbf{F}s \qquad \beta = \mathbf{F}r \circ i^{-1} \circ s.$$

2. *If* $e : C \to C$ *is a biquotient of* i *and* $e = s \circ r$ *with* $r \circ s = id_X$ *then the maps*
$$\begin{array}{rcl} \alpha & \stackrel{\text{def}}{=} & r \circ i \circ \mathbf{F}s : \mathbf{F}X \to X \\ \beta & \stackrel{\text{def}}{=} & \mathbf{F}r \circ i^{-1} \circ s : X \to \mathbf{F}X \end{array}$$
constitute an $\mathbf{F}$*-inductive retraction. Moreover, we have* $r : i \to \alpha$ *and* $s : \beta \to i^{-1}$.

Proof (1): Conditions (i) and (ii) hold by the following equational reasoning:

$$\begin{array}{rcl} e \circ i \circ \mathbf{F}e & = & s \circ r \circ i \circ \mathbf{F}s \circ \mathbf{F}r \\ & = & s \circ \alpha \circ \mathbf{F}r \circ \mathbf{F}s \circ \mathbf{F}r \\ & = & s \circ \alpha \circ \mathbf{F}r \\ & = & s \circ r \circ i \\ & = & e \circ i, \\ \mathbf{F}e \circ i^{-1} \circ e & = & \mathbf{F}s \circ \mathbf{F}r \circ i^{-1} \circ s \circ r \\ & = & \mathbf{F}s \circ \mathbf{F}r \circ \mathbf{F}s \circ \beta \circ r \\ & = & \mathbf{F}s \circ \beta \circ r \\ & = & i^{-1} \circ s \circ r \\ & = & i^{-1} \circ e. \end{array}$$

From this we get immediately that
$$e \circ i \circ \mathbf{F}e \circ i^{-1} \circ e = e \circ i \circ i^{-1} \circ e = e \circ e = e.$$

For the other implication of condition (iii), let $h : C \to C$ with $e \circ i \circ \mathbf{F}h \circ i^{-1} \circ e = h$ It follows that $r \circ i \circ \mathbf{F}h \circ i^{-1} \circ s = r \circ h \circ s$ and so we get

$$\begin{array}{rcl} r \circ h \circ s & = & r \circ i \circ \mathbf{F}h \circ i^{-1} \circ s \\ & = & \alpha \circ \mathbf{F}r \circ \mathbf{F}h \circ \mathbf{F}s \circ \beta \\ & = & \alpha \circ \mathbf{F}(r \circ h \circ s) \circ \beta, \end{array}$$

which entails $r \circ h \circ s = \mathrm{id}_X$ as α and β form an $\mathbf{F}$-inductive retract. Thus we get
$$h = e \circ h \circ e = s \circ r \circ h \circ s \circ r = s \circ r = e.$$

The proposed reconstruction of α and β from r and s can be seen as follows:

$$\begin{aligned} r \circ i \circ \mathbf{F}s &= \alpha \circ \mathbf{F}r \circ \mathbf{F}s = \alpha, \\ \mathbf{F}r \circ i^{-1} \circ s &= \mathbf{F}r \circ \mathbf{F}s \circ \beta = \beta. \end{aligned}$$

(2): We have that

$$\begin{array}{ll} \text{(a)} & r \circ i = r \circ i \circ \mathbf{F}(s \circ r), \\ \text{(b)} & i^{-1} \circ s = \mathbf{F}(s \circ r) \circ i^{-1} \circ s, \\ \text{(c)} & s \circ r \circ i \circ \mathbf{F}h \circ i^{-1} \circ s \circ r = h \text{ iff } h = e, \end{array}$$

and hence that

$$\begin{aligned} \alpha \circ \beta &= r \circ i \circ \mathbf{F}s \circ \mathbf{F}r \circ i^{-1} \circ s \\ &= r \circ i \circ \mathbf{F}(s \circ r) \circ i^{-1} \circ s \\ &= r \circ i \circ i^{-1} \circ s \\ &= r \circ s \\ &= \mathrm{id}_X. \end{aligned}$$

Let $f : X \to X$ with $f = \alpha \circ \mathbf{F}f \circ \beta$. As

$$\alpha \circ \mathbf{F}f \circ \beta = r \circ i \circ \mathbf{F}(s \circ f \circ r) \circ i^{-1} \circ s,$$

for $h \stackrel{\text{def}}{=} s \circ f \circ r$ we get

$$h = s \circ r \circ i \circ \mathbf{F}h \circ i^{-1} \circ s \circ r = e \circ i \circ \mathbf{F}h \circ i^{-1} \circ e,$$

from which we get by (c) that $h = e$. But then

$$f = r \circ s \circ f \circ s \circ r = r \circ h \circ s = r \circ e \circ s = \mathrm{id}_X$$

as desired. It remains to show that $r : i \to \alpha$ and $s : \beta \to i^{-1}$:

$$\begin{aligned} \alpha \circ \mathbf{F}r &= r \circ i \circ \mathbf{F}s \circ \mathbf{F}r = r \circ i \text{ by (a)}, \\ \mathbf{F}s \circ \beta &= \mathbf{F}s \circ \mathbf{F}r \circ i^{-1} \circ s = i^{-1} \circ s \text{ by (b)}. \; \Box \end{aligned}$$

Chapter 6

Equivalence of effectively given domains

Kanda and Park [KP79] have shown that, in general, it is possible to present an effectively given algebraic domain in essentially different ways. This fact led them to introduce and investigate a notion of equivalence of effectively given algebraic domains. This notion does not generalize to effectively given domains in the sense of Smyth [Smy77], and it is rather involved as it considers directly the enumerations of basis elements.

In this chapter we introduce a more abstract and simpler definition of equivalence, which also works for continuous domains. In some sense, our notion is not new at all, because it is simply an instance of the notion of equivalence of objects of concrete categories [AHS90]. But we claim that the idea of applying this notion to effectively given domains is original.

The results of this chapter are applied in Chapter 12 of Part IV to show that there is an essentially unique effective presentation of the real numbers type hierarchy which makes the Real PCF primitive operations formally computable. In particular, Theorem 6.12 together with the results of Chapter 10 of Part III directly entail the result.

6.1 A non-standard effective presentation of $\mathcal{B}^\omega$

An effective presentation of $\mathcal{B}^\omega$ in the sense of Definition 3.16 is given by $(\mathrm{id}_{\mathcal{B}^\omega}, \mathrm{id}_{\mathcal{B}^\omega})$. Let $f : \omega \to \omega$ be a non-recursive permutation of ω, and define $\phi : \mathcal{B}^\omega \to \mathcal{B}^\omega$ by $\phi(p) = p \circ f$. Then

(ϕ, ϕ^{-1}) *is an effective presentation of* $\mathcal{B}^\omega$,

because $\phi \circ \phi^{-1} = \mathrm{id}_{\mathcal{B}^\omega}$. But this effective presentation is intuitively not "really" effective, by construction.

One might suspect that this may have something to do with the definition of the notion of effective presentation via section-retraction pairs given in Definition 3.16. But the same phenomenon takes place for the usual notion of effectively given algebraic domain. If we define $b'_n = b_{f(n)}$, where b is the standard algebraic effective presentation of $\mathcal{B}^\omega$, it is easy to check that the axioms for effectively given algebraic domains given in Definition 3.14 are satisfied by b'.

One might conclude that the solution to this paradox would lie in the fact that $(\mathcal{B}^\omega, \mathrm{id}, \mathrm{id})$ and $(\mathcal{B}^\omega, \phi, \phi^{-1})$ are *not* isomorphic objects of **ECDom**. An examination of this possibility seems to make the paradox even worse:

$\phi : (\mathcal{B}^\omega, \mathrm{id}, \mathrm{id}) \to (\mathcal{B}^\omega, \phi, \phi^{-1})$ *is an isomorphism in* **ECDom**

Proof By definition, we have that a function $f : \mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable with respect to effective presentations (s_1, r_1) and (s_2, r_2) respectively iff $s_1 \circ f \circ r_2 : \mathcal{B}^\omega \to \mathcal{B}^\omega$ is computable with respect to the standard effective presentation of $\mathcal{B}^\omega$. If we take $(s_1, r_1) = (\mathrm{id}_{\mathcal{B}^\omega}, \mathrm{id}_{\mathcal{B}^\omega})$, $(s_2, r_2) = (\phi, \phi^{-1})$ and $f = \phi$ then $s_1 \circ f \circ r_2 = \mathrm{id} \circ \phi \circ \phi^{-1} = \mathrm{id}$ which is certainly computable with respect to the standard effective presentation. If we take $(s_1, r_1) = (\phi, \phi^{-1})$, $(s_2, r_2) = (\mathrm{id}_{\mathcal{B}^\omega}, \mathrm{id}_{\mathcal{B}^\omega})$ and $f = \phi^{-1}$ then $s_1 \circ f \circ r_2 = \phi \circ \phi^{-1} \circ \mathrm{id} = \mathrm{id}$. Therefore ϕ and its inverse are computable! $\square$

Again, in the category of effectively given algebraic domains and computable maps, $(\mathcal{B}^\omega, b)$ and $(\mathcal{B}^\omega, b')$ are isomorphic objects, with ϕ as an isomorphism. We omit the argument, which is similar.

On the other hand,

> *The identity of $\mathcal{B}^\omega$ is not computable as a morphism $(\mathcal{B}^\omega, \mathrm{id}, \mathrm{id}) \to (\mathcal{B}^\omega, \phi, \phi^{-1})$ or as a morphism $(\mathcal{B}^\omega, \phi, \phi^{-1}) \to (\mathcal{B}^\omega, \mathrm{id}, \mathrm{id})$.*

This *is* reasonable and solves the paradox, because it shows that we cannot access within $\mathcal{B}^\omega$ "correctly presented" the computable elements of $\mathcal{B}^\omega$ "incorrectly presented", which are not computable in $\mathcal{B}^\omega$ "correctly presented".

We can summarize the above remarks by saying that effective domain theory does not give an absolute notion of effectivity. In the following sections we investigate the concept introduced in the definition below, and we show that we can achieve absoluteness by imposing additional structure on effectively given domains.

Definition 6.1 We say that two effective presentations (s_1, r_1) and (s_2, r_2) of a coherently complete domain D ***induce the same notion of computability*** on D if the following conditions hold:

1. For every effectively given domain (C, s, r), a function $f : C \to D$ is computable w.r.t. (s, r) and (s_1, r_1) iff it is computable w.r.t. (s, r) and (s_2, r_2).
2. For every effectively given domain (E, s, r), a function $f : D \to E$ is computable w.r.t. (s_1, r_1) and (s, r) iff it is computable w.r.t. (s_2, r_2) and (s, r).

By taking $C = 1$ we see that if (s_1, r_1) and (s_2, r_2) induce the same notion of computability on D, then an element of D is computable w.r.t. (s_1, r_1) iff it is computable w.r.t. (s_2, r_2). $\square$

6.2 Equivalence of effectively given domains

We recall the following definitions from [AHS90], and we make explicit a notion which we call representability:
Definition 6.2

1. A ***concrete category*** over a category $\mathbf{X}$ is a category $\mathbf{A}$ together with a faithful functor $\mathbf{U} : \mathbf{A} \to \mathbf{X}$, called the ***underlying functor***.

2. Let $(\mathbf{A}, \mathbf{U})$ be a concrete category over $\mathbf{X}$. If A and B are objects in $\mathbf{A}$ then a morphism $f : \mathbf{U}A \to \mathbf{U}B$ in $\mathbf{X}$ is (A,B)***-representable*** if there is a (necessarily unique) morphism $\bar{f} : A \to B$ in $\mathbf{A}$ such that $\mathbf{U}(\bar{f}) = f$, called the (A,B)***-representation*** of f.

3. We write $f : A \to B$ to denote the fact that a morphism $f : \mathbf{U}A \to \mathbf{U}B$ is (A,B)-representable, and we notationally identify f with its (A,B)-representation $\bar{f}$.

4. The ***fibre*** of an object X in $\mathbf{X}$ is the preordered class consisting of all objects A in $\mathbf{A}$ with $\mathbf{U}(A) = X$, with order given by:

   $A \leq B$ iff $\mathrm{id}_X : A \to B$, where $\mathrm{id}_X : X \to X$ is the identity.

5. Let A and B in $\mathbf{A}$ be objects in the same fibre. The we define

   $A \equiv B$ iff $A \leq B$ and $B \leq A$,

   and in this case we say that A and B are ***equivalent objects*** of $(\mathbf{A}, \mathbf{U})$. □

Clearly, **ECDom** is concrete over **CDom**, with the underlying functor given by the forgetful functor which forgets effective presentations. Using the language of the above definition, we have that

> $(\mathcal{B}^\omega, \mathrm{id}, \mathrm{id})$ *and* $(\mathcal{B}^\omega, \phi, \phi^{-1})$ *are isomorphic objects of the category* **ECDom**, *but inequivalent objects of the concrete category* $(\mathbf{ECDom}, \mathbf{U})$.

Clearly, if D and E are objects in **ECDom**, a function $f : \mathbf{U}D \to \mathbf{U}E$ in **CDom** is (D,E)-representable iff it is computable w.r.t the effective presentations of D and E.

Definition 6.3 We say that two effective presentations (s_1, r_1) and (s_2, r_2) of a coherently complete domain D are ***equivalent*** if (D, s_1, r_1) and (D, s_2, r_2) are equivalent objects of the concrete category $(\mathbf{ECDom}, \mathbf{U})$. □

It is immediate that (s_1, r_1) and (s_2, r_2) are equivalent iff $s_1 \circ r_2$ and $s_2 \circ r_1$ are computable w.r.t. the standard effective presentation of $\mathcal{B}^\omega$.

Proposition 6.4 *Let* $(\mathbf{A}, \mathbf{U})$ *be a concrete category over* $\mathbf{X}$, *let* X *be an* $\mathbf{X}$-*object, and let* B *and* B' *be* $\mathbf{A}$-*objects in the fibre of* X. *Then the following are equivalent:*

1. $B \leq B'$.

2. *For all A in $\mathbf{A}$ and $f : \mathbf{U}A \to X$ in $\mathbf{X}$, if $f : A \to B$ then $f : A \to B'$.*

3. *For all C in $\mathbf{A}$ and $f : X \to \mathbf{U}C$ in $\mathbf{X}$ if $f : B' \to C$ then $f : B \to C$.*

Proof ($1 \Rightarrow 2$): Assume that $f : \mathbf{U}A \to X$ is (A, B)-representable, let $\bar{f} : A \to B$ be the (A, B)-representation of f, and $\overline{\mathrm{id}}_X : B \to B'$ be the (A, B')-representation of id_X. Then $f' = \overline{\mathrm{id}}_X \circ \bar{f}$ is a (B, B')-representation of f, because $\mathbf{U}(f') = \mathbf{U}(\overline{\mathrm{id}}_X \circ \bar{f}) = \mathbf{U}(\overline{\mathrm{id}}_X) \circ \mathbf{U}(\bar{f}) = \mathrm{id}_X \circ f = f$. ($1 \Rightarrow 3$): Similar. ($2 \Rightarrow 1$): Take $A = B$ and $f = \mathrm{id}_X$. ($3 \Rightarrow 1$): Similar. □

This shows that our notion of equivalence is appropriate:

Corollary 6.5 *Two effective presentations of a coherently complete domain are equivalent iff they induce the same notion of computability on the domain.*

The following definition is also taken from [AHS90]:

Definition 6.6 A concrete category $(\mathbf{A}, \mathbf{U})$ over $\mathbf{X}$ is called ***concretely cartesian closed*** provided the following hold:

1. $\mathbf{A}$ and $\mathbf{X}$ are cartesian closed,

2. $\mathbf{U}$ preserves finite products, exponentials, and evaluation.

□

Clearly, $(\mathbf{ECDom}, \mathbf{U})$ is concretely cartesian closed. The following results are routinely proved, as Proposition 6.4:

Proposition 6.7 *Let $(\mathbf{A}, \mathbf{U})$ be a concretely cartesian closed category over $\mathbf{X}$. If 1 and $1'$ are terminal objects of $\mathbf{A}$ in the same fibre then $1 \leq 1'$, and if $A \leq A'$ and $B \leq B'$ are objects of $\mathbf{A}$ in the same fibre then*

1. $A \times B \leq A' \times B'$,

2. $(A \Rightarrow B') \leq (A' \Rightarrow B)$.

We can thus say that equivalence is a "cartesian closed congruence".

Proposition 6.8 *Let $(\mathbf{A}, \mathbf{U})$ be a concretely cartesian closed category over $\mathbf{X}$, and let A and B and be $\mathbf{A}$-objects. Then*

1. *Any two products of A and B which are in the same fibre are equivalent.*

2. *Any two exponentials of B to the power A which are in the same fibre are equivalent.*

Thus, as soon as effective presentations of D and E are specified, the effective presentations of $D \times E$ and $D \Rightarrow E$ are uniquely specified up to equivalence.

It is clear that any coherently complete finite domain has *some* effective presentation.

Proposition 6.9 *Any two effective presentations of a finite coherently complete domain are equivalent.*

Proof Let D be a finite coherently complete domain and (s_1, r_1) and (s_2, r_2) be effective presentations of D. We have to show that $s_1 \circ r_2$ and $r_2 \circ s_1$ are computable w.r.t. the standard effective presentation of $\mathcal{B}^\omega$. Let $d \in D$, and $i \in \{1, 2\}$. Since $d = r_i(s_i(d))$ is finite and r_i is continuous, there is some finite $p_i^d \sqsubseteq s_i(d)$ in $\mathcal{B}^\omega$ such that already $d = r_i(p_i^d)$. Hence

$$r_i = \bigsqcup_{d \in D} p_i^d \Rrightarrow d \qquad \text{and} \qquad s_i = \bigsqcup_{d \in D} d \Rrightarrow s_i(r_i(p_i^d)),$$

because $s_i(d) = s_i(r_i(p_i^d))$. Hence, if $j \in \{1, 2\} \backslash \{i\}$,

$$s_i \circ r_j(x) = \bigsqcup_{d \sqsubseteq r_j(x)} s_i(r_i(p_i^d)) = \bigsqcup_{d \sqsubseteq \bigsqcup_{p_j^e \sqsubseteq x} e} s_i(r_i(p_i^d)).$$

But $s_i(r_i(p_i^d))$ is computable because p_i^d is finite and $s_i \circ r_i$ is computable by definition of effective presentation. Therefore $s_i \circ r_j$ is computable, because there are finitely many d's and e's. $\square$

Thus, we don't need to specify the effective presentation of a finite domain.

6.3 Characterizations of some effective presentations

How can we isolate the "correct" effective presentation of a domain D among all possible effective presentations? As we have seen, there is no absolute notion of computability for domain theory, and from the categorical point of view this is not a worry. However, in practice we need specific effective presentations. One way to achieve absoluteness is:

1. find some basic operations on D which are concretely computable (in our case these will be the Real PCF definable primitive operations),

2. find an effective presentation which makes the operations formally computable, and

3. show that any two effective presentations of D which make the operations formally computable are equivalent.

We first illustrate this idea in the case of $D = \mathcal{B}^\omega$. The operations defined in the following proposition are concretely computable, and formally computable w.r.t. the standard presentation of $\mathcal{B}^\omega$, as it is shown in [Plo78]. This establishes (1) and (2). The proposition itself establishes (3):

Proposition 6.10 *Define continuous maps*

$$\text{cons} : \mathcal{B} \times \mathcal{B}^\omega \to \mathcal{B}^\omega, \quad \text{head} : \mathcal{B}^\omega \to \mathcal{B}, \quad \text{tail} : \mathcal{B}^\omega \to \mathcal{B}^\omega$$

by

$$\begin{array}{rclcrcl} \text{cons}(t,p)(0) & = & t, & \qquad & \text{head}(p) & = & p(0), \\ \text{cons}(t,p)(n+1) & = & p(n), & & \text{tail}(p)(n) & = & p(n+1). \end{array}$$

Then any two effective presentations of $\mathcal{B}^\omega$ which make cons, head, *and* tail *computable are equivalent.*

Here we assume any effective presentation of $\mathcal{B}$, and we notationally identify the domain $\mathcal{B}$ with the domain $\mathcal{B}$ endowed with the effective presentation.

Proof Let D and D' in **ECDom** be objects in the fibre of $\mathcal{B}^\omega$ such that

$$\text{cons} : \mathcal{B} \times D \to D, \quad \text{head} : D \to \mathcal{B}, \quad \text{tail} : D \to D$$

and

$$\text{cons} : \mathcal{B} \times D' \to D', \quad \text{head} : D' \to \mathcal{B}, \quad \text{tail} : D' \to D'$$

The identity of $\mathcal{B}^\omega$ is the least fixed-point of $F : (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega) \to (\mathcal{B}^\omega \Rightarrow \mathcal{B}^\omega)$ defined by

$$F(f)(p) \quad = \quad \text{cons}(\text{head}(p), f(\text{tail}(p))).$$

Since head : $D \to \mathcal{B}$, tail : $D \to D$ and cons : $\mathcal{B} \times D' \to D'$, we have that $F : (D \Rightarrow D') \to (D \Rightarrow D')$. And since head : $D' \to \mathcal{B}$, tail : $D' \to D'$ and cons : $\mathcal{B} \times D \to D$, we have that $F : (D' \Rightarrow D) \to (D' \Rightarrow D)$. Hence $\text{id}_{\mathcal{B}^\omega} : D \to D'$ and $\text{id}_{\mathcal{B}^\omega} : D' \to D$. But $\text{id}_{\mathcal{B}^\omega} = \text{fix } F$ is computable because F is computable and the fixed-point combinator is always computable. Therefore D and D' are equivalent. □

That is, there is a unique notion of computability for $\mathcal{B}^\omega$ such that cons, head, and tail are computable.

Since cons : $\mathcal{B} \times \mathcal{B}^\omega \to \mathcal{B}^\omega$ is a bifree algebra, we see that Proposition 6.10 is a special case of Theorem 6.11 below. We already know that a locally computable functor has a computable bifree algebra.

Theorem 6.11 *Let* $\mathbf{F} : \mathbf{SDom} \to \mathbf{SDom}$ *be a locally continuous functor which restricts to a locally computable functor* $\mathbf{G} : \mathbf{ESDom} \to \mathbf{ESDom}$*, and let* $i : \mathbf{F}C \to C$ *be a bifree algebra. Then there is at most one effective presentation of* C *which makes* i *and* i^{-1} *computable, up to equivalence.*

Proof As in Proposition 6.10, observing that the identity of C is the unique algebra homomorphism from i to itself, which is the least fixed-point of the functional F defined by $F(f) = i \circ \mathbf{G}f \circ i^{-1}$. □

This theorem further generalizes to inductive retractions with the same proof, by virtue of Lemma 5.6:

Theorem 6.12 *Let* $\mathbf{F} : \mathbf{SDom} \to \mathbf{SDom}$ *be a locally continuous functor which restricts to a locally computable functor* $\mathbf{G} : \mathbf{ESDom} \to \mathbf{ESDom}$*, and let* $\langle \alpha : \mathbf{F}C \to C, \beta : C \to \mathbf{F}C \rangle$ *be an* $\mathbf{F}$*-inductive section-retraction pair. Then there is at most one effective presentation of* C *which makes* α *and* β *computable, up to equivalence.*

We apply this theorem in Section 12.1 in order to show that there an essentially unique effective presentation of the real numbers type hierarchy which makes the Real PCF definable elements computable.

Chapter 7

Coherently complete domains

In our investigation of computational completeness in Chapter 12 of Part IV, we are led to consider bounded complete domains D with the following property: for every element d of D (or perhaps of some basis of D) there is a *continuous* map $\mathrm{join}_d : D \to D$ such that

$$\mathrm{join}_d(x) = d \sqcup x \text{ for every } x \text{ consistent with } d.$$

Such a map join_d is called a ***joining map*** and such a domain D is called a ***J-domain***. In the above definition, if x is inconsistent with d, then there is a certain degree of freedom in what $\mathrm{join}_d(x)$ can be. Therefore, in general, an element can have more than one joining map, if it has a joining map at all.

In this chapter we characterize the J-domains exactly as the coherently complete domains. Moreover, we show that every element of a coherently complete domain has a least joining map. We also investigate joining maps of function spaces with respect to the joining maps of their source and target.

The results of this chapter are applied in Chapter 12 to show that Real PCF (without the existential quantifier) is first-order computationally complete.

7.1 Joining maps

Definition 7.1 A ***joining map*** of an element d of a bounded complete domain D is a continuous function $\mathrm{join}_d : D \to D$ such that

$$\mathrm{join}_d(x) = d \sqcup x \text{ for every } x \uparrow d.$$

A bounded complete domain D ***has internal joins*** if each $d \in D$ has a joining map join_d. □

Proposition 7.2 *Flat domains have internal joins.*

Proof For bottom, the unique joining map is the identity, and for a non-bottom element x, the unique joining map is the constant map x. □

Proposition 7.3 *If a bounded complete domain D has internal joins, so does D^ω.*

Proof Let $d \in D^\omega$, and define $\mathrm{join}_d : D^\omega \to D^\omega$ by $\mathrm{join}_d(x) = \left\langle \mathrm{join}_{d_i}(x_i) \right\rangle_{i \in \omega}$. Then join_d is clearly a joining map for d, because $d \uparrow x$ iff $d_i \uparrow x_i$ for every $i \in \omega$, and joins are computed componentwise. □

Proposition 7.4 *Domains with internal joins are closed under the formation of retracts.*

Proof Let D be a retract of a domain E with internal joins, with respect to a section $s : D \to E$ and a retraction $r : E \to D$, and for each $d \in D$ define $\mathrm{join}_d : D \to D$ by $\mathrm{join}_d = r \circ \mathrm{join}_{s(d)} \circ s$, where $\mathrm{join}_{s(d)}$ is a joining map for $s(d) \in E$. Let x be consistent with d. Then $s(x) \uparrow s(d)$. Hence $\mathrm{join}_d(x) = r(s(d) \sqcup s(x)) = d \sqcup x$, by a general property of section-retraction pairs. Therefore join_d is a joining map for d. $\square$

Recall that a domain is *coherently complete* if every pairwise consistent subset has a least upper bound, and recall that the coherently complete, countably based domains are characterized as the retracts of $\mathcal{B}^\omega$.

Theorem 7.5 *A bounded complete countably based domain has internal joins iff it is coherently complete.*

Proof ($\Rightarrow$): Let $\{a, b, c\}$ be a pairwise consistent subset of D, and join_a be a joining map for a. Since join_a is monotone, it preserves consistency. Thus $a \sqcup b \uparrow a \sqcup c$, because $b \uparrow c$. Hence $\{a, b, c\}$ is consistent, and by bounded completeness it has a least upper bound. By induction, every finite pairwise consistent subset has a least upper bound. Therefore, for every pairwise consistent set there is a cofinal directed set, obtained by adding the joins of finite subsets.

($\Leftarrow$) : Since $\mathcal{B}$ is a flat domain it has internal joins and hence so do $\mathcal{B}^\omega$ and every retract of $\mathcal{B}^\omega$. $\square$

We shall see that the countability hypothesis is not really necessary in the above theorem.

We shall need domains which have joining maps only for elements of a countable basis.

Definition 7.6 A ***J-domain*** is a bounded complete domain with a countable basis B such that each $b \in B$ has a joining map. $\square$

But these turn out to be the coherently complete countably based domains again. Recall that $\downarrow\uparrow d$ is the set of elements consistent with d.

Lemma 7.7 *Let D be a bounded complete domain, x and y be elements of D, and let $Z \subseteq D$ be directed. Then the following statements hold:*

1. *If $a \uparrow b$ for all $a \ll x$ and $b \ll y$ then $x \uparrow y$.*

2. *The intersection of the closed neighbourhoods of x is $\downarrow\uparrow x$.*

3. *If $x \uparrow z$ for every $z \in Z$ then $x \uparrow \bigsqcup^{\uparrow} Z$.*

Proof (1): Assume that every $a \ll x$ is consistent with every $b \ll y$, and let $Z = \{a \sqcup b | a \ll x, b \ll y\}$. Then Z is non-empty because it contains $\perp$. Let a_1 and a_2 be way-below x, and b_1 and b_2 be way-below y. Then there is some $u \ll x$ above a_1 and a_2, and there is some $v \ll y$ above b_1 and b_2. Hence $u \sqcup v$ is a member of Z above the members $a_1 \sqcup a_2$ and $b_1 \sqcup b_2$. This shows that Z is directed. Therefore x and y are consistent, because $\bigsqcup Z$ is above them.

(2): Any closed neighbourhood of x contains $\downarrow\uparrow x$, because open sets are upper sets and closed sets are lower sets. If $y \notin \downarrow\uparrow x$ then y is apart from x, and there are disjoint open neighbourhoods U and V of x and y respectively, by (1). In fact, the contrapositive of (1) states that if x and y lie apart then there are $a \ll x$ and $b \ll y$ such that a and b already lie apart; thus we can take $U = \Uparrow a$ and $V = \Uparrow b$. Therefore y is not in the closure of U.

(3): This follows from (2), because $Z \subseteq \downarrow\uparrow x$ and closed sets contain the least upper bounds of their directed subsets. □

Proposition 7.8 *Any J-domain is coherently complete.*

Proof Let B be a countable basis of a coherently complete domain D such that every element b of B has a joining map join_b, let $\{a, b, c\}$ be a pairwise consistent subset of D, and let $a' \ll a$ in B. Since $\mathrm{join}_{a'}$ is monotone, it preserves consistency. Thus $a' \sqcup b \uparrow a' \sqcup c$, because $b \uparrow c$. Hence $a' \sqcup b \uparrow a' \sqcup c$ for every $a' \ll a$ in B. Since $\bigsqcup_{a' \in \Downarrow a \cap B} a' \sqcup x = a \sqcup x$ for any $x \uparrow a$, by two applications of Lemma 7.7 we conclude that $a \sqcup b \uparrow a \sqcup c$. Therefore $\{a, b, c\}$ is consistent, and we can finish the proof as in Theorem 7.5. □

Corollary 7.9 *A countably based bounded complete domain is a J-domain iff it is coherently complete.*

7.2 Canonical joining maps

In general, an element of a coherently complete domain can have more than one joining map. For example, let D be the flat domain of truth values with a new element $\mathbf{tt}'$ above $\mathbf{tt}$. Then any joining map of $\mathbf{tt}$ has to map $\perp$ and $\mathbf{tt}$ to $\mathbf{tt}$, and $\mathbf{tt}'$ to $\mathbf{tt}'$, but we can choose to map $\mathbf{ff}$ to either $\mathbf{tt}$ or else $\mathbf{tt}'$. However, Theorem 7.13 below shows that there is a canonical way of choosing joining maps. Moreover, it shows how to construct joining maps directly instead of indirectly via the joining maps of the coherently complete domain $\mathcal{B}^\omega$.

Proposition 7.10 *If* join_d *is a joining map for an element* d *of a bounded complete domain* D*, then* join_d *is an idempotent with image* $\uparrow d$.

Proof $\mathrm{join}_d(\perp) = d$. By monotonicity, $\mathrm{join}_d(x) \in \uparrow d$. If $x \in \uparrow d$ then $x \uparrow d$ and $\mathrm{join}_d(x) = d \sqcup x = x$. □

Recall that a ***kernel operator*** on a dcpo D is a continuous idempotent below the identity of D [AJ94, GHK[+]80], and that a kernel operator is uniquely determined by its image.

Proposition 7.11 *If* join_d *is a joining map of an element* d *of a bounded complete domain* D*, then the map* $\mathrm{consist}_d : D \to D$ *defined by*

$$\mathrm{consist}_d(x) = \mathrm{join}_d(x) \sqcap x$$

is a kernel operator with image $\downarrow\uparrow d$.

Proof The binary meet operation is well-defined and continuous in any bounded complete domain. Hence so is $\mathrm{consist}_d$. For any x, $\mathrm{consist}_d(x)$ is consistent with d, because $\mathrm{join}_d(x)$ is clearly above $\mathrm{consist}_d(x)$, and it is above d by Proposition 7.10. Conversely, if x is consistent with d then $\mathrm{consist}_d(x) = (d \sqcup x) \sqcap x = x$. Therefore $\mathrm{consist}_d$ is a continuous idempotent with image $\downarrow\uparrow d$. $\square$

Proposition 7.12 *For any coherently complete domain D and any $d \in D$ there is a kernel operator* $\mathrm{consist}_d : D \to D$ *with image $\downarrow\uparrow d$, given by*

$$\mathrm{consist}_d(x) = \max\left(\downarrow x \cap \downarrow\uparrow d\right).$$

That is, $\mathrm{consist}_d(x)$ is the greatest element below x which is consistent with d.

Proof For each $x \in D$ define $Y_x = \downarrow x \cap \downarrow\uparrow d$. The set Y_x is non-empty, because it has bottom as a member. Let $a, b \in Y_x$. This means that a and b are bounded by x and consistent with d. By coherence, $\{a, b, d\}$ has a join. Hence $a \sqcup b$ is consistent with d. Therefore Y_x is directed. But it is also closed, because $\downarrow x$ and $\downarrow\uparrow x$ are closed by Lemma 7.7. Therefore it contains its least upper bound, which has to be its greatest member. In order to show that $\mathrm{consist}_d$ is continuous, we use the ϵ-δ characterization of continuity. Let $y \ll \bigsqcup Y_x$. We have to show that there is some $x' \ll x$ such that already $y \ll \bigsqcup Y_{x'}$. By a basic property of the way-below order on continuous dcpos, $y \ll a$ for some $a \in Y_x$. By interpolation, $y \ll a' \ll a$ for some a', which belongs to Y_x as Y_x is closed. By transitivity, $a' \ll x$, and again by interpolation, we can find x' such that $a' \ll x' \ll x$. But $a' \in Y_{x'}$. Hence $y \ll \bigsqcup Y_{x'}$, because $y \ll a'$. Therefore $\mathrm{consist}_d$ is a kernel operator with image $\downarrow\uparrow d$. $\square$

Theorem 7.13 *Every element d of a coherently complete domain D has a least joining map* join_d*, given by*

$$\mathrm{join}_d(x) = d \sqcup \mathrm{consist}_d(x).$$

Proof The map is well-defined because $\mathrm{consist}_d(x)$ is consistent with d by construction, and it is clearly continuous. Therefore join_d is a joining map of d, because $x = \mathrm{consist}_d(x)$ iff x is consistent with d, again by construction. It remains to show that it is below any other joining map join'_d of d. By Proposition 7.11, we know that $\mathrm{consist}_d(x) = \mathrm{join}'_d(x) \sqcap x$, because kernel operators are uniquely determined by their image. Hence $\mathrm{join}_d(x) = d \sqcup (\mathrm{join}'_d(x) \sqcap x)$. We have that d is below $\mathrm{join}'_d(x)$ by Proposition 7.10 , and we clearly have that $\mathrm{join}'_d(x) \sqcap x$ is below $\mathrm{join}'_d(x)$. Hence the join of d and $\mathrm{join}'_d(x) \sqcap x$ is itself below $\mathrm{join}'d(x)$. Therefore $\mathrm{join}_d(x) \sqsubseteq \mathrm{join}'_d(x)$. $\square$

We have already shown that a bounded complete, countably based domain has internal joins iff it is coherently complete. The above theorem implies that this is true also when the countability hypothesis is omitted:

Corollary 7.14 *A bounded complete domain has internal joins iff it is coherently complete.*

Also, notice that the proof of this more general fact does not use the universality of $\mathcal{B}^\omega$.

7.3 Universality of $\mathcal{B}^\omega$ via internal joins

We give a new proof of the fact that $\mathcal{B}^\omega$ is a universal coherently complete countably based domain (in the sense that is has every coherently complete countably based domain as a retract). The proof is based on internal joins. The advantage of the new proof is that it can be effectivized in order to establish first-order computational completeness of Real PCF and full computational completeness of extensions of PCF with ground types interpreted as algebraic domains.

Theorem 7.15 *A bounded complete countably based domain D is coherently complete iff it is a retract of $\mathcal{B}^\omega$.*

Proof Only the left-to-right implication is non-trivial. Let B be a countable basis of D and let $\{b_n | n \in \omega\}$ be an enumeration of B. The function $s : D \to \mathcal{B}^\omega$ defined by

$$s(x)(n) \quad = \quad \begin{cases} \mathbf{tt} & \text{if } b_n \ll x, \\ \mathbf{ff} & \text{if } b_n \# x, \\ \perp & \text{otherwise.} \end{cases}$$

is easily seen to be continuous. If $p = s(x)$ then $p^{-1}(\mathbf{tt})$ is an enumeration of $\twoheaddownarrow x \cap B$. We thus define $j : \mathbb{N} \to (D \Rightarrow D)$, $f : \mathcal{B}^\omega \to (\mathbb{N} \Rightarrow D)$, and $r : \mathcal{B}^\omega \to D$ by

$$\begin{aligned} j(n) &= \text{join}_{b_n} \\ f(p)(n) &= \text{pif } p(n) \text{ then } j(n)(f(p)(n+1)) \text{ else } f(p)(n+1), \\ r(p) &= f(p)(0). \end{aligned}$$

Here f is assumed to be the least fixed point of an implicitly defined functional, and pif is the parallel conditional defined in Section 3.4. The idea is that f recursively joins the elements enumerated by $p^{-1}(\mathbf{tt})$. The function j is clearly continuous ($\mathbb{N}$ with discrete topology), and f and r are continuous because they are defined by composition of continuous functions. We clearly have that $f = \bigsqcup_{k \in \omega} f^{(k)}$, for $f^{(k)}$ inductively defined by

$$\begin{aligned} f^{(0)}(p)(n) &= \perp, \\ f^{(k+1)}(p)(n) &= \text{pif } p(n) \text{ then } j(n)\left(f^{(k)}(p)(n+1)\right) \text{ else } f^{(k)}(p)(n+1). \end{aligned}$$

We prove by induction on k that for all $x \in D$ and $k, n \in \omega$,

$$f^{(k)}(s(x))(n) \quad = \quad \bigsqcup_{\substack{n \le i < k+n \\ b_i \ll x}} b_i.$$

This implies that

$$r(s(x)) \quad = \quad f(s(x))(0) = \bigsqcup_{k \in \omega} f^{(k)}(s(x))(0) = \bigsqcup_{k \in \omega} \bigsqcup_{\substack{i < k \\ b_i \ll x}} b_i = \bigsqcup_{\substack{k \in \omega \\ b_k \ll x}} b_k = x,$$

establishing that D is a retract of $\mathcal{B}^\omega$. For $k = 0$ the claim is immediate. For the inductive step, define

$$\begin{aligned} z &= \bigsqcup_{\substack{n+1\leq i<k+n+1 \\ b_i \ll x}} b_i, \\ y &= j_n(z). \end{aligned}$$

By the induction hypothesis,

$$f^{(k+1)}(s(x))(n) \quad = \quad \text{pif } s(x)(n) \text{ then } y \text{ else } z.$$

We consider the following three cases for x:

Case (1): $b_n \ll x$. Then $s(x)(n) = \mathbf{tt}$, and $b_n \uparrow z$, because $z \sqsubseteq x$. Thus

$$f^{(k+1)}(s(x))(n) \quad = \quad y = b_n \sqcup z = \bigsqcup_{\substack{n\leq i<(k+1)+n \\ b_i \ll x}} b_i.$$

Case (2): $b_n \# x$. Then $s(x)(n) = \mathbf{ff}$ and $b_n \not\ll x$. Thus

$$f^{(k+1)}(s(x))(n) \quad = \quad z = \bigsqcup_{\substack{n\leq i<(k+1)+n \\ b_i \ll x}} b_i.$$

Case (3) Otherwise. Then $s(x)(n) = \bot$, $b_n \uparrow x$, $b_n \not\ll x$, and $b_n \uparrow z$, because $z \sqsubseteq x$. Thus

$$f^{(k+1)}(s(x))(n) \quad = \quad y \sqcap z = (b_n \sqcup z) \sqcap z = z = \bigsqcup_{\substack{n\leq i<(k+1)+n \\ b_i \ll x}} b_i. \square$$

7.4 Function spaces of J-domains

We already know that the category of coherently complete countably based domains is cartesian closed. The point of the following theorem is to show how to construct joining maps for a basis of a function space $D \Rightarrow E$ from the joining maps for bases of D and E. Of course, we cannot define $\mathrm{join}_f^{D\Rightarrow E}(g)(x) = \mathrm{join}_{f(x)}^E(g(x))$, because the map $e \mapsto \mathrm{join}_e^E$ is not continuous; in fact, it is easy to see that it is not even monotone.

Lemma 7.16 *A bounded complete domain is a J-domain iff it has a countable* subbasis *S such that each $s \in S$ has a joining map.*

Proof If $A = \{s_1, \ldots, s_n\}$ is a bounded subset of S and $s_1, \ldots, s_n$ have joining maps $\mathrm{join}_{s_1}, \ldots, \mathrm{join}_{s_n}$ respectively, then the element $\bigsqcup A$ of the induced basis has a joining map $\mathrm{join}_{s_1} \circ \cdots \circ \mathrm{join}_{s_n}$. Notice that for $n = 0$ we have that $\bigsqcup A = \bot$ and that $\mathrm{join}_{s_1} \circ \cdots \circ \mathrm{join}_{s_n}$ is the identity. $\square$

Theorem 7.17 *The category of J-domains is cartesian closed.*

Proof Let D and E be J-domains w.r.t. countable bases A and B respectively.

The identity function of a one-point domain is a joining map for the unique element of the domain. Hence it is a J-domain. A countable basis of $D \times E$ is $A \times B$. If $(a, b) \in A \times B$ then a and b have joining maps join_a^D and join_b^E respectively, and it is easy to see that then (a, b) has a joining map $\mathrm{join}_{(a,b)}^{D \times E} \stackrel{\text{def}}{=} \mathrm{join}_a^D \times \mathrm{join}_b^E$. Therefore the category of J-domains is closed under the formation of finite products.

Recall that a countable subbasis of $D \Rightarrow E$ is given by the set of single-step functions $A \Rightarrow_s B = \{a \Mapsto b | a \in A, b \in B\}$, where

$$(a \Mapsto b)(x) = \begin{cases} b & \text{if } a \ll x, \\ \bot & \text{otherwise.} \end{cases}$$

Let $a \Mapsto b \in A \Rightarrow_s B$, let join_b^E be a joining map for b, and define a map $\text{way-below}_a^D : D \to \mathcal{B}$ by

$$\text{way-below}_a^D(x) = \begin{cases} \mathbf{tt} & \text{if } x \text{ is in } \Uparrow a, \\ \mathbf{ff} & \text{if } x \text{ is in the exterior of } \Uparrow a, \\ \bot & \text{if } x \text{ is in the boundary of } \Uparrow a. \end{cases}$$

Here the ***exterior*** of a set is the largest open set disjoint from it, namely the interior of its complement. Therefore this function is continuous. Define $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E} : (D \Rightarrow E) \to (D \Rightarrow E)$ by

$$\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f)(x) = \mathrm{pif}^E\ \text{way-below}_a^D(x) \text{ then } \mathrm{join}_b^E(f(x)) \text{ else } f(x),$$

and let f be an element of $D \Rightarrow E$ consistent with $a \Mapsto b$. In order to prove that $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f) = (a \Mapsto b) \sqcup f$, we show that $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f)(x) = ((a \Mapsto b) \sqcup f)(x)$ for every $x \in D$, by considering the following three cases:

(1) x is in $\Uparrow a$: In this case x is (trivially) consistent with a and hence $b = (a \Mapsto b)(x) \uparrow f(x)$. Therefore $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f)(x) = \mathrm{join}_b^E(f(x)) = b \sqcup f(x) = ((a \Mapsto b) \sqcup f)(x)$.

(2) x is in the exterior of $\Uparrow a$: Then $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f)(x) = f(x) = \bot \sqcup f(x) = ((a \Mapsto b) \sqcup f)(x)$.

(3) x is in the boundary of $\Uparrow a$: We first show that $b \uparrow f(x)$. By hypothesis, every neighbourhood of x intersects $\Uparrow a$. In particular, every $x' \ll x$ is way-consistent with a. Let $x' \ll x$ and $u \in \Uparrow x' \cap \Uparrow a$. Then $(a \Mapsto b)(u) = b \uparrow f(u)$, because function application preserves consistency. Since $x' \sqsubseteq u$ and hence $f(x') \sqsubseteq f(u)$, $b \uparrow f(x')$. This shows that $b \uparrow f(x')$ for every $x' \ll x$. Hence, by continuity of f and Lemma 7.7, $b \uparrow \bigsqcup_{x' \ll x} f(x') = f(x)$. It follows that $\mathrm{join}_{a \Mapsto b}^{D \Rightarrow E}(f)(x) = \mathrm{join}_b^E(f(x)) \sqcap f(x) = (b \sqcup f(x)) \sqcap f(x) = f(x) = ((a \Mapsto b) \sqcup f)(x)$.

Therefore, by Lemma 7.16, the category of J-domains is closed under the formation of function spaces. □

We don't pause to check whether the joining maps constructed above are minimal whenever the given joining maps are minimal. But we conjecture that this is the case.

The above theorem gives a particularly simple construction of joining maps for function spaces of algebraic J-domains. By Lemma 7.7, if d is an element of a bounded complete domain D, then the set $d^\sharp$ of elements apart from d is an open set disjoint from the open set $\Uparrow d$; that is, $d^\sharp$ is contained in the exterior of $\Uparrow d$. In general, there is no reason why these sets should be equal. In fact, it seems to be hard to characterize the exterior of $\Uparrow d$ in order-theoretical terms. But a simple characterization is available when d is compact and D is algebraic:

Lemma 7.18 *If d is a compact element of a bounded complete algebraic domain D, then the exterior of the open set $\Uparrow d = \uparrow d$ is $d^\sharp$.*

Proof Let x be an exterior point of $\uparrow d$. Then x has an open neighbourhood O disjoint from $\uparrow d$. By algebraicity of D, O contains a compact element $b \sqsubseteq x$. Hence b is apart from d, because O is an upper set, and therefore x itself is apart from d, because it is above b. $\square$

Part III

The partial real line

In Chapter 8 we introduce the partial real line. We include both standard facts and new results. In Chapter 9 we develop the idea that partial real numbers are "continuous words", which is applied to obtain an operational semantics for Real PCF in Chapter 11 of Part IV. In Chapter 10 we introduce induction principles and recursion schemes for the partial real line, which are related to the interpretation of partial real numbers as continuous words.

Chapter 8

Partial real numbers

In Section 8.1 we introduce the interval domain and its basic structure. In Section 8.2 we relate the Scott topology on the interval domain to the Euclidean topology on the real line. Since the subspace of maximal points of the interval domain is homeomorphic to the Euclidean real line, we decide to refer to is as the *partial real line*. In Section 8.3 we discuss extensions of continuous real valued functions of real valued variables to Scott continuous partial real valued functions of partial real variables. The material of these first three sections is standard.

In Section 8.4 we introduce new results about partial real valued functions. In Section 8.5 we relate *discontinuous* functions in analysis to *continuous* functions in domain theory. In Section 8.6 we discuss order of magnitude on partial real numbers. In Section 8.7 we show that every partial real number has a unique joining map (cf. Chapter 7 of Part II), and we conclude in particular that the partial real line is a coherently complete domain. In Chapter 8.8 we discuss a "parallel effect" on the partial real line which forces us to use the so-called parallel conditional, which is informally discussed in [BC90]. We give a simple topological explanation of the parallel effect.

Sections 8.9–8.13 are included as an appendix to this chapter. They can be omitted as they are not needed in the development which follows. Section 8.9 relates the partial real line to the Euclidean plane. Section 8.10 generalizes the construction of the real line by Dedekind cuts to the partial real line. Section 8.11 constructs the partial real line by round ideals, and it does not contain any new material. Section 8.12 discuss an algebraic version of the partial real line. Section 8.13 presents the partial real line as a quasi-metric space.

8.1 The interval domain

In this section we discuss standard facts about the interval domain introduced in [Sco72b], which is the main domain of interest in this thesis.

The set $\mathcal{R} = \mathbf{I}\mathbb{R}$ of non-empty compact subintervals of the Euclidean real line ordered by reverse inclusion

$$x \sqsubseteq y \text{ iff } x \supseteq y$$

is a domain, referred to as the ***interval domain***. If we add an artificial bottom element to $\mathcal{R}$, which can be concretely taken as the non-compact interval $(-\infty, +\infty)$, then $\mathcal{R}$ becomes a bounded complete domain $\mathcal{R}_\perp$.

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

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

so that $x = [\underline{x}, \overline{x}]$. A subset $A \subseteq \mathcal{R}$ has a least upper bound iff it has non-empty intersection, and in this case

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

Any subset $A \subseteq \mathcal{R}$ with a lower bound has a greatest lower bound, given by

$$\bigsqcap A = \left[\inf_{a \in A} \underline{a}, \sup_{a \in A} \overline{a}\right] \supseteq \bigcup A.$$

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

$$\begin{aligned} x \ll y \quad & \text{iff} \quad \underline{x} < \underline{y} \text{ and } \overline{y} < \overline{x} \\ & \text{iff} \quad \text{the interior of } x \text{ contains } y. \end{aligned}$$

The way-below relation of $\mathcal{R}$ is ***multiplicative***, in the sense that

$$x \ll y \text{ and } x \ll z \text{ together imply } x \ll y \sqcap z.$$

A basis of $\mathcal{R}$ is given by the intervals with distinct rational (respectively dyadic) end-points. Recall that a ***dyadic*** number is a rational number of the form $m/2^n$.

The unit interval domain

The set $\mathcal{I}= \mathbf{I}[0,1]$ of all non-empty closed intervals contained in the ***unit interval*** $[0,1]$ is a bounded complete, countably based domain, referred to as the ***unit interval domain***. The bottom element of $\mathcal{I}$ is the interval $[0,1]$. Its way-below order is given by

$$\begin{aligned} x \ll y \quad & \text{iff} \quad \underline{x} = 0 \text{ or } \underline{x} < \underline{y}, \text{ and } \overline{y} < \overline{x} \text{ or } \overline{y} = 1 \\ & \text{iff} \quad \text{the interior of } x \text{ contains } y. \end{aligned}$$

The extended interval domain

Let $\mathbb{R}^\star = [-\infty, +\infty]$ be the two-point compactification of the real line, also referred to as the ***extended real line***. Then the set $\mathcal{R}^\star = \mathbf{I}\mathbb{R}^\star$ of all non-empty compact intervals of the extended real line is a bounded complete, countably based domain, referred to as the ***extended interval domain***. Its way-below relation is given similarly.

8.2 The partial real line

Let $s : \mathbb{R} \to \mathcal{R}$ denote the ***singleton-map*** $r \mapsto \{r\}$. This is a function onto the maximal elements of $\mathcal{R}$. Since the sets $\twoheaduparrow x$, for $x \in \mathcal{R}$, form a basis of the Scott topology of $\mathcal{R}$, and since

$$\twoheaduparrow x \cap \operatorname{Max} \mathcal{R} = \{\{r\} | \underline{x} < r < \underline{x}\} = \{s(r) | r \in (\underline{x}, \overline{x})\},$$

the sets $\{s(r) | r \in (a, b)\}$, for open intervals (a, b), form a base of the relative topology on $\operatorname{Max} \mathcal{R}$. This shows that $\operatorname{Max} \mathcal{R}$ is homeomorphic to the real line, and that the singleton map is a subspace inclusion, because the open intervals (a, b) form a base of the topology of the real line.

Similarly, $\operatorname{Max} \mathcal{I}$ and $\operatorname{Max} \mathcal{R}^\star$ are homeomorphic to the unit interval and the extended real line.

Therefore non-singleton intervals can be regarded as approximations of real numbers, and we refer to them as ***partial real numbers*** and to the interval domain as the ***partial real line***. Similarly, we refer to the unit interval domain as the ***partial unit interval*** and to the extended interval domain as the ***extended partial real line***.

Remark 8.1 In interval analysis [Moo66] one works with $\mathcal{R}$ endowed with the topology induced by the Hausdorff metric on intervals. We refer to the resulting space as the ***interval space***. For topological connections between domain theory and interval analysis see [EC93]. In particular, it is shown that the Scott open sets of the interval domain are the open upper sets of the interval space. Conversely, the open sets of the interval space are the Lawson open sets of the interval domain. Since in interval analysis one restricts oneself to monotone functions, many results presented in [Moo66] go through if one replaces the topology induced by the Hausdorff metric by the Scott topology. See also Acióly [Aci91] for more connections between domain theory and interval analysis. $\square$

8.3 Canonical extensions of real valued maps of real variables

Every continuous map $f : \mathbb{R}^n \to \mathbb{R}$ extends to a continuous map $\mathbf{I}f : \mathcal{R}^n \to \mathcal{R}$ defined by

$$\mathbf{I}f(x_1, \ldots, x_n) = \{f(r_1, \ldots, r_n) | r_1 \in x_1, \ldots, r_n \in x_n\},$$

called its ***canonical extension***. For $n = 1$ we reason as follows. Since f is continuous, it maps connected sets to connected sets, and compact sets to compact sets. Hence it maps compact intervals to compact intervals. Therefore $\mathbf{I}f$ is well-defined. But extensions of maps to powersets preserve intersections of $\supseteq$-directed sets. Therefore $\mathbf{I}f$ is Scott continuous. For n arbitrary the argument is analogous.

It is easy to see that the canonical extension is the greatest monotone extension. Since it is continuous and every continuous function is monotone, it is also the greatest continuous extension. Notice however that the above extension property doesn't follow from the injectivity properties discussed in Section 2.8, because $\mathcal{R}$ is not bounded complete as it lacks a bottom element.

If the function f is increasing in each argument with respect to the natural order of $\mathbb{R}$, then $\mathbf{I}f$ is given pointwise:

$$\mathbf{I}f(x_1, \ldots, x_n) = [f(\underline{x_1}, \ldots, \underline{x_n}), f(\overline{x_1}, \ldots, \overline{x_n})].$$

If f is decreasing in each argument, then $\mathbf{I}f$ is given "antipointwise":

$$\mathbf{I}f(x_1, \ldots, x_n) = [f(\overline{x_1}, \ldots, \overline{x_n}), f(\underline{x_1}, \ldots, \underline{x_n})].$$

Convention 8.2 We often notationally identify the function $f : \mathbb{R}^n \to \mathbb{R}$ with its extension $\mathbf{I}f : \mathcal{R}^n \to \mathcal{R}$, and a number $r \in \mathbb{R}$ with its inclusion $\{r\} \in \mathcal{R}$. Moreover, we often define a function $f : \mathcal{R}^n \to \mathcal{R}$ by first defining a function $f : \mathbb{R}^n \to \mathbb{R}$ and then implicitly taking its canonical extension. The same convention applies to functions denoted by operator symbols, such as addition denoted by $+$. □

For example,

$$\begin{aligned} x + y &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\ 1 - 2x &= [1 - 2\overline{x}, 1 - 2\underline{x}]. \end{aligned}$$

Of course, the extension properties discussed for the interval domain also apply to the partial unit interval.

For the extended partial real line we can reason as follows. The closure of $\mathbb{R}$ in $\mathbb{R}^\star$ is $\mathbb{R}^\star$. Hence the singleton map $r \mapsto \{r\} : \mathbb{R} \to \mathcal{R}^\star$ embeds the real line as a dense subspace of $\mathcal{R}^\star$. Therefore, by the injective property of bounded complete domains (cf. Section 2.8), every continuous function $f : \mathbb{R} \to \mathbb{R}$ has a greatest continuous extension $f^\star : \mathcal{R}^\star \to \mathcal{R}^\star$. If the limits of f at $\pm\infty$ exist, then the values of $f^\star$ at $\pm\infty$ are these limits, as discussed in the next section.

8.4 Partial real valued functions

In this section we consider functions defined on any space with values on the extended partial real line.

The projections $\underline{\pi}, \overline{\pi} : \mathcal{R}^\star \to \mathbb{R}^\star$ defined by

$$\underline{\pi}(x) = \underline{x} \qquad \text{and} \qquad \overline{\pi}(x) = \overline{x}$$

are not continuous because they do not preserve the specialization order, as the specialization order of $\mathbb{R}^\star$ is discrete.

The set of extended real numbers endowed with its natural order $\leq$ is a continuous lattice, and so is its opposite (see [GHK$^+$80]). Moreover, for any space X, a function $f : X \to \mathbb{R}^\star$ is lower semicontinuous iff it is continuous with respect to the Scott topology on $\mathbb{R}^\star$ induced by $\leq$, and it is upper semicontinuous iff it is continuous with respect to the Scott topology on $\mathbb{R}^\star$ induced by $\geq$. It is clear from this observation that the above projections are respectively lower and upper semicontinuous.

In order to avoid the rather long terms "lower semicontinuous" and "upper semicontinuous", we denote by $\underline{\mathbb{R}}$ and $\overline{\mathbb{R}}$ the set of extended real numbers endowed with the Scott topologies induced by $\leq$ and $\geq$ respectively, and we refer to the points of these topological spaces as respectively ***lower*** and ***upper*** real numbers. Thus, the above projections are continuous functions $\underline{\pi} : \mathcal{R}^\star \to \underline{\mathbb{R}}$ and $\overline{\pi} : \mathcal{R}^\star \to \overline{\mathbb{R}}$.

The projections satisfy

$$\underline{\pi} \leq \overline{\pi}$$

pointwise. Thus, given any continuous function $f : X \to \mathcal{R}^\star$, we can define continuous functions $\underline{f} : X \to \underline{\mathbb{R}}$ and $\overline{f} : X \to \overline{\mathbb{R}}$ by composition with the projections, and we have that $\underline{f} \leq \overline{f}$ pointwise. Conversely,

Lemma 8.3 *For any space X and all continuous maps $\underline{f} : X \to \underline{\mathbb{R}}$ and $\overline{f} : X \to \overline{\mathbb{R}}$ with*

$$\underline{f} \leq \overline{f}$$

pointwise, there is a unique continuous map $f : X \to \mathcal{R}^\star$ such that

$$\underline{f} = \underline{\pi} \circ f \qquad \textit{and} \qquad \overline{f} = \overline{\pi} \circ f,$$

namely $[\underline{f}, \overline{f}]$ defined by

$$[\underline{f}, \overline{f}](x) = [\underline{f}(x), \overline{f}(x)].$$

Proof It suffices to show that $[\underline{f}, \overline{f}]$ is continuous. Given a basic open set $\Uparrow y$ in $\mathcal{R}^\star$, we have that

$$\begin{aligned}
[\underline{f}, \overline{f}]^{-1}(\Uparrow y) &= \{x \in X | y \ll [\underline{f}, \overline{f}](x)\} \\
&= \{x \in X | \underline{y} \ll_{\underline{\mathbb{R}}} \underline{f}(x) \text{ and } \overline{y} \ll_{\overline{\mathbb{R}}} \overline{f}(x)\} \\
&= \{x \in X | \underline{y} \ll_{\underline{\mathbb{R}}} \underline{f}(x)\} \cap \{x \in X | \overline{y} \ll_{\overline{\mathbb{R}}} \overline{f}(x)\} \\
&= \underline{f}^{-1}(\Uparrow_{\underline{\mathbb{R}}} \underline{y}) \cap \overline{f}^{-1}(\Uparrow_{\overline{\mathbb{R}}} \overline{y})
\end{aligned}$$

is an open set, because $\Uparrow_{\underline{\mathbb{R}}} \underline{y}$ and $\Uparrow_{\overline{\mathbb{R}}} \overline{y}$ are open sets in $\underline{\mathbb{R}}$ and $\overline{\mathbb{R}}$ respectively. Therefore $[\underline{f}, \overline{f}]$ is continuous. □

Thus, for any space X, a continuous function $f : X \to \mathcal{R}^\star$ is essentially the same as a pair of continuous maps $\langle \underline{f} : X \to \underline{\mathbb{R}}, \overline{f} : X \to \overline{\mathbb{R}} \rangle$ with $\underline{f} \leq \overline{f}$ pointwise.

We can thus say that an extended partial real number is given by a pair $\langle \underline{x}, \overline{x} \rangle$ of respectively lower and upper real numbers such that $\underline{x} \leq \overline{x}$.

Corollary 8.4 *$\mathcal{R}^\star$ is homeomorphic to the subspace of $\underline{\mathbb{R}} \times \overline{\mathbb{R}}$ consisting of pairs of extended real numbers $\langle \underline{x}, \overline{x} \rangle$ with $\underline{x} \leq \overline{x}$.*

Since $\mathcal{R}^\star$ is a bounded complete domain, it is a densely injective space (cf. Section 2.8).

Proposition 8.5 *Let X be a dense subspace of a metric space Y and $f : X \to \mathcal{R}^\star$ be a continuous map. Then the greatest continuous extension $\hat{f} : Y \to \mathcal{R}^\star$ of f is given by*

$$\hat{f}(y) = \left[\liminf_{x \to y} \underline{f}(x), \limsup_{x \to y} \overline{f}(x)\right].$$

Here $x \to y$ is a short-hand for $x \in X$ and $x \to y$.

Proof

$$\begin{aligned}
\hat{f}(y) &= \bigsqcup^{\uparrow}_{y \in O \in \Omega Y} \bigsqcap_{x \in X \cap O} f(x) \\
&= \bigsqcup^{\uparrow}_{\epsilon > 0} \bigsqcap_{0 < d(x,y) < \epsilon} [\underline{f}(x), \overline{f}(x)] \\
&= \bigsqcup^{\uparrow}_{\epsilon > 0} \left[\inf_{0 < d(x,y) < \epsilon} \underline{f}(x), \sup_{0 < d(x,y) < \epsilon} \overline{f}(x)\right] \\
&= \left[\sup_{\epsilon > 0} \inf_{0 < d(x,y) < \epsilon} \underline{f}(x), \inf_{\epsilon > 0} \sup_{0 < d(x,y) < \epsilon} \overline{f}(x)\right] \\
&= \left[\liminf_{x \to y} \underline{f}(x), \limsup_{x \to y} \overline{f}(x)\right]. \square
\end{aligned}$$

In particular, if $f : X \to \mathbb{R}^\star$ is a continuous map, then the above theorem applied to the coextension $s \circ f : X \to \mathcal{R}^\star$ of f to $\mathcal{R}^\star$, where $s : \mathbb{R}^\star \to \mathcal{R}^\star$ is the singleton embedding, produces a greatest extension $\hat{f} : Y \to \mathcal{R}^\star$ of f, given by

$$\hat{f}(y) = \left[\liminf_{x \to y} f(x), \limsup_{x \to y} f(x)\right].$$

Let $f : \mathbb{R} \to \mathbb{R}$ be continuous. By the above remark, if f has a limit at ∞, then $\hat{f}(\infty) = \lim_{x \to \infty} f(x)$. For a pathological example, consider $f : (\mathbb{R} - \{0\}) \to \mathbb{R}$ defined by $f(x) = \sin(1/x)$. Then $\hat{f}(0) = [-1, 1]$, so that $\hat{f}$ behaves as the so-called *topologist's sine curve* [HY88]. Also, $\hat{f}(\pm\infty) = 0$.

Lemma 8.6 *Every continuous map $f : \mathbb{R} \to \mathcal{R}$ has a greatest continuous extension $\hat{f} : \mathcal{R} \to \mathcal{R}$, given by*

$$\hat{f}(x) = \bigsqcap f(x).$$

Proof Since this is clearly the greatest *monotone* extension, it suffices to show that it is continuous. In this proof we make use of techniques not introduced in the background Part I, which can be found in [Sch93] (see also [Smy83, Vic89, Eda95e]). Let $\mathbf{U}$ be the endofunctor on the category of topological spaces which assigns to a space X its *upper space*, whose points are the non-empty compact saturated sets of X, and which assigns to a continuous map $f : X \to Y$ the continuous map $\mathbf{U}f : \mathbf{U}X \to \mathbf{U}Y$ defined by $\mathbf{U}f(Q) = \uparrow f(Q)$. Then for any space X the map $x \mapsto \uparrow x : X \to \mathbf{U}X$ is continuous, and for any continuous $\sqcap$-semilattice D, the meet map $\bigsqcap : \mathbf{U}D \to D$ is well-defined and continuous.

Since $\mathcal{R}$ is a continuous $\sqcap$-semilattice and a subspace of $\mathbf{U}\mathbb{R}$, the map $\hat{f}$ is continuous, because it can be expressed as the following composition of continuous maps:

$$\begin{array}{ccccccc}
\mathcal{R} & \hookrightarrow & \mathbf{U}\mathbb{R} & \xrightarrow{\mathbf{U}f} & \mathbf{U}\mathcal{R} & \xrightarrow{\quad\bigsqcap\quad} & \mathcal{R} \\
x & \mapsto & x & \mapsto & \uparrow f(x) & \mapsto & \bigsqcap \uparrow f(x) = \bigsqcap f(x).
\end{array}$$

(Note: This also shows that the assignment $f \mapsto \hat{f}$ is Scott continuous, and is a particular case of a much more general fact about injectivity established in [Esc97]). $\square$

8.5 Discontinuous functions in real analysis versus continuous functions in domain theory

This section contains new results about extensions of *arbitrary* real valued functions to *continuous* partial real valued functions.

In real analysis one often considers discontinuous functions $f : \mathbb{R} \to \mathbb{R}$, but in many cases only the points of continuity of f are interesting. For instance, a function $f : \mathbb{R} \to \mathbb{R}$ is Riemann integrable on any compact interval iff it is bounded on compact intervals and continuous almost everywhere [Roy88]. Moreover, the integral of f depends only on its points of continuity. The following theorem shows that such uses of ad hoc discontinuity can be avoided in domain theory.

Lemma 8.7 *For any function $f : X \to \mathbb{R}^\star$ defined on a metric space X there is a* continuous *map $\tilde{f} : X \to \mathcal{R}^\star$ agreeing with f at every point of continuity of f, given by*

$$\tilde{f}(x) = \left[\liminf_{y \to x} f(y), \limsup_{y \to x} f(y)\right].$$

Proof We know from classical topology and analysis that

$$\underline{g}(y) = \liminf_{x \to y} f(x)$$

is the greatest lower semicontinuous function below f, and that

$$\overline{g}(y) = \limsup_{x \to y} f(x)$$

is the least upper semicontinuous function above f (see e.g. [Bou66, Roy88]). Since $\tilde{f}$ is $[\underline{g}, \overline{g}]$, it is continuous. Since f is continuous at y iff $\lim_{x \to y} g(x)$ exists iff $\liminf_{x \to y} f(x) = \limsup_{x \to y} g(x)$, $\tilde{f}$ agrees with f at every point of continuity of f. $\square$

Theorem 8.8 *For any function $f : \mathbb{R} \to \mathbb{R}$ bounded on compact intervals there is a continuous map $\ddot{f} : \mathcal{R} \to \mathcal{R}$ agreeing with f at every point of continuity of f, given by*

$$\ddot{f}(x) = [\inf \underline{g}(x), \sup \overline{g}(x)],$$

where $\underline{g} : \mathbb{R} \to \underline{\mathbb{R}}$ and $\overline{g} : \mathbb{R} \to \overline{\mathbb{R}}$ are continuous maps defined by

$$\underline{g}(y) = \liminf_{x \to y} f(x) \qquad \textit{and} \qquad \overline{g}(y) = \limsup_{x \to y} f(x).$$

Proof Since f is bounded on compact intervals, the function $\tilde{f} : \mathbb{R} \to \mathcal{R}^{\star}$ defined in Lemma 8.7 corestricts to $\mathcal{R}$. By Lemma 8.6, the corestriction can be extended to a function $\ddot{f} : \mathcal{R} \to \mathcal{R}$, given by

$$\ddot{f}(x) = \bigsqcap \tilde{f}(x) = [\inf \underline{g}(x), \sup \overline{g}(x)]. \qquad \square$$

8.6 Order of magnitude on the partial real line

We define a strict order $<$ on partial numbers by

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

This relation is clearly irreflexive, transitive, and asymmetric (in the sense that $x < y$ together with $x > y$ is impossible). The following lemma is immediate:

Lemma 8.9 *For all partial real numbers x and y, exactly one of the relations $x < y$, $x \uparrow y$ and $x > y$ holds.*

Lemma 8.10 *For all $x, y, u, v \in \mathcal{R}$,*

1. *$u < x \sqcap y$ iff $u < x$ and $u < y$,*
2. *$x \sqcap y < v$ iff $x < v$ and $y < v$.*

Proof (1): $u < x \sqcap y$ iff $\overline{u} < \underline{x \sqcap y}$ iff $\overline{u} < \min(\underline{x}, \underline{y})$ iff $\overline{u} < \underline{x}$ and $\overline{u} < \underline{y}$ iff $u < x$ and $u < y$. (2): Similar. $\square$

Let $\mathbb{B}= \{\mathbf{tt}, \mathbf{ff}\}$ be the discrete space of truth values. The characteristic function $\chi_< : \mathbb{R} \times \mathbb{R} \to \mathbb{B}$ of the inequality predicate $<$ is discontinuous at each point $\langle x, x \rangle$. This reflects the fact that equality of real numbers is undecidable [ML70, PeR83, Wei87, Wei95]. Recall that $\mathcal{B}= \mathbb{B}_\perp$ is the flat domain of partial truth values. Since the points of continuity of $\chi_<$ form a dense set, $\chi_<$ restricted to its points of continuity has a greatest continuous extension $(x, y) \mapsto (x <_\perp y) : \mathcal{R}_\perp \times \mathcal{R}_\perp \to \mathcal{B}$. This extension is given by

$$(x <_\perp y) = \begin{cases} \mathbf{tt} & \text{if } x < y, \\ \mathbf{ff} & \text{if } x > y, \\ \perp & \text{if } x \uparrow y. \end{cases}$$

We define a relation $\leq$ on partial numbers by

$x \leq y$ iff $\overline{x} \leq \underline{y}$.

The relation $\leq$ transtive and antisymmetric, but reflexive only on total real numbers, and therefore the notation can be misleading.

8.7 The partial real line as a J-domain

Recall from Chapter 7 that a joining map of an element d of a bounded complete domain D is a continuous function $\mathrm{join}_d : D \to D$ such that

$$\mathrm{join}_d(x) = d \sqcup x \text{ for every } x{\uparrow}d,$$

and that a bounded complete domain is a J-domain iff it has a basis such that every basis element has a joining map.

Proposition 8.11 *Each $a \in \mathcal{R}_\perp$ has a unique joining map* join_a.

Proof If $a = \perp$ the identity is the only joining map. Otherwise, if $\mathrm{join}_a : \mathcal{R}_\perp \to \mathcal{R}_\perp$ is a joining map of a, then its monotonicity implies that $\mathrm{join}_a(x) = \underline{a}$ for all $x < a$ and $\mathrm{join}_a(x) = \overline{a}$ for all $x > a$. In fact, assume that $x < a$. This means that $\overline{x} < \underline{a}$. Then $[\underline{x}, \underline{a}] \sqsubseteq x$. Hence $\underline{a} = \mathrm{join}_a([\underline{x}, \underline{a}]) \sqsubseteq \mathrm{join}_a(x)$. Therefore $\underline{a} = \mathrm{join}_a(x)$, because $\underline{a}$ is maximal. The other case is similar. By Lemma 8.9, if a has a joining map then it has to be join_a defined by

$$\begin{aligned}\mathrm{join}_a(x) &= \begin{cases} \underline{a} & \text{if } x < a, \\ a \sqcup x & \text{if } x{\uparrow}a, \\ \overline{a} & \text{if } a < x \end{cases} \\ &= \max(\underline{a}, \min(x, \overline{a})).\end{aligned}$$

Notice that we are applying Convention 8.2. Since this map is continuous, it is a joining map of a. □

Since a domain is coherently complete iff each of its elements has a joining map, we have that

Corollary 8.12 *$\mathcal{R}_\perp$ is a coherently complete domain.*

8.8 A parallel effect in the partial real line

We say that a continuous predicate $p : \mathcal{R}_\perp \to \mathcal{B}$ is ***non-trivial*** if there are total real numbers x and y such that $p(x) = \mathbf{tt}$ and $p(y) = \mathbf{ff}$, and we say that a function $f : \mathcal{R}_\perp \to D$ is ***undefined*** at x if $f(x) = \perp$.

Proposition 8.13 *Let D be a domain, $p : \mathcal{R}_\perp \to \mathcal{B}$ be a continuous predicate, $g, h : \mathcal{R}_\perp \to D$ be continuous functions, and define a function $f : \mathcal{R}_\perp \to D$ by*

$$f(x) = \text{if } p(x) \text{ then } g(x) \text{ else } h(x).$$

If p is non-trivial then f is undefined at some total real number.

The proof depends only on the fact that $\mathrm{Max}(\mathcal{R}_\perp)$ is a connected space.

Proof The non-empty disjoint sets $U = p^{-1}(\mathbf{tt}) \cap \mathrm{Max}(\mathcal{R}_\perp)$ and $V = p^{-1}(\mathbf{ff}) \cap \mathrm{Max}(\mathcal{R}_\perp)$ are open in $\mathrm{Max}(R)$, because p is continuous, and $\{\mathbf{tt}\}$ and $\{\mathbf{ff}\}$ are open in $\mathcal{B}$. Hence $U \cup V \neq \mathrm{Max}(\mathcal{R}_\perp)$, because $\mathrm{Max}(\mathcal{R}_\perp)$ is connected. Therefore there is some maximal element x such that $p(x) = \perp$. □

Thus, the sequential conditional is not appropriate for definition by cases of total functions on the partial real line (or any domain with subspace of maximal points homeomorphic to the real line) because it produces non-total functions for non-trivial continuous predicates. We claim that the parallel conditional overcomes this deficiency of the sequential conditional. This "parallel effect" seems to be related to the "intensional effect" described in [Luc77].

In virtually all definitions by cases of the form

$$f(x) = \text{pif } p(x) \text{ then } g(x) \text{ else } h(x)$$

given in this work, one has that $g(x) = h(x)$ for all maximal x with $p(x) = \bot$. In such a situation, if x is maximal and $p(x) = \bot$, then

$$f(x) = g(x) \sqcap h(x) = g(x) = h(x).$$

The examples below illustrate this situation, and, moreover, they show that the parallel conditional is useful to overcome the fact that equality of real numbers is undecidable.

Example 8.14 The following definition gives an extension of the absolute value function:

$$|x| = \text{pif } x <_\bot 0 \text{ then } -x \text{ else } x.$$

For the case $x = 0$ one has

$$|0| = \text{pif } \bot \text{ then } -0 \text{ else } 0 = 0 \sqcap 0 = 0. \qquad \square$$

Example 8.15 Let E be any domain, and let $f, g : \mathcal{I} \to E$ be continuous maps. Then f and g are (generalized) paths in the domain E. (The restrictions of f and g to total real numbers are paths in the usual sense [HY88].) If $f(1) = g(0)$ the paths are said to be ***composable***. Now define a continuous map $h : \mathcal{I} \to E$ by

$$h(x) \quad = \quad \text{pif } x <_\bot \tfrac{1}{2} \text{ then } f(2x) \text{ else } g(2x-1)$$

If the paths are composable, then h is a (generalized) ***composite path***. Let us check the crucial case $x = \frac{1}{2}$:

$$\begin{aligned} h(\tfrac{1}{2}) &= \text{pif } \tfrac{1}{2} <_\bot \tfrac{1}{2} \text{ then } f(2 \cdot \tfrac{1}{2}) \text{ else } g(2 \cdot \tfrac{1}{2} - 1) \;=\; \text{pif } \bot \text{ then } f(1) \text{ else } g(0) \\ &= f(1) \sqcap g(0) \;=\; f(1) \;=\; g(0) \end{aligned}$$

Notice that the sequential conditional would produce $\bot$ instead of $f(1) \sqcap g(0)$, and therefore h would be undefined at $\frac{1}{2}$.

If f and g are not composable, then h is a generalized composite path with a *jump* at $\frac{1}{2}$, namely $f(1) \sqcap g(0)$. For instance, if $E = \mathcal{I}$ and f and g are constant maps with range 0 and 1 respectively, then h can be thought as a *switch* which turns on at time $\frac{1}{2}$. In this case, the switch is in the transition state $[0,1] = 0 \sqcap 1$ at time $\frac{1}{2}$. Notice that even in this case h *is* Scott continuous, because it is a composition of continuous maps. $\square$

The following lemma is useful to prove properties of definitions involving the parallel conditional. Recall that a function between two bounded complete domains is ***multiplicative*** iff it preserves binary meets.

Lemma 8.16 *Let $f : D \to E$ be a continuous map between bounded complete domains D and E. Then*

$$f(\text{pif } p \text{ then } x \text{ else } y) = \text{pif } p \text{ then } f(x) \text{ else } f(y)$$

for all $p \in \mathcal{B}$ and all $x, y \in D$ iff f is multiplicative.

Proof ($\Rightarrow$): Take $p = \bot$. Then the left-hand side of the equation is $f(x \sqcap y)$ and the right-hand side is $f(x) \sqcap f(y)$.

($\Leftarrow$): If $p = \mathbf{tt}$ then lhs=$f(x)$=rhs, If $p = \mathbf{ff}$ then lhs=$f(y)$=rhs, and if $p = \bot$ then lhs=$f(x \sqcap y) = f(x) \sqcap f(y)$=rhs. □

Appendix

Section 8.9 relates the partial real line to the Euclidean plane. Section 8.10 generalizes the construction of the real line by Dedekind cuts to the partial real line. Section 8.11 constructs the partial real line by round ideals, and it does not contain any new material. Section 8.12 discuss an algebraic version of the partial real line. Section 8.13 presents the partial real line as a quasi-metric space.

8.9 The unit triangle and the half-plane

The partial unit interval can be presented in a geometrically convenient form as follows. The unit square $[0,1] \times [0,1]$ under the componentwise order induced by the usual order $\leq$ on $[0,1]$ is a continuous lattice, whose Lawson topology coincides with the Euclidean topology on the unit square [GHK$^+$80] (see Remark 8.1). If we consider the points below (equivalently, on the left of) the diagonal which goes from $(0,1)$ to $(1,0)$, that is, the points (x,y) with $x + y \leq 1$, we get a triangle, which we refer to as the ***unit triangle***. The unit triangle is easily seen to be a domain. Its maximal elements are the points (x,y) with $x + y = 1$, that is, the points on the diagonal.

It turns out that the unit triangle is isomorphic to the partial unit interval. The isomorphisms can be taken as

$$\begin{aligned} (x,y) &\mapsto [x, 1-y], \\ [x,y] &\mapsto (x, 1-y). \end{aligned}$$

We can think of the unit triangle as a *coordinate system* for the partial unit interval.

We have a similar fact for the partial real line; a coordinate system for $\mathcal{R}$ is given by the ***half-plane*** consisting of the points (x,y) with $x + y \leq 0$. A coordinate system for $\mathcal{R}_\bot$ is obtained by adding a point $(-\infty, -\infty)$ to the half-plane, and a coordinate system for $\mathcal{R}^\star$ is obtained similarly.

8.10 A construction of the partial real line by Dedekind cuts

The real line can be constructed by Dedekind cuts [Rud76]. In this section we extend this construction to the extended partial real numbers domain.

A Dedekind ***cut*** is a pair $\langle L, R\rangle$ of sets of rational numbers such that

1. L and R are non-empty.

2. L is a lower set with no greatest element, and R is an upper set with no least element.

3. Every member of L is strictly below every element of R.

4. If $p < q$ are rational numbers then p is a member of L or q is a member of R.

The second axiom can be put in constructive form, by positively expressing the negation of the universal quantification.

If the set $\mathbb{R}$ of real numbers is already given, then there is a bijection between real numbers and cuts, given by the maps

$$\begin{aligned} x &\mapsto \langle \{r \in \mathbb{Q} | r < x\}, \{r \in \mathbb{Q} | x < r\}\rangle, \\ \langle L, R\rangle &\mapsto \sup L \qquad \text{(or, equivalently, } \inf R). \end{aligned}$$

Otherwise, we can *define* $\mathbb{R}$ to be the set of cuts, and, indeed, this is one of the approaches to the construction of real numbers. The idea is that a real number is uniquely determined by its sets of rational lower and upper bounds.

If the first axiom is omitted, then we obtain the ***extended cuts***. Only two new cuts arise, namely $\langle \emptyset, \mathbb{Q}\rangle$ and $\langle \mathbb{Q}, \emptyset\rangle$, which can be thought as $-\infty$ and $+\infty$.

If the last axiom is also omitted, we obtain the ***partial extended cuts***, and a construction of the partial extended real line. The information order on cuts can be defined by $\langle L, R\rangle \sqsubseteq \langle L', R'\rangle$ iff $L \subseteq L'$ and $R \subseteq R'$. We thus have an isomorphism between partial extended real numbers and partial extended cuts, given by

$$\begin{aligned} x &\mapsto \langle \{r \in \mathbb{Q} | r < \underline{x}\}, \{r \in \mathbb{Q} | \overline{x} < r\}\rangle, \\ \langle L, R\rangle &\mapsto [\sup L, \inf R]. \end{aligned}$$

The partial real line can be constructed by keeping the first axiom and omitting the last, but allowing the partial cut $\perp = \langle \emptyset, \emptyset\rangle$. The lower and upper real line $\underline{\mathbb{R}}$ and $\overline{\mathbb{R}}$ can be constructed by partial extended cuts of the form $\langle L, \emptyset\rangle$ and $\langle \emptyset, R\rangle$ respectively.

If real numbers, extended real numbers, lower and upper real numbers, partial real numbers, and partial extended real numbers are constructed in this way, all of them coexist in a single universe, namely the domain of partial extended cuts.

Cuts have a natural computational interpretation. We can say that we know a (partial extended) real number if we know which rationals are below it and which rationals are above it. Then computability can be taken as effective knowledge, and, in fact, for real numbers this coincides with the definition of computability for the real line given in e.g. [PeR83, Wei87, Wei95]. The information order as defined above gives a natural ordering between the degrees of knowledge induced by partial cuts.

8.11 A construction of the partial real line by round ideals

Here we consider the abstract basis of $\mathcal{R}$ consisting of intervals with distinct rational end-points, referred to as the ***abstract rational basis***. The order is given by

$$x \prec y \text{ iff } \underline{x} < \underline{y} \text{ and } \overline{x} > \overline{y}.$$

The treatment of the extended case is similar. It is easy to see that there is a bijection between round ideals in the abstract rational basis and partial Dedekind cuts, given by

$$\begin{array}{rcl} A & \mapsto & \langle \{p | \, [p,q] \in A \text{ for some } q\}, \{q | \, [p,q] \in A \text{ for some } p\} \rangle \\ \langle L, R \rangle & \mapsto & \{\, [p,q] | \, p \in L \text{ and } q \in R\}. \end{array}$$

Martin-Löf's approach to constructive real analysis [ML70] is based on the abstract rational basis. (Notice that a recursion-theoretic approach to constructivity is adopted in loc. cit.) He considers the elements of the abstract rational basis as *formal intervals*, refers to them as *neighbourhoods*, and reads the expression $x \prec y$ as y *is finer than* x. He refers to a round ideal as an *approximation*, to its members as its *neighbourhoods*, positively defines the apartness relation by

$$x \# y \text{ iff } \overline{x} < \underline{y} \text{ or } \underline{x} > \overline{y},$$

and defines

> *"An approximation a is* maximal *if, for every pair of neighbourhoods I and J such that I is finer than J, either I lies apart from some neighbourhood of a or J is a neighbourhood of a. The maximal approximations form the* constructive points *of the space we are considering."*

A non-constructive argument (using e.g. Zorn's Lemma) is needed to prove that maximality in his sense coincides with order-theoretic maximality.

In the domain-theoretic approach to real number computation, approximations and points coexist in a single space, and approximations are considered as partial points. The construction of the partial real line by round ideals is discussed in detail in [Aci91].

8.12 An algebraic version of the partial real line

As we have seen in Section 2.7, every domain is a retract of the ideal completion of any of its bases, in a canonical way. Here we consider the ideal completion $\tilde{\mathcal{R}}$ of the basis B of $\mathcal{R}$ consisting of intervals with (not necessarily distinct) rational end-points. Then we have a section-retraction pair

$$\mathcal{R} \underset{e}{\overset{p}{\leftrightarrows}} \tilde{\mathcal{R}}$$

defined by

$$\begin{aligned} e(x) &= \mathop{\downarrow\!\!\downarrow} x \\ p(X) &= \bigsqcup{}^{\uparrow} X. \end{aligned}$$

Since sections are subspace embeddings, if $s : \mathbb{R} \to \mathcal{R}$ is the singleton embedding defined in Section 8.2, then $e \circ s : \mathbb{R} \to \tilde{\mathcal{R}}$ is a subspace embedding, which is *not* onto the maximal elements of $\tilde{\mathcal{R}}$. In fact, if $r \in \mathcal{R}$ is rational, then the round ideal

$$r^{\circ} = e \circ s(r) = \{[a,b] \in B | a < r < b\} = \mathop{\downarrow\!\!\downarrow}\{r\}$$

has three ideals above it, namely

$$\begin{aligned} r^{\Box} &= \{[a,b] \in B | a \leq r \leq b\} = \downarrow\{r\} \\ r^{-} &= \{[a,b] \in B | a < r \leq b\} \\ r^{+} &= \{[a,b] \in B | a \leq r < b\}. \end{aligned}$$

Among these, the "square ideal" $r^{\Box}$ is maximal, and it is an open point of $\tilde{\mathcal{R}}$, simply because it is the principal ideal generated by a maximal basis element. Therefore the map $d : \mathbb{Q} \to \tilde{\mathcal{R}}$ defined by

$$d(r) = r^{\Box}$$

is an embedding of the ***discrete space of rational numbers*** into the maximal elements of $\tilde{\mathcal{R}}$.

Since the image of d is open in $\tilde{\mathcal{R}}$, the closure of the image of $e \circ s$ is disjoint from the image of d. Therefore the embedding $e \circ s$ is not dense. But it is easily seen to be strongly isochordal (cf. Section 2.8). Since $\tilde{\mathcal{R}}_{\perp}$ is easily seen to be a coherently complete domain (and hence isochordally injective), we have that every continuous function $f : \mathbb{R} \to \mathbb{R}$ extends to a continuous function $\tilde{f} : \tilde{\mathcal{R}}_{\perp} \to \tilde{\mathcal{R}}_{\perp}$. This justifies an approach to real number computation based on $\tilde{\mathcal{R}}$, or the similar algebraic domain considered by Pietro di Gianantonio [Gia93a].

8.13 The partial real line as a quasi-metric space

A ***quasi-metric space*** [Law73, Smy87, Smy89, Smy92b] is a set X together with a function

$$d : X \times X \to [0, \infty]$$

such that

1. $d(x, x) = 0$,

2. $d(x, y) + d(y, z) \geq d(x, z)$.

Any quasi-metric induces a topology, given by the basis of ***open balls***

$$B_\epsilon(x) = \{y \in X | d(x, y) < \epsilon\}.$$

The quasi-metric and specialization order induced by the topology are related by

$d(x, y) = 0$ iff $x \sqsubseteq y$.

If d is a quasi-metric then so is its ***opposite***, defined by

$$d^{\mathrm{op}}(x, y) = d(y, x),$$

The ***symmetrization*** of d defined by

$$d^*(x, y) = \max(d(x, y), d^{\mathrm{op}}(x, y))$$

is a metric, as it is immediate that

$d^*(x, y) = d^*(y, x)$.

The topology induced by d^* is the join of the topologies induced by d and d^{op}.

The examples below show that a quasi-metric d has the following intuitive interpretation:

> $d(x, y)$ tells us how much we have to degrade x so that it becomes smaller than y in the specialization order.

We can define a quasi-metric on the extended real numbers by

$$d(x, y) = x \mathbin{\dot{-}} y,$$

where $\dot{-}$ is ***truncated subtraction***, defined by

$$\begin{aligned} x \mathbin{\dot{-}} y &= \begin{cases} x - y & \text{if } x \geq y, \\ 0 & \text{if } x \leq y \end{cases} \\ &= \inf\{\epsilon > 0 \,|\, x - \epsilon \leq y\}. \end{aligned}$$

Notice that a subtraction involving infinities can be ambiguous, but that the last term of the definition eliminates the ambiguity if ϵ is assumed to be finite. We call this the ***lower quasi-metric*** on the extended real numbers, as the induced topology is the lower topology, because $d(x, y) < \epsilon$ iff $x - \epsilon \ll y$, and hence

$$B_\epsilon(x) = \mathord{\Uparrow}(x - \epsilon).$$

Dually, d^{op} induces the upper topology and is called the ***upper quasi-metric***. Hence d^* induces the the usual topology on the extended real numbers. In fact,

$$d^*(x, y) = \max((x \mathbin{\dot{-}} y), (y \mathbin{\dot{-}} x)) = |y - x|,$$

as one of $x \dot{-} y$ and $y \dot{-} x$ must be zero and the other must be $|y - x|$.

Limits of sequences and Cauchy sequences can be defined in the same way as for metric spaces. Thus, a limit of a sequence x_i on a quasi-metric space X is a point $y \in X$ such that for every $\epsilon > 0$ there is an n such that $d(x_i, y) < \epsilon$ for all $i \geq n$, and x_i is a Cauchy sequence if for every $\epsilon > 0$ there is an n such that $d(x_i, x_j) < \epsilon$ for all $i, j \geq n$. Although limits as defined above clearly coincide with limits with respect to the induced topology, it is not true that every convergent sequence is Cauchy. The following proposition is proved in a more general form in [Smy89]:

Proposition 8.17 *A sequence x_i of extended real numbers converges to y with respect to the lower quasi-metric iff*

$$y \leq \limsup_{i \in \omega} x_i.$$

In particular, if x_i is increasing then it converges to y iff

$$y \leq \sup_{i \in \omega} x_i.$$

Proof x_i converges to y

iff for every $\epsilon > 0$ there is an n such that $x_i - \epsilon < y$ for all $i \geq n$,

iff for every $\epsilon > 0$ there is an n such that $\sup_{i \geq n} x_i - \epsilon < y$,

iff for every $\epsilon > 0$, $\inf_{n \in \omega} \sup_{i \geq n} x_i - \epsilon < y$,

iff for every $\epsilon > 0$, $\limsup_{i \in \omega} x_i - \epsilon < y$,

iff for every $\epsilon > 0$, $d(\limsup_{i \in \omega} x_i, y) < \epsilon$,

iff $d(\limsup_{i \in \omega} x_i, y) \leq 0$,

iff $y \leq \limsup_{i \in \omega} x_i$. $\square$

Thus, every sequence is convergent with respect to the lower quasi-metric. This merely reflects the fact that every net is convergent in the lower real line, because $-\infty$ is the least limit of any net. The above proposition shows that there is always a greatest limit too.

Given quasi-metrics on sets X and Y, we can define a quasi-metric on the product $X \times Y$ by

$$d(\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle) = \max(d(x_1, x_2), d(y_1, y_2)).$$

Then the metric on the product induces the product topology. In particular, the product of the lower and upper quasi-metrics induces a quasi-metric on the product space $\underline{\mathbb{R}} \times \overline{\mathbb{R}}$. By the above remark, this quasi-metric induces the Scott topology on $\underline{\mathbb{R}} \times \overline{\mathbb{R}}$. In fact, we have that

$$d(x, y) = \bigsqcup \{\epsilon > 0 \mid x_\epsilon \sqsubseteq y\},$$

where the partial number

$$x_\epsilon = \langle \underline{x} - \epsilon, \overline{x} + \epsilon \rangle$$

is the ϵ-***degradation*** of x. Hence the open balls are given by

$$B_\epsilon(x) = \Uparrow x_\epsilon.$$

This quasi-metric restricts to the subspaces $\mathcal{R}^\star$ and $\mathcal{R}$ of the quasi-metric space $\underline{\mathbb{R}} \times \overline{\mathbb{R}}$ (cf. Section 8.4, where $\mathcal{R}^\star$ and $\mathcal{R}$ are identified with subspaces of $\underline{\mathbb{R}} \times \overline{\mathbb{R}}$). This quasi-metric on $\mathcal{R}$ coincides with the quasi-metric defined by Acióly [Aci91] in a more direct way, and it is also induced by the quasi-metric defined by Edalat [Eda95e] on the upper space of $\mathbb{R}$. Under the identification of real numbers and maximal elements of $\mathcal{R}$, the restriction of the standard quasi-metric to the maximal elements is the usual metric on real numbers. Therefore, the limit of a Cauchy sequence of maximal elements w.r.t. the usual metric coincides with the greatest limit w.r.t. the quasi-metric.

The symmetrization of the induced quasi-metric on $\mathcal{R}$ is the Hausdorff metric

$$d(x, y) = \max(|\underline{y} - \underline{x}|, |\overline{y} - \overline{x}|).$$

and therefore it induces the Lawson topology (see Remark 8.1 and Section 8.9).

By the above proposition and its dual, a sequence x_i in $\mathcal{R}^\star$ converges to $y \in \mathcal{R}^\star$ iff

$$y \sqsubseteq \left[\liminf_{i \in \omega} \overline{x_i}, \limsup_{i \in \omega} \underline{x_i}\right].$$

Thus, every sequence x_i converges to a least limit $[-\infty, +\infty]$ and to a greatest limit $[\liminf_{i \in \omega} \overline{x_i}, \limsup_{i \in \omega} \underline{x_i}]$. Steve Vickers [Vic96] has defined a notion of "flat completion" of a quasi-metric space. He also showed (personal communication) that the completion of the rational intervals with respect to the quasi-metric defined above produces the partial real line.

The above metric on $\mathcal{R}_\perp$ can be extended to $\mathcal{R}_\perp$ by stipulating that

1. $d(\perp, x) = 0$
2. $d(x, \perp) = \infty$ if $x \neq \perp$.

This coincides with a definition via ϵ-degradations as above if we understand that

$$\perp_\epsilon = \perp.$$

Now we consider a recursive definition of a limiting operator

$$\lim : \mathcal{R}_\perp^\omega \to \mathcal{R}_\perp.$$

Definition 8.18 Let (X, d) be a quasi-metric space. A ***fast-converging Cauchy sequence*** in X is sequence x_i such that $d(x_m, x_n) < 2^{-n}$ for all $m \leq n$. □

There is nothing special about the particular sequence 2^{-n}. We could have used the sequence $1/(n + 1)$ (as it is done in Bishop and Bridges [BB85]), or any other simple sequence of rational numbers converging to zero at a known rate. Notice that a term of a fast-converging Cauchy sequence in $\mathcal{R}_\perp$ is bottom iff all terms are bottom.

Lemma 8.19 *If x_n is a fast-converging Cauchy sequence in $\mathcal{R}_\perp$, then its greatest limit is given by*

$$\bigsqcup_n^{\uparrow} [\underline{x_n} - 2^{-n}, \overline{x_n} + 2^{-n}].$$

Proof Immediate consequence of the above observations, noting that the interval $[\underline{x_n} - 2^{-n}, \overline{x_n} + 2^{-n}]$ is the 2^{-n}-degradation of x_n. □

Define a map $j : \mathbb{R}^3 \to \mathbb{R}$ by

$$j(x, y, z) = \max(x, \min(y, z)).$$

Then, given $p \leq q \in \mathbb{R}$, the map $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = j(p, x, q)$$

is idempotent,

$$p \leq f(x) \leq q,$$

and

$$f(x) = x \text{ iff } p \leq x \leq q.$$

Let $\hat{j} : \mathcal{R}_\perp^3 \to \mathcal{R}_\perp$ be the greatest extension of $j : \mathbb{R}^3 \to \mathbb{R}$ defined by the formula of Section 2.8:

$$\begin{aligned}
\hat{j}(x, y, z) &= \bigsqcup^{\uparrow}_{(x,y,x) \in V \in \Omega \mathcal{R}_\perp^3} \quad \bigsqcap_{(p,q,r) \in (s \times s \times s)^{-1}(V)} j(p, q, r) \\
&= \bigsqcup^{\uparrow}_{(a,b,c) \ll (x,y,x)} \quad \bigsqcap_{(p,q,r) \in a \times b \times c} j(p, q, r),
\end{aligned}$$

where $s : \mathbb{R} \to \mathcal{R}_\perp$ is the singleton embedding. Hence, if x and y are non-bottom intervals with $\underline{x} \leq \underline{y}$ and $\overline{x} \leq \overline{y}$, then

$$j(x, \perp, y) \quad = \quad [\underline{x}, \overline{y}].$$

Thus, j is non-strict.

Proposition 8.20 *Define a function $L : \mathcal{R}_\perp^\omega \to \mathcal{R}_\perp$ by*

$$L(x) = M(0, 1, x)$$

where $M : \omega \times \mathcal{R}_\perp \times \mathcal{R}_\perp^\omega \to \mathcal{R}_\perp$ is recursively defined by

$$L(n, \epsilon, x) = \hat{j}(x_n - \epsilon, M(n + 1, \epsilon/2, x), x_n + \epsilon).$$

If $x \in \mathcal{R}_\perp^\omega$ is a fast-converging Cauchy sequence, then $L(x)$ is its greatest limit.

Proof (Sketch) M is the least upper bound of the sequence of functions defined by

$$\begin{aligned}
M_0(n, \epsilon, x) &= \perp \\
M_{i+1}(n, \epsilon, x) &= \hat{j}(x_n - \epsilon, M_i(n + 1, \epsilon/2, x), x_n + \epsilon).
\end{aligned}$$

By induction,

$$M_i(n, \epsilon, x) = \bigsqcup_{n \leq k < n+i} [\underline{x} - \epsilon/2^n, \overline{x} + \epsilon/2^n].$$

Therefore the result follows from Lemma 8.19. □

Chapter 9

Partial real numbers considered as continuous words

This chapter is the basis of the operational semantics of the programming language Real PCF, introduced in Chapter 11 of Part IV. In Section 9.1 we briefly motivate and outline the programme of the following sections by considering discrete words first.

9.1 Discrete words

Consider a (deterministic) device that outputs symbols of a finite alphabet in sequence. We assume that the device can either produce output forever, or else stop producing output after some (possibly none) output has been produced. But we don't assume that the device gives an indication when it stops producing output. Rather, we assume that the typical reason why the device stops producing output, if it stops at all, is that its internal machinery loops forever, without being able to produce more symbols. Such a device may have been designed, for example, to print the decimal expansion of some real number, giving as many digits as we are patient to wait for.

If the device never stops, its potentially infinite output is represented by an infinite sequence of symbols. Otherwise, its finite output is represented by a finite sequence. The empty sequence represents the situation in which the device fails to produce any output. We refer to a finite or infinite sequence of symbols as a ***word***. Recall that a word x is a ***prefix*** of a word y if x is an initial subsequence of y. In this case, if x is infinite then it must be the same as y. If two devices X and Y as above produce outputs respectively x and y such that x is a prefix of y, this means Y produces more output than X. Since the set of words ordered by prefix forms a domain, we can use domain theory in order to account for the *external* behaviour of such devices.

Now suppose that we are interested in the *internal* behaviour of such a device, and consider the case when it is specified by a program in a functional programming language. Then there are two main questions: First, what are the primitive operations on words from which we can define more complex operations? Second, how is the functional program interpreted as an algorithm?

The answer to the first question is more or less simple. Let Σ be a finite alphabet, let Σ^∞ denote the domain of words ordered by prefix, and let σ range over Σ. The primitive operations that one finds in a typical functional programming language with features for infinite word processing are essentially the functions $\mathrm{cons}_\sigma, \mathrm{tail} : \Sigma^\infty \to \Sigma^\infty$ and $\mathrm{head} : \Sigma^\infty \to \Sigma_\perp$ defined by

$$\begin{aligned}
\mathrm{cons}_\sigma(x) &= \sigma x \\
\mathrm{tail}(\epsilon) &= \epsilon \\
\mathrm{tail}(\sigma x) &= x \\
\mathrm{head}(\epsilon) &= \perp \\
\mathrm{head}(\sigma x) &= \sigma.
\end{aligned}$$

By combining these operations with the aid of the conditional and recursion, one can define more complex operations.

In order to answer the second question, suppose that such a complex combination of primitive operations denoting the intended external behaviour of some device is given. As an example, we consider the (not very complex) combination $x = \mathrm{cons}_a(x)$, where a is some symbol. Since the only word x satisfying this equation is the infinite word $aaa\cdots$, the device is supposed to repeatedly output the symbol a forever. But how can the device work out the symbols to be printed from an expression as above, in a mechanical fashion? That is, what is the operational semantics of our (informally and imprecisely specified) language? A simple idea is the following. Given an expression e, the device tries to mechanically reduce it to an expression of the form $\mathrm{cons}_\sigma(e')$. Whatever e' is, the value of the expression $\mathrm{cons}_\sigma(e')$ is some word whose first symbol is σ. Therefore the device can output the symbol σ, and proceed to evaluate the expression e', repeatedly using the same principle. We can ignore the reduction rules for the conditional and recursion, because they are the same for any data type. The rules specific to our data type can be:

$$\begin{aligned}
\mathrm{tail}(\mathrm{cons}_\sigma(x)) &\to x \\
\mathrm{tail}(e) &\to \mathrm{tail}(e') \qquad \text{if } e \to e' \\
\mathrm{head}(\mathrm{cons}_\sigma(x)) &\to \sigma \\
\mathrm{head}(e) &\to \mathrm{head}(e') \qquad \text{if } e \to e'.
\end{aligned}$$

In the following sections we introduce similar primitive operations for the partial real line. As opposed to the case of words, it is perhaps not obvious how to combine these primitive operations with the aid of the conditional and recursion in order to obtain more complex operations. We develop some techniques to systematically derive recursive definitions in the next chapter.

9.2 The prefix preorder of a monoid

Recall that a monoid is a set together with a binary associative operation and a neutral element for this operation. It is customary to refer to this operation as multiplication, but in this work it is convenient to refer to it as ***concatenation***.

By a word over an alphabet Σ we mean either a finite word in Σ^* or an infinite word in Σ^ω. We denote the set of all words by Σ^∞. Usual concatenation of words, with the convention that $xy = x$ if x is infinite, makes Σ^∞ into a monoid with neutral element the empty word ε.

In any monoid $(M, \cdot, e)$ we can define a preorder, called its ***prefix preorder***, by $x \sqsubseteq z$ iff $xy = z$ for some y. In this case x is called a ***prefix*** of z and y is called a ***suffix***. This relation is reflexive because e is right neutral, and it is transitive because the concatenation operation is associative. It has e as its least element because e is left neutral. Monoid homomorphisms preserve the least element and the prefix preorder, by the very definition of monoid homomorphism. An element x is maximal iff it is ***left dominant***, in the sense that $xy = x$ for every y. The meet of a set, when it exists, is the greatest common prefix of the elements of the set.

The prefix preorder of the monoid Σ^∞ makes the set Σ^∞ into a Scott domain [Smy92b]. In particular, the prefix preorder is a partial order; that is, it is antisymmetric. An element of Σ^∞ is maximal iff it is an infinite word, and it is finite in the domain-theoretic sense iff it is a finite word. The concatenation operation seen as a function $\Sigma^\infty \times \Sigma^\infty \to \Sigma^\infty$ is not continuous, because it is not even monotone. But it is continuous in its second argument. This can be expressed by saying that left translations $x \mapsto ax$ are continuous for all words a.

9.3 Left translations of a monoid

We denote a left translation $x \mapsto ax$ of a monoid $(M, \cdot, e)$ by cons_a. Left translations are monotone and $\mathrm{cons}_a(M) = {\uparrow} a$, where ${\uparrow} a = \{x \in M | a \sqsubseteq x\}$. An element a is left-cancelable iff the translation cons_a is injective.

If x is left-cancelable and $x \sqsubseteq z$, we denote the unique y such that $xy = z$ by $z \backslash x$, so that $x(z \backslash x) = z$. The basic properties of this (partially defined) quotient operation are:

$$
\begin{array}{rcll}
x \backslash e & = & x & \text{for all } x, \\
x \backslash x & = & e & \text{for all left-cancelable } x, \\
(y \backslash x)(z \backslash y) & = & z \backslash x & \text{for all } z \text{ and all left-cancelable } x, y \text{ with } x \sqsubseteq y \sqsubseteq z, \\
(xy) \backslash z & = & (x \backslash z) y & \text{for all } x, y \text{ and all left-cancelable } z \text{ with } z \sqsubseteq x.
\end{array}
$$

Let a be a left-cancelable element of a monoid M. Then the corestriction of cons_a to its image is a bijection between the sets M and ${\uparrow} a$, with inverse $x \mapsto x \backslash a$. Therefore M is a preordered set isomorphic to the set ${\uparrow} a$ under the inherited order, because cons_a is monotone.

The left-cancelable elements of Σ^∞ are the finite words. It follows that ${\uparrow} a$ is a domain isomorphic to Σ^∞ for every finite word a. The subsection below shows that the partial unit interval has the same property for every non-maximal partial number a, for a suitable concatenation operation on partial numbers.

9.4 Concatenation of partial real numbers

Define a binary operation $(x, y) \mapsto xy$ on the *partial unit interval* by

$$xy = [(\overline{x} - \underline{x})\underline{y} + \underline{x},\ (\overline{x} - \underline{x})\overline{y} + \underline{x}].$$

That is, given $x, y \in \mathcal{I}$, rescale and translate the unit interval so that it becomes x, and define xy to be the interval which results from applying the same rescaling and translation to y. Then it is immediate that xy is a subinterval of x. The rescaling factor is the diameter of x, namely $\overline{x} - \underline{x}$, and the translation constant is the left end-point of x. If x is maximal, then its diameter is zero, so that $xy = x$. In order to simplify notation, we let κ_x stand for the diameter of x and μ_x stand for the left end-point of x, so that

$$x = [\mu_x, \mu_x + \kappa_x].$$

Thus, a left translation $\mathrm{cons}_a : \mathcal{I} \to \mathcal{I}$ is the canonical extension of the unique increasing affine map $[0, 1] \to [0, 1]$ with image a, namely

$$r \mapsto \kappa_a r + \mu_a : [0, 1] \to [0, 1].$$

Recall that an ***affine map*** is a function of the form $r \mapsto pr + q$. Hence, by Convention 8.2, we can write

$$xy = \kappa_x y + \mu_x.$$

Notice that in some cases there can be some ambiguity with the usual notation for multiplication. In these cases we will denote multiplication by $\times$.

Theorem 9.1 *$(\mathcal{I}, \cdot, \perp)$ is a monoid with the following properties:*

1. *Its prefix preorder coincides with the information order of the domain $\mathcal{I}$.*
2. *Its left-cancelable elements are the non-maximal ones.*
3. *Its left translations preserve all meets and all existing joins.*

Proof Let $x, y, z \in \mathcal{I}$. Then

$$xy = \kappa_x y + \mu_x = \kappa_x[\mu_y,\ \mu_y + \kappa_y] + \mu_x = [\kappa_x \mu_y + \mu_x,\ \kappa_x(\mu_y + \kappa_y) + \mu_x].$$

Hence $\mu_{xy} = \kappa_x \mu_y + \mu_x$ and $\kappa_{xy} = \kappa_x \kappa_y$. Therefore

$$(xy)z = \kappa_{xy} z + \mu_{xy} = \kappa_x \kappa_y z + \kappa_x \mu_y + \mu_x = \kappa_x(\kappa_y z + \mu_y) + \mu_x = x(yz).$$

Now, clearly $x\perp = x$, because by definition $\perp = [0, 1]$ is rescaled and translated so that it becomes x. Also, $\perp x = x$ because by definition $\perp = [0, 1]$ is rescaled and translated so that it becomes itself; hence $\perp x$ is the application of the identity to x. Therefore $(\mathcal{I}, \cdot, \perp)$ is a monoid.

(1) We have already seen that a maximal element is not left-cancelable. The proof of item (2) shows that if $x \sqsubseteq z$ in the information order and x is non-maximal then there is a (unique) y such that $xy = z$.

(2) A rescaling is reversible iff the rescaling factor is non-zero, a translation is always reversible, and an element has diameter zero iff it is maximal.

(3) The linear map $r \mapsto \kappa_a r + \mu_a : [0,1] \to [0,1]$ is either an increasing order isomorphisms between $[0,1]$ and its image (if $\kappa_x > 0$) or else constant map (if $\kappa_x = 0$), and the canonical extension of such a function is computed pointwise (cf. Section 8.3). $\square$

Remark 9.2 The fact that a left translation preserves all existing meets does not imply that it has a lower adjoint, which would preserve all joins and hence would be continuous, because $\mathcal{I}$ is not a complete lattice, as it lacks a top element [AJ94, GHK+80]. In fact, if cons_a had a lower adjoint, it would be given by $x \mapsto \bigsqcap \mathrm{cons}_a^{-1}(\uparrow x)$; but $\mathrm{cons}_a^{-1}(\uparrow x)$ is empty if x is apart from a, and in this case the meet would be the missing top element. In Section 9.7 we show that if a is non-maximal then cons_a has a continuous left inverse tail_a, however. $\square$

The above proof shows that the concatenation operation on partial numbers "multiplies diameters", in the sense that $\kappa_{xy} = \kappa_x \kappa_y$. Therefore $\kappa_{xy} \leq \kappa_x$ and $\kappa_{xy} \leq \kappa_y$, the equalities holding iff $x = \perp$ or $y = \perp$.

Concatenation of partial numbers has the following geometrical and computational interpretations. In a concatenation xy, the interval y refines the information given by x by selecting a subinterval of x. For example, the partial numbers $[0,1]$, $[0,\frac{1}{2}]$, $[\frac{1}{2},1]$, and $[\frac{1}{3},\frac{2}{3}]$ respectively select the whole interval, the first half, the second half, and the middle third part. Thus, concatenation of partial numbers allows for incremental computation on partial real numbers, also known as ***lazy evaluation*** (cf. Proposition 9.4).

There is yet another geometrical interpretation of concatenation, induced by the isomorphism between the partial unit interval and the unit triangle. The upper set of any non-maximal element x of the unit triangle is clearly a triangle, isomorphic to the unit triangle via a rescaling of the unit triangle followed by a translation. Thus, any element y can be interpreted either as an absolute address of a point in the unit triangle or else as a relative address of a point in the smaller triangle generated by x, namely the point xy, obtained by applying the same rescaling and translation to y.

We finish this section with the following lemma:

Lemma 9.3 *The bases of the partial unit interval consisting of respectively all non-maximal partial numbers and all non-maximal partial numbers with rational end-points are submonoids of $(\mathcal{I}, \cdot, \perp)$ closed under existing quotients, in the sense that if b and c are basis elements with $b \sqsubseteq c$ then $c \backslash b$ is a basis element too.*

Proof Existing quotients are given by

$$y \backslash x = (y - \mu_x)/\kappa_x = [(\underline{y} - \mu_x)/\kappa_x, (\overline{y} - \mu_x)/\kappa_x].$$

Therefore, if x and y have distinct (rational) end-points, so does $y \backslash x$. $\square$

The basis consisting of all non-maximal partial numbers with dyadic end-points is a submonoid, but it is not closed under existing quotients.

9.5 Infinitely iterated concatenations

Let M be a monoid with a partial prefix order and joins of increasing ω-chains, and let $\langle x_n \rangle_{n \geq 1}$ be a sequence of elements of M. Then we have that

$$x_1 \sqsubseteq x_1 x_2 \sqsubseteq \cdots \sqsubseteq x_1 x_2 \cdots x_n \sqsubseteq \cdots .$$

The ***infinitely iterated concatenation*** of $\langle x_n \rangle_{n \geq 1}$ is defined to be the join of these partial concatenations, informally denoted by $x_1 x_2 \cdots x_n \cdots$. It is also convenient to use the informal notation $y_1 \sqcup y_2 \sqcup \cdots \sqcup y_n \sqcup \cdots$ for the join of a chain $y_1 \sqsubseteq y_2 \sqsubseteq \cdots \sqsubseteq y_n \sqsubseteq \cdots$.

An ***interval expansion*** of a partial real number x is a sequence of intervals $\langle x_n \rangle_{n \geq 1}$ such that $x = x_1 x_2 \cdots x_n \cdots$. For example, interval expansions formed from the intervals

$$\left[0, \frac{1}{10}\right], \left[\frac{1}{10}, \frac{2}{10}\right], \ldots, \left[\frac{8}{10}, \frac{9}{10}\right], \left[\frac{9}{10}, 1\right]$$

are essentially *decimal expansions.*

If we think of an infinitely iterated concatenation as a computation, the following proposition shows that we can compute in an incremental fashion if left translations are continuous:

Proposition 9.4 (Infinite associativity) *Let M be a monoid with infinitely iterated concatenations. Then M satisfies the ω-associativity law*

$$x_1(x_2 \cdots x_n \cdots) \quad = \quad x_1 x_2 \cdots x_n \cdots$$

iff left translations preserve joins of increasing ω-chains.

Proof $(\Rightarrow)$ Assume that the ω-associativity law holds, let $\langle y_n \rangle_{n \geq 1}$ be an increasing ω-chain of elements of M, and let $\langle x_n \rangle_{n \geq 1}$ be a sequence of elements of M such that $y_n x_n = y_{n+1}$. Then the join of the chain is the same as the infinitely iterated concatenation of the sequence $\langle x_n \rangle_{n \geq 1}$ with y_1 added as a new first element, because $y_1 x_1 x_2 \cdots x_n = y_n$, as an inductive argument shows. Therefore

$$\begin{aligned}
a(y_1 \sqcup y_2 \sqcup \cdots \sqcup y_n \sqcup \cdots) &= a(y_1 x_1 x_2 \cdots x_n \cdots) \\
&= a y_1 x_1 x_2 \cdots x_n \cdots \\
&= a \sqcup a y_1 \sqcup a y_1 x_1 \sqcup \cdots \sqcup a y_1 x_1 \cdots x_n \sqcup \cdots \\
&= a \sqcup a y_1 \sqcup a y_2 \sqcup \cdots \sqcup a y_n \sqcup \cdots \\
&= a y_1 \sqcup a y_2 \sqcup \cdots \sqcup a y_n \sqcup \cdots .
\end{aligned}$$

($\Leftarrow$) Assume that left translations preserve joins of increasing ω-chains. Then we have that

$$\begin{aligned} x_1(x_2 \cdots x_n \cdots) &= x_1(x_2 \sqcup x_2x_3 \sqcup \cdots \sqcup x_2x_3 \cdots x_n \sqcup \cdots) \\ &= x_1x_2 \sqcup x_1x_2x_3 \sqcup \cdots \sqcup x_1x_2x_3 \cdots x_n \sqcup \cdots \\ &= x_1 \sqcup x_1x_2 \sqcup x_1x_2x_3 \sqcup \cdots \sqcup x_1x_2x_3 \cdots x_n \sqcup \cdots \\ &= x_1x_2 \cdots x_n \cdots . \square \end{aligned}$$

We denote the infinitely iterated concatenation $xx \cdots x \cdots$ of a constant sequence with range x by x^ω. An immediate consequence of the above proposition is that x^ω is the least fixed point of cons_x for any monoid with infinite concatenations.

Now let $y_1 \sqsubseteq y_2 \sqsubseteq \cdots \sqsubseteq y_n \sqsubseteq \cdots$ be a chain of left-cancelable elements. Then the sequence $y_1,\, y_2\backslash y_1,\, y_3\backslash y_2, \ldots,\, y_{n+1}\backslash y_n, \ldots$ has the elements of the chain as partial concatenations. Therefore the join of the chain is the same as the infinite concatenation of the induced sequence.

Proposition 9.5 *Consider the bases of $\mathcal{I}$ consisting of respectively all non-maximal partial numbers and all non-maximal partial numbers with rational end-points. Then there is a bijection between ω-*chains *of basis elements and* sequences *of basis elements, taking any chain to a sequence whose infinitely iterated concatenation is the join of the chain.* $\square$

Therefore we can replace ω-chains of basis elements by (arbitrary) sequences of basis elements and work with infinitely iterated concatenations instead of joins.

For monoids with infinitely iterated concatenations, it is natural to ask homomorphisms to preserve them.

Proposition 9.6 *A monoid homomorphism preserves infinitely iterated concatenations iff it preserves joins of increasing ω-chains*

Proof Let $h : L \to M$ be a monoid homomorphism between monoids L and M.
($\Rightarrow$) Let $\langle y_n \rangle_{n \geq 1}$ be an increasing ω-chain of elements of L having a join and let $\langle x_n \rangle_{n \geq 1}$ be a sequence of elements of L such that $y_n x_n = y_{n+1}$. Then we have that

$$\begin{aligned} & h(y_1 \sqcup \cdots \sqcup y_n \sqcup \cdots) \\ &= h(y_1x_1 \cdots x_n \cdots) \\ &= h(y_1)h(x_1) \cdots h(x_n) \cdots \\ &= h(y_1) \sqcup h(y_1)h(x_1) \sqcup \cdots \sqcup h(y_1)h(x_1) \cdots h(x_n) \sqcup \cdots \\ &= h(y_1) \sqcup \cdots \sqcup h(y_n) \sqcup \cdots . \end{aligned}$$

because $h(y_n)h(x_n) = h(y_{n+1})$.
($\Leftarrow$) Let $\langle x_i \rangle_{i \geq 1}$ be a sequence of elements of L having an infinitely iterated

concatenation. Then we have that

$$\begin{aligned} h\left(x_1 x_2 \cdots x_n \cdots\right) &= h\left(x_1 \sqcup x_1 x_2 \sqcup \cdots \sqcup x_1 x_2 \cdots x_n \sqcup \cdots\right) \\ &= h(x_1) \sqcup h(x_1 x_2) \sqcup \cdots \sqcup h(x_1 x_2 \cdots x_n) \sqcup \cdots \\ &= h(x_1) \sqcup h(x_1) h(x_2) \sqcup \cdots \sqcup h(x_1) h(x_2) \cdots h(x_n) \sqcup \cdots \\ &= h(x_1) h(x_2) \cdots h(x_n) \cdots . \square \end{aligned}$$

9.6 Partial real numbers considered as continuous words

Concatenation of partial numbers in the partial unit interval generalizes concatenation of words over any finite alphabet, in the following sense:

Proposition 9.7 *For every finite n-letter alphabet Σ, the monoid Σ^∞ is isomorphic to a finitely generated submonoid of $\mathcal{I}$. Moreover, the induced embedding of Σ^∞ into $\mathcal{I}$ is a continuous monoid homomorphism.*

Proof Without essential loss of generality, we prove the claim for the two-letter case $\Sigma = \{0, 2\}$. Let C be the submonoid of $\mathcal{I}$ finitely generated by the partial numbers $[0, 1/3]$ and $[2/3, 1]$. Then C clearly contains the partial numbers corresponding to the intervals successively produced in the construction of the Cantor set. Hence the joins of strictly increasing ω-chains of elements of C are the elements of the Cantor set. But these joins are the infinitely iterated concatenations of the generators. The set $\mathcal{C}$ of finite and infinite concatenations of the generators is also a submonoid of $\mathcal{I}$, which can be considered as the monoid of *partial Cantor numbers*. It is easy to see that $\{0, 2\}^\infty$ is isomorphic to $\mathcal{C}$. We know that $\{0, 2\}^*$ is the free monoid over $\{0, 2\}$. Hence there is a unique monoid homomorphism $h : \{0, 2\}^* \to \mathcal{C}$ such that $h(0) = [0, 1/3]$ and $h(2) = [2/3, 1]$. But h has a unique continuous extension to $\{0, 2\}^\infty$, because monoid homomorphisms are monotone and $\{0, 2\}^*$ consists of the finite elements (in the domain-theoretic sense) of $\{0, 2\}^\infty$. The resulting extension is a monoid homomorphism. Since it takes an infinite word x to the element of the Cantor set whose ternary expansion is x, it follows that h is a bijection, and therefore a monoid isomorphism.

(For general n, we repeat the above construction for n non-overlapping intervals of the same diameter.) $\square$

Therefore, the elements of the partial unit interval can be considered as "continuous words". For any word $x \in \Sigma^\infty$ let $\text{length}(x) \in [0, \infty]$ denote the length of x defined in the usual way. For any continuous word $x \in \mathcal{I}$ and any real number $b > 1$ we define $\text{length}_b(x) \in [0, \infty]$ to be $-\log_b(\kappa_x)$. Thus, $\text{length}_b(x) = \infty$ iff x is maximal, and $\text{length}_b(x) = 0$ iff x is bottom.

If a continuous word x is a partial realization of an unknown real number y, then $\text{length}_b(x)$ is the number of correct digits of an expansion to base b of y that x allows us to know. For example, if we consider $x = [0.143, 0.145]$ as a partial realization of an unknown real number y then $\text{length}_{10}(x)$ is roughly 2.7, meaning that x allows us to know two decimal digits of y.

Although it is possible to show that Proposition 9.7 holds even for countably infinite alphabets (by considering countably many non-overlapping intervals), the following proposition holds only for the finite case.

Proposition 9.8 *Consider $[0, \infty]$ as a monoid under addition and as a domain under its prefix preorder (which coincides with its natural order).*

1. *For every real number $b > 1$, $\mathrm{length}_b : \mathcal{I} \to [0, \infty]$ is a continuous monoid homomorphism.*
2. *Let h be the embedding of the monoid of words over a finite n-letter alphabet into $\mathcal{I}$ defined in Proposition 9.7. Then $\mathrm{length} = \mathrm{length}_{n+1} \circ h$.*

Proof Routine verification. The first part follows from the fact that the concatenation operation multiplies diameters and that logarithms take multiplication to addition. □

Corollary 9.9 *The infinitely iterated concatenation of a sequence of continuous words is maximal iff the sum of the lengths of the continuous words is ∞.*

Proof The length function is a continuous monoid homomorphism and a partial number has infinite length iff it is maximal. □

The following proposition allows us to prove some properties of real number programs. We say that a map $f : \mathcal{I} \to \mathcal{I}$ is ***guarded*** if there is a real number $\delta > 0$ such that $\mathrm{length}(f(x)) \geq \mathrm{length}(x) + \delta$, called a ***guarding constant*** for f. Clearly, left translations cons_a with $a \neq \perp$ are guarded, with guarding constant $\mathrm{length}(a)$.

Proposition 9.10 *Any continuous guarded map $f : \mathcal{I} \to \mathcal{I}$ has a maximal partial number as its unique fixed point.*

Proof Since $\mathrm{length}(f^n(\perp)) \geq n\delta$ for every n and length is a continuous homomorphism, $\mathrm{length}\left(\bigsqcup_n f^n(\perp)\right) \geq \sup_n n\delta = \infty$. This means that the least fixed point of f is maximal. Therefore it is the unique fixed point of f. □

9.7 Heads and tails of continuous words

The tail map on discrete words removes a prefix of length 1 from its argument, if such a prefix exists. The length 1 plays no distinguished rôle in continuous words, and, moreover, length is not an absolute notion in the case of continuous words as it depends on the choice of a base. Thus, in order to obtain an analogue of the tail map for continuous words, we consider other properties of the tail map on discrete words.

The main property of the tail map on discrete words is that

$$\mathrm{tail}(\mathrm{cons}_\sigma(x)) = x.$$

In fact, it gives rise to the computation rule for tail discussed in Section 9.1. In other words, this property says that tail is a left inverse of cons_σ (for every symbol σ). However, it is not hard to see that there is no continuous map tail on continuous words which is a left inverse of cons_a for every continuous word a. Nevertheless, it will be enough to consider a continuous left inverse tail_a of cons_a for each left-cancelable continuous word a. That is, we consider a map tail_a which removes the prefix a of its argument, if such a prefix exists:

$$\mathrm{tail}_a(ax) = x.$$

This specification fails to uniquely characterize tail_a, because it doesn't say what $\mathrm{tail}_a(x)$ should be if x doesn't have a as a prefix. A naïve attempt would be to let $\mathrm{tail}_a(x)$ be bottom in this case. But then tail_a wouldn't be continuous, because we would have that $\mathrm{tail}_a(\twoheaddownarrow a) = \{\bot\}$ as $a \not\sqsubseteq x$ for all $x \ll a$ with $x \neq \bot$.

Since the equation $\mathrm{tail}_a(ax) = x$ means that $\mathrm{tail}_a \circ \mathrm{cons}_a = \mathrm{id}$, we see that tail_a is a retraction of cons_a.

Proposition 9.11 *Let $a \in \mathcal{I}$ be left-cancelable. The retractions of cons_a are the maps tail_a of the form*

$$\mathrm{tail}_a(x) = e_a(x) \backslash a,$$

for $e_a : \mathcal{I} \to \mathcal{I}$ a continuous idempotent with image $\uparrow a$.

Proof Lemmas 9.12 and 9.13 below. □

Lemma 9.12

1. *Let $f : D \to E$ be a continuous map such that its corestriction $f^\circ : D \to f(D)$ is an isomorphism. If the inclusion $f_\circ : f(D) \to E$ has a retraction $r : E \to f(D)$ then f has a retraction $g \stackrel{\mathrm{def}}{=} (f^\circ)^{-1} \circ r$, as illustrated in the following diagram:*

$$D \underset{f}{\overset{g}{\leftrightarrows}} E = D \underset{f_\circ}{\overset{(f^\circ)^{-1}}{\leftrightarrows}} f(D) \underset{f^\circ}{\overset{r}{\leftrightarrows}} E$$

2. *Any retraction $r : E \to D$ of a continuous map $s : D \to E$ is uniquely determined by its induced idempotent $s \circ r$.*

Proof (1) This follows from the fact that for all arrows $X \underset{r}{\overset{s}{\leftrightarrows}} Y \underset{r'}{\overset{s'}{\leftrightarrows}} Z$ of any category, if r, r' are retractions of s, s' respectively then $r' \circ r$ is a retraction of $s \circ s'$.

(2) Assume that $r' : E \to D$ is a retraction of s with the same induced idempotent, i.e., $s \circ r' = s \circ r$. Since $r \circ s = \mathrm{id}$, we conclude that $r \circ s \circ r' = r \circ s \circ r$ and hence $r' = r$. □

Lemma 9.13 *Let $a \in \mathcal{I}$ be left-cancelable.*

1. *If tail_a is a retraction of cons_a then the induced idempotent $\mathrm{cons}_a \circ \mathrm{tail}_a$ has image $\uparrow a$.*

2. *Conversely, for every idempotent $e_a : \mathcal{I} \to \mathcal{I}$ with image $\uparrow a$, there is a unique retraction* tail_a *of* cons_a *with* $e_a = \mathrm{cons}_a \circ \mathrm{tail}_a$*, given by*

$$\mathrm{tail}_a(x) = e_a(x)\backslash a$$

Proof (1) $a \sqsubseteq \mathrm{cons}_a(\mathrm{tail}_a(x))$ by Theorem 9.1, which shows that the image is contained in $\uparrow a$. If $a \sqsubseteq x$ then $x = a(x\backslash a)$ and $\mathrm{cons}_a(\mathrm{tail}_a(a(x\backslash a))) = \mathrm{cons}_a(x\backslash a) = x$, which establishes the inclusion in the other direction.

(2) $x \mapsto x\backslash a : \uparrow a \to \mathcal{I}$ is the inverse of cons_a corestricted to its image. Hence tail_a is a retraction of cons_a by Lemma 9.12(1), because every idempotent factors through its image as a retraction followed by a section. Uniqueness follows from Lemma 9.12(2). □

Recall that $\mathrm{join}_a : \mathcal{I} \to \mathcal{I}$ constructed in Proposition 8.11 is an idempotent with image $\uparrow a$ by virtue of Proposition 7.10.

Definition 9.14 For every left cancelable $a \in \mathcal{I}$, we denote by $\mathrm{tail}_a : \mathcal{I} \to \mathcal{I}$ the unique continuous map such that

$$\mathrm{tail}_a \circ \mathrm{cons}_a = \mathrm{id} \qquad \text{and} \qquad \mathrm{cons}_a \circ \mathrm{tail}_a = \mathrm{join}_a.$$

That is,

$$\mathrm{tail}_a(x) = \mathrm{join}_a(x)\backslash a.$$

□

By Proposition 8.11, using the fact that

$$(\max(\underline{a}, \min(r, \overline{a})) - \mu_a)/\kappa_a = \max(0, \min((r - \mu_a)/\kappa_a, 1)),$$

we see that the map tail_a is the canonical extension of the map

$$r \mapsto \max(0, \min((r - \mu_a)/\kappa_a, 1)).$$

Remark 9.15 If in the case of discrete words we define $\mathrm{tail}_a(x) = \mathrm{tail}^n(x)$, where n is the length of the finite word a, then tail_a is a retraction of cons_a and the induced idempotent is the unique joining map of a. □

Lemma 9.16 tail_a *is a multiplicative function.*

Proof This follows from the fact the tail_a is the canonical extension of an increasing continuous function, and any such canonical extension preserves all meets. □

Let $h : \Sigma^\infty \to \mathcal{I}$ be the submonoid embedding defined in Proposition 9.7, for the two-letter alphabet $\Sigma = \{0, 2\}$. If we identify 0 with **tt** and 2 with **ff** so that $\Sigma_\perp = \mathcal{B}$, then for every $x \in \Sigma^\infty$ we have that

$$\mathrm{head}(x) = (h(x) <_\perp 1/2).$$

We are thus tempted to define $\mathrm{head} : \mathcal{I} \to \mathcal{B}$ by $\mathrm{head}(x) = (x <_\perp 1/2)$. Since the number $1/2$ is an accidental feature of our construction of h, and since we don't want to commit ourselves to this arbitrary choice at this stage, we instead define $\mathrm{head}_r : \mathcal{I} \to \mathcal{B}$ for each real number $r \in [0, 1]$ by

$$\mathrm{head}_r(x) = (x <_\perp r).$$

Lemma 9.17 head_r *is a multiplicative function.*

Proof Immediate consequence of Lemma 8.10. □

In the next section we show that the generalized versions of the primitive operations on discrete words have simple computation rules. In the next chapter we show how to combine them by means of the (parallel) conditional and recursion in order to obtain more complex operations.

9.8 Computation rules for continuous words

The following lemma shows how to reduce expressions e denoting non-bottom partial numbers to expressions of the form $\mathrm{cons}_a(e')$ with $a \neq \bot$. Such an expression is called a ***head-normal form***. The idea is that if an expression e has a head-normal form $\mathrm{cons}_a(e')$ then we know that its value is contained in the interval a. Thus, a head-normal form is a ***partially evaluated*** expression. A better partial evaluation of e is obtained by partially evaluating e', obtaining a head-normal form $\mathrm{cons}_b(e'')$, and applying rule (1) below to $\mathrm{cons}_a(\mathrm{cons}_b(e''))$ in order to obtain the more informative head-normal form $\mathrm{cons}_{ab}(e'')$ of e, and so on.

Lemma 9.18 *For all non-maximal $a, b \in \mathcal{I}$ and all $x \in \mathcal{I}$,*

1. $\mathrm{cons}_a(\mathrm{cons}_b(x)) = \mathrm{cons}_{ab}(x)$
2. $\mathrm{tail}_a(\mathrm{cons}_b(x)) = 0$ *if* $b \leq a$
3. $\mathrm{tail}_a(\mathrm{cons}_b(x)) = 1$ *if* $b \geq a$
4. $\mathrm{tail}_a(\mathrm{cons}_b(x)) = \mathrm{cons}_{b\backslash a}(x)$ *if* $a \sqsubseteq b$
5. $\mathrm{tail}_a(\mathrm{cons}_b(x)) = \mathrm{cons}_{(a\sqcup b)\backslash a}(\mathrm{tail}_{(a\sqcup b)\backslash b}(x))$ *if* $a\uparrow b$
6. $\mathrm{head}_r(\mathrm{cons}_a(x)) = \mathbf{tt}$ *if* $a < r$
7. $\mathrm{head}_r(\mathrm{cons}_a(x)) = \mathbf{ff}$ *if* $a > r$
8. $\mathrm{pif}\ \mathbf{tt}\ \mathrm{then}\ x\ \mathrm{else}\ y = x$
9. $\mathrm{pif}\ \mathbf{ff}\ \mathrm{then}\ x\ \mathrm{else}\ y = y$
10. $$\begin{aligned}\mathrm{pif}\ p\ \mathrm{then}\ \mathrm{cons}_a(x)\ \mathrm{else}\ \mathrm{cons}_b(y) =\ & \\ \mathrm{cons}_{a\sqcap b}(\mathrm{pif}\ p\ \ \mathrm{then}\ \ & \mathrm{cons}_{a\backslash(a\sqcap b)}(x) \\ \mathrm{else}\ \ & \mathrm{cons}_{b\backslash(a\sqcap b)}(y)).\end{aligned}$$

Proof (1) This is the associativity law expressed in terms of left translations.
(2)–(3) Immediate

(4) This is a special case of (5) below. But it is also equivalent to the fact that $(bx)\backslash a = (b\backslash a)x$.

(5) For all $c\uparrow x$, $\mathrm{cons}_b(\mathrm{join}_c(x)) = \mathrm{cons}_b(c \sqcup x) = \mathrm{cons}_b(c) \sqcup \mathrm{cons}_b(x) = \mathrm{join}_{bc}(\mathrm{cons}_b(x))$, because left-translations preserve existing joins. Hence

$$\begin{aligned}\mathrm{tail}_a(\mathrm{cons}_b(x)) &= \mathrm{join}_a(\mathrm{cons}_b(x))\backslash a = \mathrm{join}_{a\sqcup b}(\mathrm{cons}_b(x))\backslash a \\ &= \mathrm{join}_{b((a\sqcup b)\backslash b)}(\mathrm{cons}_b(x))\backslash a = \mathrm{cons}_b(\mathrm{join}_{(a\sqcup b)\backslash b}(x))\backslash a \\ &= \mathrm{cons}_b(\mathrm{cons}_{(a\sqcup b)\backslash b}(\mathrm{tail}_{(a\sqcup b)\backslash b}(x)))\backslash a \\ &= \mathrm{cons}_{a\sqcup b}(\mathrm{tail}_{(a\sqcup b)\backslash b}(x))\backslash a = \mathrm{cons}_{(a\sqcup b)\backslash a}(\mathrm{tail}_{(a\sqcup b)\backslash b}(x)).\end{aligned}$$

The second step follows from the fact that $\mathrm{join}_a(y) = \mathrm{join}_{a \sqcup b}(y)$ if $a \uparrow b$ and $b \sqsubseteq y$. The last step follows from the fact that $(cy)\backslash a = (c\backslash a)y$.

(6–9) Immediate. (10) It suffices to show that

$$\mathrm{cons}_a(x) \sqcap \mathrm{cons}_b(y) \quad = \quad \mathrm{cons}_{a \sqcap b}(\mathrm{cons}_{a\backslash(a \sqcap b)}(x) \sqcap \mathrm{cons}_{b\backslash(a\sqcap b)}(y)).$$

Since left translations preserve meets and $a = c(a\backslash c)$ for all $c \sqsubseteq a$, the result follows by taking $c = a \sqcap b$. □

9.9 The partial unit interval acting on the partial real line

In this section we extend the results of the previous subsections from the partial unit interval to the partial real line.

The concatenation operation

$$xy \quad = \quad \kappa_x y + \mu_x,$$

originally defined for x and y in the partial unit interval, makes sense for x and y ranging over the partial real line. With this extension, $\mathcal{R}$ becomes a monoid too. But its prefix preorder does not coincide with its information order, and it is due to this reason that we initially restricted ourselves to the partial unit interval. In fact, if $y = ax$ this does not mean that y is contained in a. However, if we know that x is in the partial unit interval then $y = ax$ does imply that y is contained in a, even if y is not in the partial unit interval. This is the content of the following theorem.

Theorem 9.19 *The map $(x, y) \mapsto xy : \mathcal{R} \times \mathcal{I} \to \mathcal{R}$ is a (right) action of the monoid $(\mathcal{I}, \cdot, \perp)$ on the monoid $(\mathcal{R}, \cdot, [0, 1])$, inducing the information order of $\mathcal{R}$:*

1. *For all $x \in \mathcal{R}$ and all $y, z \in \mathcal{I}$, $\quad x\perp = x \quad$ and $\quad (xy)z = x(yz)$.*
2. *For all $x, z \in \mathcal{R}$, $\quad x \sqsubseteq z$ iff $xy = z$ for some $y \in \mathcal{I}$, such a y being unique iff x is non-maximal.*

Moreover, for all $a \in \mathcal{R}$, the map $\mathrm{ricons}_a : \mathcal{I} \to \mathcal{R}_\perp$ defined by $\mathrm{ricons}_a(x) = ax$ preserves all meets and all existing joins. □

Given $x, z \in \mathcal{R}$ with $x \sqsubseteq z$ and x non-maximal, denote the unique $y \in \mathcal{I}$ such that $xy = z$ by $z\backslash x$.

For each non-maximal $a \in \mathcal{R}$, define a strict continuous map $\mathrm{irtail}_a : \mathcal{R}_\perp \to \mathcal{I}$ as in Definition 9.14 so that for all non-maximal $a \in \mathcal{R}$, the maps ricons_a and irtail_a form a section-retraction pair, with join_a as the induced idempotent, and

$$\mathrm{irtail}_a(x) = x\backslash a \text{ if } a \sqsubseteq x.$$

Finally, for each non-maximal $a \in \mathcal{R}$ define a strict continuous map $\mathrm{rrcons}_a : \mathcal{R}_\perp \to \mathcal{R}_\perp$ by

$$\mathrm{rrcons}_a(x) \quad = \quad ax.$$

Thus, $\mathcal{I}$ is a retract of $\mathcal{R}_\perp$, in many ways. In particular,

1. $\mathrm{ricons}_{[0,1]} : \mathcal{I} \to \mathcal{R}_\perp$ is the inclusion,
2. $\mathrm{irtail}_{[0,1]} : \mathcal{R}_\perp \to \mathcal{I}$ is a "truncated projection", and
3. $\mathrm{join}_{[0,1]} : \mathcal{R}_\perp \to \mathcal{R}_\perp$ is the coextension of the truncated projection.

The head-normal forms for $\mathcal{R}_\perp$ are taken as the expressions of the form $\mathrm{ricons}_a(e)$.

Lemma 9.20 *For all non-maximal $a \in \mathcal{R}$ and $b \in \mathcal{I}$, and all $x \in \mathcal{I}$,*

1. *(a)* $\mathrm{rrcons}_a(\mathrm{rrcons}_b(x)) = \mathrm{rrcons}_{ab}(x)$
   (b) $\mathrm{rrcons}_a(\mathrm{ricons}_b(x)) = \mathrm{ricons}_{ab}(x)$
   (c) $\mathrm{ricons}_a(\mathrm{cons}_b(x)) = \mathrm{ricons}_{ab}(x)$
2. $\mathrm{irtail}_a(\mathrm{ricons}_b(x)) = 0$ *if* $b \leq a$
3. $\mathrm{irtail}_a(\mathrm{ricons}_b(x)) = 1$ *if* $b \geq a$
4. $\mathrm{irtail}_a(\mathrm{ricons}_b(x)) = \mathrm{cons}_{b \backslash a}(x)$ *if* $a \sqsubseteq b$
5. $\mathrm{irtail}_a(\mathrm{ricons}_b(x)) = \mathrm{cons}_{(a \sqcup b) \backslash a}(\mathrm{tail}_{(a \sqcup b) \backslash b}(x))$ *if* $a \uparrow b$
6. $\mathrm{rhead}_r(\mathrm{ricons}_a(x)) = \mathbf{tt}$ *if* $a < r$
7. $\mathrm{rhead}_r(\mathrm{ricons}_a(x)) = \mathbf{ff}$ *if* $a > r$
8. $\mathrm{pif}\ \mathbf{tt}\ \mathrm{then}\ x\ \mathrm{else}\ y = x$
9. $\mathrm{pif}\ \mathbf{ff}\ \mathrm{then}\ x\ \mathrm{else}\ y = y$
10. $$\begin{aligned} &\mathrm{pif}\ p\ \mathrm{then}\ \mathrm{ricons}_a(x)\ \mathrm{else}\ \mathrm{ricons}_b(y) = \\ &\quad \mathrm{ricons}_{a \sqcap b}(\mathrm{pif}\ p\ \ \mathrm{then}\ \ \mathrm{ricons}_{a \backslash (a \sqcap b)}(x) \\ &\qquad\qquad\qquad\quad \mathrm{else}\ \ \mathrm{ricons}_{b \backslash (a \sqcap b)}(y)), \end{aligned}$$

where the parallel conditional, the comparison map, and the head map for $\mathcal{R}_\perp$ are defined in the same way as for $\mathcal{I}$. Notice that the above lemma is Lemma 9.18 with "r" and "i" inserted in appropriate places.

Chapter 10

Induction and recursion on the partial real line

At this point, it is possible to proceed directly to Chapter 11 of Part IV, which introduces Real PCF. This chapter has Chapter 5 of Part II as a prerequisite. It introduces recursion on the real line, a technique which can be used to derive Real PCF programs. This technique is used in Chapter 12 of Part IV in order to respectively establish existence of a unique sound effective presentation of the partial real line, up to equivalence, and computational completeness of Real PCF (cf. Section 1.1). Several examples of recursive definitions given in the present chapter immediately give rise to Real PCF programs for computing the recursively defined entities. Thus, this chapter is implicitly about programming in Real PCF.

In Section 10.1 we characterize the partial unit interval by axioms similar to the so-called Peano axioms for natural numbers. In Section 10.2 we introduce structural recursion for the partial unit interval. In Section 10.3 we generalize binary expansions of real numbers to what we call *bifurcated binary expansions* of partial real numbers; this material is needed in Chapter 12 of Part IV. In Section 10.4 we briefly discuss coinduction on the partial unit interval. In Section 10.5 we briefly introduce structural recursion on the partial real line. Finally, in Section 10.6 we give some examples of recursive definitions (in addition to those given in Section 10.3).

10.1 Peano-like axioms for the partial unit interval

Recall that $\mathrm{cons}_L, \mathrm{cons}_R : \mathcal{I} \to \mathcal{I}$ are defined by

$$\mathrm{cons}_L(x) = x/2 \qquad \mathrm{cons}_R(x) = (x+1)/2.$$

and that these are the unique increasing affine maps such that

$$\mathrm{cons}_L(\perp) = L \qquad \mathrm{cons}_R(\perp) = R,$$

where

$$L = [0, 1/2] \qquad R = [1/2, 1].$$

In this section we consider cons_L and cons_R as partial real number "constructors", in a similar fashion as zero and successor are considered as natural number constructors. A main difference is that the natural numbers together with zero and successor form a free algebra, whereas $\mathcal{I}$ together with cons_L and cons_R will form a kind of quotient of a bifree algebra with respect to some equations. The main such equation is

$$\mathrm{cons}_L(1) = \mathrm{cons}_R(0),$$

which simply means that $1/2 = (0+1)/2$.

Another main difference is that natural numbers are constructed from zero by finitely many applications of the successor function, whereas elements of $\mathcal{I}$ are constructed "from nothing" by infinitely many applications of cons_L and cons_R. This observation will be made precise in the development that follows. For the moment, we remark that 0 and 1 are constructed by infinitely many applications of cons_L and cons_R in the sense that:

$$0 = \bigsqcup_n{}^{\uparrow} \mathrm{cons}_L^n(\bot) \quad \text{and} \quad 1 = \bigsqcup_n{}^{\uparrow} \mathrm{cons}_R^n(\bot).$$

In fact, 0 and 1 are the unique fixed points of cons_L and cons_R respectively: $\mathrm{cons}_L(x) = x$ iff $x/2 = x$ iff $x = 0$, and $\mathrm{cons}_R(y) = y$ iff $(y+1)/2 = y$ iff $y = 1$. More generally, every *total* $x \in \mathcal{I}$ can be written in the form

$$x = \bigsqcup_n{}^{\uparrow} \mathrm{cons}_{a_1} \circ \cdots \circ \mathrm{cons}_{a_n}(\bot)$$

for some sequence $a_i \in \{L, R\}$, corresponding to a binary expansion of x. Partial real numbers are obtained by iterating cons_L and cons_R in a more complicated way (see Section 10.3 below).

Pursuing our analogy with natural numbers, we now observe that every natural number is either zero or else the successor of a unique number.

Lemma 10.1 *For every $x \in \mathcal{I}$,*

1. *if $x \leq 1/2$ then $x = \mathrm{cons}_L(y)$ for a unique y,*
2. *if $x \geq 1/2$ then $x = \mathrm{cons}_R(z)$ for a unique z,*
3. *if $x \uparrow 1/2$ then $x = \mathrm{cons}_L(y) \sqcap \mathrm{cons}_R(z)$ for unique $y \sqsubseteq 1$ and $z \sqsubseteq 0$.*

Notice that $x \uparrow 1/2$ iff $x \sqsubseteq 1/2$, because $1/2$ is maximal.

Proof (1): cons_L is bijective onto its image. Hence there is at most one y with $x = \mathrm{cons}_L(y)$. The image of cons_L is $\uparrow L$ by definition. But $x \leq 1/2$ iff $L \sqsubseteq x$. (2): Similar. (3): In this case $1/2 \in x$. Hence $x = [\underline{x}, 1/2] \sqcap [1/2, \overline{x}] = \mathrm{cons}_L([2\underline{x}, 1]) \sqcap \mathrm{cons}_L([0, 2\overline{x} - 1])$. $\square$

The fact that every number is either zero or a successor can be expressed by the equation

$$n = \text{if } n = 0 \text{ then } 0 \text{ else } \mathrm{succ}(\mathrm{pred}(n)).$$

where

$$\mathrm{pred}(0) = 0 \quad \text{(say)} \qquad \text{and} \qquad \mathrm{pred}(\mathrm{succ}(n)) = n.$$

Recall that in Section 9.7 we defined maps $\mathrm{tail}_a : \mathcal{I} \to \mathcal{I}$ for $a \in \mathcal{I}$ non-maximal, and maps $\mathrm{head}_r : \mathcal{I} \to \mathcal{B}$ for $r \in [0,1]$. For notational simplicity we write

$$\mathrm{head} = \mathrm{head}_{1/2}.$$

Lemma 10.2

$$\begin{array}{rclrcl}
\mathrm{tail}_L(\mathrm{cons}_L(x)) & = & x & \mathrm{tail}_R(\mathrm{cons}_L(x)) & = & 0 \\
\mathrm{tail}_L(\mathrm{cons}_R(y)) & = & 1 & \mathrm{tail}_R(\mathrm{cons}_R(y)) & = & y \\
\mathrm{tail}_L(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) & = & x \sqcap 1 & \mathrm{tail}_R(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) & = & 0 \sqcap y,
\end{array}$$

$$\begin{array}{rclrcll}
\mathrm{head} \circ \mathrm{cons}_L(x) & \sqsubseteq & \mathbf{tt} & \mathrm{head} \circ \mathrm{cons}_L(x) & = & \perp & \textit{iff} \quad x \sqsubseteq 1 \\
\mathrm{head} \circ \mathrm{cons}_R(y) & \sqsubseteq & \mathbf{ff} & \mathrm{head} \circ \mathrm{cons}_R(y) & = & \perp & \textit{iff} \quad y \sqsubseteq 0
\end{array}$$

$$x = \mathrm{pif}\ \mathrm{head}(x) \text{ then } \mathrm{cons}_L(\mathrm{tail}_L(x)) \text{ else } \mathrm{cons}_R(\mathrm{tail}_R(x)).$$

Proof Routine verification, included for the sake of completeness. The equation $\mathrm{tail}_a \circ \mathrm{cons}_a = \mathrm{id}$ holds by construction. By definition,

$$\mathrm{tail}_a(x) = \mathrm{join}_a(x) \backslash a = \max(0, \min((x - \mu_a)/\kappa_a, 1)) = \max(0, \min((x - \underline{a})/(\overline{a} - \underline{a}), 1)).$$

Hence

$$\begin{array}{rcl}
\mathrm{tail}_L(x) & = & \max(0, \min((x-0)/(1/2), 1)) = \max(0, \min(2x, 1)) = \min(2x, 1), \\
\mathrm{tail}_R(x) & = & \max(0, \min((x-1/2)/(1/2), 1)) = \max(0, \min(2x-1, 1)) = \max(0, 2x-1),
\end{array}$$

because x lies in the unit interval. Hence

$$\begin{array}{l}
\mathrm{tail}_L(\mathrm{cons}_R(y)) = \min(2((y+1)/2), 1) = \min(y+1, 1) = 1 \\
\mathrm{tail}_R(\mathrm{cons}_L(x)) = \max(0, 2(x/2) - 1) = \max(0, x-1) = 0.
\end{array}$$

Since for all $p, q \in [0,1]$ we have that $p/2 \leq (q+1)/2$, as $p/2 \in L$ and $(q+1)/2 \in R$, it follows that

$$\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y) = [\underline{x}/2, \overline{x}/2] \sqcap [(\underline{y}+1)/2, (\overline{y}+1)/2] = [\underline{x}/2, (\overline{y}+1)/2].$$

Hence

$$\mathrm{tail}_L(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) = \mathrm{tail}_L([\underline{x}/2, (\overline{y}+1)/2]) \;=\; [\underline{x}, 1] = x \sqcap 1.$$

Similarly, $\mathrm{tail}_R(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) = 0 \sqcap y$. The statements about head follow from the fact that $\mathrm{head}(x) = (x <_\perp 1/2)$ by definition, $\mathrm{cons}_L(x) \leq 1/2$, and $\mathrm{cons}_L(x) < 1/2$ iff $x/2 < 1/2$ iff $x < 1$ iff $x \not\sqsubseteq 1$. Similarly, $\mathrm{cons}_R(x) \geq 1/2$, and $\mathrm{cons}_R(y) < 1/2$ iff $y \not\sqsubseteq 0$. For the last equation, only the case $\mathrm{head}(x) = \perp$

is not immediate. In this case $x\uparrow 1/2$, which is equivalent to $x \sqsubseteq 1/2$ as $1/2$ is maximal and means that $1/2 \in x$. Hence

$$\begin{aligned}
x &= [\underline{x}, 1/2] \sqcap [1/2, \overline{x}] \\
&= ([0, 1/2] \sqcup [\underline{x}, \overline{x}]) \sqcap ([1/2, 1] \sqcup [\underline{x}, \overline{x}]) \\
&= \mathrm{join}_L(x) \sqcap \mathrm{join}_R(x) \\
&= \mathrm{cons}_L(\mathrm{join}_L(x) \backslash L) \sqcap \mathrm{cons}_R(\mathrm{join}_R(x) \backslash R) \\
&= \mathrm{cons}_L(\mathrm{tail}_L(x)) \sqcap \mathrm{cons}_R(\mathrm{tail}_R(x)) \\
&= \mathrm{pif\ head}(x) \mathrm{\ then\ cons}_L(\mathrm{tail}_L(x)) \mathrm{\ else\ cons}_R(\mathrm{tail}_R(x)). \square
\end{aligned}$$

Given a set X, an element $x \in X$, and a function $g : \mathbb{N} \to X$, there is a unique function $f : \mathbb{N} \to X$ such that $f(0) = x$ and $f(n+1) = g(n)$. A similar fact holds for $\mathcal{I}$ equipped with cons_L and cons_R, but we have to take into account the equation $\mathrm{cons}_L(0) = \mathrm{cons}_R(1)$.

Lemma 10.3 (Definition by cases) *Let D be a bounded complete domain and $g_L, g_R : \mathcal{I} \to D$ be continuous maps such that*

$$g_L(1) = g_R(0).$$

Then there is a unique continuous map $f : \mathcal{I} \to D$ such that

$$\begin{aligned}
f(\mathrm{cons}_L(x)) &= g_L(x) \\
f(\mathrm{cons}_R(y)) &= g_R(y) \\
f(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) &= g_L(x) \sqcap g_L(y) \qquad \text{for } x \sqsubseteq 1 \text{ and } y \sqsubseteq 0,
\end{aligned}$$

namely the function f defined by

$$f(x) = \mathrm{pif\ head}(x) \mathrm{\ then\ } g_L(\mathrm{tail}_L(x)) \mathrm{\ else\ } g_R(\mathrm{tail}_R(x)).$$

Proof It is clear from Lemma 10.1 that there is at most one such function. We show that f as defined above is such a function. If $x \not\sqsubseteq 1$ then $\mathrm{head}(\mathrm{cons}_L(x)) = \mathbf{tt}$ and

$$f(\mathrm{cons}_L(x)) = g_L(\mathrm{tail}_L(\mathrm{cons}_L(x))) = g_L(x).$$

Otherwise $\mathrm{head}(\mathrm{cons}_L(x)) = \bot$ and

$$\begin{aligned}
f(\mathrm{cons}_L(x)) &= g_L(\mathrm{tail}_L(\mathrm{cons}_L(x))) \sqcap g_R(\mathrm{tail}_R(\mathrm{cons}_L(x))) \\
&= g_L(x) \sqcap g_R(0) = g_L(x) \sqcap g_L(1) = g_L(x),
\end{aligned}$$

because $g_L(x) \sqsubseteq g_L(1)$, by monotonicity. Similarly, a case analysis on y shows that $f(\mathrm{cons}_R(y)) = g_R(y)$. Assume that $x \sqsubseteq 1$ and $y \sqsubseteq 0$. Then

$$\begin{aligned}
&f(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) \\
&= g_L(\mathrm{tail}_L(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y))) \sqcap g_R(\mathrm{tail}_R(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y))) \\
&= g_L(x \sqcap 1) \sqcap g_R(0 \sqcap y) = g_L(x) \sqcap g_R(y),
\end{aligned}$$

which concludes the proof. $\square$

Remark 10.4 Definition by cases is the same as path composition as defined in Example 8.15. □

The natural numbers enjoy an induction principle, which can be expressed by saying that if a set of natural numbers contains zero and is closed under the successor operation, then it contains all natural numbers. A similar principle is enjoyed by the partial unit interval endowed with the operations cons_L and cons_R.

Definition 10.5 Let D be a domain. A set $A \subseteq D$ is called ***inductive*** if it closed under the formation of least upper bounds of directed subsets; that is, if $X \subseteq A$ for X directed implies $\bigsqcup^{\uparrow} X \in A$. □

Lemma 10.6 (Dyadic induction)
Let $A \subseteq \mathcal{I}$ be inductive, and assume that the following conditions hold:

1. *(Base case)* $\perp \in A$.
2. *(Inductive step)* $x \in A$ *and* $y \in A$ *together imply*
   (a) $\mathrm{cons}_L(x) \in A$,
   (b) $\mathrm{cons}_R(y) \in A$,
   (c) $\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y) \in A$ *if* $x \sqsubseteq 1$ *and* $y \sqsubseteq 0$.

Then $A = \mathcal{I}$.

Proof Let B be the basis of $\mathcal{I}$ consisting of intervals with *distinct* dyadic end-points. In order to show that $A = \mathcal{I}$, it suffices to conclude that $B \subseteq A$, because B is a basis of $\mathcal{I}$. But since

$$B = \bigcup_n B_n, \quad \text{where} \quad B_n = \{[l/2^n, m/2^n] | 0 \leq l < m \leq 2^n\},$$

it suffices to show that $B_n \subseteq A$ for all n by induction on n. For $n = 0$ this is immediate because $B_0 = \{\perp\}$ and $\perp \in A$ by hypothesis. Assume that $B_n \subseteq A$, and define

$$\begin{aligned} L_n &= \mathrm{cons}_L(B_n), \\ R_n &= \mathrm{cons}_R(B_n), \\ C_n &= \{\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y) | x, y \in B_n \wedge x \sqsubseteq 1 \wedge y \sqsubseteq 0\}. \end{aligned}$$

Then $L_n \subseteq \mathrm{cons}_L(A) \subseteq A$ because $x \in A$ implies $\mathrm{cons}_L(x) \in A$ by hypothesis.

Similarly, $R_n \subseteq A$ and $C_n \subseteq A$. Hence $L_n \cup R_n \cup C_n \subseteq A$. But

$$
\begin{aligned}
& L_n \cup R_n \cup C_n \\
= \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 0 \le l < m \le 2^n \right\} \cup \left\{ \left[\frac{l+2^n}{2^{n+1}}, \frac{m+2^n}{2^{n+1}}\right] \middle| \, 0 \le l < m \le 2^n \right\} \\
\cup \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{1}{2}\right] \sqcap \left[\frac{1}{2}, \frac{m+2^n}{2^{n+1}}\right] \middle| \, 0 \le l < 2^n \wedge 0 < m \le 2^n \right\} \\
= \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 0 \le l < m \le 2^n \right\} \cup \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 2^n \le l < m \le 2^{n+1} \right\} \\
\cup \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 0 \le l < 2^n < m \le 2^{n+1} \right\} \\
= \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 0 \le l < m \le 2^n \vee 2^n \le l < m \le 2^{n+1} \vee 0 \le l < 2^n < m \le 2^{n+1} \right\} \\
= \; & \left\{ \left[\frac{l}{2^{n+1}}, \frac{m}{2^{n+1}}\right] \middle| \, 0 \le l < m \le 2^{n+1} \right\} \\
= \; & B_{n+1}.
\end{aligned}
$$

Therefore $B_{n+1} \subseteq A$, which concludes the inductive argument. □

Corollary 10.7 *Let D be a domain and $f, g : \mathcal{I} \to D$ be continuous maps. In order to show that $f = g$ it suffices to show that the following conditions hold:*

1. *(Base case) $f(\bot) = g(\bot)$.*
2. *(Inductive step) $f(x) = g(x)$ and $f(y) = g(y)$ together imply*
   (a) *$f(\mathrm{cons}_L(x)) = g(\mathrm{cons}_L(x))$,*
   (b) *$f(\mathrm{cons}_R(y)) = g(\mathrm{cons}_R(y))$,*
   (c) *$f(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y)) = g(\mathrm{cons}_L(x) \sqcap \mathrm{cons}_R(y))$ if $x \sqsubseteq 1$ and $y \sqsubseteq 0$.*

Proof If f and g are continuous, then the set $A = \{x | f(x) = g(x)\}$ is inductive. □

Remark 10.8 If we omit condition (c) of the inductive step, then in Lemma 10.6 we conclude that $\mathrm{Max}\,\mathcal{I} \subseteq A$, and in Corollary 10.7 we conclude that $f_{|\mathrm{Max}\,\mathcal{I}} = g_{|\mathrm{Max}\,\mathcal{I}}$. A slight reformulation of Corollary 10.7 in these lines may be useful to prove that a continuous function $\bar{f} : \mathcal{I} \to \mathcal{I}$ is an extension of a continuous function $f : [0,1] \to [0,1]$, but not necessarily the *canonical* extension of f. □

Lemma 10.9 *Let D be a bounded complete domain, $g_L, g_R : D \to D$ be continuous maps, and $f : \mathcal{I} \to D$ be a continuous solution to the functional equation*

$$f(x) = \mathrm{pif\ head}(x) \mathrm{\ then\ } g_L(f(\mathrm{tail}_L(x))) \mathrm{\ else\ } g_R(f(\mathrm{tail}_R(x))).$$

Then the following statements hold:

1. *$f(0)$, $f(1)$ and $f(\bot)$ are fixed points of g_L, g_R and $g_L \sqcap g_R$ respectively.*
2. *f is uniquely determined by the values that it assumes at 0, 1, and $\bot$.*

3. *f is the least solution iff*

$$f(0) = \text{fix } g_L \qquad f(1) = \text{fix } g_R \qquad f(\bot) = \text{fix } (g_L \sqcap g_R).$$

4. *In particular, if g_L, g_R, and $g_L \sqcap g_R$ have unique fixed points then f is the unique solution.*

Proof (1): $f(0) = g_L(f(0))$ because $\text{head}(0) = \mathbf{tt}$ and $\text{tail}_L(0) = 0$. Similarly, $f(1) = g_R(f(1))$. Since $\text{head}(\bot) = \bot$ and $\text{tail}_L(\bot) = \text{tail}_R(\bot)$, we have that $f(\bot) = g_L(f(\bot)) \sqcap g_R(f(\bot)) = (g_L \sqcap g_R)(f(\bot))$.

(2): We show by dyadic induction as in Corollary 10.7 that if f' is a continuous solution agreeing with f at 0, 1 and $\bot$ then $f = f'$. (Base case): By hypothesis. (Inductive step): Assume that $f(x) = f'(x)$ and $f(y) = f'(y)$. If $\text{head} \circ \text{cons}_L(x) = \mathbf{tt}$ then

$$\begin{aligned} f(\text{cons}_L(x)) &= g_L(f(\text{tail}_L(\text{cons}_L(x)))) = g_L(f(x)) \\ &= g_L(f'(x)) = g_L(f'(\text{tail}_L(\text{cons}_L(x)))) = f'(\text{cons}_L(x)). \end{aligned}$$

Otherwise $\text{head}(\text{cons}_L(x)) = \bot$ and $x \sqsubseteq 1$. Therefore,

$$\begin{aligned} f(\text{cons}_L(x)) &= g_L(f(\text{tail}_L(\text{cons}_L(x)))) \sqcap g_R(f(\text{tail}_R(\text{cons}_L(x)))) \\ &= g_L(f(x)) \sqcap g_R(f(0)) = g_L(f'(x)) \sqcap g_R(f'(0)) \\ &= g_L(f'(\text{tail}_L(\text{cons}_L(x)))) \sqcap g_R(f'(\text{tail}_R(\text{cons}_L(x)))) \\ &= f'(\text{cons}_L(x)). \end{aligned}$$

Similarly, a case analysis on $\text{head} \circ \text{cons}_R(y)$ shows that $f(\text{cons}_R(y)) = f'(\text{cons}_R(y))$. Assume that $x \sqsubseteq 1$ and $y \sqsubseteq 0$. Then

$$\begin{aligned} &f(\text{cons}_L(x) \sqcap \text{cons}_R(y)) \\ &= g_L(f(\text{tail}_L(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \sqcap g_R(f(\text{tail}_R(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \\ &= g_L(f(x \sqcap 1)) \sqcap g_R(f(0 \sqcap y)) = g_L(f(x)) \sqcap g_R(f(y)) \\ &= g_L(f'(x)) \sqcap g_R(f'(y)) = g_L(f'(x \sqcap 1)) \sqcap g_R(f'(0 \sqcap y)) \\ &= g_L(f'(\text{tail}_L(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \sqcap g_R(f'(\text{tail}_R(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \\ &= f'(\text{cons}_L(x) \sqcap \text{cons}_R(y)). \end{aligned}$$

(3): In view of (1) and (2), it suffices to show that if f is the least solution then it satisfies the condition. Assume that f is the least solution. Then $f = \bigsqcup^{\uparrow}_n f_n$ where

$$\begin{aligned} f_0(x) &= \bot \\ f_{n+1}(x) &= \text{pif head}(x) \text{ then } g_L(f_n(\text{tail}_L(x))) \text{ else } g_R(f_n(\text{tail}_R(x))). \end{aligned}$$

But $f_n(0) = g_L^n(\bot)$ because $\text{head}(0) = \mathbf{tt}$ and $\text{tail}_L(0) = 0$. Hence $f(0) = \text{fix } g_L$. Similarly, $f(1) = \text{fix } g_R$. Also, $f_{n+1}(\bot) = (g_L \sqcap g_R)^n(\bot)$, because

$$f_{n+1}(\bot) = g_L(f_n(\bot)) \sqcap g_R(f_n(\bot)) = (g_L \sqcap g_R)(f_n(\bot)).$$

Therefore $f(\bot) = \text{fix } (g_L \sqcap g_R)$.

(4): Immediate consequence of (1), (2) and (3) above. $\square$

We can define functions on natural numbers by iteration. If X is a set, x is an element of X, and $g : X \to X$ is a function, then there is a unique function $f : \mathbb{N} \to X$ such that $f(0) = x$ and $f(n+1) = g(f(n))$. A similar fact holds for the partial unit interval equipped with cons_L and cons_L.

Lemma 10.10 (Dyadic recursion) *Let D be a bounded complete domain, and $g_L, g_R : D \to D$ be continuous maps such that*

$$g_L(\text{fix } g_R) = g_R(\text{fix } g_L).$$

Then there is a unique continuous map $f : \mathcal{I} \to D$ satisfying the equations

(Base case)

$$\begin{aligned} f(0) &= \text{fix } g_L \\ f(1) &= \text{fix } g_R \\ f(\bot) &= \text{fix } (g_L \sqcap g_R) \end{aligned}$$

(Recursion step)

$$\begin{aligned} f(\mathrm{cons}_L(x)) &= g_L(f(x)) \\ f(\mathrm{cons}_R(y)) &= g_R(f(y)) \\ f(\mathrm{cons}_R(x) \sqcap \mathrm{cons}_R(y)) &= g_L(f(x)) \sqcap g_R(f(y)) \qquad \textit{if } x \sqsubseteq 1 \textit{ and } y \sqsubseteq 0, \end{aligned}$$

namely the least continuous solution to the equation

$$f(x) \quad = \quad \text{pif head}(x) \text{ then } g_L(f(\mathrm{tail}_L(x))) \text{ else } g_R(f(\mathrm{tail}_R(x))).$$

Proof Define $F : (\mathcal{I} \Rightarrow D) \to (\mathcal{I} \Rightarrow D)$ by

$$F(f)(x) = \text{pif head}(x) \text{ then } g_L(f(\mathrm{tail}_L(x))) \text{ else } g_R(f(\mathrm{tail}_R(x))),$$

and let $f : \mathcal{I} \to D$ be a continuous map satisfying the base case. By Lemma 10.9, it suffices to show that f satisfies the recursion step iff $f = F(f)$. But, by Lemma 10.3, this is equivalent to show that

$$\begin{aligned} F(f)(\mathrm{cons}_L(x)) &= g_L(f(x)) \\ F(f)(\mathrm{cons}_R(y)) &= g_R(f(y)) \\ F(f)(\mathrm{cons}_R(x) \sqcap \mathrm{cons}_R(y)) &= g_L(f(x)) \sqcap g_R(f(y)) \qquad \text{if } x \sqsubseteq 1 \text{ and } y \sqsubseteq 0. \end{aligned}$$

In fact, assuming that these equations hold, if $f = F(f)$ then f satisfies the recursion step; and, conversely, if f satisfies the recursion step both f and $F(f)$ satisfy the same definition by cases, and therefore they have to be the same by Lemma 10.3. If $x \not\sqsubseteq 1$ then $\mathrm{head}(\mathrm{cons}_L(x)) = \mathbf{tt}$ and

$$F(f)(\mathrm{cons}_L(x)) \quad = \quad g_L(f(\mathrm{tail}_L(\mathrm{cons}_L(x)))) = g_L(f(x)).$$

Otherwise $\text{head}(\text{cons}_L(x)) = \perp$ and

$$\begin{aligned}
F(f)(\text{cons}_L(x)) &= g_L(f(\text{tail}_L(\text{cons}_L(x)))) \sqcap g_R(f(\text{tail}_R(\text{cons}_L(x)))) \\
&= g_L(f(x)) \sqcap g_R(f(0)) = g_L(f(x)) \sqcap g_R(\text{fix } g_L) \\
&= g_L(f(x)) \sqcap g_L(\text{fix } g_R) = g_L(f(x)) \sqcap g_L(f(1)) \\
&= g_L(f(x)),
\end{aligned}$$

because $g_L(f(x)) \sqsubseteq g_L(f(1))$, by monotonicity. Similarly, a case analysis on y shows that $F(f)(\text{cons}_R(y)) = g_R(f(y))$. Assume that $x \sqsubseteq 1$ and $y \sqsubseteq 0$. Then

$$\begin{aligned}
&F(f)(\text{cons}_L(x) \sqcap \text{cons}_R(y)) \\
&= g_L(f(\text{tail}_L(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \sqcap g_R(f(\text{tail}_R(\text{cons}_L(x) \sqcap \text{cons}_R(y)))) \\
&= g_L(f(x \sqcap 1)) \sqcap g_R(f(0 \sqcap y)) = g_L(f(x)) \sqcap g_R(f(y)),
\end{aligned}$$

which concludes the proof. □

Finally, the set of natural numbers is uniquely specified, up to isomorphism, by the so called Peano axioms, which are essentially the properties that we informally considered above for the sake of motivation. This idea is made formal in e.g. Stoll [Sto66], where *unary systems* are used as a tool (a unary system is a set X together with an element $x \in X$ and a function $s : X \to X$).

In the following definition, the domain D generalizes the partial unit interval and the maps a_L and a_R generalize the maps cons_L and cons_R respectively.

Definition 10.11 A ***binary system*** is a bounded complete domain D equipped with a pair of continuous maps $a_L, a_R : D \to D$ such that

$$a_L(1) = a_R(0),$$

where

$$0 = \text{fix } a_L \qquad 1 = \text{fix } a_R.$$

We also impose the technical condition

$$\text{fix } (a_L \sqcap a_R) = \perp,$$

which ensures that homomorphisms defined below, as Lemma 10.10 suggests, make binary systems into a category under ordinary function composition.

A ***homomorphism*** from a binary system (D, a_L, a_R) to a binary system (E, b_L, b_R) is a continuous map $f : D \to E$ such that

$$\begin{aligned}
f(0) &= 0 \\
f(1) &= 1 \\
f(\perp) &= \perp \\[1ex]
f(a_L(x)) &= b_L(f(x)) \\
f(a_R(y)) &= b_R(f(y)) \\
f(a_L(x) \sqcap a_R(y)) &= b_L(f(x)) \sqcap b_R(f(y)) \qquad \text{if } x \sqsubseteq 1 \text{ and } y \sqsubseteq 0.
\end{aligned}$$ □

Binary systems were introduced and investigated in the context of uniform spaces in the extended abstract [Esc94], which contains some mistakes.

Remark 10.12 Recall that an ***iterated function system (IFS)*** [Hut81] consists of a complete metric space X together with a finite list of contracting maps $f_1, \ldots, f_n : X \to X$. Several generalizations of this concept have been introduced and investigated. Hayaschi [Hay85] has investigated the case where X is a compact Hausdorff space. Edalat [Eda95e, Eda96a] has investigated the case where X is the upper space of a complete metric space. In this generalized setting, a binary system is an IFS. Moreover, the notion of homomorphism between binary systems coincides with the notion of ***conjugacy*** of IFS's investigated in Devaney [Dev89]. □

Lemma 10.10 can be formulated as

Theorem 10.13 *$(\mathcal{I}, \mathrm{cons}_L, \mathrm{cons}_R)$ is an initial object in the category of binary systems.*

Compare the following lemma to Lemmas 10.2 and 10.3, were c_L, c_R, h, t_L, t_R play the rôle of cons_L, cons_R, head, tail_L, and tail_R respectively:

Lemma 10.14 *Let $D = (D, c_L, c_R)$ be a binary system. Then there is at most one triple of continuous maps $h : D \to \mathcal{B}$ and $t_L, t_R : D \to D$ such that*

$$\begin{array}{rcl@{\qquad}rcl} t_L(c_L(x)) & = & x & t_R(c_L(x)) & = & 0 \\ t_L(c_R(y)) & = & 1 & t_R(c_R(y)) & = & y \\ t_L(c_L(x) \sqcap c_R(y)) & = & x \sqcap 1 & t_R(c_L(x) \sqcap c_R(y)) & = & 0 \sqcap y \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} h(c_L(x)) & \sqsubseteq & \mathbf{tt}, & h(c_L(x)) & = & \bot \quad \textit{iff} \quad x \sqsubseteq 1 \\ h(c_R(y)) & \sqsubseteq & \mathbf{ff}, & h(c_R(y)) & = & \bot \quad \textit{iff} \quad y \sqsubseteq 0 \end{array}$$

$$x = \mathrm{pif}\ h(x)\ \mathrm{then}\ c_L(t_L(x))\ \mathrm{else}\ c_R(t_R(x)).$$

Moreover, in this case D admits definition by cases, in the sense that for each bounded complete domain E and each pair of continuous maps $g_L, g_L : D \to E$ with

$$g_L(1) = g_R(0)$$

there is a unique map $f : D \to E$ such that

$$\begin{array}{rcl} f(c_L(x)) & = & g_L(x) \\ f(c_R(y)) & = & g_R(y) \\ f(c_R(x) \sqcap c_L(y)) & = & g_L(x) \sqcap g_R(y) \qquad \textit{if } x \sqsubseteq 1 \textit{ and } y \sqsubseteq 0 \end{array}$$

Proof Assume that such maps exist. Claim: For every $x \in \mathcal{I}$,

1. if $h(x) = \mathbf{tt}$ then $x = c_L(y)$ for a unique y,

2. if $h(x) = \mathbf{ff}$ then $x = c_R(z)$ for a unique z,

3. if $h(x) = \bot$ then $x = c_L(y) \sqcap c_R(z)$ for unique $y \sqsubseteq 1$ and $z \sqsubseteq 0$.

(1): There is at most one such y because t_L is a left inverse of c_L. Since

$$x = \text{pif } \mathbf{tt} \text{ then } c_L(t_L(x)) \text{ else } c_R(t_R(x)) = c_L(t_L(x)),$$

We can take $y = t_L(x)$. (2): Similar. (3): (Uniqueness) Assume that $c_L(y) \sqcap c_R(z) = c_L(y') \sqcap c_R(z')$ for $y, y' \sqsubseteq 1$ and $z, z' \sqsubseteq 0$. Then, by applying t_L to both sides, we obtain $y \sqcap 1 = y' \sqcap 1$. But $y \sqcap 1 = y$ and $y' \sqcap 1 = y'$. Therefore $y = y'$. Similarly, $z = z'$. (Existence):

$$x = \text{pif } \perp \text{ then } c_L(t_L(x)) \text{ else } c_R(t_R(x)) = c_L(t_L(x) \sqcap c_R(t_R(x))).$$

Hence $t_L(x) = t_L(x) \sqcap 1$ and $t_R(x) = 0 \sqcap t_R(x)$. We can thus let $y = t_L(x)$ and $z = t_R(x)$, and the proof of the claim is concluded.

It follows that there is at most one f satisfying the definition-by-cases scheme. Therefore there is at most one triple of maps as specified above, because t_L and t_R satisfy the definition-by-cases scheme, and h is completely specified by the above clauses by virtue of the claim (the inequality $h(c_L(x) \sqcap c_R(y)) \sqsubseteq \perp$ holds by monotonicity). Finally, a function f satisfying the definition-by-cases scheme can be constructed as in Lemma 10.3, because it only uses the abstract properties of cons_L, cons_R, head, tail_L, and tail_R considered in the statement of the present lemma (and proved in Lemma 10.2). □

Definition 10.15 If such maps exist, then they are called the ***destructors*** of (D, c_L, c_R). □

Definition 10.16 A binary system (D, c_L, c_R) satisfies the ***dyadic induction principle*** if for any inductive set $A \subseteq D$ the conditions

1. (Base case) $\perp \in A$,
2. (Inductive step) $x \in A$ and $y \in A$ together imply
   (a) $c_L(x) \in A$,
   (b) $c_R(y) \in A$,
   (c) $c_L(x) \sqcap c_R(y) \in A$ if $x \sqsubseteq 1$ and $y \sqsubseteq 0$,

together entail that $A = D$. □

Definition 10.17 A binary system (D, c_L, c_R) is ***inductive*** if

1. 0 and 1 are the unique fixed points of c_L and c_R.
2. $0 \neq 1$ and $0 \sqcap 1 = \perp$.
3. It has destructors.
4. It satisfies the dyadic induction principle. □

The following theorem axiomatically characterizes the binary system $(\mathcal{I}, \text{cons}_L, \text{cons}_R)$ up to isomorphism, without explicit reference to real numbers or intervals.

Theorem 10.18 *A binary system is inductive iff it is initial. In particular, any two inductive binary systems are isomorphic.*

Proof The inductiveness property is clearly preserved by binary system isomorphisms. Since $(\mathcal{I}, \mathrm{cons}_L, \mathrm{cons}_R)$ is an initial inductive binary system, every initial binary system is inductive, because any two initial objects are isomorphic. The proof of Lemma 10.10 only uses the axioms for inductive binary systems, without mentioning any particular property of $(\mathcal{I}, \mathrm{cons}_L, \mathrm{cons}_R)$, except for the equations

$$\begin{array}{rclcrcl} \mathrm{head}(0) & = & \mathbf{tt} & \quad & \mathrm{head}(1) & = & \mathbf{ff} \\ \mathrm{tail}_L(0) & = & 0 & & \mathrm{tail}_R(1) & = & 1 \\ \mathrm{tail}_L(\bot) & = & \bot & & \mathrm{tail}_R(\bot) & = & \bot, \end{array}$$

indirectly in Lemma 10.9, which easily follow from the axioms. □

10.2 Structural recursion on the partial unit interval

Definition 10.19 Define $\mathbf{T} : \mathbf{SDom} \to \mathbf{SDom}$ by

$$\mathbf{T}\, D = \mathcal{B} \times D \times D,$$

and define

$$\mathcal{I} \underset{\mathrm{destr}}{\overset{\mathrm{cons}}{\leftrightarrows}} \mathbf{T}\,\mathcal{I}$$

by

$$\begin{array}{rcl} \mathrm{cons} & = & \mathrm{pif} \circ (\mathrm{id} \times \mathrm{cons}_L \times \mathrm{cons}_R), \\ \mathrm{destr} & = & \langle \mathrm{head}, \mathrm{tail}_L, \mathrm{tail}_R \rangle; \end{array}$$

that is,

$$\begin{array}{rcl} \mathrm{cons}(p, y, z) & = & \mathrm{pif}\ x\ \mathrm{then}\ \mathrm{cons}_L(y)\ \mathrm{else}\ \mathrm{cons}_R(z), \\ \mathrm{destr}(x) & = & \langle \mathrm{head}(x), \mathrm{tail}_L(x), \mathrm{tail}_R(x) \rangle. \square \end{array}$$

Theorem 10.20 $\langle \mathrm{destr}, \mathrm{cons} \rangle$ *is a* $\mathbf{T}$*-inductive section-retraction pair.*

Proof The equation

$$f = \mathrm{cons} \circ \mathbf{T}\, f \circ \mathrm{destr}.$$

is equivalent to the equation

$$f(x) = \mathrm{pif}\ \mathrm{head}(x)\ \mathrm{then}\ \mathrm{cons}_L(f(\mathrm{tail}_L(x)))\ \mathrm{else}\ \mathrm{cons}_R(f(\mathrm{tail}_R(x))).$$

By Lemma 10.2, $f = \mathrm{id}$ is a solution. But cons_L and cons_R and $\mathrm{cons}_R \sqcap \mathrm{cons}_L$ have unique fixed-points. Therefore there is a unique solution by virtue of Lemma 10.9. □

Recall from Section 5.2 that this gives rise to structural recursion and corecursion on $\mathcal{I}$.

We leave as an open problem to characterize the $\mathbf{T}$-algebras a such that there is a (necessarily unique) homomorphism from cons to a in a neat way, without referring to the bifree algebra for the functor $\mathbf{T}$.

We now relate the algebra cons : $\mathbf{T}\,\mathcal{I} \to \mathcal{I}$ to the binary system $(\mathcal{I}, \mathrm{cons}_L, \mathrm{cons}_R)$.

Definition 10.21 A ***binary*** $\mathbf{T}$***-algebra*** is a $\mathbf{T}$-algebra $a : \mathbf{T}\,D \to D$ of the form

$$\mathrm{pif} \circ (\mathrm{id} \times a_L \times a_R)$$

for (necessarily unique) $a_L, a_R : D \to D$. □

Such maps are necessarily unique because they have to satisfy the equations

$$\begin{aligned} a_L(x) &= a(\mathbf{tt}, x, \bot), \\ a_R(y) &= a(\mathbf{ff}, \bot, y). \end{aligned}$$

Compare the following proposition to Lemma 10.10 and Definition 10.11:

Proposition 10.22 *Let $a : \mathbf{T}\,D \to D$ be a binary $\mathbf{T}$-algebra. Then a strict continuous map $f : \mathcal{I} \to D$ is a homomorphism from* cons *to a iff*

$$\begin{aligned} f(\mathrm{cons}_L(x)) &= a_L(f(x)) \\ f(\mathrm{cons}_R(y)) &= a_R(f(y)) \\ f(\mathrm{cons}_R(x) \sqcap \mathrm{cons}_R(y)) &= a_L(f(x)) \sqcap a_R(f(y)). \end{aligned}$$

Proof f is a homomorphism from cons to a iff

$$f(\text{pif } p \text{ then } \mathrm{cons}_L(x) \text{ else } \mathrm{cons}_R(y)) = \text{pif } p \text{ then } a_L(f(x)) \text{ else } a_R(f(y))$$

for all p, x, y. By considering the cases $p = \mathbf{tt}, \mathbf{ff}, \bot$ respectively, this equation is seen to be equivalent to the above equations together. □

Compare the following proposition to Lemmas 10.9 and 10.10.

Proposition 10.23 *If there is a homomorphism from* cons *to a binary $\mathbf{T}$-algebra $a : \mathbf{T}\,D \to D$ then it is the least continuous map $f : \mathcal{I} \to D$ such that*

$$f(x) = \text{pif head}(x) \text{ then } a_L(f(\mathrm{tail}_L(x))) \text{ else } a_R(f(\mathrm{tail}_R(x))).$$

Proof By Proposition 5.8 we know that if there is a homomorphism from cons to a, then it is the least continuous function f such that

$$f = a \circ \mathbf{T}\,f \circ \mathrm{destr},$$

which is equivalent to the above equation. □

Compare the following proposition to Definition 10.11:

Proposition 10.24 *Let $a : \mathbf{T}\, D \to D$ be a binary $\mathbf{T}$-algebra. If there is a homomorphism from* cons *to a then*

$$a_L(\text{fix } a_R) = a_R(\text{fix } a_L).$$

Proof Let f be a homomorphism. Then f satisfies the equations of Lemma 10.22. Hence

$$\begin{aligned} f(\text{cons}_L(1)) &= a_L(f(1)) = a_L(\text{fix } a_R) \\ f(\text{cons}_R(0)) &= a_R(f(0)) = a_R(\text{fix } a_L) \end{aligned}$$

by Lemma 10.23. But $\text{cons}_L(1) = \text{cons}_R(0)$. □

Is the converse true? No, because cons makes more identifications than cons_L and cons_R do. For example, $\text{cons}(\bot, x, y) = \text{cons}(\mathbf{tt}, x, \bot)$ if $y \sqsubseteq 1$, and $\text{cons}(\bot, x, y) = \text{cons}(\mathbf{ff}, \bot, y)$ if $x \sqsubseteq 0$. We leave as an open problem to give a necessary and sufficient condition on a binary algebra a for the existence of a homomorphism from cons to a. This amounts to the classification of these identifications, which shouldn't be very difficult.

In order to consider recursive definitions of functions $f : \mathcal{I} \times \mathcal{I} \to E$, we can try to develop the analogue of Lemma 10.10 for two variables, as in the following proposition. Unfortunately, in practice it turns that one of the hypotheses is hardly ever satisfied (namely that the transpose of f is strict).

Proposition 10.25 *Let $f : \mathcal{I} \times \mathcal{I} \to D$ be a continuous map multiplicative in each argument such that its transpose $g : \mathcal{I} \to (\mathcal{I} \Rightarrow D)$ is strict, let*

$$a_{ij} : E \to E, \quad i, j \in \{L, R\}$$

be continuous functions such that

$$f \circ (\text{cons}_i \times \text{cons}_j) = a_{ij} \circ f$$

and define $a : \mathbf{T}\,(\mathcal{I} \Rightarrow E) \to (\mathcal{I} \Rightarrow E)$ by

$$a = \text{id} \times (\text{pif } \circ (\text{id} \times a_{LL} \times a_{LR}) \circ \text{destr}) \times (\text{pif } \circ (\text{id} \times a_{RL} \times a_{RR}) \circ \text{destr}).$$

Then g is the unique homomorphism from cons *to the algebra a. In particular, f is the least solution to the functional equation*

$$\begin{aligned} f(x,y) = \text{pif head}(x) \text{ then pif head}(y) &\text{ then } a_{LL}(f(\text{tail}_L(x), \text{tail}_L(y))) \\ &\text{ else } \ a_{LR}(f(\text{tail}_L(x), \text{tail}_R(y))) \\ \text{else } \ \text{pif head}(y) &\text{ then } a_{RL}(f(\text{tail}_R(x), \text{tail}_L(y))) \\ &\text{ else } \ a_{RR}(f(\text{tail}_R(x), \text{tail}_R(y))). \end{aligned}$$

Proof The condition on f implies that

$$g(\text{cons}_i(x))(\text{cons}_j(y)) = a_{ij}(g(x)(y)),$$

and by Lemma 10.2 we know that the identity

$$z = \text{pif head}(z) \text{ then cons}_L(\text{tail}_L(z)) \text{ else cons}_R(\text{tail}_R(z))$$

holds. This and the fact that f is multiplicative in its second argument together imply that

$$
\begin{aligned}
&g(\mathrm{cons}_i(x))(z)\\
&\quad= g(\mathrm{cons}_i(x))(\mathrm{pif\ head}(z)\ \mathrm{then\ cons}_L(\mathrm{tail}_L(z))\ \mathrm{else\ cons}_R(\mathrm{tail}_R(z)))\\
&\quad= \mathrm{pif\ head}(z)\ \mathrm{then}\ g(\mathrm{cons}_i(x))(\mathrm{cons}_L(\mathrm{tail}_L(z)))\ \mathrm{else}\ g(\mathrm{cons}_i(x))(\mathrm{cons}_R(\mathrm{tail}_R(z)))\\
&\quad= \mathrm{pif\ head}(z)\ \mathrm{then}\ a_i L(g(x)(z))\ \mathrm{else}\ a_i R(g(x)(z)).
\end{aligned}
$$

Since the definition of a can be written

$$
\begin{aligned}
a(p,h,k)(z) = \mathrm{pif}\ p\ \mathrm{then\ pif\ head}(z)\ &\mathrm{then}\ a_{LL}(h(tail_L(z)))\\
&\mathrm{else}\ \ a_{LR}(h(\mathrm{tail}_R(z)))\\
\mathrm{else}\ \ \mathrm{pif\ head}(z)\ &\mathrm{then}\ a_{RL}(k(\mathrm{tail}_L(z)))\\
&\mathrm{else}\ \ a_{RR}(k(\mathrm{tail}_R(z)))
\end{aligned}
$$

and f is multiplicative in its first argument, we have that

$$
\begin{aligned}
(g \circ \mathrm{cons}(p,x,y))(z) &= g(\mathrm{pif}\ p\ \mathrm{then\ cons}_L(x)\ \mathrm{else\ cons}_R(y))(z)\\
&= (\mathrm{pif}\ p\ \mathrm{then}\ g(\mathrm{cons}_L(x))\ \mathrm{else}\ g(\mathrm{cons}_R(y)))(z)\\
&= \mathrm{pif}\ p\ \mathrm{then}\ g(\mathrm{cons}_L(x))(z)\ \mathrm{else}\ g(\mathrm{cons}_R(y))(z)\\
&= a(p, g(x), g(y))(z)\\
&= (a \circ \mathbf{T}\, g(p,x,y))(z).
\end{aligned}
$$

Therefore $g \circ \mathrm{cons} = a \circ \mathbf{T}\, g$, which means that g is an algebra homomorphism from cons to a. From Proposition 5.8, we know that g is the least solution of the equation

$$g = a \circ \mathbf{T}\, g \circ \mathrm{destr}.$$

Therefore g is the least solution to the equation

$$
\begin{aligned}
g(x)(y) = \mathrm{pif\ head}(x)\ \mathrm{then\ pif\ head}(y)\ &\mathrm{then}\ a_{LL}(g(\mathrm{tail}_L(x))(\mathrm{tail}_L(y)))\\
&\mathrm{else}\ \ a_{LR}(g(\mathrm{tail}_L(x))(\mathrm{tail}_R(y)))\\
\mathrm{else}\ \ \mathrm{pif\ head}(y)\ &\mathrm{then}\ a_{RL}(g(\mathrm{tail}_R(x))(\mathrm{tail}_L(y)))\\
&\mathrm{else}\ \ a_{RR}(g(\mathrm{tail}_R(x))(\mathrm{tail}_R(y)))
\end{aligned}
$$

and f is the least solution to the above equation. □

Notice that the multiplicativity hypothesis in the statement of the proposition is stronger than necessary.

10.3 Bifurcated binary expansions

The canonical solution to the domain equation

$$D \cong \mathbf{T}\, D,$$

for the functor $\mathbf{T}\, D = \mathcal{B} \times D \times D$ considered in Definition 10.19, is the domain $\mathcal{B}$Tree of infinite binary trees with nodes labelled by truth values, ordered nodewise, together with the bifree algebra

$$\mathrm{mktree} : \mathbf{T}\, \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$$

which sends a list $\langle p, s, t\rangle$ to the tree with root labelled by p, and with left and right subtrees s and t respectively:

$$\mathrm{mktree}(p,s,t) = \begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}$$

Definition 10.26 Let $\mathrm{num} : \mathcal{B}\mathrm{Tree} \to \mathcal{I}$ be the unique algebra homomorphism from mktree to cons, and $\mathrm{bin} : \mathcal{I} \to \mathcal{B}\mathrm{Tree}$ be the unique coalgebra homomorphism from destr to mktree^{-1}, as indicated in the following diagrams:

$$\begin{tikzcd} \mathbf{T}\,\mathcal{B}\mathrm{Tree} \arrow[r, "\mathrm{mktree}"] \arrow[d, "\mathbf{T}\,\mathrm{num}"'] & \mathcal{B}\mathrm{Tree} \arrow[d, "\mathrm{num}"] \\ \mathbf{T}\,\mathcal{I} \arrow[r, "\mathrm{cons}"'] & \mathcal{I} \end{tikzcd} \qquad \begin{tikzcd} \mathcal{I} \arrow[r, "\mathrm{destr}"] \arrow[d, "\mathrm{bin}"'] & \mathbf{T}\,\mathcal{I} \arrow[d, "\mathbf{T}\,\mathrm{bin}"] \\ \mathcal{B}\mathrm{Tree} \arrow[r, "\mathrm{mktree}^{-1}"'] & \mathbf{T}\,\mathcal{B}\mathrm{Tree} \end{tikzcd}$$

$\square$

Then we have that

$$\mathrm{num}\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) = \mathrm{pif}\ p\ \mathrm{then}\ \mathrm{cons}_L(\mathrm{num}(s))\ \mathrm{else}\ \mathrm{cons}_R(\mathrm{num}(t)),$$

$$\mathrm{bin}(x) = \begin{array}{ccccc} & & \mathrm{head}(x) & & \\ & \swarrow & & \searrow & \\ \mathrm{bin}(\mathrm{tail}_L(x)) & & & & \mathrm{bin}(\mathrm{tail}_R(x)). \end{array}$$

By Proposition 5.5,

$$\mathrm{num} \circ \mathrm{bin} = \mathrm{id}.$$

Definition 10.27 We refer to the tree $\mathrm{bin}(x)$ as the ***bifurcated binary expansion*** of x. $\square$

By the above recursive definition of bin, we have that

$$\mathrm{bin}(0) = \begin{array}{ccccc} & & \mathbf{tt} & & \\ & \swarrow & & \searrow & \\ \mathrm{bin}(0) & & & & \mathrm{bin}(0) \end{array}$$

$$\mathrm{bin}(1) = \begin{array}{ccccc} & & \mathbf{ff} & & \\ & \swarrow & & \searrow & \\ \mathrm{bin}(1) & & & & \mathrm{bin}(1) \end{array}$$

$$\mathrm{bin}(1/2) = \begin{array}{ccccc} & & \bot & & \\ & \swarrow & & \searrow & \\ \mathrm{bin}(1) & & & & \mathrm{bin}(0) \end{array}$$

Thus the bifurcated binary expansions of 0 and 1 are the binary trees with all nodes labelled by **tt** and **ff** respectively. Also,

$$\mathrm{bin}(x) = \begin{cases} \begin{array}{ccc} & \mathbf{tt} & \\ \swarrow & & \searrow \\ \mathrm{bin}(\mathrm{tail}_L(x)) & & \mathrm{bin}(0) \end{array} & \text{if } x < 1/2, \\[2ex] \begin{array}{ccc} & \mathbf{ff} & \\ \swarrow & & \searrow \\ \mathrm{bin}(1) & & \mathrm{bin}(\mathrm{tail}_R(x)) \end{array} & \text{if } x > 1/2, \\[2ex] \begin{array}{ccc} & \perp & \\ \swarrow & & \searrow \\ \mathrm{bin}(\mathrm{tail}_L(x)) & & \mathrm{bin}(\mathrm{tail}_R(x)) \end{array} & \text{if } x \uparrow 1/2, \end{cases}$$

because $\mathrm{tail}_L(\mathrm{cons}_R(x)) = 1$ and $\mathrm{tail}_R(\mathrm{cons}_L(x)) = 0$.

Now, recall that a ***binary expansion*** of a real number $x \in [0, 1]$ is a sequence $\langle a_n \rangle_{n \geq 1}$ of bits (binary digits 0 and 1) with

$$\sum_{n \geq 1} a_n 2^{-n} = x$$

(see e.g. Bourbaki [Bou66]). A binary expansion $\langle a_n \rangle_{n \geq 1}$ of a number $x \in [0, 1]$ can be found by the following non-deterministic inductive procedure, which is also non-effective as equality of real numbers is undecidable:

1. Let $x_0 = x$.
2. If $x_n \leq 1/2$ let $a_n = 0$ and $x_{n+1} = \mathrm{tail}_L(x_n)$.
3. If $x_n \geq 1/2$ let $a_n = 1$ and $x_{n+1} = \mathrm{tail}_R(x_n)$.

If we denote the multi-valued function which sends a number to its set of binary expansions by Bin, and if we write sequences vertically, the above process can be described by the following informal recursive definition:

$$\mathrm{Bin}(x) = \begin{cases} \begin{array}{c} 0 \\ \downarrow \\ \mathrm{Bin}(\mathrm{tail}_L(x)) \end{array} & \text{if } x \leq 1/2, \\[2ex] \begin{array}{c} 1 \\ \downarrow \\ \mathrm{Bin}(\mathrm{tail}_R(x)) \end{array} & \text{if } x \geq 1/2. \end{cases}$$

Thus, the main differences between bifurcated binary expansions and usual binary expansions are that

1. bifurcated binary expansions capture partial numbers in addition to total numbers,

2. bifurcated binary expansions are obtained by a deterministic process (the above recursive definition of bin),

3. bifurcated binary expansions are obtained by an effective process, which replaces the non-effective tests $x \leq 1/2$ and $x \geq 1/2$ by the effective test $\mathrm{head}(x) = (x <_{\perp} 1/2)$.

On the other hand, not every binary tree is a bifurcated binary expansion. However, the idempotent

$$\mathrm{norm} = \mathrm{bin} \circ \mathrm{num} : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$$

has the bifurcated binary expansions as its fixed-points. Moreover, by general properties of retracts, the set of bifurcated binary expansions endowed with the order inherited from $\mathcal{B}\mathrm{Tree}$ is a domain isomorphic to $\mathcal{I}$.

In the remaining of this section we look for a recursive definition of

$$\mathcal{B}\mathrm{Tree} \xleftarrow{\mathrm{norm}} \mathcal{B}\mathrm{Tree} = \mathcal{B}\mathrm{Tree} \xleftarrow{\mathrm{bin}} \mathcal{I} \xleftarrow{\mathrm{num}} \mathcal{B}\mathrm{Tree}$$

not involving the intermediate domain $\mathcal{I}$. Such a recursive definition is applied in Section 12.1 in order to find an appropriate effective presentation of the partial real line.

If we have an algebra $\mathrm{cons}' : \mathbf{T}\,\mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ such that bin is a homomorphism from cons to cons$'$, as indicated in the diagram

$$\begin{tikzcd}
\mathbf{T}\,\mathcal{I} \arrow[r, "\mathrm{cons}"] \arrow[d, "\mathbf{T}\,\mathrm{bin}"'] & \mathcal{I} \arrow[d, "\mathrm{bin}"] \\
\mathbf{T}\,\mathcal{B}\mathrm{Tree} \arrow[r, "\mathrm{cons}'"'] & \mathcal{B}\mathrm{Tree}
\end{tikzcd}$$

then norm is a homomorphism from mktree to cons$'$, because num is a homomorphism from mktree to cons, by definition, and homomorphisms compose. Therefore norm can be recursively defined by

$$\mathrm{norm} \circ \mathrm{mktree} = \mathrm{cons}' \circ \mathbf{T}\,\mathrm{norm},$$

or, equivalently, by

$$\mathrm{norm}\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) = \mathrm{cons}'(p, \mathrm{norm}(s), \mathrm{norm}(t)).$$

Moreover, if cons$'$ is a binary $\mathbf{T}$-algebra (Definition 10.21), in the sense that we can find maps $\mathrm{cons}'_L, \mathrm{cons}'_R : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ such that

$$\mathrm{cons}' = \mathrm{pif} \circ (\mathrm{id} \times \mathrm{cons}'_L \times \mathrm{cons}'_R),$$

then we can recursively define norm by

$$\mathrm{norm}\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) = \mathrm{pif}\ p\ \mathrm{then}\ \mathrm{cons}'_L(\mathrm{norm}(s))\ \mathrm{else}\ \mathrm{cons}'_R(\mathrm{norm}(t)).$$

Lemma 10.28 bin *is a multiplicative function.*

Proof bin is the least fixed-point of the continuous functional F defined by

$$F(f) = \mathrm{mktree} \circ \mathbf{T}\, f \circ \mathrm{destr}.$$

The function mktree is multiplicative because binary meets of triples are computed componentwise, and binary meets of trees are computed nodewise. If f is multiplicative so is $\mathbf{T}\, f$. Also destr $= \langle \mathrm{head}, \mathrm{tail}_L, \mathrm{tail}_R \rangle$ is multiplicative because head, tail_R and tail_L are multiplicative by Lemmas 9.17 and 9.16 respectively. Hence, since a composition of multiplicative functions is multiplicative, if f is multiplicative so is $F(f)$. Therefore the least fixed-point of F is multiplicative, because the binary meet operation preserves directed joins. $\square$

Lemma 10.29 *Let* $\mathrm{cons}'_L, \mathrm{cons}'_R : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ *be continuous maps, and define*

$$\mathrm{cons}' = \mathrm{pif} \circ (\mathrm{id} \times \mathrm{cons}'_L \times \mathrm{cons}'_R).$$

If the diagrams

$$\begin{array}{ccc} \mathcal{I} & \xrightarrow{\mathrm{cons}_L} & \mathcal{I} \\ {\scriptstyle \mathrm{bin}}\downarrow & & \downarrow{\scriptstyle \mathrm{bin}} \\ \mathcal{B}\mathrm{Tree} & \xrightarrow[\mathrm{cons}'_L]{} & \mathcal{B}\mathrm{Tree} \end{array} \qquad \begin{array}{ccc} \mathcal{I} & \xrightarrow{\mathrm{cons}_R} & \mathcal{I} \\ {\scriptstyle \mathrm{bin}}\downarrow & & \downarrow{\scriptstyle \mathrm{bin}} \\ \mathcal{B}\mathrm{Tree} & \xrightarrow[\mathrm{cons}'_R]{} & \mathcal{B}\mathrm{Tree} \end{array}$$

commute then bin *is an algebra homomorphism from* cons *to* cons$'$.

Proof By Lemmas 8.16 and 10.28, if the diagrams commute then

$$\begin{aligned} \mathrm{bin} \circ \mathrm{cons}(p, y, z) &= \mathrm{bin}(\mathrm{pif}\ p \text{ then } \mathrm{cons}_L(y) \text{ else } \mathrm{cons}_R(z)) \\ &= \mathrm{pif}\ p \text{ then } \mathrm{bin}(\mathrm{cons}_L(y)) \text{ else } \mathrm{bin}(\mathrm{cons}_R(z)) \\ &= \mathrm{pif}\ p \text{ then } \mathrm{cons}'_L(\mathrm{bin}(y)) \text{ else } \mathrm{cons}'_R(\mathrm{bin}(z)) \\ &= \mathrm{cons}'(p, \mathrm{bin}(y), \mathrm{bin}(z)) \\ &= \mathrm{cons}' \circ \mathbf{T}\, \mathrm{Bin}(p, y, z), \end{aligned}$$

and hence bin is a homomorphism from cons to cons$'$. $\square$

In order to construct such maps cons'_L and cons'_R, we first consider the following construction:

Lemma 10.30 *Let* $\mathrm{head}'_0, \mathrm{head}'_1 : \mathcal{B}\mathrm{Tree} \to \mathcal{B}$ *be recursively defined by*

$$\begin{aligned} \mathrm{head}'_0\left(\begin{array}{ccc} & p & \\ \swarrow & & \searrow \\ s & & t \end{array} \right) &= \mathrm{pif}\ p \text{ then } \mathrm{head}'_0(s) \text{ else } \mathbf{ff}, \\ \mathrm{head}'_1\left(\begin{array}{ccc} & p & \\ \swarrow & & \searrow \\ s & & t \end{array} \right) &= \mathrm{pif}\ p \text{ then } \mathbf{tt} \text{ else } \mathrm{head}'_1(t). \end{aligned}$$

Then

$$\mathrm{head}'_0 = \mathrm{head}_0 \circ \mathrm{num} \qquad \textit{and} \qquad \mathrm{head}'_1 = \mathrm{head}_1 \circ \mathrm{num}.$$

Proof Define

$$
\begin{array}{llll}
f_0 & = \bot & f_{n+1}\,(\mathrm{mktree}(p,s,t)) & = \mathrm{pif}\ p\ \mathrm{then}\ f_n(s)\ \mathrm{else}\ \mathbf{ff},\\
\mathrm{id}_0 & = \bot & \mathrm{id}_{n+1}\,(\mathrm{mktree}(p,s,t)) & = \mathrm{mktree}(p, \mathrm{id}_n(s), \mathrm{id}_n(t)).
\end{array}
$$

Then head_0' and $\mathrm{id} : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ are the least upper bounds of the chains f_n and id_n respectively. We show by induction on n that $\mathrm{head}_0 \circ \mathrm{num} \circ \mathrm{id}_n = f_n$. For the base case this is immediate, because head_0 and num are both strict. For the inductive step, recall that head_0 is multiplicative by Lemma 9.17, and that hence it distributes over the parallel conditional in the sense of Lemma 8.16. Also, notice that $\mathrm{head}_0 \circ \mathrm{cons}_L = \mathrm{head}_0$, because $x < 0$ iff $x/2 < 0$, and that $\mathrm{head}_0 \circ \mathrm{cons}_R(x) = \mathbf{ff}$, because $\mathrm{cons}_R(x) \geq 1/2$. Hence

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathrm{head}_0 \circ \mathrm{num} \circ \mathrm{id}_{n+1}\,(\mathrm{mktree}(p,s,t))}\\
= & \mathrm{head}_0 \circ \mathrm{num}\,(\mathrm{mktree}(p, \mathrm{id}_n(s), \mathrm{id}_n(t)))\\
= & \mathrm{head}_0(\mathrm{pif}\ p\ \mathrm{then}\ \mathrm{cons}_L(\mathrm{num}(\mathrm{id}_n(s)))\ \mathrm{else}\ \mathrm{cons}_R(\mathrm{num}(\mathrm{id}_n(s))))\\
= & \mathrm{pif}\ p\ \mathrm{then}\ \mathrm{head}_0(\mathrm{cons}_L(\mathrm{num}(\mathrm{id}_n(s))))\ \mathrm{else}\ \mathrm{head}_0(\mathrm{cons}_R(\mathrm{num}(\mathrm{id}_n(s))))\\
= & \mathrm{pif}\ p\ \mathrm{then}\ \mathrm{head}_0(\mathrm{num}(\mathrm{id}_n(s)))\ \mathrm{else}\ \mathbf{ff}\\
= & \mathrm{pif}\ p\ \mathrm{then}\ f_n(s)\ \mathrm{else}\ \mathbf{ff}\\
= & f_{n+1}\,(\mathrm{mktree}(p,s,t))\,,
\end{array}
$$

which finishes our inductive argument. Therefore

$$
\mathrm{head}_0' = \bigsqcup_n^{\uparrow} f_n = \bigsqcup^{\uparrow} \mathrm{head}_0 \circ \mathrm{num} \circ \mathrm{id}_n = \mathrm{head}_0 \circ \mathrm{num} \circ \bigsqcup^{\uparrow} \mathrm{id}_n = \mathrm{head}_0 \circ \mathrm{num}.
$$

The proof for head_1' is symmetric. □

The continuous maps of the hypothesis of Lemma 10.29 can be constructed as follows:

Lemma 10.31 *Let* $\mathrm{cons}_L', \mathrm{cons}_R' : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ *be defined by*

$$
\mathrm{cons}_L'(t) \quad = \quad \begin{array}{ccc} & \mathrm{head}_1'(t) & \\ \swarrow & & \searrow \\ t & & \mathrm{bin}(0) \end{array}
$$

$$
\mathrm{cons}_R'(t) \quad = \quad \begin{array}{ccc} & \mathrm{head}_0'(t) & \\ \swarrow & & \searrow \\ \mathrm{bin}(1) & & t \end{array}
$$

Then the diagrams displayed in Lemma 10.29 commute.

Proof Since $\mathrm{head}_1' = \mathrm{head}_1 \circ \mathrm{num}$ by Lemma 10.30, and since $\mathrm{num} \circ \mathrm{bin}$, we conclude that $\mathrm{head}_1 = \mathrm{head}_1' \circ \mathrm{bin}$. Since $\mathrm{head} \circ \mathrm{cons}_L(x) = (x/2 <_\bot 1/2) =$

$(x{<_\perp}1) = \mathrm{head}_1(x)$,

$$\begin{array}{ll} & \mathrm{bin} \circ \mathrm{cons}_L(x) \\ = & \mathrm{mktree}(\mathrm{head} \circ \mathrm{cons}_L(x), \mathrm{bin}(\mathrm{tail}_L(\mathrm{cons}_L(x))), \mathrm{bin}(\mathrm{tail}_R(\mathrm{cons}_L(x)))) \\ = & \mathrm{mktree}(\mathrm{head}_1(x), \mathrm{bin}(x), \mathrm{bin}(0)) \\ = & \mathrm{mktree}(\mathrm{head}'_1(\mathrm{bin}(x)), \mathrm{bin}(x), \mathrm{bin}(0)) \\ = & \mathrm{cons}'_L \circ \mathrm{bin}(x). \end{array}$$

The proof for cons_R is symmetric. □

We have thus established

Theorem 10.32 *The idempotent* norm *can be recursively defined by*

$$\mathrm{norm}\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) = \text{pif } p \text{ then } \mathrm{cons}'_L(\mathrm{norm}(s)) \text{ else } \mathrm{cons}'_R(\mathrm{norm}(t)).$$

Compare the following lemma to Definition 3.16(3b), where $\mathcal{B}\mathrm{Tree}$ plays the rôle of $\mathcal{B}^\omega$:

Lemma 10.33 *Let* $\mathrm{tail}'_L, \mathrm{tail}'_R : \mathcal{B}\mathrm{Tree} \to \mathcal{B}\mathrm{Tree}$ *and* $\mathrm{head}' : \mathcal{B}\mathrm{Tree} \to \mathcal{B}$ *be defined by*

$$\begin{array}{rcl} \mathrm{head}'\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) & = & \text{pif } p \text{ then } \mathrm{head}'_1(s) \text{ else } \mathrm{head}'_0(t) \\ \mathrm{tail}'_L\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) & = & \text{pif } p \text{ then } s \text{ else } \mathrm{bin}(1) \\ \mathrm{tail}'_R\left(\begin{array}{ccccc} & & p & & \\ & \swarrow & & \searrow & \\ s & & & & t \end{array}\right) & = & \text{pif } p \text{ then } \mathrm{bin}(0) \text{ else } t. \end{array}$$

Then, for each $a \in \{L, R\}$,

$$\begin{array}{rcl} \mathrm{cons}_a & = & (\mathrm{bin} \Rightarrow \mathrm{num})(\mathrm{cons}'_a) \\ \mathrm{head} & = & (\mathrm{bin} \Rightarrow \mathrm{id})(\mathrm{head}') \\ \mathrm{tail}_a & = & (\mathrm{bin} \Rightarrow \mathrm{num})(\mathrm{tail}'_a). \end{array}$$

Proof By Lemma 10.31, we have that $\mathrm{bin} \circ \mathrm{cons}_a = \mathrm{cons}'_a \circ \mathrm{bin}$. Hence

$$\mathrm{cons}_a = \mathrm{num} \circ \mathrm{cons}_a \circ \mathrm{bin} = (\mathrm{bin} \Rightarrow \mathrm{num})(\mathrm{cons}'_a),$$

because $\mathrm{num} \circ \mathrm{bin} = \mathrm{id}$. For head we have that

$$\begin{array}{ll} & (\mathrm{bin} \Rightarrow \mathrm{id})(\mathrm{head}')(x) \\ = & \mathrm{head}' \circ \mathrm{bin}(x) \\ = & \mathrm{head}'(\mathrm{mktree}(\mathrm{head}(x), \mathrm{bin}(\mathrm{tail}_L(x)), \mathrm{bin}(\mathrm{tail}_R(x)))) \\ = & \text{pif } \mathrm{head}(x) \text{ then } \mathrm{head}'_1(\mathrm{bin}(\mathrm{tail}_L(x))) \text{ else } \mathrm{head}'_0(\mathrm{bin}(\mathrm{tail}_R(x))) \\ = & \text{pif } \mathrm{head}(x) \text{ then } \mathrm{head}_1(\mathrm{tail}_L(x)) \text{ else } \mathrm{head}_0(\mathrm{tail}_R(x)), \end{array}$$

by virtue of Lemma 10.30. If $\mathrm{head}(x) = \mathbf{tt}$ then $x < 1/2$ and hence $\mathrm{tail}_L(x) < 1$. Thus, in this case the last term is

$$\mathrm{head}_1(\mathrm{tail}_L(x)) = \mathbf{tt} = \mathrm{head}(x).$$

Similarly, if $\mathrm{head}(x) = \mathbf{ff}$ then

$$\mathrm{head}_1(\mathrm{tail}_R(x)) = \mathbf{ff} = \mathrm{head}(x).$$

Otherwise $\mathrm{head}(x) = \bot$. Then $x \sqsubseteq 1/2$ and hence $\mathrm{tail}_L(x) \sqsubseteq 1$ and $\mathrm{tail}_R(x) \sqsubseteq 0$. Therefore in this case the last term is

$$\mathrm{head}_1(\mathrm{tail}_L(x)) \sqcap \mathrm{head}_0(\mathrm{tail}_R(x)) = \bot \sqcap \bot = \bot = \mathrm{head}(x).$$

For tail_L we have that

$$\begin{aligned}
&(\mathrm{bin} \Rightarrow \mathrm{num})(\mathrm{tail}'_L)(x) \\
&= \mathrm{num} \circ \mathrm{tail}'_L \circ \mathrm{bin}(x) \\
&= \mathrm{num} \circ \mathrm{tail}'_L(\mathrm{mktree}(\mathrm{head}(x), \mathrm{bin}(\mathrm{tail}_L(x)), \mathrm{bin}(\mathrm{tail}_R(x)))) \\
&= \mathrm{num}(\mathrm{pif}\ \mathrm{head}(x)\ \mathrm{then}\ \mathrm{bin}(\mathrm{tail}_L(x))\ \mathrm{else}\ \mathrm{bin}(1)) \\
&= \mathrm{num} \circ \mathrm{bin}(\mathrm{pif}\ \mathrm{head}(x)\ \mathrm{then}\ \mathrm{tail}_L(x)\ \mathrm{else}\ 1) \\
&= \mathrm{pif}\ \mathrm{head}(x)\ \mathrm{then}\ \mathrm{tail}_L(x)\ \mathrm{else}\ 1
\end{aligned}$$

If $\mathrm{head}(x) = \mathbf{tt}$ then the last term is $\mathrm{tail}_L(x)$. If $\mathrm{head}(x) = \mathbf{ff}$ then the last term is $1 = \mathrm{tail}_L(x)$. Otherwise, $\mathrm{head}(x) = \bot$. Then $x \sqsubseteq 1/2$ and $\mathrm{tail}_L(x) \sqsubseteq 1$. Hence in this case the last term is

$$\mathrm{tail}_L(x) \sqcap 1 = \mathrm{tail}_L(x).$$

For tail_R we have a symmetric proof. □

10.4 Coinduction on the partial unit interval

Dana Scott suggested to the author that he should also consider a characterization of the partial unit interval by "co-Peano axioms", based on some form of coinduction and corecursion. Although we don't have such a characterization yet, we have a coinduction principle based on the ideas of Smyth [Smy92a] and Fiore [Fio93, Fio96].

Definition 10.34 A ***bisimulation*** on $\mathcal{I}$ is a binary relation $\sim \subseteq \mathcal{I} \times \mathcal{I}$ such that

> $x \sim y$ implies that $\mathrm{head}(x) = \mathrm{head}(y)$ and $\mathrm{tail}_a(x) \sim \mathrm{tail}_a(y)$ for $a \in \{R, L\}$.

We say that x and y are ***bisimilar*** if they are related by some bisimulation. □

Theorem 10.35 (Coinduction) *If $x, y \in \mathcal{I}$ are bisimilar then $x = y$.*

Proof (Sketch) Assume that x and y are bisimilar. Then $\mathrm{bin}(x) = \mathrm{bin}(y)$. Since bin is split mono, $x = y$. □

So far we don't have any application of this coinduction principle.

Of course, we can replace equalities by inequalities thus obtaining the notion of ***simulation*** and a more general coinduction principle for establishing inequalities.

10.5 Structural recursion on the partial real line

The domain $\mathcal{R}_\perp$ is treated similarly, by considering the primitive operations

$$\begin{array}{rclcrcl}
\mathrm{rcons}_L(x) & = & x/2, & & \mathrm{rtail}_L(x) & = & 2x, \\
\mathrm{rcons}_R(x) & = & (x+1)/2, & & \mathrm{rtail}_R(x) & = & 2x-1, \\
\mathrm{trunc}(x) & = & \max(0, \min(x, 1)), & & \mathrm{incl}(x) & = & x.
\end{array}$$

Here $\mathrm{incl} : \mathcal{I} \to \mathcal{R}$ and $\mathrm{trunc} : \mathcal{R}_\perp \to \mathcal{I}$ form a section-retraction pair.

We omit the routine details, because they are formally similar to those of the treatment of $\mathcal{I}$. The crucial idea is that the identity of $\mathcal{R}_\perp$ is the unique continuous function such that

$$\begin{array}{rclll}
f(x) & = & \mathrm{pif}\ x <_\perp 0 & \mathrm{then} & \mathrm{rcons}_R(f(\mathrm{rtail}_R(x))) \\
 & & \mathrm{pif}\ 1 <_\perp x & \mathrm{then} & \mathrm{rcons}_L(f(\mathrm{rtail}_L(x))) \\
 & & & \mathrm{else} & \mathrm{incl}(\mathrm{trunc}(x)).
\end{array}$$

Roughly, the idea is that if $x < 0$ then there is some n such that $\mathrm{rtail}_R^n(x)$ belongs to $\mathcal{I}$, and, similarly, if $x > 1$ then there is some n such that $\mathrm{rtail}_L^n(x)$ belongs to $\mathcal{I}$, so that we can reduce the treatment to the previous case. This leads us to define a functor $\mathbf{T} : \mathbf{SDom} \to \mathbf{SDom}$ by

$$\mathbf{T}\, D = \mathcal{B} \times \mathcal{B} \times D \times D \times \mathcal{I}$$

functions

$$\mathcal{R}_\perp \underset{\mathrm{destr}}{\overset{\mathrm{cons}}{\leftrightarrows}} \mathbf{T}\, \mathcal{R}_\perp$$

by

$$\begin{array}{rclll}
\mathrm{cons}(p, q, x, y, z) & = & \mathrm{pif}\ p & \mathrm{then} & \mathrm{rcons}_R(x) \\
 & & \mathrm{pif}\ q & \mathrm{then} & \mathrm{rcons}_L(y) \\
 & & & \mathrm{else} & \mathrm{incl}(z).
\end{array}$$

$$\mathrm{destr}(x) = \langle x <_\perp 0, 1 <_\perp x, \mathrm{rtail}_R(x), \mathrm{rtail}_L(x), \mathrm{trunc}(x) \rangle.$$

Theorem 10.36 $\langle \mathrm{destr}, \mathrm{cons} \rangle$ *is a* $\mathbf{T}$*-inductive section-retraction pair.*

It is easy to derive the analogue of Theorem 10.32 and Lemma 10.33, which are necessary to establish computational completeness of Real PCF.

10.6 Examples of recursive definitions

We have already seen several examples of recursive definitions in Section 10.3. In this section we show how to derive recursive definitions of basic real functions using the principles introduced in Sections 10.1, 10.2, and 10.5. For the sake of brevity, some routine proofs and constructions are only sketched.

Unless otherwise stated, when we refer to a real function as a partial real function, we mean its canonical extension, as in Convention 8.2.

Proposition 10.37 *The complement map* $\mathrm{compl} : \mathcal{I} \to \mathcal{I}$ *defined by*

$$\mathrm{compl}(x) = 1 - x$$

can be recursively defined by

$$\begin{aligned}\mathrm{compl}(x) = \mathrm{pif}\ \mathrm{head}(x) \quad & \mathrm{then} \quad \mathrm{cons}_R(\mathrm{compl}(\mathrm{tail}_L(x))) \\ & \mathrm{else} \quad \mathrm{cons}_L(\mathrm{compl}(\mathrm{tail}_R(x))).\end{aligned}$$

Proof There are two ways to prove this fact, based on Sections 10.1 and 10.2 respectively, but they are essentially equivalent:

First way: If (D, c_L, c_R) is a binary system, so is (D, c_R, c_L), because the definition is symmetric. Thus, it suffices to show that compl is a homomorphism from $(\mathcal{I}, \mathrm{cons}_L, \mathrm{cons}_R)$ to $(\mathcal{I}, \mathrm{cons}_R, \mathrm{cons}_L)$, and apply Lemma 10.10, obtaining the above recursive definition. The base case clearly holds because it simply says that $\mathrm{compl}(0) = 1$, $\mathrm{compl}(1) = 0$, and $\mathrm{compl}(\bot) = \bot$, which is immediately true. The first two equations of the recursion step are proved by

$$\begin{aligned}\mathrm{compl}(\mathrm{cons}_L(x)) &= 1 - x/2 = (2-x)/2 = ((1-x)+1)/2 \\ &= \mathrm{cons}_R(\mathrm{compl}(x)), \\ \mathrm{compl}(\mathrm{cons}_R(x)) &= 1 - (x+1)/2 = (2-(x+1))/2 = (1-x)/2 \\ &= \mathrm{cons}_L(\mathrm{compl}(x)).\end{aligned}$$

For the third equation of the recursion step, it suffices to observe that compl is multiplicative, simply because it is an isomorphism.

Second way: If compl is a homomorphism from the algebra cons to the algebra cons' defined by

$$\mathrm{cons}' = \mathrm{pif}\ \circ (\mathrm{id} \times \mathrm{cons}_R \times \mathrm{cons}_L),$$

then the result follows from Proposition 10.23. But the hypothesis follows from the arguments above and Proposition 10.22. □

Proposition 10.38

1. *The logarithm function* $\log_2 : \mathbf{I}[1,2] \to \mathbf{I}[0,1]$ *can be recursively defined by*

$$\begin{aligned}\log_2(x) = \mathrm{pif}\ x <_\bot \sqrt{2} \quad & \mathrm{then} \quad \mathrm{cons}_L(\log_2(x^2)) \\ & \mathrm{else} \quad \mathrm{cons}_R(\log_2(x^2/2)),\end{aligned}$$

where x^2 *and* $x^2/2$ *stand for* $\min(x^2, 2)$ *and* $\max(1, x^2/2)$ *respectively, by an abuse of notation.*

2. The inverse $\exp_2 : \mathbf{I}[0,1] \to \mathbf{I}[1,2]$ *of* $\log_2$ *can be recursively defined by*

$$\begin{array}{rll}\exp_2(x) = \text{pif head}(x) & \text{then} & \sqrt{\exp_2(\text{tail}_L(x))} \\ & \text{else} & \sqrt{2\exp_2(\text{tail}_L(x))}.\end{array}$$

Proof Define $a_L, a_R : \mathbf{I}[1,2] \to \mathbf{I}[1,2]$ by

$$a_L(x) = \sqrt{x} \qquad a_R(x) = \sqrt{2x}.$$

Then

$$\begin{aligned}\log_2 \circ a_L(x) &= \log_2(\sqrt{x}) = \log_2(x)/2 = \text{cons}_L \circ \log_2(x) \\ \log_2 \circ a_R(x) &= \log_2(\sqrt{2x}) = (\log_2(x)+1)/2 = \text{cons}_R \circ \log_2(x).\end{aligned}$$

Since $\exp_2$ is the inverse of $\log_2$,

$$\begin{aligned}\exp_2 \circ \text{cons}_L &= a_L \circ \exp_2 \\ \exp_2 \circ \text{cons}_R &= a_R \circ \exp_2 .\end{aligned}$$

It follows that $(\mathbf{I}[1,2], a_L, a_R)$ is a binary system isomorphic to $(\mathcal{I}, \text{cons}_L, \text{cons}_R)$ and hence it is also initial by Theorem 10.18. Hence

$$a = \text{pif } \circ (\text{id} \times a_L \times a_R)$$

is a bifree algebra. But since $\log_2$ and $\exp_2$ are isomorphisms, they are multiplicative. Hence Proposition 10.22 shows that $\log_2 : a \to \text{cons}$ and $\exp_2 : \text{cons} \to a$. The destructors of $(\mathcal{I}, a_L, a_L)$ are

$$\begin{aligned}h(x) &= x <_\perp \sqrt{2} \\ t_L(x) &= \min(x^2, 1) \\ t_R(x) &= \max(0, x^2/2).\end{aligned}$$

Therefore the result follows from Proposition 10.23. □

Proposition 10.39 *The the average map* $\oplus : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$ *defined by*

$$x \oplus y = (x+y)/2$$

can be recursively defined by

$$\begin{array}{rlll}x \oplus y = \text{pif head}(x) & \text{then pif head}(y) & \text{then} & \text{cons}_L(\text{tail}_L(x) \oplus \text{tail}_L(y)) \\ & & \text{else} & \text{cons}_C(\text{tail}_L(x) \oplus \text{tail}_R(y)) \\ & \text{else } \ \text{pif head}(y) & \text{then} & \text{cons}_C(\text{tail}_R(x) \oplus \text{tail}_L(y)) \\ & & \text{else} & \text{cons}_R(\text{tail}_R(x) \oplus \text{tail}_R(y)).\end{array}$$

where

$$C = [1/4, 3/4].$$

Proof First, notice that average satisfies the following exchange law:

$$\begin{aligned}(a \oplus x) \oplus (b \oplus y) &= ((a+x)/2 + (b+y)/2)/2 = ((a+b)/2 + (x+y)/2)/2 \\ &= (a \oplus b) \oplus (x \oplus y).\end{aligned}$$

Since

$$\mathrm{cons}_L(x) = 0 \oplus x, \qquad \mathrm{cons}_C(x) = 1/2 \oplus x, \qquad \mathrm{cons}_R(x) = 1 \oplus x,$$

and since $0 \oplus 0 = 0$, $1/2 = 0 \oplus 1 = 1 \oplus 0$, and $1 \oplus 1 = 1$, it follows that

$$\begin{aligned}\mathrm{cons}_L(x) \oplus \mathrm{cons}_L(y) &= \mathrm{cons}_L(x \oplus y) \\ \mathrm{cons}_L(x) \oplus \mathrm{cons}_R(y) &= \mathrm{cons}_C(x \oplus y) \\ \mathrm{cons}_R(x) \oplus \mathrm{cons}_L(y) &= \mathrm{cons}_C(x \oplus y) \\ \mathrm{cons}_R(x) \oplus \mathrm{cons}_R(y) &= \mathrm{cons}_R(x \oplus y).\end{aligned}$$

A routine verification shows that average is multiplicative in each argument (it suffices to check one argument because it is commutative). Hence, if the transpose of average were strict, which unfortunately it isn't, the result would immediately follow from Proposition 10.25. We thus proceed as follows: (1) We show that average satisfies the recursive equation, and (2) we show that any two functions satisfying the recursive equation are equal. Step (1) can be done directly using the above equations and the fact that average is multiplicative. Step (2) can be done by nested dyadic induction. We omit the details. $\square$

Ternary average, needed in the recursive definition of multiplication given in Proposition 10.40 below, can be recursively defined in essentially the same way as binary average (with an extra level in the nesting of conditionals).

A routine verification shows that the multiplication map $(x, y) \mapsto x \times y : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$ satisfies the following equations:

$$\begin{aligned}\mathrm{cons}_L(x) \times \mathrm{cons}_L(y) &= \mathrm{cons}_{LL}(x \times y) \\ \mathrm{cons}_L(x) \times \mathrm{cons}_R(y) &= \mathrm{cons}_L\left(\frac{x + x \times y}{2}\right) \\ \mathrm{cons}_R(x) \times \mathrm{cons}_L(y) &= \mathrm{cons}_L\left(\frac{x \times y + y}{2}\right) \\ \mathrm{cons}_R(x) \times \mathrm{cons}_R(y) &= \mathrm{cons}_{RL}\left(\frac{x + x \times y + y}{3}\right)\end{aligned}$$

If we informally apply Proposition 10.25 in order to reduce these four equations to a single equation, and formally proceed as in the proof of Proposition 10.39, we obtain:

Proposition 10.40 *The multiplication map $(x, y) \mapsto x \times y : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$ can be recursively defined by*

$x \times y =$ pif head(x)
    then pif head(y) then $\mathrm{cons}_{LL}\,(\mathrm{tail}_L(x) \times \mathrm{tail}_L(y))$

$$\begin{array}{l} \qquad\qquad\qquad\qquad\text{else}\quad \mathrm{cons}_L\left(\dfrac{\mathrm{tail}_L(x)+\mathrm{tail}_L(x)\times\mathrm{tail}_R(y)}{2}\right)\\ \text{else}\quad \text{pif head}(y)\ \text{then}\ \mathrm{cons}_L\left(\dfrac{\mathrm{tail}_R(x)\times\mathrm{tail}_L(y)+\mathrm{tail}_L(y)}{2}\right)\\ \qquad\qquad\qquad\qquad\text{else}\quad \mathrm{cons}_{RL}\left(\dfrac{\mathrm{tail}_R(x)+\mathrm{tail}_R(x)\times\mathrm{tail}_R(y)+\mathrm{tail}_R(y)}{3}\right). \end{array}$$

Similarly, we have that

$$\begin{array}{rcl} \max(\mathrm{cons}_L(x),\mathrm{cons}_L(y)) & = & \mathrm{cons}_L(\max(x,y))\\ \max(\mathrm{cons}_L(x),\mathrm{cons}_R(y)) & = & \mathrm{cons}_L(x)\\ \max(\mathrm{cons}_R(x),\mathrm{cons}_L(y)) & = & \mathrm{cons}_L(y)\\ \max(\mathrm{cons}_R(x),\mathrm{cons}_R(y)) & = & \mathrm{cons}_R(\max(x,y)), \end{array}$$

and we conclude that

Proposition 10.41 max *can be recursively defined by*

$$\begin{array}{llll} \max(x,y) = \text{pif head}(x) & \text{then pif head}(y) & \text{then} & \mathrm{cons}_L(\max(\mathrm{tail}_L(x),\mathrm{tail}_L(y)))\\ & & \text{else} & y\\ & \text{else}\ \ \text{pif head}(y) & \text{then} & x\\ & & \text{else} & \mathrm{cons}_R(\max(\mathrm{tail}_R(x),\mathrm{tail}_R(y))). \end{array}$$

The operation min can be defined in a similar way, or simply by

$$\min(x,y) = 1-\max(1-x,1-y).$$

Proposition 10.42 *The function* $(x,y)\mapsto(x{<_\perp}y):\mathcal{I}\times\mathcal{I}\to\mathcal{I}$ *can be recursively defined by*

$$\begin{array}{llll} (x{<_\perp}y) = \text{pif head}(x) & \text{then pif head}(y) & \text{then} & \mathrm{tail}_L(x){<_\perp}\mathrm{tail}_L(y)\\ & & \text{else} & \mathbf{tt}\\ & \text{else}\ \ \text{pif head}(y) & \text{then} & \mathbf{ff}\\ & & \text{else} & \mathrm{tail}_R(x){<_\perp}\mathrm{tail}_R(y) \end{array}$$

We now consider some recursive definitions on $\mathcal{R}_\perp$. We obtain a recursive definition of addition in $\mathcal{R}_\perp$ by reducing addition to the average operation on $\mathcal{I}$ defined in Proposition 10.39. In the following proposition, the symbol $\vee$ denotes ***parallel-or***, defined by

$$p\vee q = \text{pif } p \text{ then } \mathbf{tt} \text{ else } q.$$

Also, the affine maps in the recursive definition correspond to operations rrcons_a for appropriate a.

Proposition 10.43 *Addition on* $\mathcal{R}_\perp$ *can be recursively defined by*

$$\begin{array}{ll} x+y = & \text{pif } x<_\perp 0\vee y<_\perp 0 \text{ then } 2\left(\frac{x+1}{2}+\frac{y+1}{2}\right)-2\\ \quad\text{else} & \text{pif } x>_\perp 1\vee y>_\perp 1 \text{ then } 2\left(\frac{x}{2}+\frac{y}{2}\right)\\ & \qquad\qquad\qquad\qquad\ \text{else}\ \ 2\,\mathrm{trunc}(\mathrm{incl}(x)\oplus\mathrm{incl}(y)). \end{array}$$

Proof We only observe that the above equation is equivalent to

$$\begin{aligned}
x + y = {} & \text{pif } x < 0 \vee y < 0 \text{ then } 2\left(\tfrac{x+1}{2} + \tfrac{y+1}{2}\right) - 2 \\
\text{else} \quad & \text{pif } x > 1 \vee y > 1 \text{ then } 2\left(\tfrac{x}{2} + \tfrac{y}{2}\right) \\
& \qquad\qquad\qquad\quad \text{else } \ 2\left(\tfrac{x+y}{2}\right),
\end{aligned}$$

with some slight abuse of notation. The idea is to "normalize" x and y to fractions, compute average, and then rescale the result back. □

Maximum, minimum, and multiplication on $\mathcal{R}_\perp$ can be obtained in a similar way from the corresponding operations on $\mathcal{I}$.

Proposition 10.44 *If $b > 1$ then the logarithm function $\log_b : \mathcal{R}_\perp \to \mathcal{R}_\perp$, with the convention that $\log_b(x) = \perp$ if x is consistent with a non-positive number, can be recursively defined by*

$$\begin{aligned}
\log_b(x) = {} & \text{pif } x >_\perp b \text{ then} \log_b(x/b) + 1 \\
\text{else} \quad & \text{pif } x <_\perp 1 \text{ then} \log_b(bx) - 1 \\
\text{else} \quad & \text{join}_{[0,1]} \left(\text{pif } x^2 <_\perp b \text{ then } \tfrac{\log_b(x^2)}{2} \text{ else } \tfrac{\log_b(x^2/b)+1}{2}\right)
\end{aligned}$$

For a treatment of some trigonometric functions see [Esc94].

Part IV

Computation on the partial real line

In Chapter 11 we introduce Real PCF and its operational and denotational semantics, and we establish computational adequacy. In Chapter 12 we show that the real numbers type hierarchy has a unique sound effective presentation, and we establish computational completeness of Real PCF.

Chapter 11

The programming language Real PCF

We introduce two successive extensions of PCF, first with a type for the partial unit interval (Section 11.1) and then with a further type for the partial real line (Section 11.2). We prove that the operational semantics enjoys the following ***adequacy property***: a program of real number type evaluates to a head-normal form iff its value is different from $\perp$; if its value is different from $\perp$ then it successively evaluates to head-normal forms giving better and better partial results converging to its value.

11.1 The programming language $\mathbf{PCF}^I$

We let $\mathcal{L}$ range over the languages $\mathcal{L}_{\mathrm{DA}}$, $\mathcal{L}_{\mathrm{PA}}$ and $\mathcal{L}_{\mathrm{PA}+\exists}$, and $\mathcal{L}^I$ denote the extension of $\mathcal{L}$ with a new ground type $\mathbf{I}$ and the following new constants:

1. $\mathbf{cons}_a : \mathbf{I} \to \mathbf{I}$
2. $\mathbf{tail}_a : \mathbf{I} \to \mathbf{I}$
3. $\mathbf{head}_r : \mathbf{I} \to \mathbf{T}$
4. $\mathbf{pif}_{\mathbf{I}} : (\mathbf{T}, \mathbf{I}, \mathbf{I}, \mathbf{I})$

for each non-bottom $a \in \mathcal{I}$ with distinct rational end-points, so that the interpretation of $\mathbf{tail}_a$ given below is well-defined, and each rational $r \in (0,1)$. We refer to the ground type $\mathbf{I}$ as the ***real number type***, and to programs of real number type as ***real programs***.

Denotational semantics

We let $D_{\mathbf{I}} = \mathcal{I}$ and we extend the standard interpretation $\mathcal{A}$ of $\mathcal{L}$ to $\mathcal{L}^I$ by

1. $\mathcal{A}[\![\mathbf{cons}_a]\!] = \mathrm{cons}_a$
2. $\mathcal{A}[\![\mathbf{tail}_a]\!] = \mathrm{tail}_a$
3. $\mathcal{A}[\![\mathbf{head}_r]\!] = \mathrm{head}_r$

4. $\mathcal{A}[\![\mathbf{pif_I}]\!]pxy = \text{pif } p \text{ then } x \text{ else } y$.

Operational semantics

We extend the immediate reduction relation $\rightarrow$ of $\mathcal{L}$ to $\mathcal{L}^I$ by the following rules:

1. $\mathbf{cons}_a(\mathbf{cons}_b M) \rightarrow \mathbf{cons}_{ab}\, M$

2. $\mathbf{tail}_a(\mathbf{cons}_b M) \rightarrow \mathbf{Y}\mathbf{cons}_L$ if $b \leq a$

3. $\mathbf{tail}_a(\mathbf{cons}_b M) \rightarrow \mathbf{Y}\mathbf{cons}_R$ if $b \geq a$

4. $\mathbf{tail}_a(\mathbf{cons}_b M) \rightarrow \mathbf{cons}_{b\backslash a}\, M$ if $a \sqsubseteq b$ and $a \neq b$

5. $\mathbf{tail}_a(\mathbf{cons}_b M) \rightarrow \mathbf{cons}_{(a\sqcup b)\backslash a}(\mathbf{tail}_{(a\sqcup b)\backslash b}\, M)$
   if $a\uparrow b$, $a \not\sqsubseteq b$, $b \not\sqsubseteq a$, $b \not\leq a$, and $a \not\leq b$

6. $\mathbf{head}_r(\mathbf{cons}_a M) \rightarrow \mathbf{tt}$ if $a < r$

7. $\mathbf{head}_r(\mathbf{cons}_a M) \rightarrow \mathbf{ff}$ if $a > r$

8. $\mathbf{pif}\,\mathbf{tt} MN \rightarrow M, \quad \mathbf{pif}\,\mathbf{ff} MN \rightarrow N$

9. $\mathbf{pif}\ L\ (\mathbf{cons}_a M)\ (\mathbf{cons}_b N) \rightarrow$
   $$\mathbf{cons}_{a\sqcap b}(\mathbf{pif}\ L\ (\mathbf{cons}_{a\backslash(a\sqcap b)}\, M)\ (\mathbf{cons}_{b\backslash(a\sqcap b)}\, N))$$
   if $a \sqcap b \neq \bot$

10. $\dfrac{N \rightarrow N'}{MN \rightarrow MN'}$ if M is $\mathbf{cons}_a$, $\mathbf{tail}_a$, $\mathbf{head}_r$ or $\mathbf{pif}$

11. $\dfrac{L \rightarrow L'}{\mathbf{pif} L \rightarrow \mathbf{pif} L'} \quad \dfrac{M \rightarrow M'}{\mathbf{pif} LM \rightarrow \mathbf{pif} LM'} \quad \dfrac{N \rightarrow N'}{\mathbf{pif} LMN \rightarrow \mathbf{pif} LMN'}$.

These rules are well-defined by virtue of Lemma 9.3 and because the extra conditions on them which are not present in Lemma 9.18 ensure that no bottom elements and no maximal elements are produced as subscripts of **cons** or **tail**. It may seem that there is a missing self-evident reduction rule for $\mathbf{tail}_a$, namely the rule $\mathbf{tail}_a(\mathbf{cons}_a M) \rightarrow M$; see remark after Lemma 11.6.

> The basic idea behind the above reduction rules is to reduce computations on real numbers to computations on rational numbers, namely the subscripts of the constants **cons**, **head**, and **tail**.

Notice that the immediate reduction rules for the parallel conditional are non-deterministic. The following lemma shows that this non-determinism does not produce inconsistencies:

Lemma 11.1 *$M \to N$ implies $[\![M]\!](\rho) = [\![N]\!](\rho)$ for all terms M and N and any environment ρ.*

Proof This is true for the language $\mathcal{L}$, and remains true for the extended language $\mathcal{L}^I$ by virtue of Lemma 9.18. $\square$

The partial map Eval on programs M of truth-value and natural number type is defined in the same way as for $\mathcal{L}$. It is well-defined by virtue of Lemma 11.1, because no two different constants have the same interpretation. We extend Eval to a multi-valued map on real programs M by

$$\mathrm{Eval}(M) = \{a \in \mathcal{I} | M \to^* \mathbf{cons}_a M' \text{ for some } M'\};$$

that is, $a \in \mathrm{Eval}(M)$ iff M has a head-normal form $\mathbf{cons}_a M'$.

Soundness of the operational semantics

Lemma 11.2 *For all real programs M, $a \in \mathrm{Eval}(M)$ implies $a \sqsubseteq [\![M]\!]$.*

Proof By Lemma 11.1, if $M \to^* \mathbf{cons}_a M'$ then $[\![M]\!] = [\![\mathbf{cons}_a M']\!] = a[\![M']\!]$. Therefore $a \sqsubseteq [\![M]\!]$, because the information order on partial numbers coincides with the prefix preorder. $\square$

Theorem 11.3 (Soundness) *For any real program M,*

$$\bigsqcup \mathrm{Eval}(M) \sqsubseteq [\![M]\!].$$

Proof The join exists because $[\![M]\!]$ is an upper bound of the set $\mathrm{Eval}(M)$. Therefore the result follows from Lemma 11.2. $\square$

Completeness of the operational semantics

By virtue of Lemma 11.2, for any real program M we have that $\mathrm{Eval}(M)$ is empty if $[\![M]\!] = \bot$. The following theorem states a stronger form of the converse:

Theorem 11.4 (Completeness) *For any real program M,*

$$\bigsqcup \mathrm{Eval}(M) \sqsupseteq [\![M]\!]$$

Proof Lemma 11.6 below. $\square$

The soundness and completeness properties in the senses of Theorems 11.3 and 11.4 can be referred together as the ***computational adequacy*** property of $\mathcal{L}^I$.

Theorem 11.5 (Adequacy) *$\mathcal{L}^I$ is computationally adequate.*

Several recursive definitions given in Section 10.6 immediately give rise to PCF programs. It would be a formidable task to syntactically prove the correctness of the resulting programs by appealing to the operational semantics given by the reduction rules. However, a mathematical proof of the correctness of a recursive definition allows us to conclude that the induced programs indeed produce correct results, via an application of the Adequacy Theorem. In fact, this is one of the main points of denotational semantics.

We extend the inductive definition of the predicates Comp_σ given in Section 4.3 of Part I by the following clause:

> A real program M has property $\mathrm{Comp}_{\mathbf{I}}$ if for every non-bottom partial number $x \ll [\![M]\!]$ (as close to $[\![M]\!]$ as we please) there is some $a \in \mathrm{Eval}(M)$ with $x \sqsubseteq a$.

If we read the relation $x \ll y$ as "x is a piece of information about y" then the above definition says that a real program is computable if every piece of information about its value can be produced in a finite number of reduction steps.

The domain-theoretic justificative for continuity of computable functions is that it is a finiteness condition [Plo80, Sco72b, Smy83, Smy92b]; a function f is continuous if a finite amount of information about $f(x)$ depends only on a finite amount of information about x. The continuity of a domain, which amounts to its way-below order being well-behaved, gives us a general and abstract framework for consistently talking about "pieces of information". Then it should come as no surprise that Lemma 11.6 makes essential use of the way-below order and of the continuity of the primitive functions.

The following lemma, which extends Lemma 4.6, establishes the Completeness Theorem:

Lemma 11.6 *Every term is computable.*

Proof It suffices to extend the inductive proof of Lemma 4.6. We have to:

1. slightly modify the proof for the case of abstractions, because it mentions constants and we have added new constants;
2. extend the proof of computability of the terms $\mathbf{Y}_\sigma$ to the new types; and
3. show that the new constants are computable.

(1) If M is computable so is $\lambda\alpha M$:

It is enough to show that the ground term $LN_1 \cdots N_n$ is computable when $N_1, \cdots, N_n$ are closed computable terms and L is a closed instantiation of $\lambda\alpha M$ by computable terms. Here L must have the form $\lambda\alpha\tilde{M}$ where $\tilde{M}$ is an instantiation of all free variables of M, except α, by closed computable terms. Since $[N_1/\alpha]\tilde{M}$ is computable, so is $[N_1/\alpha]\tilde{M}N_2 \cdots N_n$. Since $LN_1 \cdots N_n \to [N_1/\alpha]\tilde{M}N_2 \cdots N_n$ and the reduction relation preserves meaning, in order to evaluate $LN_1 \cdots N_n$ it suffices to evaluate $[N_1/\alpha]\tilde{M}N_2 \cdots N_n$.

(2) $\mathbf{Y}_\sigma$ is computable for all new types:

In order to prove that $\mathbf{Y}_\sigma$ is computable for all new types it suffices to show that the term $\mathbf{Y}_{(\sigma_1,\ldots,\sigma_k,\mathbf{I})}N_1 \cdots N_k$ is computable whenever $N_1 : \sigma_1, \ldots, N_k : \sigma_k$ are closed computable terms.

It follows from (1) above that the terms $\mathbf{Y}_\sigma^{(n)}$ are computable for all new types, because the proof of computability of $\mathbf{Y}_\sigma^{(n)}$ for old types depends only on the fact that variables are computable and that the combination and abstraction formation rules preserve computability.

The syntactic information order $\preccurlyeq$ on $\mathcal{L}$ extends to $\mathcal{L}^I$, and Lemma 4.5 is easily seen to remain true by a routine extension of its inductive proof.

Let $x \ll [\![\mathbf{Y} N_1 \cdots N_k]\!]$ be a non-bottom partial number. By a basic property of the way-below order of any *continuous* dcpo, there is some n such that $x \ll [\![\mathbf{Y}^{(n)} N_1 \cdots N_k]\!]$, because $[\![\mathbf{Y}]\!] = \bigsqcup_n [\![\mathbf{Y}^{(n)}]\!]$. Since $\mathbf{Y}^{(n)}$ is computable, there is a $c \in \mathrm{Eval}(\mathbf{Y}^{(n)} N_1 \cdots N_k)$ with $x \sqsubseteq c$. Since there is a term M with $\mathbf{Y}^{(n)} N_1 \cdots N_k \to^* \mathrm{cons}_c M$ and $\mathbf{Y}^{(n)} \preccurlyeq \mathbf{Y}$, it follows from Lemma 4.5 that $\mathbf{Y} N_1 \cdots N_k \to^* \mathrm{cons}_c M$ for some M and therefore $c \in \mathrm{Eval}(\mathbf{Y} N_1 \cdots N_k)$.

(3) The new constants are computable:

In order to prove that one of the new constants c is computable it suffices to show that if $M_1, \ldots, M_n$ are closed computable terms such that $c M_1 \ldots M_n$ has real number type, then $c M_1 \ldots M_n$ is computable.

(3)(a) $\mathbf{cons}_a$ is computable:

Let M be a computable real program and let $x \ll [\![\mathbf{cons}_a M]\!] = a[\![M]\!]$ be a non-bottom partial number. We have to produce $c \in \mathrm{Eval}(\mathbf{cons}_a M)$ with $x \sqsubseteq c$. Let $b \ll [\![M]\!]$ with $x \ll ab$. If $b = \bot$ then we can take $c = a$. Otherwise, by computability of M we can find $b' \in \mathrm{Eval}(M)$ with $b \sqsubseteq b'$. Then we can take $c = ab'$, because $\mathbf{cons}_a M \to^* \mathbf{cons}_a(\mathbf{cons}_{b'} M') \to \mathbf{cons}_{ab'} M'$ for some M'.

(3)(b) $\mathbf{tail}_a$ is computable:

Let M be a computable real program and let $y \ll [\![\mathbf{tail}_a M]\!] = \mathrm{tail}_a([\![M]\!])$ be a non-bottom partial number. We have to produce $c \in \mathrm{Eval}(\mathbf{tail}_a M)$ with $y \sqsubseteq c$. Since tail_a is continuous, there is some $x \ll [\![M]\!]$ such that $y \ll \mathrm{tail}_a(x)$. Let $b \in \mathrm{Eval}(M)$ with $x \sqsubseteq b$. It follows that $b \not\sqsubseteq a$, because $y \neq \bot$ and if $b \sqsubseteq a$ then $y \ll \mathrm{tail}_a(x) \sqsubseteq \mathrm{tail}_a(b) \sqsubseteq \mathrm{tail}_a(a) = \bot$. Then exactly one of the following four cases holds:

(t1) $b \leq a$

(t2) $a \leq b$

(t3) $a \sqsubseteq b$

(t4) $a \uparrow b$, $a \not\sqsubseteq b$, $b \not\leq a$, and $a \not\leq b$

which have the following proofs:

(t1) In this case $\mathbf{tail}_a M \to^* \mathbf{tail}_a(\mathbf{cons}_b M') \to \mathbf{Y}\mathbf{cons}_L$ for some M'. Hence $y \ll [\![\mathbf{tail}_a M]\!] = [\![\mathbf{Y}\mathbf{cons}_L]\!]$. Since $\mathbf{Y}\mathbf{cons}_L$ is a computable term, there is some $c \in \mathrm{Eval}(\mathbf{Y}\mathbf{cons}_L) \subseteq \mathrm{Eval}(\mathbf{tail}_a M)$ with $y \sqsubseteq c$.

(t2) This case is handled similarly.

(t3) Since $x \sqsubseteq b$, we have that $\mathrm{tail}_a(x) \sqsubseteq \mathrm{tail}_a(b) = b \backslash a$, and since $y \sqsubseteq \mathrm{tail}_a(x)$, we have that $y \sqsubseteq b \backslash a$. Therefore we can take $c = b \backslash a$, because $\mathbf{tail}_a M \to^* \mathbf{tail}_a(\mathbf{cons}_b M') \to \mathbf{cons}_{b \backslash a} M'$ for some M'.

(t4) Since $x \sqsubseteq b$, we have that $\mathrm{tail}_a(x) \sqsubseteq \mathrm{tail}_a(b) = (a \sqcup b) \backslash a$, and since $y \sqsubseteq \mathrm{tail}_a(x)$, we have that $y \sqsubseteq (a \sqcup b) \backslash a$. Therefore we can take $c = (a \sqcup$

$b)\backslash a$, because $\mathbf{tail}_a M \to^* \mathbf{tail}_a(\mathbf{cons}_b M') \to \mathbf{cons}_{(a \sqcup b)\backslash a}(\mathbf{tail}_{(a \sqcup b)\backslash b} M')$ for some M'.

(3)(c) $\mathbf{head}_r$ is computable:

Assume that $[\![\mathbf{head}_r M]\!] = \mathrm{head}_r([\![M]\!]) \neq \bot$ for a computable real program M. Then there is an $x \ll [\![M]\!]$ such that either $x < r$ or else $x > r$, because head_r is continuous. Let $c \in \mathrm{Eval}(M)$ with $x \sqsubseteq c$. Then either $c < r$ or else $c > r$. Hence there is some M' such that either $\mathbf{head}_r M \to^* \mathbf{head}_r(\mathbf{cons}_c M') \to \mathbf{tt}$ or else $\mathbf{head}_r M \to^* \mathbf{head}_r(\mathbf{cons}_c M') \to \mathbf{ff}$ respectively. Therefore if $x < r$ then $\mathrm{Eval}(\mathbf{head}_r M) = \mathbf{tt}$ and if $x > r$ then $\mathrm{Eval}(\mathbf{head}_r M) = \mathbf{ff}$.

(3)(d) $\mathbf{pif}$ is computable:

It suffices to show that $\mathbf{pif} LMN$ is computable whenever $[\![L]\!] = \bot$ and M and N are computable programs, because the case $[\![L]\!] \neq \bot$ is immediate. Let $x \ll [\![\mathbf{pif} LMN]\!]$. Then $x \ll [\![M]\!] \sqcap [\![N]\!]$. Hence $x \ll [\![M]\!]$ and $x \ll [\![N]\!]$. Let $a \in \mathrm{Eval}(M)$ and $b \in \mathrm{Eval}(N)$ with $x \sqsubseteq a$ and $x \sqsubseteq b$. Then $x \sqsubseteq a \sqcap b$ and $a \sqcap b \in \mathrm{Eval}(\mathbf{pif} LMN)$, because $\mathbf{pif} LMN \to^* \mathbf{pif} L(\mathbf{cons}_a M)(\mathbf{cons}_b N) \to \mathbf{cons}_{a \sqcap b}(\mathbf{pif} L(\mathbf{cons}_{a\backslash(a \sqcap b)} M)(\mathbf{cons}_{b\backslash(a\sqcap b)} N))$.

This concludes the proof of Lemma 11.6. □

The proof for tail_a implicitly shows that a reduction rule $\mathbf{tail}_a(\mathbf{cons}_a M) \to M$ would be useless. This can be explicitly explained as follows. If M denotes $\bot$, so does $\mathbf{tail}_a(\mathbf{cons}_a M)$. Hence there is no point in reducing $\mathbf{tail}_a(\mathbf{cons}_a M)$ before reducing M to a head-normal form. If M has a head-normal form $\mathrm{cons}_c M'$, then $\mathbf{tail}_a(\mathbf{cons}_a M)$ reduces to $\mathbf{tail}_a(\mathbf{cons}_{ac} M')$, which in turn reduces to $\mathrm{cons}_c M'$. In practice, however, this rule (and possibly more rules) can be included for efficiency reasons.

11.2 The programming language Real PCF

We let $\mathcal{L}^R$ denote the extension of $\mathcal{L}^I$ with a new ground type $\mathbf{R}$ and the following new constants:

1. $\mathbf{rrcons}_a : \mathbf{R} \to \mathbf{R}$
2. $\mathbf{ricons}_a : \mathbf{I} \to \mathbf{R}$
3. $\mathbf{irtail}_a : \mathbf{R} \to \mathbf{I}$
4. $\mathbf{rhead}_r : \mathbf{R} \to \mathbf{T}$
5. $\mathbf{pif}_{\mathbf{R}} : (\mathbf{T}, \mathbf{R}, \mathbf{R}, \mathbf{R})$

for each $a \in \mathcal{R}$ with distinct rational end-points and each rational number r.

We let $D_{\mathbf{R}} = \mathcal{R}_\bot$ and we extend the standard interpretation $\mathcal{A}$ of $\mathcal{L}^I$ to $\mathcal{L}^R$ in the obvious way:

1. $\mathcal{A}[\![\mathbf{rrcons}_a]\!] = \mathrm{rrcons}_a$
2. $\mathcal{A}[\![\mathbf{ricons}_a]\!] = \mathrm{ricons}_a$
3. $\mathcal{A}[\![\mathbf{irtail}_a]\!] = \mathrm{irtail}_a$

4. $\mathcal{A}[\![\mathbf{rhead}_r]\!]x = \mathrm{rhead}_r$

5. $\mathcal{A}[\![\mathbf{pif}_{\mathbf{R}}]\!]pxy = \mathrm{pif}\ p\ \mathrm{then}\ x\ \mathrm{else}\ y.$

The reduction rules for $\mathcal{L}^R$ are given by Lemma 9.20, with the same restrictions as for the reduction rules for $\mathcal{L}^I$. The Computability Lemma 11.6 and the Adequacy Theorem 11.5 routinely generalize to $\mathcal{L}^R$.

Definition 11.7 ***Real PCF*** is the programming language $\mathcal{L}^R$ for $\mathcal{L} = \mathcal{L}_{DA}$. □

Theorem 11.8 (Adequacy) *Real PCF is computationally adequate.*

Chapter 12

Computational completeness of Real PCF

In Section 12.1 we show that there is a unique effective presentation of the partial real line such that the basic operations for real number computation are computable, up to equivalence in the sense of Chapter 6 of Part II. In Section 12.2 we discuss a technique introduced by Thomas Streicher in order to establish computational completeness; the technique relies of the fact that PCF extended with $\exists$ is known to be computationally complete. In Section 12.3 we apply the technique to show that Real PCF extended with $\exists$ is computationally complete. A limitation of this technique is that it does not give information about definability in the case that $\exists$ is not available. In Section 12.4 we discuss a new technique for establishing computational completeness. As a corollary, in Section 12.5 we obtain a very simple new proof of computational completeness of PCF extended with $\exists$. The technique works in its full generality only for algebraic domains in its present form. However, in Section 12.6 we are able to apply the technique to show that $\exists$ is not necessary to define the *first-order* computable functions in Real PCF, as it is the case for PCF, thus obtaining more information about definability.

12.1 The unique sound effective presentation of the partial real line

Recall that we defined a functor $\mathbf{T} : \mathbf{SDom} \to \mathbf{SDom}$ by

$$\mathbf{T}\, D = \mathcal{B} \times D \times D$$

(Definition 10.19). As we have seen in Chapter 10, the carrier of the bifree $\mathbf{T}$-algebra is the domain of infinite binary trees with nodes labelled by truth-values. Since $\mathcal{B}^\omega$ is isomorphic to this domain, there is a bifree $\mathbf{T}$-algebra with $\mathcal{B}^\omega$ as its carrier.

Lemma 12.1 *There is a computable bifree algebra* $\mathrm{mktree} : \mathbf{T}\,\mathcal{B}^\omega \to \mathcal{B}^\omega$.

Proof $\mathcal{B}^\omega$ is the canonical solution to the domain equation $D \cong \mathcal{B} \times D$. A bifree algebra is given by cons : $\mathcal{B} \times \mathcal{B}^\omega \to \mathcal{B}^\omega$ defined in Proposition 6.10.

Since Pair : $\mathcal{B}^\omega \to \mathcal{B}^\omega \times \mathcal{B}^\omega$ defined in Section 3.4 is an isomorphism, so is $\mathrm{mktree}(t,p,q) = \mathrm{cons}(t, \mathrm{Pair}^{-1}(p,q))$. This isomorphism is clearly computable. A routine verification shows that $\langle \mathrm{mktree}^{-1}, \mathrm{mktree}\rangle$ is a $\mathbf{T}$-inductive isomorphism pair (cf. Definition 5.2). By Proposition 5.3, mktree is an initial algebra. □

Now, recall that we defined maps

$$\mathcal{I} \underset{\mathrm{destr}}{\overset{\mathrm{cons}}{\leftrightarrows}} \mathbf{T}\mathcal{I}$$

by

$$\begin{aligned} \mathrm{cons}(p,y,z) &= \mathrm{pif}\ x \ \mathrm{then}\ \mathrm{cons}_L(y)\ \mathrm{else}\ \mathrm{cons}_R(z), \\ \mathrm{destr}(x) &= \langle \mathrm{head}(x), \mathrm{tail}_L(x), \mathrm{tail}_R(x)\rangle \end{aligned}$$

(Definition 10.19) and that these maps were shown to form a $\mathbf{T}$-inductive section-retraction pair in Theorem 10.20. Recall also that we defined num to be the unique algebra homomorphism from mktree to cons, and bin to be the unique coalgebra homomorphism from destr to mktree^{-1} (Definition 10.26), and that $\langle \mathrm{bin}, \mathrm{num}\rangle$ is a section-retraction pair by virtue of Proposition 5.5.

Theorem 12.2 *There is an effective presentation of $\mathcal{I}$ which makes* cons_L, cons_R, head, tail_L, *and* tail_R *computable.*

Proof By Theorem 10.32, the idempotent norm = bin ∘ num is computable w.r.t. the standard effective presentation of $\mathcal{B}^\omega$. Therefore $\langle \mathrm{bin}, \mathrm{num}\rangle$ is an effective presentation of $\mathcal{I}$. By Lemmas 10.31 and 10.33, the above functions are computable. □

Theorem 12.3 *Any two effective presentations of $\mathcal{I}$ which make* cons_L, cons_R, head, tail_L, *and* tail_R *computable are equivalent.*

Proof Any two effective presentations which make the above maps computable also make destr and cons computable. Therefore the result follows from Theorem 6.12. □

Similarly,

Theorem 12.4 *There is an effective presentation of $\mathcal{R}_\perp$ which makes* rcons_L, rcons_R, rhead_0, rhead_1, rtail_L, rtail_R, incl *and* trunc *computable.*

Theorem 12.5 *Any two effective presentations of $\mathcal{R}_\perp$ which make* rcons_L, rcons_R, rhead_0, rhead_1, rtail_L, *and* rtail_R, incl *and* trunc *computable are equivalent.*

Convention 12.6 Whenever we implicitly or explicitly speak about computability on $\mathcal{I}$ or $\mathcal{R}_\perp$, we assume any effective presentation which make the above operations computable. □

12.2 Computational completeness via universal domains

Definition 12.7 We say that a programming language is ***computationally complete*** if every computable element of its universe of discourse is definable in the language. □

The term *universal* is also used to refer to this property (because of universal Turing machines), but we avoid it as it is very overloaded (e.g. universal domain, universal property of a categorical construction).

We prove that Real PCF extended with $\exists$ is computationally complete by means of a technique introduced by Thomas Streicher [Str94] in order to prove that PCF extended with recursive types, parallel-or and $\exists$ is computationally complete. Here are the main steps of his proof:

1. Show that every definable element is computable.
2. Take a universal domain $\mathcal{U}$ of PCF, e.g. $(\mathcal{N} \Rightarrow \mathcal{B})$.
3. Show that for every domain D in the extended language there is a definable section-retraction pair
$$D \underset{s_D}{\overset{r_D}{\leftrightarrows}} \mathcal{U}$$
4. Given $d \in D$ computable, $s_D(d) \in \mathcal{U}$ is computable because s_D is computable.
5. Since PCF extended with parallel-or and $\exists$ is computationally complete [Plo77] and $\mathcal{U}$ is a PCF domain, $s_D(d)$ is definable.
6. Hence so is d, because $d = r_D(s_D(d))$.
7. Therefore every computable element is definable.

The crucial step consists in showing that D is a definable retract of $\mathcal{U}$, and this is not so simple in the presence of recursive types. But in our case the situation is very simple:

Lemma 12.8 *Let PCF' be any extension of PCF with parallel-or, $\exists$, new ground types and new constants such that*

1. *Every PCF'-definable element is computable.*
2. *Every ground domain of PCF' is a definable retract of a PCF universal domain $\mathcal{U}$ such that $\mathcal{U} \Rightarrow \mathcal{U}$ is a computable retract of $\mathcal{U}$.*

Then PCF' is computationally complete.

Proof In order to apply Streicher's argument, it suffices to show by induction on types that every domain D in PCF' is a definable retract of $\mathcal{U}$. The base case holds by hypothesis. For the inductive step, assume that D and E are definable retracts of $\mathcal{U}$. We have to show that $D \Rightarrow E$ is a definable retract of $\mathcal{U}$. The function space $D \Rightarrow E$ is a retract of $\mathcal{U} \Rightarrow \mathcal{U}$ as indicated below,

$$D \underset{r_E \Rightarrow s_D}{\overset{s_E \Rightarrow r_D}{\leftrightarrows}} (\mathcal{U} \Rightarrow \mathcal{U}),$$

because

$$\begin{aligned} (s_E \Rightarrow r_D) \circ (r_E \Rightarrow s_D)(f) &= (s_E \Rightarrow r_D)(s_D \circ f \circ r_E) \\ &= r_D \circ s_D \circ f \circ r_E \circ s_E \\ &= \mathrm{id} \circ f \circ \mathrm{id} = f. \end{aligned}$$

Let

$$(\mathcal{U} \Rightarrow \mathcal{U}) \underset{s_\mathcal{U}}{\overset{r_\mathcal{U}}{\leftrightarrows}} \mathcal{U},$$

be a computable section-retraction pair. Then

$$D \underset{s_\mathcal{U} \circ (r_E \Rightarrow s_D)}{\overset{(s_E \Rightarrow r_D) \circ r_\mathcal{U}}{\leftrightarrows}} \mathcal{U},$$

is a section-retraction pair, because section-retraction pairs compose. Therefore $D \Rightarrow E$ is a definable retract of $\mathcal{U}$, because

$$\begin{aligned} (s_E \Rightarrow r_D) \circ r_\mathcal{U}(u) &= r_D \circ r_\mathcal{U}(u) \circ s_E, \\ s_\mathcal{U} \circ (r_E \Rightarrow s_D)(f) &= s_\mathcal{U}(s_D \circ f \circ r_E), \end{aligned}$$

and $s_\mathcal{U}$ and $r_\mathcal{U}$ are definable by computational completeness of PCF extended with parallel-or and $\exists$.

$\square$

12.3 Computational completeness of Real PCF

Definition 12.9 In PCF we don't have product types, and hence we have to work with curried maps in order to capture maps $C \times D \to E$, and we have to work with pairs of maps in order to capture maps $C \to D \times E$. In order to avoid unnecessary detailed constructions in the following proofs, we say that a map is ***essentially definable*** if we can define it by eliminating product types in this (standard) way. $\square$

The uniqueness part of the following theorem appears to be new, but it is certainly not unexpected:

Theorem 12.10 *There is a unique notion of computability for PCF such that every PCF definable element is computable.*

By this we mean that there is a unique effective presentation of the PCF domains with the above property, up to equivalence.

Proof In [Plo77] it is shown that the following enumerations are effective presentation of the PCF ground domains:

$$\begin{array}{llll} B_{\mathcal{B}} = \mathcal{B} & b_0^{\mathcal{B}} = \bot & B_{\mathcal{N}} = \mathcal{N} & b_0^{\mathcal{N}} = \bot \\ & b_1^{\mathcal{B}} = \mathbf{tt} & & b_{n+1}^{\mathcal{N}} = n, \\ & b_{n+2}^{\mathcal{B}} = \mathbf{ff} & & \end{array}$$

and it is also shown that the elements of $\mathcal{L}_{\mathrm{DA}}$, $\mathcal{L}_{\mathrm{PA}}$ and $\mathcal{L}_{\mathrm{PA}+\exists}$ are computable with respect to it.

Uniqueness for $\mathcal{B}$ follows from the fact that it is a finite domain. In order to establish uniqueness for $\mathcal{N}$, recall that it is the canonical solution of the domain equation $\mathcal{N} \cong 1+\mathcal{N}$. Then we can apply Theorem 6.11, because the (standard) bifree algebra and its inverse are essentially definable.

Finally we can apply Proposition 6.8 to lift the effective presentations to higher-order types in a unique way, up to equivalence. □

This result extends to Real PCF:

Theorem 12.11 *There is a unique notion of computability for the real numbers type hierarchy such that every Real PCF definable element is computable.*

Proof Immediate consequence of Theorems 12.2, 12.3, 12.4 and 12.5, and Proposition 6.8. □

Proposition 12.12 *$\exists$ is computable but not Real PCF definable.*

Proof The inductive proof given in [Plo77] for the fact that $\exists$ is not definable in PCF extended with parallel-or goes through for Real PCF, by adding obvious inductive steps for the Real PCF constants. □

Theorem 12.13 *Real PCF extended with $\exists$ is computationally complete.*

Proof By virtue of Lemma 12.8, it suffices to show that every ground type is a definable retract of some universal domain of PCF. The maps

$$\mathcal{B}_\omega \underset{e}{\overset{p}{\leftrightarrows}} (\mathcal{N} \Rightarrow \mathcal{B})$$

defined by

$$\begin{aligned} e(x)(n) &= \begin{cases} \bot & \text{if } n = \bot, \\ x_n & \text{otherwise,} \end{cases} \\ p(f)(i) &= f(i) \end{aligned}$$

form a section-retraction pair (with the strict functions $\mathcal{N} \to \mathcal{B}$ as the fixed-points of the induced idempotent). Hence $\mathcal{U} = (\mathcal{N} \Rightarrow \mathcal{B})$ is a universal domain. It is immediate that $\mathcal{B}$ and $\mathcal{N}$ are definable retracts of $\mathcal{U}$.

In order to show that $\mathcal{I}$ is a definable retract of $\mathcal{U}$, we apply Lemma 5.7. Define

$$\begin{aligned} \mathrm{mktree}' &= e \circ \mathrm{mktree} \circ \mathbf{T}\, p, \\ \mathrm{dstree}' &= \mathbf{T}\, e \circ \mathrm{dstree} \circ p, \end{aligned}$$

where dstree $=$ mktree^{-1}. Then mktree$'$ and dstree$'$ are easily seen to be essentially definable. Define bin : $\mathcal{I} \to \mathcal{U}$ and num : $\mathcal{U} \to \mathcal{I}$ recursively by

$$\begin{array}{rcl} \mathrm{bin}' & = & \mathrm{cons} \circ \mathbf{T}\,\mathrm{bin}' \circ \mathrm{dstree}' \\ \mathrm{num}' & = & \mathrm{mktree}' \circ \mathbf{T}\,\mathrm{num}' \circ \circ \mathrm{destr}. \end{array}$$

By Lemma 5.7, $\langle \mathrm{bin}', \mathrm{num}' \rangle$ is a section-retraction pair. Since cons, destr, mktree$'$ and dstree are essentially definable, bin$'$ and num$'$ are (strictly) definable. Therefore $\mathcal{I}$ is a definable retract of $\mathcal{U}$.

$\mathcal{R}_\perp$ is treated similarly. □

There is another way of showing that $\mathcal{I}$ and $\mathcal{R}_\perp$ are definable retracts of $\mathcal{U}$, based on Theorems 7.15 and 7.17. If we show that the maps join_a^D and way-below_a^D, for $D \in \{\mathcal{I}, \mathcal{R}_\perp\}$ are definable uniformly in the Gödel number of a, then we can define

$$s(n)(x) = \text{way-below}_n^D(x)$$

and the proof of Theorem 7.15 gives a construction of the desired definable retraction. This is done in Section 12.6 below, with the purpose of obtaining more information about Real PCF definability.

12.4 Computational completeness via joining maps

In this section we introduce a general technique for establishing universality of extensions of PCF with ground types interpreted as algebraic coherently complete domains. It particular, we obtain a new proof of universality of PCF extended with the parallel conditional and the existential quantifier. It is an open problem to extend the technique to the continuous case, but the technique is intrinsically restricted to coherently complete domains, due to the results of Chapter 7.1.

In this section we work with effectively given algebraic domains in the sense of Definition 3.5.

Let $\mathcal{L}_P$ be an extension of $\mathcal{L}_{\mathrm{DA}}$ with new ground types and a set of new constants $P \supseteq \mathrm{DA}$, and consider any extension of the standard interpretation of $\mathcal{L}_{\mathrm{DA}}$ to $\mathcal{L}_P$ such that

1. The interpretation of ground types is given by coherently complete algebraic domains.

2. The interpretation is ***computationally feasible***, in the sense that there is an effective presentation of the new ground domains such that every $\mathcal{L}_P$-definable element is computable w.r.t. the induced effective presentations at higher-order types.

Definition 12.14 A J_P-domain is a domain D for which there is an $\mathcal{L}_P$-definable function

$$\begin{array}{ccc} \mathcal{N} & \to & (D \Rightarrow D) \\ n & \mapsto & \mathrm{join}_n^D \\ \perp & \mapsto & \perp \end{array}$$

such that join_n^D is a joining map of b_n^D. $\square$

Lemma 12.15 *If D is a J_P-domain then every computable element of D is $\mathcal{L}_P$-definable.*

Proof Let $x \in D$ be computable. Then there is a primitive recursive function $k \mapsto n_k$ such that $x = \bigsqcup_{k \in \mathbb{N}} b_{n_k}^D$. Let Φ be the endomap on the function space

$$(\mathcal{N} \Rightarrow (D \Rightarrow D)) \Rightarrow D$$

defined by

$$\Phi(F)(f) = f(0)(F(n \mapsto f(n+1))),$$

for $F \in (\mathcal{N} \Rightarrow (D \Rightarrow D)) \Rightarrow D$ and $f \in \mathcal{N} \Rightarrow (D \Rightarrow D)$. Then $\Phi^m(\bot)(f) = f(0) \circ f(1) \circ \cdots \circ f(m)(\bot)$. Hence

$$\begin{aligned}
\bigsqcup_{m \in \mathbb{N}} \Phi^m(\bot)\left(k \mapsto \mathrm{join}_{n_k}^D\right) &= \bigsqcup_{m \in \mathbb{N}} \mathrm{join}_{n_0}^D \circ \mathrm{join}_{n_1}^D \circ \cdots \circ \mathrm{join}_{n_m}^D(\bot) \\
&= \bigsqcup_{m \in \mathbb{N}} b_{n_0}^D \sqcup b_{n_1}^D \sqcup \cdots \sqcup b_{n_m}^D \\
&= x,
\end{aligned}$$

and thus x is the least fixed point of Φ applied to the map $k \mapsto \mathrm{join}_{n_k}^D$. Therefore x is $\mathcal{L}_P$-definable, because Φ is $\mathcal{L}_P$-definable. $\square$

We let way-below_n^D be a short hand for $\text{way-below}_{b_n^D}^D$. Recall that the way-below maps were defined in the proof of Theorem 7.17.

Definition 12.16 A W_P-domain is a domain D such that the map

$$\begin{array}{rcl}
\mathcal{N} & \to & (D \Rightarrow \mathcal{B}) \\
n & \mapsto & \text{way-below}_n^D \\
\bot & \mapsto & \bot
\end{array}$$

is $\mathcal{L}_P$-definable. $\square$

Lemma 12.17 *If D is a W_P-domain and E is a J_P-domain then $D \Rightarrow E$ is a $J_{P+\mathrm{pif}}$-domain.*

Proof The construction given in Theorem 7.17 shows that there is a definable map $n \mapsto \mathrm{join}_n^{D \Rightarrow E}$ with the desired property. First, there is a $\mathcal{L}_{P+\mathrm{pif}}$-definable map $n \mapsto \mathrm{sjoin}_n^{D \Rightarrow E}$ such that $\mathrm{sjoin}_n^{D \Rightarrow E}$ is a joining map of the subbasis element s_n^D, because

$$\mathrm{sjoin}_{\langle n,m \rangle}^{D \Rightarrow E}(f)(x) = \mathrm{pif}^E\ \text{way-below}_n^D(x)\ \text{then}\ \mathrm{join}_m^E(f(x))\ \text{else}\ f(x).$$

Hence $n \mapsto \mathrm{join}_n^{D \Rightarrow E}$ can be recursively given by

$$\mathrm{join}_n^{D \Rightarrow E}(f)(x) \quad = \quad \text{if ``} b_n^{D \Rightarrow E} = \bot \text{'' then } f(x) \text{ else } \mathrm{sjoin}_{\delta(n)}^{D \Rightarrow E}(\mathrm{join}_{\rho(n)}^{D \Rightarrow E}(f)(x)).$$

where δ and ρ are recursive functions specified immediately after Definition 3.1, and "$b_n^{D \Rightarrow E} = \bot$" is some term defining the recursive predicate $b_n^{D \Rightarrow E} = \bot$ of n. $\square$

Define the ***bounded universal quantifier***

$$\begin{array}{ccccc} \mathcal{N} & \to & (\mathcal{N} \;\Rightarrow\; \mathcal{B}) & \to & \mathcal{B} \\ n & \mapsto & p & \mapsto & \forall_{x\in\Delta_n}\ p(x) \end{array}$$

by

$$\forall_{x\in\Delta_n}\ p(x) = \begin{cases} \mathbf{tt} & \text{if } p(x) = \mathbf{tt} \text{ for all } x \in \Delta_n, \\ \mathbf{ff} & \text{if } p(x) = \mathbf{ff} \text{ for some } x \in \Delta_n, \\ \perp & \text{otherwise,} \end{cases}$$

where Δ_n is specified in Definition 3.1.

Lemma 12.18 *The bounded universal quantifier is $\mathcal{L}_{\mathrm{PA}}$-definable.*

Proof Recursively define the bounded quantifier by

$$\forall_{x\in\Delta_n}\ p(x) = \text{“}\Delta_n = \emptyset\text{”} \vee \left(p(\delta(n)) \wedge \forall_{x\in\Delta_{\rho(n)}}\ p(x)\right),$$

where δ and ρ are specified immediately after Definition 3.1, “$\Delta_n = \emptyset$” stands for the equality test $n =_\perp 0$, and $\vee$ and $\wedge$ are parallel-or and parallel-and [Plo77]. Actually, a sequential version of disjunction would be enough. But parallel conjunction is necessary, because $p(x)$ can be $\perp$ for some $x \in \Delta_n$ and $\mathbf{ff}$ for another $x \in \Delta_n$, and one has to ensure that in such cases $\forall_{x\in\Delta_n}\ p(x) = \mathbf{ff}$. □

The following definability lemma depends on the fact that the function space $D \Rightarrow E$ is an algebraic domain, as it makes use of Lemmas 7.18 and 3.10:

Lemma 12.19 *If D is a J_P-domain and E is a W_P-domain then $D \Rightarrow E$ is a $W_{P+\mathrm{pif}+\exists}$-domain.*

Proof If D is a J_P-domain then there is a $\mathcal{L}_P$-definable function

$$\begin{array}{ccc} \mathcal{N} & \to & (\mathcal{N} \to D) \\ n & \mapsto & \mathrm{up}_n^D \end{array}$$

such that the range of up_n^D is $\uparrow b_n^D \cap B_D$ (and therefore $\mathrm{up}_n^D(\perp) = b_n^D$), given by

$$\mathrm{up}_n^D(m) \quad = \quad \mathrm{join}_n^D \circ \mathrm{join}_m^D(\perp).$$

By Lemmas 7.18 and 3.10,

$$\text{way-below}_n^{D\Rightarrow E}(f) = \forall_{\langle a,b\rangle\in\Delta_n}\ \forall_{k\in\mathcal{N}}\ \text{way-below}_b^E(f(\mathrm{up}_a^D(k))),$$

where $\forall_{k\in\mathcal{N}} e = \neg\exists\lambda k.\neg e$. This establishes the $\mathcal{L}_{P+\mathrm{pif}+\exists}$-definability of the map $n \mapsto \text{way-below}_n^{D\Rightarrow E}$. □

In joint work with Thomas Erker and Klaus Keimel [EEK97], we have obtained a characterization of the way-below order on function spaces $X \to E$ where X is a coherent space and E is a bounded complete domain. However, we haven't had the opportunity to consider the application of this characterization to the extension of the method of proof introduced in this section to continuous domains.

12.5 Computational completeness of PCF revisited

In [Plo77] it is shown that $\mathcal{L}_{\mathrm{DA}}$ and $\mathcal{L}_{\mathrm{PA}}$ are not computationally complete, but that $\mathcal{L}_{PA+\exists}$ is. The proof given in [Plo77] is based on the fact that the ground domains are flat. We offer another proof, based on the general results of the previous section.

Lemma 12.20 *Each PCF ground domain is a J_{DA}-domain and a W_{DA}-domain.*

Proof Recall from Proposition 7.2 that every element of a flat domain has a unique joining map, and that it is the identity for the bottom element and a constant function for a non-bottom element. The map $\mathrm{join}_n^{\mathcal{B}}$ is given by

$$\mathrm{join}_n^{\mathcal{B}}(x) = \text{if } x =_\perp 0 \text{ then } x \text{ else } ((n-1) =_\perp 0).$$

The map $\mathrm{join}_n^{\mathcal{N}}$ is given by

$$\mathrm{join}_n^{\mathcal{N}}(x) = \text{if } x =_\perp 0 \text{ then } x \text{ else } n-1.$$

Therefore $n \mapsto \mathrm{join}_n^D$ is DA-definable for D ground. The map $\text{way-below}_n^{\mathcal{B}}$ is given by

$$\text{way-below}_n^{\mathcal{B}}(x) = n =_\perp 0 \vee \text{if } n =_\perp 1 \text{ then } x \text{ else } \neg x.$$

The map $\text{way-below}_n^{\mathcal{N}}$ is given by

$$\text{way-below}_n^{\mathcal{N}}(x) = (n =_\perp 0) \vee ((n-1) =_\perp x).$$

Therefore $n \mapsto \text{way-below}_n^D(x)$ is DA-definable for D ground. $\square$

Theorem 12.21 *$\mathcal{L}_{PA+\exists}$ is computationally complete. Moreover, every first-order computable function is $\mathcal{L}_{\mathrm{PA}}$-definable.*

Proof Every domain is a $\mathrm{J}_{\mathrm{PA}+\exists}$-domain and a $\mathrm{W}_{\mathrm{PA}+\exists}$-domain, by a simultaneous induction on the formation rules of the PCF domains; Lemma 12.20 establishes the base cases, and Lemmas 12.17 and 12.19 establish the inductive steps. Then the result follows from Lemma 12.15. First-order computational completeness follows from the fact that the existential quantifier is needed only in the hypothesis of Lemma 12.19. $\square$

12.6 Computational completeness of Real PCF revisited

We know that Real PCF extended with $\exists$ is computationally complete. In this section we show that $\exists$ is not necessary to define the first-order computable functions, as it is the case for PCF. For the sake of brevity, we only treat the type $\mathcal{I}$, as the treatment of $\mathcal{R}_\perp$ is analogous.

We assume the notion of effectively given domain given in [Smy77] via enumerations of basis elements. Essentially, the axioms are the same as for effectively given algebraic domains via enumerations of basis elements as in Definition 3.5, with some technical axioms which ensure that we obtain a cartesian closed category. Since these technical axioms are not important for our purposes as we are concerned with first-order types only, we adopt the naïve approach.

Recall that an enumeration $\{r_n\}_{n \in \mathbb{N}}$ of the rational numbers is standard if there are recursive functions $n \mapsto s_n$, $n \mapsto p_n$ and $n \mapsto q_n$ such that $r_n = (-1)^{s_n}\, p_n/q_n$.

Lemma 12.22 *An enumeration $\{r_n\}_{n \in \mathbb{N}}$ of the rational numbers is standard iff the basic four operations and the (in)equality predicates are recursive with respect to it.*

For example, Cantor's enumeration of the rational numbers is standard. In this section $\{r_n\}_{n \in \mathbb{N}}$ is any standard enumeration of the rational numbers contained in the unit interval.

Let B be the basis of $\mathcal{I}$ consisting of intervals with (not necessarily distinct) rational end-points. Then the following enumeration of B gives rise to an effective presentation of $\mathcal{I}$:

$$b_{\langle m,n \rangle} = [\min(r_m, r_n), \max(r_m, r_n)]$$

By Lemma 12.22, we see that the basic primitives for real number computation are computable w.r.t. this presentation.

Lemma 12.23 *The (strict) map $n \mapsto b_n : \mathcal{N} \to \mathcal{I}$ is Real PCF definable.*

Proof By Lemma 12.22, for each $a \in \{L, R\}$ there are recursive functions g_a and h such that

$$\begin{aligned} b_{g_a(n)} &= \mathrm{tail}_a(b_n^{\mathcal{I}}) \\ h(n) &= \mathrm{head}(b_n^{\mathcal{I}}), \end{aligned}$$

and we have that

$$b_n \;=\; \text{if } h(n) \text{ then } \mathrm{cons}_L(b_{g_L(n)}) \text{ else } \mathrm{cons}_R(b_{g_R(n)}). \;\Box$$

Lemma 12.24 *$\mathcal{I}$ is a J_{RA}-domain and W_{RA}-domain.*

Proof It is not hard to see that for every a,

$$\begin{aligned}\text{join}_a^{\mathcal{I}}(x) &= \max(\underline{a}, \min(x, \overline{a})),\\ \text{way-below}_a^{\mathcal{I}}(x) &= (\underline{a} = 0 \vee \underline{a} <_\perp x) \wedge (x <_\perp \overline{a} \vee \overline{a} = 1),\end{aligned}$$

and also that there are recursive maps f and g such that $b_{f_D(n)} = \underline{b_n}$ and $b_{g_D(n)} = \overline{b_n}$. Then $n \mapsto \text{join}_n^{\mathcal{I}}$ and $n \mapsto \text{way-below}_n^{\mathcal{I}}$ are given by

$$\begin{aligned}\text{join}_n^{\mathcal{I}}(x) &= \max(b_{f(n)}, \min(x, b_{g(n)})),\\ \text{way-below}_n^{\mathcal{I}}(x) &= (b_{f(n)}^{\mathcal{I}} = 0 \vee b_{f(n)}^{\mathcal{I}} <_\perp x) \wedge (x <_\perp b_{g(n)} \vee b_{g(n)} = 1). \square\end{aligned}$$

Since $\mathcal{R}_\perp$ can be treated similarly, Real PCF is ***first-order computationally complete***, in the following sense:

Theorem 12.25 *All first-order computable elements of the real numbers type hierarchy are Real PCF definable.*

Proof Lemmas 12.24, 12.17 and 12.15. $\square$

In joint work with Abbas Edalat [EE96b], we have extended the above method to second-order functions, by obtaining a suitable characterization of the way-below order on the function space $\mathcal{R}_\perp \to \mathcal{R}_\perp$; but notice that the existential quantifier is still needed for second-order functions, because function spaces are handled by Lemma 12.19, which uses the existential quantifier.

Part V

Integration on the partial real line

This part is based on joint work with Abbas Edalat [EE95, EE96a]. In Chapter 13 we introduce interval Riemann integration. In Chapter 14 we show how to handle interval Riemann integration in Real PCF.

Chapter 13

Interval Riemann integrals

A generalization of the Riemann theory of integration based on domain theory was introduced in [Eda95b]. Essentially, a domain-theoretic framework for the integration of real-valued functions w.r.t. any finite measure on a compact metric space was constructed using the probabilistic power domain of the upper space of the metric space. In this work we are only concerned with integration w.r.t. the Lebesgue measure (uniform distribution) in $\mathbb{R}^n$.

In order to extend Real PCF with integration, we embark on a novel approach compared to [Eda95b] for integration w.r.t. the Lebesgue measure in $\mathbb{R}$, in that we consider integration of maps of type $\mathcal{R}^n \to \mathcal{R}$ rather than $\mathbb{R}^n \to \mathbb{R}$. We deduce various properties of integration defined in this way, which are interesting in their own right as well.

In Section 13.1 we introduce simple interval Riemann integration. In Section 13.2 we introduce multiple Riemann integration, which is related to simple interval Riemann integration via a generalization of the so-called Fubini's rule. In Section 13.3 we introduce a supremum operator, which is used in Chapter 14 to obtain a fixed-point definition of Riemann integration.

13.1 Simple interval Riemann Integrals

Recall that addition in $\mathcal{R}$ is defined by

$$x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}],$$

and that given a real number $\alpha \geq 0$ and an interval x, we have that

$$x\alpha = \alpha x = [\underline{x}\alpha, \overline{x}\alpha].$$

In this chapter it will be convenient to denote the diameter κ_x of an interval $x \in \mathcal{R}$ by $\mathrm{d}x$:

$$\mathrm{d}x = \overline{x} - \underline{x}.$$

A ***partition*** of an interval $[a, b]$ is a finite set of the form

$$P = \{[a, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n], [x_n, b]\}.$$

We denote by $\mathcal{P}[a,b]$ the set of all partitions of $[a,b]$. A partition Q ***refines*** a partition P if Q is obtained by partitioning some elements of P, in the sense that there is a (necessarily unique) family $\{Q_x\}_{x \in P}$ such that Q is its disjoint union and Q_x is a partition of x for each $x \in P$. Such a family is called the ***refinement witness***.

Lemma 13.1 *$\mathcal{P}[a,b]$ is directed by the refinement order. That is, for any two partitions of $[a,b]$ there is a third partition refining both.*

Definition 13.2 Let $f : \mathcal{R} \to \mathcal{R}$ be a map and $[a,b]$ be an interval. An ***interval Riemann sum*** of f on $[a,b]$ is a sum of the form

$$\sum_{x \in P} f(x)\mathrm{d}x \qquad \text{for } P \in \mathcal{P}[a,b]. \ \square$$

Lemma 13.3 *Let $f : \mathcal{R} \to \mathcal{R}$ be a* monotone *map* (w.r.t. the information order). *If a partition Q of an interval $[a,b]$ refines a partition P then*

$$\sum_{x \in P} f(x)\mathrm{d}x \sqsubseteq \sum_{x \in Q} f(x)\mathrm{d}x.$$

Therefore, the set of interval Riemann sums of f on $[a,b]$ is directed.

Proof If two compact intervals x_1 and x_2 just touch then

$$\begin{aligned} f(x_1 \sqcap x_2)\mathrm{d}(x_1 \sqcap x_2) &= f(x_1 \sqcap x_2)(\mathrm{d}x_1 + \mathrm{d}x_2) \\ &= f(x_1 \sqcap x_2)\mathrm{d}x_1 + f(x_1 \sqcap x_2)\mathrm{d}x_2 \\ &\sqsubseteq f(x_1)\mathrm{d}x_1 + f(x_2)\mathrm{d}x_2. \end{aligned}$$

By induction, if successive elements of the sequence $x_1, \ldots, x_n$ just touch then

$$f\Big(\prod_{k=1}^{n} x_k\Big)\mathrm{d}\Big(\prod_{k=1}^{n} x_k\Big) \quad \sqsubseteq \quad \sum_{k=1}^{n} f(x_k)\mathrm{d}x_k.$$

Hence, if $\{Q_x\}_{x \in P}$ is the refinement witness, then for any $x \in P$,

$$f(x)\mathrm{d}x \quad \sqsubseteq \quad \sum_{y \in Q_x} f(y)\mathrm{d}y,$$

because $x = \prod Q_x$. By monotonicity of addition and induction on the size of P,

$$\sum_{x \in P} f(x)\mathrm{d}x \quad \sqsubseteq \quad \sum_{x \in P} \sum_{y \in Q_x} f(y)\mathrm{d}y.$$

Since Q is the disjoint union of the sets Q_x and addition is associative,

$$\sum_{x \in P} \sum_{y \in Q_x} f(y)\mathrm{d}y \quad = \quad \sum_{y \in Q} f(y)\mathrm{d}y. \ \square$$

Definition 13.4 The ***interval Riemann integral*** of a *monotone* map $f : \mathcal{R} \to \mathcal{R}$ on an interval $[a, b]$ is defined by

$$\int_a^b f = \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} f(x)\mathrm{d}x.$$

We sometimes denote $\int_a^b f$ by $\int_a^b f(x)\mathrm{d}x$. □

Proposition 13.5 *For all continuous maps $f, g : \mathcal{R} \to \mathcal{R}$ and all real numbers α and β,*

$$\begin{aligned} \int_a^a f &= 0, \\ \int_a^b f + \int_b^c f &= \int_a^c f, \\ \int_a^b (\alpha f + \beta g) &= \alpha \int_a^b f + \beta \int_a^b g. \end{aligned}$$

Proof The first equation follows from the fact that $\{[a, a]\}$ is the only partition of $[a, a]$. If P and Q are partitions of $[a, b]$ and $[b, c]$ respectively, then $P \cup Q$ is a partition of $[a, c]$. Conversely, if R is partition of $[a, c]$, then there are partitions P and Q of $[a, b]$ and $[b, c]$ respectively such that $P \cup Q$ refines R. Therefore

$$\begin{aligned} \int_a^b f + \int_b^c f &= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} f(x)\mathrm{d}x + \bigsqcup^{\uparrow}_{Q \in \mathcal{P}[b,c]} \sum_{y \in Q} f(y)\mathrm{d}y \\ &= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \bigsqcup^{\uparrow}_{Q \in \mathcal{P}[b,c]} \sum_{x \in P} f(x)\mathrm{d}x + \sum_{y \in Q} f(y)\mathrm{d}y \\ &= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \bigsqcup^{\uparrow}_{Q \in \mathcal{P}[b,c]} \sum_{z \in P \cup Q} f(z)\mathrm{d}z \\ &= \bigsqcup^{\uparrow}_{R \in \mathcal{P}[a,c]} \sum_{z \in R} f(z)\mathrm{d}z \\ &= \int_a^c f. \end{aligned}$$

We omit the routine proof of the third equation. □

Clearly, $\int_a^b f$ depends only on the values that f assumes on $\mathbf{I}[a, b]$.

Theorem 13.6 *For every interval $[a, b]$, the integration map*

$$f \mapsto \int_a^b f \; : \; (\mathbf{I}[a, b] \Rightarrow \mathcal{R}) \to \mathcal{R}$$

is continuous.

Proof Let $\mathcal{F}$ be a directed subset of the domain $\mathbf{I}[a,b] \Rightarrow \mathcal{R}$. Then

$$\begin{aligned}
\int_a^b \bigsqcup{}^{\uparrow} \mathcal{F} &= \bigsqcup_P{}^{\uparrow} \sum_{x \in P} \left(\bigsqcup{}^{\uparrow} \mathcal{F} \right)(x) \mathrm{d}x \\
&= \bigsqcup_P{}^{\uparrow} \sum_{x \in P} \left(\bigsqcup_{f \in \mathcal{F}}{}^{\uparrow} f(x) \right) \mathrm{d}x \\
&= \bigsqcup_P{}^{\uparrow} \sum_{x \in P} \bigsqcup_{f \in \mathcal{F}}{}^{\uparrow} f(x) \mathrm{d}x \\
&= \bigsqcup_P{}^{\uparrow} \bigsqcup_{f \in \mathcal{F}}{}^{\uparrow} \sum_{x \in P} f(x) \mathrm{d}x \\
&= \bigsqcup_{f \in \mathcal{F}}{}^{\uparrow} \bigsqcup_P{}^{\uparrow} \sum_{x \in P} f(x) \mathrm{d}x \\
&= \bigsqcup_{f \in \mathcal{F}}{}^{\uparrow} \int_a^b f. \square
\end{aligned}$$

Any dense subset A of $[a,b]$ induces a basis

$$B = \{[x,y] | x \le y \text{ in } A\}$$

of $\mathbf{I}[a,b]$.

Lemma 13.7 *Let $[a,b]$ be an interval, let B be any basis of $\mathbf{I}[a,b]$ induced by a dense subset of $[a,b]$, and denote by $\mathcal{P}_B[a,b]$ the partitions of $[a,b]$ consisting of basis elements. Then for any* continuous *function $f : \mathbf{I}[a,b] \to \mathcal{R}$,*

$$\int_a^b f = \bigsqcup_{Q \in \mathcal{P}_B[a,b]}{}^{\uparrow} \sum_{x \in Q} f(x) \mathrm{d}x.$$

Proof Let $\mathbf{u} \ll \int_a^b f$. It suffices to conclude that

$$\mathbf{u} \sqsubseteq \bigsqcup_{P \in \mathcal{P}_B[a,b]}{}^{\uparrow} \sum_{x \in P} f(x) \mathrm{d}x.$$

Let $P = \{x_1, \ldots, x_n\} \in \mathcal{P}[a,b]$ such that $\mathbf{u} \ll \sum_{x \in P} f(x) \mathrm{d}x$. W.l.o.g., we can assume that $[a,b]$ has non-zero diameter and that P consists of intervals of non-zero diameter. Then for each $x \in P$ there is some $x' \ll x$ in B such that already

$$\mathbf{u} \ll \sum_{x \in P} f(x') \mathrm{d}x$$

because f, addition, and scalar multiplication are continuous. Wlog we can assume that only successive elements of the sequence $x'_1, \ldots, x'_n$ do not overlap, because otherwise we can shrink the intervals x'_i in such a way that that the above inequality still holds. Then the unique partition Q of $[a,b]$ consisting of intervals of non-zero diameter with the end-points of the intervals $x'_1, \ldots, x'_n$ is of the form $\{y_1, z_1, y_2, \ldots, z_{n-1}, y_n\}$ with

1. $z_i = x'_i \sqcup x'_{i+1} \quad$ for $1 \le i \le n$

2. (a) $y_1 \sqcap z_1 = x'_1$ and $z_{n-1} \sqcap y_n = x'_n$

   (b) $z_{i-1} \sqcap y_i \sqcap z_i = x'_i \quad$ for $1 < i < n$.

We claim that

$$\sum_{x \in P} f(x)\mathrm{d}x \sqsubseteq \sum_{y \in Q} f(y)\mathrm{d}y,$$

which implies that

$$\mathbf{u} \ll \bigsqcup^{\uparrow}_{Q \in \mathcal{P}_B[a,b]} \sum_{x \in Q} f(x)\mathrm{d}x,$$

by transitivity, and concludes the proof. For notational simplicity and w.l.o.g., we prove the claim for the case $P = \{x_1, x_2\}$. In this case the claim reduces to

$$f(x'_1)\mathrm{d}x_1 + f(x'_2)\mathrm{d}x_2 \quad \sqsubseteq \quad f(y_1)\mathrm{d}y_1 + f(z_1)\mathrm{d}z_1 + f(y_2)\mathrm{d}y_2,$$

and is proved by

$$\begin{array}{ll} & f(x'_1)\mathrm{d}x_1 + f(x'_2)\mathrm{d}x_2 \\ = & f(x'_1)(\mathrm{d}y_1 + \mathrm{d}x_1 - \mathrm{d}y_1) + f(x'_2)(\mathrm{d}x_2 - \mathrm{d}y_2 + \mathrm{d}y_2) \\ = & f(x'_1)\mathrm{d}y_1 + f(x'_1)(\mathrm{d}x_1 - \mathrm{d}y_1) + f(x'_2)(\mathrm{d}x_2 - \mathrm{d}y_2) + f(x'_2)\mathrm{d}y_2 \\ \sqsubseteq & f(y_1)\mathrm{d}y_1 + f(z_1)(\mathrm{d}x_1 - \mathrm{d}y_1) + f(z_1)(\mathrm{d}x_2 - \mathrm{d}y_2) + f(y_2)\mathrm{d}y_2 \\ = & f(y_1)\mathrm{d}y_1 + f(z_1)\mathrm{d}z_1 + f(y_2)\mathrm{d}y_2. \square \end{array}$$

Remark 13.8 Moore [Moo66] handles integration by considering sums which are essentially interval Riemann sums for partitions consisting of n intervals of the same length, but he restricts his definition to rational functions. The integrand is assumed to be monotone w.r.t. inclusion and continuous w.r.t. the Hausdorff metric on intervals. Since the Hausdorff metric induces the Lawson topology on $\mathcal{R}$, the integrand is Scott continuous [EC93, GHK$^+$80] (cf. Remark 8.1). Therefore Lemma 13.7 above and Theorem 13.13 below show that our definition generalizes that of Moore to all Scott continuous functions, and Theorem 13.9 below shows that our definition captures all Riemann integrable functions. □

Recall that given any continuous function $f : \mathbb{R} \to \mathbb{R}$, the function $\mathbf{I}f : \mathcal{R} \to \mathcal{R}$ defined by

$$\mathbf{I}f(x) = f(x)$$

is a continuous extension of f. Since continuous maps preserve connectedness and compactness,

$$\mathbf{I}f(x) = [\inf f(x), \sup f(x)].$$

Hence the end-points of an interval Riemann sum are given by lower and upper Darboux sums respectively:

$$\sum_{x \in P} \mathbf{I}f(x)\mathrm{d}x = \left[\sum_{x \in P} \inf f(x)\mathrm{d}x, \sum_{x \in P} \sup f(x)\mathrm{d}x\right].$$

Therefore the interval Riemann integral is a singleton in this case:

$$\int_a^b \mathbf{I}f = \left[\underline{\int_a^b} f, \overline{\int_a^b} f\right] = \left\{\int_a^b f\right\},$$

where the symbols $\underline{\int_a^b}$ and $\overline{\int_a^b}$ denote lower and upper Riemann integrals respectively. Any continuous map $f : \mathbb{R} \to \mathbb{R}$ has infinitely many distinct continuous extensions to $\mathcal{R} \to \mathcal{R}$. Recall that the extension $\mathbf{I}f$ is characterized as the greatest one. Theorem 13.13 below shows that $\mathbf{I}f$ can be replaced by *any* continuous extension of f in the above equation. Moreover, Theorem 13.9 below shows that the above equation can be generalized to any Riemann integrable function. Recall that a function is Riemann integrable on compact intervals iff it is bounded on compact intervals and continuous almost everywhere [Roy88], and that in this case the value of the integral depends only on the points of continuity of the function. Recall also that Theorem 8.8 shows that for any function $f : \mathbb{R} \to \mathbb{R}$ bounded on compact intervals there is a continuous map $\ddot{f} : \mathcal{R} \to \mathcal{R}$ agreeing with f at every point of continuity of f, given by

$$\ddot{f}(x) = [\inf \underline{g}(x), \sup \overline{g}(x)],$$

where $\underline{g} : \mathbb{R} \to \underline{\mathbb{R}}$ and $\overline{g} : \mathbb{R} \to \overline{\mathbb{R}}$ are continuous maps defined by

$$\underline{g}(y) = \liminf_{x \to y} f(x) \qquad \text{and} \qquad \overline{g}(y) = \limsup_{x \to y} f(x).$$

The following theorem shows that interval Riemann integration, even when restricted to Scott continuous functions, captures all Riemann integrable functions.

Theorem 13.9 *Let $f : \mathbb{R} \to \mathbb{R}$ be Riemann integrable on compact intervals $[a, b]$, and let $\ddot{f} : \mathcal{R} \to \mathcal{R}$ be the Scott continuous function defined in Theorem 8.8. Then*

$$\int_a^b \ddot{f} = \left\{\int_a^b f\right\}.$$

Proof By Theorem 8.8,

$$\begin{aligned}
\int_a^b \ddot{f} &= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} \ddot{f}(x) \mathrm{d}x \\
&= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} [\inf \underline{g}(x), \sup \overline{g}(x)] \mathrm{d}x \\
&= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \left[\sum_{x \in P} \inf \underline{g}(x) \mathrm{d}x, \sum_{x \in P} \sup \overline{g}(x) \mathrm{d}x\right] \\
&= \left[\sup_{P \in \mathcal{P}[a,b]} \sum_{x \in P} \inf \underline{g}(x) \mathrm{d}x, \inf_{P \in \mathcal{P}[a,b]} \sum_{x \in P} \sup \overline{g}(x) \mathrm{d}x\right] \\
&= \left[\underline{\int_a^b} \underline{g}, \overline{\int_a^b} \overline{g}\right] \\
&= \left\{\int_a^b f\right\},
\end{aligned}$$

because $\underline{g}$ and $\overline{g}$ agree with f at every point of continuity of f. $\square$

Lemma 13.10 *For every continuous function $f : \mathcal{R} \to \mathcal{R}$ there is a greatest continuous function $\hat{f} : \mathcal{R} \to \mathcal{R}$ such that*

$$f_{|\operatorname{Max}\mathcal{R}} = \hat{f}_{|\operatorname{Max}\mathcal{R}},$$

given by

$$\hat{f}(x) = \bigsqcap f(\uparrow x \cap \operatorname{Max}\mathcal{R})$$

Proof We first restrict f to $\operatorname{Max}\mathcal{R} \cong \mathbb{R}$, and then we find the greatest continuous extension to $\mathcal{R}$ by an application of Lemma 8.6, obtained by a formula which is essentially the same as the above one. $\square$

Lemma 13.11 *For any continuous $f : \mathcal{R} \to \mathcal{R}$,*

$$\int_a^b f \quad = \quad \int_a^b \hat{f}.$$

Proof $\int_a^b f \sqsubseteq \int_a^b \hat{f}$ because $f \sqsubseteq \hat{f}$. For the other direction, we first prove that

$$\hat{f}(x)\mathrm{d}x \sqsubseteq \int_{\underline{x}}^{\overline{x}} f$$

for all $x \in \mathcal{R}$. Let $b \ll \hat{f}(x)\mathrm{d}x$. It suffices to conclude that

$$b \sqsubseteq \int_{\underline{x}}^{\overline{x}} f.$$

Since $\hat{f}(x) = \bigsqcap_{r \in x} f(\{r\})$, by Lemma 13.10, we have that $b \ll f(\{r\})\mathrm{d}x$ for all $r \in x$. By continuity of f, for each $r \in x$ there is a $w_r \ll \{r\}$ such that already $b \ll f(w_r)\mathrm{d}x$. Since the interiors of the intervals w_r form an open cover of the compact interval x, there is a finite subset $\mathcal{C}$ of $\{w_r\}_{r \in x}$ such that the interiors of the members of $\mathcal{C}$ already cover x. Since the way-below order of $\mathcal{R}$ is multiplicative, $b \ll \bigsqcap_{y \in \mathcal{C}} f(y)\mathrm{d}x$. Now, there is a unique partition P of x, consisting of non-singleton intervals, such that the set of end-points of elements of P is the set of end-points of elements of $\mathcal{C}$ belonging to x, together with the two points $\underline{x}$ and $\overline{x}$. Since

$$f(z) \sqsubseteq \bigsqcap_{y \in P, z \sqsubseteq y} f(y),$$

we have that

$$\bigsqcap_{z \in \mathcal{C}} f(z) \sqsubseteq \bigsqcap_{z \in \mathcal{C}} \quad \bigsqcap_{y \in P, z \sqsubseteq y} f(y) = \bigsqcap_{y \in P} f(y).$$

Hence $b \ll \bigsqcap_{y \in P} f(y)\mathrm{d}x$. But

$$\bigsqcap_{y \in P} f(y)\mathrm{d}x \sqsubseteq \sum_{z \in P} f(z)\mathrm{d}z,$$

because

$$\prod_{y \in P} f(y)\mathrm{d}x = \left[\min_{y \in P} f(y)\mathrm{d}x, \max_{y \in P} f(y)\mathrm{d}x\right],$$

and hence the weighted average $\sum_{z \in P} f(z)\mathrm{d}z$ has to be contained in the latter interval. Therefore $b \ll \sum_{z \in P} f(z)\mathrm{d}z \sqsubseteq \int_{\underline{x}}^{\overline{x}} f$, which yields $b \sqsubseteq \int_{\underline{x}}^{\overline{x}} f$, as desired. Finally, we have that

$$\begin{aligned}
\int_a^b \hat{f} &= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} \hat{f}(x)\mathrm{d}x \sqsubseteq \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \sum_{x \in P} \int_{\underline{x}}^{\overline{x}} f \\
&= \bigsqcup^{\uparrow}_{P \in \mathcal{P}[a,b]} \int_a^b f = \int_a^b f,
\end{aligned}$$

which concludes the proof. □

Theorem 13.12 *The interval Riemann integral of a continuous function $f : \mathcal{R} \to \mathcal{R}$ depends only on the value that f assumes at maximal elements, in the sense that for any continuous function $g : \mathcal{R} \to \mathcal{R}$,*

$$f_{|\mathrm{Max}\,(\mathcal{R})} = g_{|\mathrm{Max}\,(\mathcal{R})} \quad \textit{implies} \quad \int_a^b f = \int_a^b g.$$

Proof By Lemma 13.10, $f_{|\mathrm{Max}\,(\mathcal{R})} = g_{|\mathrm{Max}\,(\mathcal{R})}$ implies $\hat{f} = \hat{g}$. Therefore the result follows from Lemma 13.11. □

Theorem 13.13 *If $f : \mathbb{R} \to \mathbb{R}$ is Riemann integrable on compact intervals and $\tilde{f} : \mathcal{R} \to \mathcal{R}$ is any Scott continuous map agreeing with f at points of continuity of f, then*

$$\int_a^b \tilde{f} = \left\{\int_a^b f\right\}.$$

Proof By Theorem 13.9, we know that this is true for the greatest such $\tilde{f}$, namely $\ddot{f}$. Therefore the result follows from Theorem 13.12. □

The significance of Theorems 13.12 and 13.13 is that sometimes it is easy to obtain a Real PCF program for an extension of a function f but it is difficult or undesirable to obtain a program for its greatest continuous extension. For instance, the distributive law does not hold for the greatest continuous extensions of addition and multiplication, so that two different definitions of the same function can give rise to two different extensions and two different programs [Moo66].

13.2 Multiple interval Riemann integrals

A ***partition*** of a ***hyper-cube***

$$\vec{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathcal{R}^n$$

is a cartesian product

$$\vec{P} = P_1 \times \cdots \times P_n$$

of partitions of $\mathbf{a}_1, \ldots, \mathbf{a}_n$ respectively. We denote the set of partitions of $\vec{\mathbf{a}}$ by $\mathcal{P}\vec{\mathbf{a}}$. Refinements are defined coordinatewise. The ***volume*** of an n-dimensional hyper-cube $\vec{x}$ is

$$\mathrm{d}\vec{x} = \mathrm{d}x_1 \cdots \mathrm{d}x_n.$$

Definition 13.14 Let $f : \mathcal{R}^n \to \mathcal{R}$ be a map and $\vec{\mathbf{a}}$ be an n-dimensional hyper-cube. A ***multiple interval Riemann sum*** of f on $\vec{\mathbf{a}}$ is a sum of the form

$$\sum_{\vec{x} \in \vec{P}} f(\vec{x})\mathrm{d}\vec{x} \qquad \text{for } \vec{P} \in \mathcal{P}\vec{\mathbf{a}}. \square$$

Definition 13.15 The ***multiple interval Riemann integral*** of a monotone map $f : \mathcal{R}^n \to \mathcal{R}$ on a hyper-cube $\vec{\mathbf{a}}$ is defined by

$$\int_{\vec{\mathbf{a}}} f = \bigsqcup_{\vec{P} \in \mathcal{P}\vec{\mathbf{a}}}^{\uparrow} \sum_{\vec{x} \in \vec{P}} f(\vec{x})\mathrm{d}\vec{x}. \qquad \square$$

For $n = 1$ this definition reduces to our previous definition:

$$\int_{(\mathbf{a})} f = \int_{\underline{\mathbf{a}}}^{\overline{\mathbf{a}}} f.$$

Theorem 13.16 (Fubini's Rule) *For every natural number $n > 1$, every continuous function $f : \mathcal{R}^n \to \mathcal{R}$, and every n-dimensional hyper-cube $\vec{\mathbf{a}}$,*

$$\int_{\vec{\mathbf{a}}} f = \int_{(\mathbf{a}_1)} \left(\int_{\vec{\mathbf{a}}'} f(x)\mathrm{d}\vec{x}' \right) \mathrm{d}x_1,$$

where $\vec{\mathbf{a}}' = (\mathbf{a}_2, \ldots, \mathbf{a}_n)$ and $\vec{x}' = (x_2, \ldots, x_n)$.

Proof For notational simplicity and without essential loss of generality, we prove the claim for $n = 2$, which corresponds to the inductive step of the claim for arbitrary n by induction on n:

$$\begin{aligned}
\int_{(\mathbf{a},b)} f &= \bigsqcup_{P \times Q \in \mathcal{P}(\mathbf{a},b)}^{\uparrow} \sum_{(x,y) \in P \times Q} f(x,y)\mathrm{d}(x,y) \\
&= \bigsqcup_{P \in \mathcal{P}\mathbf{a}}^{\uparrow} \bigsqcup_{Q \in \mathcal{P}b}^{\uparrow} \sum_{x \in P} \sum_{y \in Q} f(x,y)\mathrm{d}x\mathrm{d}y \\
&= \bigsqcup_{P \in \mathcal{P}\mathbf{a}}^{\uparrow} \sum_{x \in P} \left(\bigsqcup_{Q \in \mathcal{P}b}^{\uparrow} \sum_{y \in Q} f(x,y)\mathrm{d}y \right) \mathrm{d}x \\
&= \int_{(\mathbf{a})} \lambda x \int_{(b)} \lambda y f(x,y). \square
\end{aligned}$$

13.3 A supremum operator

In this section we define a supremum operator, which is used in Section 14.2 to obtain a fixed-point definition of interval Riemann integration. The presentation follows the same pattern as Section 13.1, and therefore we omit the proofs which are reworkings of previous proofs.

Lemma 13.17 *Let $f : \mathcal{R} \to \mathcal{R}$ be a* monotone *map* (w.r.t. the information order). *If a partition Q of an interval $[a,b]$ refines a partition P then*

$$\max_{x \in P} f(x) \sqsubseteq \max_{x \in Q} f(x).$$

Definition 13.18 For a function $f : \mathbb{R} \to \mathbb{R}$ we write

$$\sup_{[a,b]} f = \sup_{x \in [a,b]} f(x).$$

The ***supremum*** of a *monotone* map $f : \mathcal{R} \to \mathcal{R}$ on an interval $[a,b]$ is defined by

$$\sup_{[a,b]} f = \bigsqcup_{P \in \mathcal{P}[a,b]}^{\uparrow} \max_{x \in P} f(x). \qquad \square$$

Proposition 13.19 *For all continuous maps $f, g : \mathcal{R} \to \mathcal{R}$ and all real numbers α and β,*

$$\begin{aligned} \sup_{[a,a]} f &= f(a), \\ \max(\sup_{[a,b]} f, \sup_{[b,c]} f) &= \sup_{[a,c]} f, \\ \sup_{[a,b]} \max(\alpha f, \beta g) &= \max(\alpha \sup_{[a,b]} f, \beta \sup_{[a,b]} g). \end{aligned}$$

Clearly, $\sup_{[a,b]} f$ depends only on the values that f assumes on $\mathbf{I}[a,b]$.

Theorem 13.20 *For every interval $[a,b]$, the supremum map*

$$f \mapsto \sup_{[a,b]} f \; : \; (\mathbf{I}[a,b] \Rightarrow \mathcal{R}) \to \mathcal{R}$$

is continuous.

Lemma 13.21 *Let $[a,b]$ be an interval, and let B be any basis of $\mathbf{I}[a,b]$. Then for any* continuous *function $f : \mathbf{I}[a,b] \to \mathcal{R}$,*

$$\sup_{[a,b]} f = \bigsqcup_{Q \in \mathcal{P}_B[a,b]}^{\uparrow} \max_{x \in Q} f(x).$$

Clearly, for $f : \mathbb{R} \to \mathbb{R}$ continuous we have that

$$\max_{x \in P} \mathbf{I}f(x) = \left[\max_{x \in P} \inf f(x), \max_{x \in P} \sup f(x)\right].$$

Therefore

$$\sup_{[a,b]} \mathbf{I}f = \left\{ \sup_{[a,b]} f \right\}.$$

Lemma 13.22 *For any continuous $f : \mathcal{R} \to \mathcal{R}$,*

$$\sup_{[a,b]} f \quad = \quad \sup_{[a,b]} \hat{f}.$$

Theorem 13.23 *The supremum of a continuous function $f : \mathcal{R} \to \mathcal{R}$ depends only on the value that f assumes at maximal elements.*

Theorem 13.24 *If $f : \mathbb{R} \to \mathbb{R}$ is continuous and $f : \mathcal{R} \to \mathcal{R}$ is a continuous extension of f then*

$$\sup_{[a,b]} f \quad = \quad \left\{ \sup_{[a,b]} f \right\}.$$

An infimum operator inf is defined similarly, by replacing max by min.

Chapter 14

Integration in Real PCF

In Section 14.1 we extend Real PCF with a primitive for interval Riemann integration, and we establish computational adequacy for the extension. Similarly, in Section 14.3 we extend Real PCF with a primitive for supremum, and we establish computational adequacy for the extension. In Section 14.2 we show how to recursively define integration from the supremum operator. Finally, in Section 14.4 we discuss computational completeness of Real PCF extended with integration or supremum.

For simplicity, in this chapter we are deliberately informal concerning syntax.

14.1 Real PCF extended with interval Riemann integration

Again, for simplicity and without essential loss of generality, we restrict ourselves to the unit interval. Clearly, the map $\int_0^1 : (\mathcal{I} \Rightarrow \mathcal{R}) \to \mathcal{R}$ restricts to $(\mathcal{I} \Rightarrow \mathcal{I}) \to \mathcal{I}$. We denote the restriction by $\int$.

The programming language Real PCF$^\int$

Instead of introducing integration as a constant, we introduce it as a term-formation rule. This treatment of primitive operations is taken from Gunter [Gun92]. We could treat *all* primitive operations in this way, as he does, but we treat only integration in this way, for simplicity.

Definition 14.1

1. Real PCF$^\int$ is Real PCF extended by the following term-formation rule:

   If $Y : \mathcal{I}$ is a term and $\mathtt{x} : \mathcal{I}$ is a variable, then $\int Y \mathrm{d}\mathtt{x} : \mathcal{I}$ is a term, with the same free variables as Y, except for $\mathtt{x}$, which becomes bound. Terms of this form are called ***integrals***, whereas Y is called the ***integrand***.

   Here the term Y must not be confused with the fixed-point combinator `Fix`.

2. The meaning of the term $\int Y \mathrm{d}\mathtt{x}$ in an environment ρ is $\int f$, where f is the meaning of $\lambda\mathtt{x}.Y$ in ρ. □

Convention 14.2 We denote α-congruence by $\equiv$. Following Barendregt [Bar92], we identify α-congruent terms, and we adopt the inductive definition of substitution given in page 125 of loc. cit., extended by the rules

- $c[\alpha := M] \equiv c$ for any constant c.
- $(\int Y \mathrm{d}\mathtt{x})[\alpha := M] \equiv (\int Y[\alpha := M]\mathrm{d}\mathtt{x})$ provided $\alpha \not\equiv \mathtt{x}$.
- $(\int Y \mathrm{d}\mathtt{x})[\mathtt{x} := M] \equiv (\int Y \mathrm{d}\mathtt{x})$.

Notice that this definition assumes the so-called "[bound] variable convention" in order to omit the cumbersome proviso which prevents the capture of free variables [Bar84]. □

Operational semantics

Recall that $\oplus$ denotes binary average, which is a Real PCF definable operation.

Lemma 14.3 *For any continuous map $f : \mathcal{I} \to \mathcal{I}$,*

$$\int \mathrm{cons}_{\mathbf{a}} \circ f = \mathrm{cons}_{\mathbf{a}} \left(\int f\right),$$

$$\int f = \int f \circ \mathrm{cons}_L \oplus \int f \circ \mathrm{cons}_R.$$

Proof The first equation is linearity. For the second equation we have

$$\begin{aligned} \int f &= \int_0^1 f \\ &= \int_0^{\frac{1}{2}} f + \int_{\frac{1}{2}}^1 f \\ &= \int_0^1 f\left(\tfrac{x}{2}\right) \tfrac{1}{2}\mathrm{d}x + \int_0^1 f\left(\tfrac{x+1}{2}\right) \tfrac{1}{2}\mathrm{d}x \\ &= \int f \circ \mathrm{cons}_L \oplus \int f \circ \mathrm{cons}_R. \;\square \end{aligned}$$

Definition 14.4 The ***immediate reduction rules for integration*** are:

1. **(Production)** $\int Y \mathrm{d}\mathtt{x} \to \int Z \mathrm{d}\mathtt{x}$ if $Y \to Z$,
2. **(Output)** $\int \mathrm{cons}_a Y \mathrm{d}\mathtt{x} \to \mathrm{cons}_a \left(\int Y \mathrm{d}\mathtt{x}\right)$,
3. **(Input)** $\int Y \mathrm{d}\mathtt{x} \to \int Y_L \mathrm{d}\mathtt{x} \oplus \int Y_R \mathrm{d}\mathtt{x}$,

where

$$Y_a \equiv Y[\mathtt{x} := \mathrm{cons}_a \mathtt{x}].$$ □

Intuitively, the output rule produces partial output, the input rule supplies partial input, and the production rule partially evaluates the integrand (with no input or with the partial input supplied by the input rule in previous reduction steps).

Computational adequacy

Lemma 14.5 *For every natural number n define a map $\int^{(n)} : (\mathcal{I} \Rightarrow \mathcal{I}) \to \mathcal{I}$ by*

$$\int^{(n)} f = \sum_{k=1}^{2^n} f\left(\left[\frac{k-1}{2^n}, \frac{k}{2^n}\right]\right) \frac{1}{2^n}.$$

Then $\int^{(n)}$ is continuous, and

$$\int f = \bigsqcup_{n \geq 0}^{\uparrow} \int^{(n)} f.$$

Proof The right-hand side of the equation can be expressed as

$$\bigsqcup_{n \geq 0}^{\uparrow} \sum_{y \in Q_n} f(y) \mathrm{d}y,$$

where

$$Q_n = \left\{ \left[\frac{k-1}{2^n}, \frac{k}{2^n}\right] \middle| 1 \leq k \leq 2^n \right\}.$$

Let $D_n = \{k/2^n | 0 \leq k \leq 2^n\}$. Then $\bigcup_{n \geq 0} D_n$ is the set of dyadic numbers, which is dense in $[0,1]$. Hence intervals with distinct dyadic end-points form a basis of $\mathcal{I}[0,1]$, say B. Moreover, the end-points of the intervals in Q_n are contained in D_n. Hence for every partition $P \in \mathcal{P}_B[0,1]$ there is some n such that Q_n refines P. Therefore the result follows from Lemma 13.7. $\square$

Lemma 14.6 *For every natural number n,*

$$\begin{aligned} \int^{(0)} f &= f(\perp), \\ \int^{(n+1)} f &= \int^{(n)} f \circ \mathrm{cons}_L \oplus \int^{(n)} f \circ \mathrm{cons}_R. \end{aligned}$$

Proof For the first equation we have

$$\int^{(0)} f = \sum_{k=1}^{2^0} f\left(\left[\frac{k-1}{2^0}, \frac{k}{2^0}\right]\right) \frac{1}{2^0} = f([0,1]) = f(\perp).$$

For the second equation we have

$$
\begin{aligned}
\int^{(n+1)} f &= \sum_{k=1}^{2^{n+1}} f\left(\left[\frac{k-1}{2^{n+1}}, \frac{k}{2^{n+1}}\right]\right) \frac{1}{2^{n+1}} \\
&= \sum_{k=1}^{2^{n}} f\left(\left[\frac{k-1}{2^{n+1}}, \frac{k}{2^{n+1}}\right]\right) \frac{1}{2^{n+1}} + \sum_{k=2^n+1}^{2^{n+1}} f\left(\left[\frac{k-1}{2^{n+1}}, \frac{k}{2^{n+1}}\right]\right) \frac{1}{2^{n+1}} \\
&= \frac{1}{2}\sum_{k=1}^{2^{n}} f\left(\left[\frac{k-1}{2^{n+1}}, \frac{k}{2^{n+1}}\right]\right) \frac{1}{2^{n}} + \frac{1}{2}\sum_{k=1}^{2^{n}} f\left(\left[\frac{(k+2^n)-1}{2^{n+1}}, \frac{k+2^n}{2^{n+1}}\right]\right) \frac{1}{2^{n}} \\
&= \sum_{k=1}^{2^{n}} f\left(\frac{\left[\frac{k-1}{2^n}, \frac{k}{2^n}\right]}{2}\right) \frac{1}{2^{n}} \oplus \sum_{k=1}^{2^{n}} f\left(\frac{\left[\frac{k-1}{2^n}, \frac{k}{2^n}\right]+1}{2}\right) \frac{1}{2^{n}} \\
&= \int^{(n)} f\left(\frac{x}{2}\right) \mathrm{d}x \oplus \int^{(n)} f\left(\frac{x+1}{2}\right) \mathrm{d}x \\
&= \int^{(n)} f \circ \mathrm{cons}_L \oplus \int^{(n)} f \circ \mathrm{cons}_R. \, \square
\end{aligned}
$$

As a corollary, we have that for every n there is a program in Real PCF (without the integration primitive) defining $\int^{(n)}$. But, in order to establish computational adequacy, it will prove simpler to introduce $\int^{(n)}$ as a primitive construction.

Definition 14.7

1. Real PCF$^{\int^{(n)}}$ is Real PCF$^{\int}$ extended with a constant $\Omega_\sigma : \sigma$ for each type σ and the following term-formation rule for each natural number n:

   If $Y : \mathcal{I}$ is a term and $\mathtt{x} : \mathcal{I}$ is a variable, then $\int^{(n)} Y \mathrm{d}\mathtt{x} : \mathcal{I}$ is a term, with the same free variables as Y, except for $\mathtt{x}$, which becomes bound.

2. The meaning of Ω_σ is the bottom element of the domain of interpretation of σ. Notice that Ω_σ is PCF-definable. We introduce it as a constant so as to be able to prove Lemma 14.11 below.

3. The meaning of $\int^{(n)} Y \mathrm{d}\mathtt{x}$ in an environment ρ is $\int^{(n)} f$, where f is the meaning of $\lambda \mathtt{x}.Y$ in ρ.

4. There is no reduction rule for Ω_σ.

5. The immediate reduction rules for $\int^{(n)}$ are:

   (a) **(Production)** $\int^{(0)} Y \mathrm{d}\mathtt{x} \to \int^{(0)} Z \mathrm{d}\mathtt{x} \quad$ if $Y \to Z$,

   (b) **(Output)** $\int^{(0)} \mathrm{cons}_a Y \mathrm{d}\mathtt{x} \to \mathrm{cons}_a \left(\int^{(0)} Y \mathrm{d}\mathtt{x}\right)$,

   (c) **(Input)** $\int^{(n+1)} Y \mathrm{d}\mathtt{x} \to \int^{(n)} Y_L \mathrm{d}\mathtt{x} \oplus \int^{(n)} Y_R \mathrm{d}\mathtt{x}$.

$\square$

Definition 14.8 A ***sublanguage*** of a language $\mathcal{L}$ is a subset of $\mathcal{L}$-terms which is closed under reduction. $\square$

The following lemma is immediate:

Lemma 14.9 *If every $\mathcal{L}$-term is computable, so is every term of any sublanguage of $\mathcal{L}$.*

Thus, in order to prove that every term of Real PCF$^{\int}$ is computable it suffices to prove that every term of Real PCF$^{\int^{(n)}}$ is computable.

Definition 14.10 Let $\preccurlyeq$ be the least *relation* on terms such that:

1. If $M : \sigma$ then $\Omega_\sigma \preccurlyeq M$.
2. If $Y \preccurlyeq Y' : \mathcal{I}$ then $\int^{(n)} Y d\mathtt{x} \preccurlyeq \int Y' d\mathtt{x}$ and also $\int Y d\mathtt{x} \preccurlyeq \int Y' d\mathtt{x}$.
3. $M \preccurlyeq M$.
4. If $M \preccurlyeq M' : \sigma \to \tau$ and $N \preccurlyeq N' : \sigma$, then $(MN) \preccurlyeq (M'N')$.
5. If $M \preccurlyeq M'$ are terms (of the same type) then $\lambda\alpha.M \preccurlyeq \lambda\alpha.M'$.

$\square$

This relation turns out to be reflexive and transitive, which justifies the notation, but we do not need this fact.

The following lemma is analogous to Lemma 4.5:

Lemma 14.11 *If $M \preccurlyeq N$ and $M \to M'$ then $M' \preccurlyeq N'$ and $N \to N'$ for some N'.*

This situation is summarized by the diagram below:

$$\begin{tikzcd} M \arrow[d] \arrow[r, phantom, "\preccurlyeq"] & N \arrow[d, dotted] \\ M' \arrow[r, phantom, "\preccurlyeq"] & N' \end{tikzcd}$$

Proof By structural induction on M, according to why $M \to M'$. $\square$

Corollary 14.12 *If $M \preccurlyeq N$ and $M \to^* M'$ then $M' \preccurlyeq N'$ and $N \to^* N'$ for some N'.*

Proof By induction on the length of the reduction. $\square$

Corollary 14.13 *For each natural number n and all $Y : \mathcal{I}$ and $\mathtt{x} : \mathcal{I}$,*

$$\mathrm{Eval}\left(\int^{(n)} Y d\mathtt{x}\right) \subseteq \mathrm{Eval}\left(\int Y d\mathtt{x}\right).$$

Proof Immediate consequence of Corollary 14.12, noticing that, by definition of $\preccurlyeq$, if one has that $\mathrm{cons}_a Z \preccurlyeq Z'$ then Z' has to be of the form $\mathrm{cons}_a Z''$. $\square$

The term Ω_σ is trivially computable.

Lemma 14.14 *For every n, if $Y : \mathcal{I}$ is computable so is $\int^{(n)} Y \mathrm{d}\mathtt{x}$ for all $\mathtt{x} : \mathcal{I}$.*

Proof By induction on n.

Base: Since the terms $\int^{(0)} Y \mathrm{d}\mathtt{x}$ and $Y[\mathtt{x} := \Omega_\mathcal{I}]$ have the same meaning in any environment, namely $\int^{(0)} f = f(\bot)$ where f is the meaning of $\lambda\mathtt{x}.Y$, and since $Y[\mathtt{x} := \Omega_\mathcal{I}]$ is computable as it is an instantiation of a computable term by computable terms, it suffices to conclude that $\mathrm{Eval}(Y[\mathtt{x} := \Omega_\mathcal{I}]) \subseteq \mathrm{Eval}(\int^{(0)} Y \mathrm{d}\mathtt{x})$. Assume that $Y[\mathtt{x} := \Omega_\mathcal{I}] \to^* \mathrm{cons}_a Z$. One easily checks by structural induction that $Y[\mathtt{x} := \Omega_\mathcal{I}] \preccurlyeq Y$. Hence, by Corollary 14.12 we conclude that $Y \to^* Z'$ for some Z' with $\mathrm{cons}_a Z \preccurlyeq Z'$. By definition of $\preccurlyeq$, Z' has to be of the form $\mathrm{cons}_a Z''$. Hence, by some applications of the production rule followed by an application of the output rule, $\int^{(0)} Y \mathrm{d}\mathtt{x} \to^* \int^{(0)} \mathrm{cons}_a Z'' \mathrm{d}\mathtt{x} \to \mathrm{cons}_a(\int^{(0)} Z'' \mathrm{d}\mathtt{x})$.

Induction step: If Y is computable so is Y_a for any a. Hence, by the induction hypothesis, $\int^{(n)} Y_a \mathrm{d}\mathtt{x}$ is computable. Since $\oplus$ is a Real PCF term, it is computable. By definition of computability, it follows that $\int^{(n)} Y_L \mathrm{d}\mathtt{x} \oplus \int^{(n)} Y_R \mathrm{d}\mathtt{x}$ is computable. Therefore $\int^{(n+1)} Y \mathrm{d}\mathtt{x}$ is computable, because every reduction from $\int^{(n+1)} Y \mathrm{d}\mathtt{x}$ factors through $\int^{(n)} Y_L \mathrm{d}\mathtt{x} \oplus \int^{(n)} Y_R \mathrm{d}\mathtt{x}$ via the input rule, and $\int^{(n+1)} Y \mathrm{d}\mathtt{x}$ has the same meaning as $\int^{(n)} Y_L \mathrm{d}\mathtt{x} \oplus \int^{(n)} Y_R \mathrm{d}\mathtt{x}$, in any environment. □

Lemma 14.15 *Every Real PCF$^{\int^{(n)}}$ term is computable.*

Proof Extend the proof of Lemma 11.6 by including Lemma 14.14 as one of the inductive steps. □

Theorem 14.16 (Adequacy) *Every Real PCF$^{\int}$ term is computable.*

Proof Lemmas 14.5, Corollary 14.13, and Lemmas 14.15 and 14.9. □

14.2 A fixed-point definition of integration

It is natural to ask if the integration operator, added in Section 14.1 as primitive, is already recursively definable in Real PCF.

Let $D = (\mathcal{I} \Rightarrow \mathcal{I}) \Rightarrow \mathcal{I}$. Then the second equation of Lemma 14.3 leads one to consider the map $G : D \to D$ defined by

$$G(F)(f) = F(f \circ \mathrm{cons}_L) \oplus F(f \circ \mathrm{cons}_R).$$

Thus the integration operator $\int$ is a fixed point of G. However, the least fixed point is the bottom element of D.

Peter Freyd suggested that if we restrict ourselves to the subspace $D' \subseteq D$ of functions $F \in D$ such that

$$\inf f \leq F(f) \leq \sup f,$$

then G restricts to a map $G' : D' \to D'$, and $\int$ is the least fixed point of G'. We use this idea in a modified form, obtaining $\int$ directly as the least fixed point of a function $H : D \to D$.

Define a map $p : [0,1]^3 \to [0,1]$ by

$$j(x, y, z) = \max(x, \min(y, z)).$$

Then, given $a \leq b$, the map $g : [0,1] \to [0,1]$ defined by

$$g(x) = j(a, x, b)$$

is idempotent,

$$a \leq f(x) \leq b,$$

and

$$g(x) = x \text{ iff } a \leq x \leq b.$$

(Cf. Section 8.13). Also, define a function $H : D \to D$ by

$$H(F)(f) = j\left(\inf f, F(f \circ \mathrm{cons}_L) \oplus F(f \circ \mathrm{cons}_R), \sup f\right).$$

Lemma 14.17 *For every continuous function $f : \mathcal{I} \to \mathcal{I}$,*

$$H^n(\bot)(f) = \int^{(n)} \hat{f},$$

where $\hat{f}$ is defined as in Lemma 13.10.

Proof By induction on n. For the base case use the fact that $\hat{f}(\bot) = j(\inf f, \bot, \sup f)$. $\square$

Proposition 14.18 *$\int$ is the least fixed point of H.*

Proof Immediate consequence of Lemmas 14.5 and 14.17. $\square$

14.3 Real PCF extended with supremum

This section follows the same pattern as Section 14.1. Due to this reason, we omit the proofs which are reworking of proofs given earlier. Again, for simplicity and without essential loss of generality, we restrict ourselves to the unit interval. Clearly, the map $\sup_{[0,1]} : (\mathcal{I} \Rightarrow \mathcal{R}) \to \mathcal{R}$ restricts to $(\mathcal{I} \Rightarrow \mathcal{I}) \to \mathcal{I}$. We denote the restriction by sup.

Definition 14.19 Real PCF$^{\mathrm{sup}}$ is Real PCF extended with a construction $\sup_{\mathsf{x}} Y$, as in Definition 14.1, denoting the operation $\sup : (\mathcal{I} \Rightarrow \mathcal{I}) \to \mathcal{I}$. $\square$

Lemma 14.20 *For any continuous map $f : \mathcal{I} \to \mathcal{I}$,*

$$\sup \mathrm{cons}_{\mathbf{a}} \circ f = \mathrm{cons}_{\mathbf{a}}(\sup f),$$

$$\sup f = \max\left(\sup f \circ \mathrm{cons}_L, \sup f \circ \mathrm{cons}_R\right).$$

Definition 14.21 The ***immediate reduction rules for supremum*** are:

1. **(Production)** $\sup_{\mathtt{x}} Y \to \sup_{\mathtt{x}} Z \quad \text{if } Y \to Z$,
2. **(Output)** $\sup_{\mathtt{x}} \mathrm{cons}_a Y \to \mathrm{cons}_a (\sup_{\mathtt{x}} Y)$,
3. **(Input)** $\sup_{\mathtt{x}} Y \to \max(\sup_{\mathtt{x}} Y_L, \sup_{\mathtt{x}} Y_R)$,

where

$$Y_a \equiv Y[\mathtt{x} := \mathrm{cons}_a \mathtt{x}].$$ □

Notice that these are the reduction rules for $\int$ with $\int$ and $\oplus$ replaced by sup and max respectively. We obtain the following similar results, whose proofs are omitted because they are similar too:

Lemma 14.22 *For every natural number n define a map $\sup^{(n)} : [\mathcal{I} \to \mathcal{I}] \to \mathcal{I}$ by*

$$\sup^{(n)} f = \max_{k=1}^{2^n} f\left(\left[\frac{k-1}{2^n}, \frac{k}{2^n}\right]\right).$$

Then $\sup^{(n)}$ is continuous, and

$$\sup f = \bigsqcup_{n \geq 0}^{\uparrow} \sup^{(n)} f.$$

Lemma 14.23 *For every natural number n,*

$$\begin{aligned} \sup^{(0)} f &= f(\perp), \\ \sup^{(n+1)} f &= \max\left(\sup^{(n)} f \circ \mathrm{cons}_L, \sup^{(n)} f \circ \mathrm{cons}_R\right). \end{aligned}$$

As a corollary, we have that for every n there is a Real PCF program defining $\sup^{(n)}$. But, as we did for integration, we add the partial supremum operators $\sup^{(n)}$ as primitive, and we conclude that:

Theorem 14.24 (Adequacy) *Every Real PCF$^{\sup}$ term is computable.*

The operation inf is definable from sup by

$$\inf f = 1 - \sup_x (1 - f(x)),$$

so there is no need to include it as primitive too.

Corollary 14.25 *The integration operator is definable in Real PCF$^{\sup}$.*

Proof The function H in Lemma 14.17 is Real PCF$^{\sup}$ definable. □

Corollary 14.26 *For every natural number n there is a program in Real PCF extended with either integration or supremum which computes the multiple integration operator $\int : (\mathcal{I}^n \Rightarrow \mathcal{I}) \to \mathcal{I}$ of order n.*

Proof (Since PCF does not have cartesian products, we have to use curried maps.) Our primitive or program for integration takes care of the case $n = 1$. Fubini's Rule (Theorem 13.16) can be read as a definition of a program for the case $n + 1$ from a program for the case n. By the adequacy theorems, these programs indeed compute multiple integrals of order n. □

This application of the adequacy theorems shows that computational adequacy is a powerful property. In fact, it allows us to derive correct programs from analytical results, in a representation-independent fashion. Of course, this is precisely the idea behind denotational semantics.

14.4 Computational completeness of Real PCF extended with supremum

Although Theorem 12.13 implies that sup is definable in Real PCF extended with $\exists$, we do not know a neat fixed-point definition of sup.

Proposition 14.27 *The existential quantification operator $\exists$ is definable in Real PCF extended with* sup.

Proof For $D \in \{\mathcal{N}, \mathcal{B}\}$, define continuous maps

$$D \underset{s_D}{\overset{r_D}{\leftrightarrows}} \mathcal{I}$$

by

$$\begin{aligned} s_{\mathcal{N}}(n) &= \text{if } n = 0 \text{ then } 0 \text{ else } \operatorname{cons}_R(s_{\mathcal{N}}(n-1)) \\ r_{\mathcal{N}}(x) &= \text{if } x <_\perp 1/4 \text{ then } 0 \text{ else } r_{\mathcal{N}}(\operatorname{tail}_R(x)) + 1, \end{aligned}$$

$$\begin{aligned} s_{\mathcal{B}}(t) &= \text{if } t \text{ then } 1 \text{ else } 0 \\ r_{\mathcal{B}}(x) &= \text{if } x <_\perp 1/2 \text{ then false else true} \end{aligned}$$

Then (s_D, r_D) is a section-retraction pair. This is immediate for $D = \mathcal{B}$. For $D = \mathcal{N}$, we prove by induction on n that $r_{\mathcal{N}} \circ s_{\mathcal{N}}(n) = n$. If $n = \perp$ or $n = 0$ this is immediate. For the inductive step we have that

$$\begin{aligned} r_{\mathcal{N}} \circ s_{\mathcal{N}}(n+1) &= r_{\mathcal{N}}(\operatorname{cons}_R(s_{\mathcal{N}}(n)) \\ &= r_{\mathcal{N}}(\operatorname{tail}_R \circ \operatorname{cons}_R(s_{\mathcal{N}}(n))) + 1 \\ &= r_{\mathcal{N}}(s_{\mathcal{N}}(n)) + 1 \\ &= n + 1 \quad \text{by the induction hypothesis.} \end{aligned}$$

It follows that the diagram below commutes:

$$\begin{array}{ccc} (\mathcal{I} \Rightarrow \mathcal{I}) & \xrightarrow{\sup} & \mathcal{I} \\ {\scriptstyle r_{\mathcal{N}} \to s_{\mathcal{B}}} \uparrow & & \downarrow {\scriptstyle r_{\mathcal{B}}} \\ (\mathcal{N} \Rightarrow \mathcal{B}) & \xrightarrow[\exists]{} & \mathcal{B} \end{array}$$

In fact, let $p \in [\mathcal{N} \to \mathcal{B}]$ and define $f : \mathcal{I} \to \mathcal{I}$ by

$$f = (r_{\mathcal{N}} \Rightarrow s_{\mathcal{B}})(p) = s_{\mathcal{B}} \circ p \circ r_{\mathcal{N}}.$$

If there is some n such that $p(n) = \text{true}$, then there is some x such that $f(x) = 1$, namely $x = s_{\mathcal{N}}(n)$, and in this case we have that $\sup f = 1$. If $p(\perp) = \text{false}$, then $f(\perp) = 0$, and in this case we have that $\sup f = 0$. Therefore $\exists$ is definable in Real PCF extended with sup. $\square$

Corollary 14.28 *Real PCF extended with* sup *is computationally complete.*

We don't know whether Real PCF extended with integration is computationally complete. Moreover, we don't know whether integration is definable in Real PCF with no extensions, but conjecture that this is not the case.

For applications of Real PCF to real analysis, it seems more natural to include the supremum operator as a primitive operation than to include the existential quantification operator.

Remark 14.29 Notice that the section-retraction pairs defined in the proof of Proposition 14.27 are expressed in terms of sequential primitives only, and that the maximum operation on $\mathcal{I}$ represents the "parallel or" operation $\vee$ on $\mathcal{B}$, in the sense that the following diagram commutes:

$$\begin{tikzcd}
\mathcal{I} \times \mathcal{I} \arrow[r, "\max"] & \mathcal{I} \arrow[d, "r_{\mathcal{B}}"] \\
\mathcal{B} \times \mathcal{B} \arrow[u, "s_{\mathcal{B}} \times s_{\mathcal{B}}"] \arrow[r, "\vee"'] & \mathcal{B}
\end{tikzcd}$$

$\square$

Part VI

Concluding remarks

Chapter 15

Summary of results

Recall that this thesis consists of the following five parts:

I Background

II Domain theory revisited

III The partial real line

IV Computation on the partial real line

V Integration on the partial real line

15.1 Continuous words, Real PCF, and computational adequacy

In Chapter 9 of Part III we showed that partial real numbers can be considered as continuous words. Continuous words and their basic properties were used in Chapter 11 of Part IV to extend PCF with real numbers, obtaining Real PCF, and to show that Real PCF is computationally adequate, in the sense that its operational and denotational semantics coincide.

15.2 Induction and recursion on the partial real line

In Chapter 10 of Part III we introduced induction principles and recursion schemes for the partial real line, similar to the so-called Peano axioms for natural numbers. In particular, we obtained an abstract characterization of the partial unit interval, up to isomorphism, which does not mention real numbers or intervals. In order to formulate this characterization, we introduced the category of binary systems.

Based on the results of Chapter 5 of Part II, we showed that the partial real line can be given the structure of an inductive retraction, thus obtaining structural recursion and corecursion schemes. These schemes were formulated in terms of "generalized bifree algebras", which are quotients of bifree algebras in a special way. As a byproduct, we obtained an effective version of binary expansions for partial real numbers, which we called "bifurcated binary expansions".

We gave many applications of induction and recursion. First, we obtained several examples of recursive definitions on the partial real line, most of which immediately give rise to Real PCF programs. Second, these induction principles and recursion schemes were used to establish the existence of a unique sound effective presentation of the Real PCF domains, up to equivalence. Third, they were used to establish computational completeness of Real PCF. These applications are discussed below.

15.3 Computability on the real numbers type hierarchy

In our approach to computation on real numbers, we introduced two new ingredients, which are very natural for an approach based on domain theory. First, we made partially successful computations official, via partial real numbers. Second, we admitted higher-order functions (such as integration operators). From the point of view of computability, these two new ingredients are not problematic, because domain theory was designed to support them, via effectively given domains and function spaces. Thus, in order to define computability on the real numbers type hierarchy, one has to simply choose a natural basis (such as the rational intervals) and a natural enumeration of the basis (such as Cantor enumeration), and then lift the effective presentation to higher types. Since computability is a robust notion, the choice of basis and its enumeration should not make any difference. We proved that this is indeed the case, as follows.

First, we needed a notion of equivalence of effective presentations of a continuous domain. Since such a notion was available only for the algebraic case, and since it does not immediately generalize to the continuous case, we were forced to introduce a new notion in Chapter 6 of Part II. We argued that the natural notion of equivalence is just the existing notion of equivalence of objects in concrete categories, specialized to the category of effectively given domains, considered concrete over the category of domains, via the forgetful functor which forgets effective presentations. We established two main results: (1) two effective presentations on the same domain are equivalent iff they induce the same notion of computability on the domain, and (2) effective presentations can be lifted uniquely, up to equivalence, to function spaces. These are really instances of two results on concrete categories and concretely cartesian closed categories.

Second, we observed that all Real PCF definable functions should be regarded as computable, because Real PCF has an effective operational semantics. We thus called "sound" an effective presentation of the real numbers domain which makes all definable elements computable. In Chapter 12 of Part IV we showed that there is a unique sound effective presentation, up to equivalence, obtaining an absolute notion of computability for the real numbers type hierarchy.

Absoluteness was achieved by means of the inductive principles and recursion schemes discussed above, in the following way. In Chapter 6 of Part II, we showed that for any locally computable functor $\mathbf{F}$ and any domain D, there is at most one effective presentation of D which makes D an $\mathbf{F}$-inductive retract, up to equivalence. In Chapter 10 of Part III, we exhibited a functor which gives the the partial real line the structure of an inductive retract. In Chapter 12 of Part IV, we observed that this functor is locally computable, and we showed that there is an effective presentation of the partial real line which makes the section and retraction both computable and Real PCF definable, thus establishing soundness and absoluteness.

15.4 Computational completeness of Real PCF

In Chapter 12 of Part IV, we observed that a computable existential quantifier, which is not PCF definable, is not Real PCF definable either. Again making use of the induction principles and recursion schemes discussed above, we adapted a technique by Thomas Streicher, establishing that Real PCF extended with the existential quantifier is computationally complete, in the sense that all computable elements are definable.

It is well-known that the existential quantifier is not needed to define all first-order computable functions in PCF, and one would expect Real PCF to enjoy the same property. Unfortunately, Thomas Streicher's technique does not give information about definability at first-order types. We thus developed a new general technique for establishing computational completeness of extensions of PCF with new ground types, which does give such information. In particular, we obtained as a corollary a new proof of computational completeness of PCF extended with parallel-if and the existential quantifier.

In its present form, the technique works in its full generality only when the ground types are interpreted as algebraic domains. Nevertheless, we were able to apply it to establish first-order computational completeness of Real PCF.

The technique makes use of what we called "joining maps" in Chapter 7 of Part II, and also of general properties of joining maps developed in that chapter. By the results of that chapter, a bounded complete domain admits joining maps iff it is coherently complete. Therefore, the technique is intrinsically restricted to coherently complete domains, of which the partial real line is an example anyway.

15.5 Partial real valued functions

In Chapter 8 of Part III, we showed that for any space X, a continuous partial real valued function $f : X \to \mathcal{R}$ is essentially the same as a pair of respectively lower and upper semicontinuous functions $\underline{f} : X \to \mathbb{R}$ and $\overline{f} : X \to \mathbb{R}$, bounded on compact intervals, with $\underline{f} \leq \overline{f}$ pointwise. Moreover, we showed that for any (not necessarily continuous) function $f : \mathbb{R} \to \mathbb{R}$ bounded on compact intervals there is a *continuous* function $\ddot{f} : \mathcal{R} \to \mathcal{R}$ which agrees with f at every point of continuity of f.

15.6 Interval Riemann integration

In Chapter 13 of Part V, we generalized Riemann integration of real valued functions of real variables to "interval Riemann integration" of partial real valued functions of partial real variables. We showed that interval Riemann integration fully captures Riemann integration, in the sense that for every Riemann integrable function $f : \mathbb{R} \to \mathbb{R}$ (i.e., bounded on compact intervals and continuous almost everywhere), the Riemann integral of $f : \mathbb{R} \to \mathbb{R}$ coincides with the Interval Riemann integral of any continuous $\tilde{f} : \mathcal{R} \to \mathcal{R}$ agreeing with f at every point of continuity of f. We also proved basic properties of interval Riemann integration, such as linearity and Fubini's rule for multiple Riemann integration.

15.7 Integration in Real PCF

In Chapter 14, we introduced an extension of Real PCF with simple interval Riemann integration as a primitive operation, via reduction rules, and we established computational adequacy of the extension. Fubini's rule and computational adequacy together imply definability of multiple interval Riemann integration. Based on a suggestion by Peter Freyd, we obtained a fixed-point definition of integration based on the reduction rules. Since the fixed-point definition involves a supremum operator, we showed how to extend Real PCF with it, again via reduction rules, retaining computational adequacy. We also showed that the existential quantifier can be replaced by the supremum operator in the computational completeness result, which is more natural for applications to real number computation.

15.8 New results on domain theory

As it is clear by the above summary, we developed new results on domain theory in order to obtain our main results on Real PCF, higher-order real number computability, and Real PCF definability. Since these results are interesting in their own right, we decided to collect them together in Part II. Running the risk of being somewhat repetitive, we briefly summarize them independently of their particular applications to domain-theoretic higher-order real number computation via Real PCF.

Bifree algebras generalized In Chapter 5 we defined inductive retracts as generalizations of bifree algebras, and we showed that any functor with a bifree algebra has any of is inductive retractions as a retract of the bifree algebra, in a canonical way. Moreover, we showed that inductive sections and retractions behave as respectively final coalgebras and initial algebras, except that the existence part in the definition of finality and initiality is missing. We gave a general criterion for the existence of homomorphisms, which amounted to saying that the involved (co)algebras (co)respect the congruence induced on the bifree algebra. Moreover, we gave two explicit constructions of the (unique) homomorphisms, when they exist. Except for one of the last two constructions, which holds only in certain order-enriched categories, all results hold in any category.

Equivalence of effectively given domains We remarked in Chapter 6 that, as far as we know, only equivalence of effectively given *algebraic* domains was considered in the literature. We showed that the notion of equivalence of objects in concrete categories gives rise to an appropriate notion of equivalence of effective presentations of domains, when we consider the category of effectively given domains and computable maps as a concrete category over the category of domains and continuous maps, via the forgetful functor which forgets effective presentations. We proved that two effective presentations of the same domain are equivalent iff they induce the same notion of computability on the domain, and we also showed that effective presentations lift uniquely, up to equivalence, to cartesian products and function spaces. Also, we showed that, for any functor $\mathbf{F}$ on the category of domains which extends to a locally computable functor on effectively given domains, and any $\mathbf{F}$-inductive retraction (D, α, β), there is at most one effective presentation of D which makes retraction α and the section β computable, up to equivalence. Moreover, if α and β are isomorphisms (and hence form a bifree algebra), then there is exactly one such effective presentation.

Coherently complete domains We characterized the coherently complete domains as the bounded complete domains which admit joining maps; we showed that every element of a coherently complete domain admits a least joining map; we obtained a new proof of universality of the coherently complete domain $\mathcal{B}^\omega$; and we showed how to construct joining maps of function spaces from joining maps of the target domain, and from the best continuous approximation to the characteristic function of the way below order on the source domain. The last two items gave rise to the general technique for establishing computational completeness of PCF extended with coherently complete domains, which works in its full generality only for the algebraic case as we lack information about the best continuous approximation to the way-below relation.

Chapter 16

Open problems and further work

16.1 Equivalence of effectively given domains

Since the original notion of equivalence of effectively given domains introduced in [KP79] applies only to the algebraic case and doesn't immediately generalize to the continuous case, we introduced a new notion, based on concrete categories, in Chapter 6 of Part II. Moreover, we showed that our definition is appropriate for our purposes. Nevertheless, it is an open problem whether our definition restricted to the algebraic case coincides with that of Kanda and Park. We conjecture that this is the case.

16.2 Real number computability

In Section 12.1 of Part IV, we showed that there is a unique sound effective presentation of the real numbers type hierarchy, thus providing an absolute notion of higher-order real number computability. However, we didn't relate the resulting notion of computability to existing notions for total real numbers [Grz57, KV65, Ko91, ML70, Myh53, PeR83, Ric54, Tur37, Wei87, Wei95]. We conjecture that our notion of computability, when restricted to real numbers and total real functions of total real variables, coincides with at least those of [Ko91, PeR83, Wei95][1]. See [Wei95] for comments about (in)equivalence of the existing notions. If our conjecture is true, we have that a function $f : \mathbb{R} \to \mathbb{R}$ is computable in the sense of [Ko91, PeR83, Wei95] iff it has an extension $\tilde{f} : \mathcal{R} \to \mathcal{R}$ computable in our sense iff the lifting $(\mathbf{I}f)_\perp : \mathcal{R}_\perp \to \mathcal{R}_\perp$ of its canonical extension $\mathbf{I}f : \mathcal{R} \to \mathcal{R}$ is computable in our sense. The reason is that there is a computable idempotent $c : (\mathcal{R}_\perp \Rightarrow \mathcal{R}_\perp) \to (\mathcal{R}_\perp \Rightarrow \mathcal{R}_\perp)$ such that $c(g)$ is the lifting of the canonical extension of $f : \mathbb{R} \to \mathbb{R}$, for any extension $g : \mathcal{R}_\perp \to \mathcal{R}_\perp$ of f. Notice however that this closure operator is interdefinable with the existential quantifier (or the supremum operator).

[1] Philipp Sünderhauff [Sün97] has shown that this is the case

16.3 Injective spaces

Originally, we had planned a chapter in Part II containing new results about injective spaces [Esc97]. Since most results are unrelated or vaguely related to this thesis, we eventually decided to omit it. However, we discussed some instances of some of the results. For instance, the idempotent $c : (\mathcal{R}_\perp \Rightarrow \mathcal{R}_\perp) \to (\mathcal{R}_\perp \Rightarrow \mathcal{R}_\perp)$ is an instance of a much more general construction for injective spaces. Its continuity depends only on the fact that the singleton-embedding $\mathbb{R} \to \mathcal{R}$ is proper in the sense of [HL82]. The fact that the idempotent has to make functions strict is explained by the fact that the smallest dense subspace of $\mathcal{R}_\perp$ whose embedding into $\mathcal{R}_\perp$ is a proper map is the subspace $\mathbb{R} \cup \{\perp\}$. Any sober space has such a smallest subspace and, moreover, has an associated closure operator defined roughly as above. Also, Lemma 8.6 (including the note at the end of its proof) are vastly generalized in loc. cit.

16.4 Way-below order

A new technique for proving computational completeness of extensions of PCF with new ground types was introduced in Section 12.4 of Part IV. In its present form, it only works in its full generality when the ground types are interpreted as algebraic domains. The generalization to the continuous case depends on a suitable constructive characterization of the (characteristic function of) the way-below order on function spaces of continuous domains. More precisely, we need a characterization which allows us to prove the crucial Lemma 12.17 in the continuous case. Some steps in this direction were already taken in joint work with Thomas Erker and Klaus Keimel [EEK97], where we obtained several characterizations of the way-below order on function spaces.

16.5 Full abstraction

Is Real PCF fully abstract? We conjecture that it is (recall that Real PCF includes a parallel conditional). A solution to the problem of the previous section would positively answer this question.

16.6 Sequentiality on the real numbers type hierarchy

Is there any language-independent notion of sequentiality for the real numbers type hierarchy? Does the notion of sequentiality for real numbers depend on the model of computation? It seems that the answer to the last question is positive.

We know that $\mathcal{R}_\perp$ is a coherently complete domain and that $\mathcal{B}^\omega$ is a universal coherently complete domain, which is essentially the same as the PCF domain $\mathcal{U} = (\mathcal{N} \to \mathcal{B})$. It is thus natural to attempt to reduce sequentiality on $\mathcal{R}_\perp$ to sequentiality on $\mathcal{U}$. Let us say that a (computable) function $\phi : \mathcal{U} \to \mathcal{U}$ is sequential if it is PCF definable. Given any section-retraction pair $s : \mathcal{R}_\perp \to \mathcal{U}$ and $r : \mathcal{U} \to \mathcal{R}_\perp$ with $s \circ r : \mathcal{U} \to \mathcal{U}$ computable, we could say that a (computable) function $f : \mathcal{R}_\perp \to \mathcal{R}_\perp$ is $s - r$-sequential if the function $r \circ f \circ s : \mathcal{U} \to \mathcal{U}$ is sequential. This notion generalizes to functions of several variables in the obvious way. It is not hard to see that there are distinct section-retraction pairs give which rise to distinct notions of sequentiality.

Is $\mathcal{R}_\perp$ a sequential retract of $\mathcal{U}$ (or of any universal PCF domain), in the sense that the induced idempotent on $\mathcal{U}$ is sequential? We conjecture that this is not the case. Are there s and r such that (some extension of) the addition operation is sequential? We conjecture that the answer is also negative.

16.7 Joining maps

Some questions about joining maps were raised in Chapter 7 of Part II. Here we add the following question: if join_x denotes the least joining map of an element x of a coherently complete domain D, and if we define $x \,\Box\, y = \mathrm{join}_x(y)$, does it follow that $(D, \Box, \perp)$ is a monoid? It is enough to check the associativity law, which can be expressed by the equation

$$\mathrm{join}_x(\mathrm{join}_y(z)) = \mathrm{join}_{\mathrm{join}_x(y)}(z).$$

These questions should be easy to answer [2].

16.8 Continued fractions and Möbius transformations

Peter Potts [Pot96] has recently found a common generalization of the ideas presented in this thesis about PCF extended with real numbers and the ideas of Vuillemin [Viu90, Vui88] on continued fractions. The idea is to use Möbius transformations, that is, functions of the form

$$x \mapsto \frac{px + q}{rx + s},$$

which include our affine transformations $x \mapsto px + q$ when $r = 0$ and $s = 1$. In ongoing work with him and Abbas Edalat, we have generalized the monoid of continuous words so that left actions are now Möbius transformations instead of affine transformations. We obtained an extension of PCF with real numbers, whose operational semantics is essentially the same as Real PCF's [PEE97]. The advantage of such an approach is that the parallel conditional can be avoided in many important instances, such as programs for a wide class of analytic functions.

[2] Michal Konecny has proved that this is *not* the case in general.

Bibliography

[Aci91] B.M. Acióly. *A Computational Foundation of Interval Mathematics.* PhD thesis, Universidade Federal do Rio Grande do Sul, Instituto de Informatica, Porto Alegre, Brazil, 1991. (In Portuguese).

[AHS90] J Adamek, H Herrlich, and G.E. Strecker. *Abstract and Concrete Categories.* John Wiley & Sons, Inc., 1990.

[AJ94] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D.M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, Oxford, 1994.

[Bar84] H.P. Barendregt. *The Lambda-Calculus: its Syntax and Semantics.* North-Holland, Amsterdam, 1984.

[Bar92] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Clarendon Press, Oxford, 1992.

[BB85] E. Bishop and D. Bridges. *Constructive Analysis.* Springer-Verlag, Berlin, 1985.

[BC90] H.J. Boehm and R. Cartwright. Exact real arithmetic: Formulating real numbers as functions. In Turner. D., editor, *Research Topics in Functional Programming*, pages 43–64. Addison-Wesley, 1990.

[BCRO86] H.J. Boehm, R. Cartwright, M. Riggle, and M.J. O'Donnel. Exact real arithmetic: A case study in higher order programming. In *ACM Symposium on Lisp and Functional Programming*, 1986.

[Boe87] H.J. Boehm. Constructive real interpretation of numerical programs. *SIGPLAN Notices*, 22(7):214–221, 1987.

[Bou66] N. Bourbaki. *General Topology*, volume 1. Addison-Wesley, London, 1966.

[Bra96] V. Brattka. Recursive characterization of computable real-valued functions and relations. *Theoretical Computer Science*, 162(1):45–77, August 1996.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers. *Bull. Amer. Math. Soc.*, 21:1–46, 1989.

[Cro93] R.L. Crole. *Categories for Types.* Cambridge University Press, Cambridge, 1993.

[Dev89] R. L. Devaney. *An Introduction to Chaotical Dynamical Systems.* Addison-Wesley, California, 2nd edition, 1989.

[EC76] H. Egli and R.L. Constable. Computability concepts for programming languages. *Theoretical Computer Science*, 2:133–145, 1976.

[EC93] M.H. Escardó and D.M. Claudio. Scott domain theory as a foundation for interval analysis. Technical Report 218, UFRGS/II, Porto Alegre, Brazil, 1993.

[Eda95a] A. Edalat. Domain of computation of a random field in statistical physics. In C. Hankin, I. Mackie, and R. Nagarajan, editors, *Theory and formal methods 1994: Proceedings of the second Imperial College Department of Computing Workshop on Theory and Formal Methods*, Mller Centre, Cambridge, 11-14 September 1995. IC Press.

[Eda95b] A. Edalat. Domain theory and integration. *Theoretical Computer Science*, 151:163–193, 1995.

[Eda95c] A. Edalat. Domain theory in learning processes. In S. Brookes, M. Main, A. Melton, and M. Mislove, editors, *Proceedings of MFPS*, volume one of *Electronic Notes in Theoretical Computer Science.* Elsevier, 1995.

[Eda95d] A. Edalat. Domain theory in stochastic processes. In *Proceedings of Logic in Computer Science (LICS), Tenth Annual IEEE Symposium, 26-19 June, 1995, San Diego.* IEEE Computer Society Press, 1995.

[Eda95e] A. Edalat. Dynamical systems, measures and fractals via domain theory. *Information and Computation*, 120(1):32–48, July 1995.

[Eda96a] A. Edalat. Power domains and iterated function systems. *Information and Computation*, 124:182–197, 1996.

[Eda96b] A. Edalat. The Scott topology induces the weak topology. *Mathematical Structures in Computer Science*, 1996. To appear.

[EE] A. Edalat and M.H. Escardó. Integration in Real PCF. *Information and Computation.* To appear.

[EE95] A. Edalat and M.H. Escardó. Integration in Real PCF (full version). http://theory.doc.ic.ac.uk/people/Escardo, December 1995.

[EE96a] A. Edalat and M.H. Escardó. Integration in Real PCF (extended abstract). In *Proceedings of the Eleventh Annual IEEE Symposium on Logic In Computer Science*, pages 382–393, New Brunswick, New Jersey, USA, July 1996.

[EE96b] A. Edalat and M.H. Escardó. The way-below relation on the function space $\mathcal{R} \to \mathcal{R}$. Department of Computing, Imperial College of Science and Technology, University of London, February 1996.

[EEK] T. Erker, M.H. Escardó, and K. Keimel. The way-below relation on function spaces of semantics domains. *Topology and its applications.* To appear.

[EEK97] T. Erker, M.H. Escardó, and K. Keimel. The way-below relation of function spaces of semantic domains. *Topology and Its Applications*, 1997. To appear.

[EH96] A. Edalat and R. Heckmann. A computational model for metric spaces. *Theoretical Computer Science*, 1996. To appear.

[EN96] A. Edalat and S. Negri. Generalised Riemann integral on locally compact spaces. In A. Edalat, S. Jourdan, and G. McCusker, editors, *Advanced in theory and formal methods of computing: Proceedings of the third Imperial College workshop April 1996.* Imperial College Press, 1996.

[ES] M.H. Escardó and T. Streicher. Induction and recursion on the rartial real line with applications to Real PCF. *Theoretical Computer Science.* To appear.

[ES97] M.H. Escardó and T. Streicher. Induction and recursion on the partial real line via biquotients of bifree algebras. In *Proceedings of the Twelveth Annual IEEE Symposium on Logic In Computer Science*, Warsaw, Polland, Jun 1997.

[Esc] M.H. Escardó. Properly injective spaces and function spaces. *Topology and its Applications.* To appear.

[Esc94] M.H. Escardó. Induction and recursion on the real line. In C. Hankin, I. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, pages 259–282, Mller Centre, Cambridge, 11–14 September 1994. IC Press. 1995.

[Esc96a] M.H. Escardó. PCF extended with real numbers. *Theoretical Computer Science*, 162(1):79–115, August 1996.

[Esc96b] M.H. Escardó. Real PCF extended with $\exists$ is universal. In A. Edalat, S. Jourdan, and G. McCusker, editors, *Advances in Theory and Formal Methods of Computing: Proceedings of the Third Imperial College Workshop, April 1996*, pages 13–24, Christ Church, Oxford, 1996. IC Press.

[Esc97] M.H. Escardó. Properly injective spaces and function spaces. *Topology and Its Applications*, 1997. To appear (available at `http://theory.doc.ic.ac.uk/people/Escardo`).

[Fio93] M.P. Fiore. A coinduction principle for recursive data types based on bisimulation (extended abstract). In *8*th *LICS Conf.* IEEE, Computer Society Press, 1993. (Full version in Information and Computation special issue for LICS 93).

[Fio96] M.P. Fiore. A coinduction principle for recursive data types based on bisimulation. *Information and Computation*, 127(2):186–198, 1996.

[Fre90] P. J. Freyd. Recursive types reduced to inductive types. In *Proceedings of the Fifth Annual IEEE Symposium on Logic In Computer Science*, 1990.

[Fre91] P. J. Freyd. Algebraically complete categories. In A. Carboni et al., editors, *Proc. 1990 Como Category Theory Conference*, pages 95–104, Berlin, 1991. Springer-Verlag. Lecture Notes in Mathematics Vol. 1488.

[Fre92] P. J. Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham, 1991.* Cambridge University Press, 1992. LMS Lecture Notes Series, 177.

[GHK+80] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, 1980.

[Gia93a] P. Di Gianantonio. *A functional approach to computability on real numbers.* PhD thesis, University of Pisa, 1993. Technical Report TD 6/93.

[Gia93b] P. Di Gianantonio. Real number computability and domain theory. In *18th symposioum on Mathematical Foundation of Compueter Science*, number 711 in LNCS, pages 413–422. Springer-Verlag, 1993.

[Gia96a] P. Di Gianantonio. A golden ratio notation for the real numbers. Technical Report CS-R9602, CWI Amsterdam, 1996.

[Gia96b] P. Di Gianantonio. Real number computability and domain theory. *Information and Computation*, 127(1):11–25, 1996.

[Gru] K. Grue. Unrestricted lazy numerical algorithms. Unpublished manuscript.

[Grz57] A. Grzegorczyk. On the definition of computable real continuous functions. *Fund. Math.*, 44:61–77, 1957.

[Gun92] C. A. Gunter. *Semantics of Programming Languages – Structures and Techniques*. The MIT Press, London, 1992.

[Hay85] S. Hayashi. Self-similar sets as Tarski's fixed points. *Publications of the Research Institute for Mathematical Sciences*, 21(5):1059–1066, 1985.

[HL82] K.H. Hofmann and J.D. Lawson. On the order theoretical foundation of a theory of quasicompactly generated spaces without separation axiom. In R.-E. Hoffmann, editor, *Continuous Lattices and Related Topics*, volume 27 of *Mathematik-Arbeitspapiere*, pages 143–160. University of Bremen, 1982.

[Hut81] J. E. Hutchinson. Fractals and self-similarity. *Indiana University Mathematics Journal*, 30:713–747, 1981.

[HY88] J.G. Hocking and G.S. Young. *Topology*. Dover Publications, 1988.

[JS96a] A. Jung and Ph. Sünderhauf. On the duality of compact vs. open. In S. Andima, R. C. Flagg, G. Itzkowitz, P. Misra, Y. Kong, and R. Kopperman, editors, *Papers on General Topology and Applications: Eleventh Summer Conference at University of Southern Maine*, Annals of the New York Academy of Sciences, 1996. To appear in Topology and its Applications.

[JS96b] A. Jung and Ph. Sünderhauf. Uniform approximation of topological spaces. Technical Report CSR-96-3, School of Computer Science, The University of Birmingham, 1996. 19pp., available from `http://www.cs.bham.ac.uk/`.

[Jun90] A. Jung. The classification of continuous domains. In *Logic in Computer Science*, pages 35–40. IEEE Computer Society Press, 1990.

[Kel55] J.L. Kelley. *General Topology*. D. van Nostrand, New York, 1955.

[Kle52] S.C. Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[Ko91] Ker-I Ko. *Complexitity Theory of Real Functions*. Birkhauser, Boston, 1991.

[KP79] A. Kanda and D. Park. When are two effectively given domains identical? In K. Weihrauch, editor, *Theoretical Computer Science* 4^{th} *GI Conference*, LNCS, 1979.

[KV65] S.C. Kleene and R.E Vesley. *The Foundations of Intuitionistic Mathematics: Especially in Relation to Recursive Functions*. North-Holland, Amsterdam, 1965.

[Law73] F.W. Lawvere. Metric spaces, generalized logic, and closed categories. In *Rend. del Sem. Mat. e Fis. di Milano*, volume 43, pages 135–166, 1973.

[LS81] D.J. Lehmann and M.B. Smyth. Algebraic specification of data types: a synthetic approach. *Math. Syst. Theory*, 14:97–139, 1981.

[LS86] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, Cambridge, 1986.

[Luc77] H. Luckhardt. A fundamental effect in computations on real numbers. *Theoretical Computer Science*, 5:321–324, 1977.

[McL92] C. McLarty. *Elementary Categories, Elementary Toposes*. Clarendon Press, Oxford, 1992.

[ML70] P. Martin-Löf. *Notes on Constructive Mathematics*. Almqvist & Wiksell, Stockholm, 1970.

[ML71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[MM96] V. Ménissier-Morain. Arbitrary precision real arithmetic: design and alogrithms. Submitted to J. Symbolic Computation, 1996.

[Moo66] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.

[Myh53] J. Myhill. Criteria of constructivity of real numbers. *J. Symbolic Logic*, 18:7–10, 1953.

[Pau87] L.C. Paulson. *Logic and Computation: Interactive Proof with LCF*. Cambridge University Press, Cambridge, 1987.

[PEE97] P.J. Potts, A. Edalat, and M.H. Escardó. Semantics of exact real arithmetic. In *Proceedings of the Twelveth Annual IEEE Symposium on Logic In Computer Science*, Warsaw, Polland, Jun 1997.

[PeR83] M.B. Pour-el and I. Richards. Computability and noncomputability in classical analysis. *Trans. Am. Math. Soc.*, pages 539–560, 1983.

[Phi92] I.C.C. Phillips. Recursion theory. In S. Abramsky, D.M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 79–187. Clarendon Press, Oxford, 1992.

[Plo77] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(1):223–255, 1977.

[Plo78] G. Plotkin. $\mathcal{B}^\omega$ as a universal domain. *Journal of Computer and System Sciences*, 17:209–236, 1978.

[Plo80] G. Plotkin. Domains. Post-graduate Lectures in advanced domain theory, University of Edinburgh, Department of Computer Science. http:// ida.dcs.qmw.ac.uk/ sites/ other/ domain.notes.other, 1980.

[Poi92] A. Poigné. Basic category theory. In S. Abramsky, D.M. Gabbay, and T.S.E Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 413–640. Clarendon Press, Oxford, 1992.

[Pot95] P.J. Potts. The storage capacity of forgetful neural networks. Master's thesis, Department of Computing, Imperial College, 1995.

[Pot96] P.J. Potts. Computable real arithmetic using linear fractional transformations. Manuscript presented during the second Computation and Approximation Workshop, held in Darmstadt, September 1996.

[Ric54] H.G. Rice. Recursive real numbers. *Proc. Amer. Math. Soc.*, pages 784–791, 1954.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[Roy88] H.L. Royden. *Real Analysis.* Collier Macmillan Publishing, third edition, 1988.

[Rud76] W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, New Yourk, 3rd edition, 1976.

[Sch93] A. Schalk. *Algebras for Generalized Power Constructions.* PhD thesis, Technische Hochschule Darmstadt, July 1993. ftp://ftp.cl.cam.ac.uk/papers/as213/diss.dvi.gz.

[Sco72a] D. S. Scott. Continuous lattices. In F.W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lectures Notes in Mathematics*, pages 97–136. Springer-Verlag, 1972.

[Sco72b] D. S. Scott. Lattice theory, data types and semantics. In *Formal semantics of programming languages*, pages 66–106, Englewood Cliffs, 1972. Prentice-Hall.

[Sco76] D. S. Scott. Data types as lattices. *SIAM Journal on Computing*, 5:522–587, September 1976.

[Smy77] M.B. Smyth. Effectively given domains. *Theoretical Computer Science*, 5(1):256–274, 1977.

[Smy83] M.B. Smyth. Power domains and predicate transformers: a topological view. In J. Diaz, editor, *Automata, Languages and Programming*, pages 662–675. Springer-Verlag, 1983. LNCS 154.

[Smy87] M.B. Smyth. Quasi-uniformities: Reconciling domains with metric spaces. In M. Main, A. Melton, M. Mislove, and D. Schimidt, editors, *Mathematical Foundations of Programming Languages*, pages 236–253, London, 1987. Springer-Verlag. LNCS 298.

[Smy89] M.B. Smyth. Totally bounded spaces and compact ordered spaces as domains of computation. In *Oxford Topology Symposium'89*, Oxford, 1989. Oxford University Press.

[Smy92a] M.B. Smyth. I-categories and duality. In M.P. Fourman, P.T. Johnstone, and Pitts A.M., editors, *Applications of Categories in Computer Science*, pages 270–287, Cambridge, 1992. Cambridge University Press. London Mathematical Society Lecture Notes Series 177.

[Smy92b] M.B. Smyth. Topology. In S. Abramsky, D. M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 641–761. Clarendon Press, Oxford, 1992.

[SP82] M.B. Smyth and G. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11(4):761–783, 1982.

[Sto66] R. Stoll. *Set Theory and Logic*. W.H. Freeman and Company, San Fransisco, 1966.

[Str94] T. Streicher. A universality theorem for PCF with recursive types, parallel-or and ∃. *Mathematical Structures for Computing Science*, 4(1):111 – 115, 1994.

[Sün97] Ph. Sünderhauf. A domain-theoretic approach to computable analysis. Department of Computing, Imperial College, March 1997.

[Tur37] A. Turing. On computable numbers, with an application to the Entscheindungproblem. *The London Mathematical Society*, 42:230–265, 1937.

[Vic89] S. Vickers. *Topology via Logic*. Cambridge University Press, Cambridge, 1989.

[Vic96] S. Vickers. Flat completion of quasi-metric spaces. Department of Computing, Imperial College of Science and Technology, University of London, March 1996.

[Viu90] J. Viullemin. Exact real computer arithmetic with continued fractions. *IEEE Transactions on Computers*, 39(8):1087–1105, 1990.

[Vui88] J. Vuillemin. Exact real arithmetic with continued fractions. In *Proc. ACM Conference on Lisp and Functional Programming*, pages 14–27, 1988.

[Wei87] K. Weihrauch. *Computability.* Springer-Verlag, Berlin, 1987.

[Wei95] K. Weihrauch. A simple introduction to computable analysis. Technical Report 171 – 7/1995, FernUniversitat, 1995.

[Wie80] E. Wiedmer. Computing with infinite objects. *Theoretical Computer Science*, 10:133–155, 1980.

Index