CWI

Computable Analysis with Applications to Dynamic Systems

P.J. Collins

MAC-1002

Centrum Wiskunde & Informatica

Centrum Wiskunde & Informatica (CWI) is the national research institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organisation for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Computing (MAC)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Science Park 123, 1098 XG Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3703

Computable Analysis with Applications to Dynamic Systems

Pieter Collins

19 May 2010

Abstract

In this article we develop a theory of computation for continuous mathematics. The theory is based on earlier developments of computable analysis, especially that of the school of Weihrauch, and is presented as a model of intuitionistic type theory. Every effort has been made to keep the presentation as simple yet general as possible. The core subject matter of Turing computability and computable analysis should be accessible to non-specialists with a solid background in general topology and analysis, but important technical results are also proved. To show the potential use of the theory, some simple applications are given to dynamical systems and control theory.

Keywords: Computable analysis, dynamic systems, intuitionistic type theory, Turing machines.

Mathematics Subject Classification: 03D78; 26E40; 54H20.

Contents

Chapter 1

Introduction

In this paper, we develop a theory of computation for continuous mathematics. The aim of this paper is to give an exposition which is explicitly based on Turing machine model of computation, is powerful enough to study real computational problems arising in practise (taken from the areas of dynamical systems and control theory), yet is as straightforward and direct as possible, and uses terminology which is natural in classical topology and analysis.

The main idea is to consider which mathematical operations are *computable*, by which we mean that the result can be computed by a program running on a digital computer. Since we are dealing with objects from continuous mathematics, we are typically dealing with uncountable sets of objects (such as the real numbers), so we cannot specify an arbitrary object with a finite amount of data. However, the objects we consider form topological or metric spaces, and can be *approximated* arbitrarily accurately with a finite amount of data. Hence we describe an object by an infinite *stream* of data, but in such a way that useful information can be obtained from a finite amount of data. The inherent use of approximations means that there is a very closed link between topology and representations; indeed any representation of a set of objects induces a natural topology on that set.

An operation is *computable* if a description of the result can be computed from a description of the arguments. At any stage of computation, only a finite amount of memory can be used, but most computations need an unbounded amount of internal memory to compute the complete output. The input of a computation can in principle be an arbitrary valid sequence of characters, we do not require that there be an effective procedure to determine whether a string is valid. In practise, inputs will typically be taken from a countable set of *computable* sequences i.e. those which can be computed by a Turing machine, or from a subset which can be described symbolically e.g. the rational numbers. One of the fundamental results is that only continuous functions can be computable; however which operations are continuous depends on the topology, since this affects the amount of information which is present in the input or required in the output.

The exact details as to *how* objects of a set are described is immaterial in determining which operations are possible, as long as a description in one representation can be effectively converted to a description in another representation. We therefore look at equivalence classes of representations as defining a *type* of object, and consider the computable operations between types. In general, there are inequivalent representations

of a given topological space, but typically in practise, one of these is canonical. For example, there is a unique equivalence class of representations of the real numbers $\mathbb{R}$ for which arithmetic is computable, strict comparison is verifiable, and limits of strongly-convergent Cauchy sequences are computable. Hence there is a canonical type $\mathcal{R}$ of the real numbers. From this type, we can canonically build up other types, such as Euclidean space $\mathcal{R}^n$, continuous functions $\mathcal{C}(\mathcal{R}^n; \mathcal{R}^m)$, and open sets $\mathcal{O}(\mathcal{R}^n)$.

A canonical way of describing an element of a countably-based topological space is to list the basic open sets containing it. In this way, the basic open sets become the fundamental objects describing the space, rather than the points. This observation indicates strong links with *locale theory*, which can be seen as a kind of "pointless topology".

The theory presented is a model of *intuitionistic type theory* [40]. This means that it is always possible to construct finite products and function types, with the corresponding natural computable operations. However, not all types and computable operations can be constructed from a finite collection of base types; arbitrary subtypes are allowable, and sometimes it is necessary to return to first principles to show that an operation is computable, or that a constructive definition is well-defined and matches the classical definition. For example, to prove computability of the solution of an ordinary differential equation, it is necessary to appeal to results of classical analysis to prove that an construction based on Euler time-steps has the classical properties of a solution.

The ideas in this theory can be traced back to the intuitionistic logic of Brouwer [51] and the constructive mathematics of Markov [39] and Bishop [8]. Although these theories deal with constructive mathematics rather than explicitly with computation, the idea of constructive existence is clearly linked with that of an algorithmic procedure for computing a mathematical object.

A first link with computability was via the theory of Scott domains [27], which were initially developed to give a semantics for programming languages, but which were soon recognised as a possible foundation for real analysis. There is a considerable body of work applying domain theory to various bodies of real analysis e.g. [21, 20]. However, the foundations of domain theory are based on lattice theory, and the notation and terminology is still heavily based on these foundations, rather than on the natural language for topology and analysis. Further, domain theory is usually not directly presented in terms of Turing computation, and an extra level of theory is still needed to give an explicit relationship with computation (though intuitively it is clear for experts how to proceed).

An attempt to provide a simplified theory of computable analysis based on type-two effectivity, which is explicitly based on Turing computation and using natural language was given by Weihrauch [53]. In this theory, *representations* are used to give a computational meaning to objects from continuous mathematics. Unfortunately, the elegance of the framework tends to get lost in a plethora of subtly different representations for different classes of object, and in the necessity to always specify the representation used explicitly. From this point of view, the use of types represents an important notational simplification, which we hope also improves the readability and accessibility of the theory. A further drawback of the presentation in [53] is that the exposition is mostly restricted to Euclidean spaces, and not all the results extend to spaces which are not Hausdorff or not locally-compact.

An exposition of computable analysis focusing on complexity theory of real functions

was given by Ko, based on the notion of an *oracle Turing machine*. Although the framework is equivalent to that of Weihrauch, the use of the term "oracle" is rather confusing; in computer science, an oracle is a machine which can provide the answer to an unsolvable problem, in the work of Ko, an oracle may be used to provide an input to a computation (e.g. in the form of a decimal expansion of an uncomputable real number) but is not used in the computation itself.

This article is organised as follows. In Chapter 2, we give an overview of Turing computability theory for discrete computations on words over some alphabet Σ, and show how this can be extended to computations over sequences. We then give a formal definition of *naming systems*, by which elements of some arbitrary set can be related to objects with some "computational meaning". Chapter 3 is the heart of the paper. Here we give a complete exposition of computable analysis for elementary classes of objects, including points, sets and functions. In Chapter 4, we relate the material of Chapter 3 to concepts from classical topology and locale theory. We give a characterisation of topological spaces which have a representation which adequately preserves the topological structure, describe the Scott topology on open sets, explain the concept of a sober space, and give an overview of the theory of core-compact and locally-compact spaces. Finally, in Chapter 5, we give some applications to dynamical systems and control theory.

There have been a number of excellent Ph.D. thesis in the area of computable analysis, especially those of Brattka [10], Bauer [6] and Schröder [46] and Battenfeld [4]. In particular, it was Schröder who first classified the topological spaces which can be given a representation capturing the topology. Indeed, Schröder's classification extends to weak limit spaces, a generalisation of topological spaces which may have some applications in probability theory. Other important articles giving an exposition of a large part of the theory include those of Escardo [22] and Blanck [9] and Taylor [49]. Books specifically relating to computability in analysis include those of Pour-El and Richards [42], Ko [38] and Weihrauch [53]. Other interesting books which contain deeper material in logic, domain theory and topos theory include those of Vickers [52], Johnstone [34, 35], Gierz et al. [27], Clementino, Giula and Tholen [12]. For an introduction to type theory, see [40]. Books relating to rigorous computation include Jaulin et al. [33], Hansen [30], Aberth [1], and Moore et al. [41].

We reiterate that it is the aim of this paper to give as concise an introduction to the theory as possible, while still providing proofs of the most important theorems. For this reason, we have restricted ourselves to computability theory for topological spaces, and not for the more general class of weak limit spaces, since in almost applications we use topological spaces. However, we have given most of the development of the theory for general topological spaces, and have only restricted to Hausdorff and local-compact spaces where necessary. This is important, since types of open/closed/compact subsets of a space are not Hausdorff, and non-locally-compact spaces quickly arise as function spaces, and need to be covered to discuss solution sets of dynamic systems.

We have not used the language of Scott domains, though we have given an explicit exposition of the Scott topology on the open sets, as this is the topology induced by the canonical representation. We have given an introduction to point-free topology, since this is the natural way to obtain a representation for a countably-based topological space, but have not used the language of locale theory. We have introduced the notion of a sober space, since this is required to give a link between a type-theoretic construction of compact sets by the subset predicate and the classical notion of a compact point-set.

Chapter 2

Turing Computability

In this chapter we give an outline of the theory of computability based on the standard notion of *Turing machine*, an abstract digital computing device with unlimited memory. The Turing model of computation is the standard accepted model of digital computation. There are many variants of the basic Turing machine, and generalisations to computational models which are closer to the architecture of modern digital computers, but they all yield the same computable functions. When the model was introduced by Turing in [50], it was envisaged that a human was performing the calculations, but the theory is completely appropriate for mechanical devices. The only non-realistic assumption is that the machine has enough memory available to complete the task. The *Church-Turing thesis* asserts that any algorithmic procedure for performing a calculation is given by a Turing-computable function. It seems reasonable that the thesis even holds for *reliable* analogue computing devices, due to external noise and constraints on space and energy. See e.g. [7, 55] for more detailed discussions of analogue computation.

2.1 Turing machines

** Probably better to have moving tape heads **

The standard model of computation is that of the *Turing* machine. This is a model of a process for performing computations in which only a fixed finite amount of information can be used at any stage, but for which an arbitrarily amount of storage is available. In the standard model, computations are performed on one or more infinite or bi-infinite *tapes* whose cells contain symbols from an alphabet Γ. For a digital computer, one might expect the alphabet to be $\{0, 1\}$, but for expository purposes we can take a more expressive alphabet, such as the ASCII or Unicode character sets. It will also be useful to distinguish a *tape alphabet* Γ from the *input/output* alphabet Σ.

We first give a description of a multiple-tape Turing machine as a dynamical system; later we will see what it means for the machine to compute a function.

Definition 2.1 (Turing machine)**.** A k-tape Turing machine is described by a tuple (Γ, Q, τ), where Γ and Q are finite sets, and $\tau : Q \times \Gamma^k \to Q \times \Gamma^k \times \{-1, 0, +1\}^k$ describes the *transition relation*.

The action of a Turing machine is as follows. The state of the machine is given by $(q, h_1, \ldots, h_k, \vec{s}_1, \ldots, \vec{s}_k)$, where $q \in Q$, each $h_i \in \mathbb{Z}$ and each $\vec{s}_i \in \Gamma^{\mathbb{Z}}$. The next state

$(q', h'_1, \ldots, h'_k, \vec{s}'_1, \ldots, \vec{s}'_k)$ is determined by taking

$$\big(q', (s'_{1,h_1}, \ldots, s'_{k,h_k}), (\delta_1, \ldots, \delta_n)\big) = \tau\big(q, (s_{1,h_1}, \ldots, s_{k,h_k})\big), \tag{2.1}$$

and setting

$$h'_i = h_i + \delta_i \text{ for } i = 1, \ldots, k. \tag{2.2}$$

** Fix definition for tape heads **

Informally, the value of $q \in Q$ is the *register state* of the machine, and is a kind of "program counter". The h_i represent the position of the "tape head" for the i^{th} tape. For each computational step starting at register state q, the machine scans the values $s_{1,h_1}, \ldots, s_{k,h_k}$ at the tape head, replaces them with new values $s'_{1,h_1}, \ldots, s'_{k,h_k}$, shifts the tape head left or right depending on the value of $\delta_1, \ldots, \delta_k$, and updates the register state to q'.

Remark 2.2. In the definition given here, we "shift" the tapes heads left and right to scan new symbols. Sometimes, the symbols $\{\mathtt{L}, \mathtt{N}, \mathtt{R}\}$ are used instead of $\{-1, 0, +1\}$ for the δ_i. An equivalent model sometimes used in the literature is to have shiftable tapes rather than movable tapes. The state of the system then does not need the integer variables h_i. This model yields the same computable functions in Definition 2.3.

Now suppose Γ is an alphabet containing a special blank symbol $\sqcup$, and $\Sigma \subset \Gamma \setminus \{\sqcup\}$. Define an encoding $\iota : \Sigma^* \to \Gamma^{\mathbb{Z}}$ by taking $(\iota(w))_j = w_j$ for $j = 0, \ldots, |w| - 1$, and $(\iota(w))_j = \sqcup$ otherwise. Since $\sqcup \notin \Sigma$, the encoding ι is injective, so w can be unambiguously recovered from $\vec{s} = \iota(w)$.

When using multiple tapes, we can separately consider *input*, *output* and *work* tapes. An input tape can only be read from and contains the initial input, the output tape can only be written to, but not altered, and contains the final output. The work tape and output tape start off completely blank. Formally, the i^{th} tape is *unidirectional* if for any transition, we always have $\delta_i \in \{0, +1\}$, is *read-only* if we always have $s'_i = s_i$, and is *write-only* if $s'_i = s_i$ or $s_i = \sqcup$, and *write-once*. A unidirectional read-only tape is an *input tape*. A unidirectional write-only tape such that $\delta = +1$ implies $s' \neq \sqcup$ is an *output tape*.

We now show how Turing machines can be used to compute partial word functions $(\Sigma^*)^m \rightharpoonup (\Sigma^*)^n$. We need to consider partial functions since not all words should be considered valid inputs; for example, the string `2/3/4` is not a valid description of a rational number. To distinguish "ordinary" computation on words from computation on sequences (which will be introduced in Section 2.2), we call this *type-one* Turing computation.

Definition 2.3 (Type-one Turing computation)**.** Let (Γ, Q, τ) be an k-tape Turing machine where Γ contains a special *blank* character $\sqcup$, and $\Sigma \subset \Gamma \setminus \{\sqcup\}$. Let $q_0, q_f \in Q$ be the *initial* and *final* states, respectively. Let $m, n \in \mathbb{N}$ be such that $m + n \leq k$ define the number of *input* and *output* tapes respectively.

Then $M = (\Sigma, m, n, \Gamma, Q, q_0, q_f, \tau)$ defines a partial function $(\Sigma^*)^m \to (\Sigma^*)^n$ as follows: For input $(w_1, \ldots, w_m) \in (\Sigma^*)^m$, the initial state is given by $q = q_0$, $\vec{s}_i = \iota(w_i)$ for $i = 1, \ldots, m$ and $\vec{s}_i = \sqcup^{\mathbb{Z}}$ otherwise. The computation proceeds as given by Definition 2.1 until the register state q is equal to q_f, at which point the machine halts. The computation is *valid* if the machine halts, and in the halting state, there

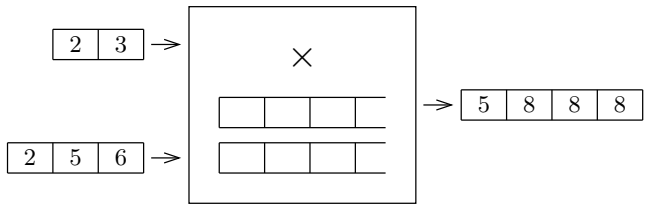

Figure 2.1: A Turing machine computing the product of two integers in decimal form.

exist $v_i \in \Sigma^*$ such that $\vec{s}_{m+i} = \iota(v_i)$ for $i = 1, \ldots, n$. The result of the computation is $(v_1, \ldots, v_n)$.

A partial function $(\Sigma^*)^m \rightharpoonup (\Sigma^*)^n$ is *computable* if it is the function computed by some Turing machine.

Remark 2.4. A theory of computable word functions can be developed without separate "input", "output" and "work" tapes. Indeed, the computability theory can be developed for single-tape machines which replace the initial contents of the tape (the input) by the output. We use the definition above for compatibility with the definition of type-two computability in Section 2.2.

Remark 2.5. The computability theory could equally well be presented using general recursive functions in $\mathbb{N}^{\mathbb{N}}$ instead of Turing computable functions on Σ^ω. However, for a meaningful *complexity* theory, we need to use a finite alphabet Σ.

Remark 2.6. In order to show that a function is computable, we in principle need to explicitly construct a Turing machine which computes the function. In general, this is a tedious exercise in Turing machine programming. In this article, in the few cases where this is necessary, we shall merely give a sketch of how the function could be computed, without explicitly describing a Turing machine.

The most important properties of Turing computability are summarised in the following theorem

Theorem 2.7.

1. *If* $\xi : (\Sigma^*)^l \rightharpoonup (\Sigma^*)^m$ *and* $\eta : (\Sigma^*)^m \rightharpoonup (\Sigma^*)^n$ *are computable, then* $\zeta = \eta \circ \xi : (\Sigma^*)^l \rightharpoonup (\Sigma^*)^n$ *is computable, where we take* $\mathrm{dom}(\eta \circ \xi) = \xi^{-1}(\mathrm{dom}(\eta))$.

2. *There is a computable* tupling *function* $\tau : \Sigma^* \times \Sigma^* \to \Sigma^*$*, and computable* projections $\pi_1, \pi_2 : \Sigma^* \rightharpoonup \Sigma^*$ *such that* $\pi_i(\tau(w_1, w_2)) = w_i$ *for* $i = 1, 2$.

3. *There is a* universal *Turing machine* $\mathcal{U}$ *computing a function* $\mu : \Sigma^* \times \Sigma^* \rightharpoonup \Sigma^*$ *such that for every Turing machine* $\mathcal{M}$ *computing a function* $\phi : \Sigma^* \rightharpoonup \Sigma^*$*, there is a word* $a \in \Sigma^*$ *such that* $\mu(a, w) = \phi(w)$ *for all* $w \in \Sigma^*$.

4. *There is a computable function* $\lambda : \Sigma^* \times \Sigma^* \rightharpoonup \Sigma^*$ *such that* $\mu(\lambda(a, v), w) = \mu(a, \tau(v, w))$ *for all* $a, v, w \in \Sigma^*$.

Part (1) shows that the composition of computable functions is computable. Roughly speaking, the computation can be carried out by running a machine computing ξ until it halts, and then running the computation of η. Part (2) shows that the contents of

two input tapes can be combined into one, and the contents recovered later. A (slightly) simpler case is that of computing a function $\tau : \Sigma^* \times \Sigma^* \to \hat{\Sigma}^*$, where $\hat{\Sigma}$ contains a symbol `,` not present in Σ, and $\tau(w_1, w_2) = w_1\texttt{,}w_2$. To avoid introducing a new symbol, we can take $\tau(w_1, w_2) = \texttt{1}^{|w_1|}\texttt{0}w_1w_2$ to recover the break between w_1 and w_2. Part (3) shows how to construct a machine with two inputs, one of which is a "program" for the computation of another machine. For example, assuming $Q \subset \Sigma$ and $\texttt{L}, \texttt{N}, \texttt{R} \in \Sigma$, we can encode a transition $(q, s) \to (q', s', \delta)$ by the string $q\texttt{,}s\texttt{,}q'\texttt{,}s'\texttt{,}\delta$. Part (4) shows that given a computable function of two variables $\xi(v, w)$ with program a, we can compute the program of the function taking w to $\xi(v, w)$ from a and v.

For more details on type-one Turing computation, see Sipser [48].

2.2 Type-two effectivity

When dealing with computation on objects from continuous mathematics, we shall need functions on sequences $\Sigma^\omega \rightharpoonup \Sigma^\omega$ or, more generally, $(\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$. In order to compute such a function, a machine needs to run forever, reading symbols from the *input* tape(s) and writing to the *output* tape, with access to finitely many internal *work* tapes of infinite size. Symbols from the output tape may not be overwritten once they have been produced. A computational run is *valid* if the machine writes infinitely many symbols to the output tape. A formal definition is given below. See Weihrauch [53, Chapter 2] for a comprehensive treatment.

Definition 2.8 (Type-two computation)**.** Let (Γ, Q, τ) be an k-tape Turing machine where Γ contains a special *blank* character $\sqcup$, and $\Sigma \subset \Gamma \setminus \{\sqcup\}$. Let $q_0, q_f \in Q$ be the *initial* and *final* states, respectively. Let $m, n \in \mathbb{N}$ be such that $m + n \leq k$ define the number of *input* and *output* tapes respectively. We require that the input tapes $\vec{s}_1, \ldots, \vec{s}_m$ and output tapes $\vec{s}_{m+1}, \ldots, \vec{s}_{m+n}$ are unidirectional, that the input tapes are read-only and the output tapes are write-only. Define an encoding $\iota : \Sigma^\omega \to \Gamma^{\mathbb{Z}}$ by taking $(\iota(p))_j = p_j$ for $j \geq 0$ and $(\iota(w))_j = \sqcup$ for $j < 0$.

Then M defines a partial function $(\Sigma^\omega)^m \to (\Sigma^\omega)^n$ as follows: For input $(p_1, \ldots, p_m) \in (\Sigma^\omega)^m$, the initial state is given by $q = q_0$, $\vec{s}_i = \iota(p_i)$ for $i = 1, \ldots, m$ and $\vec{s}_i = \sqcup^{\mathbb{Z}}$ otherwise. The computation proceeds as given by Definition 2.1. The computation is *valid* if the machine does not halt, and also writes infinitely many symbols on each output tape. The result of the computation is $(q_1, \ldots, q_n)$, where each q_i is defined by $(q_i)_j = s_{m+i,j}$ for sufficiently large j.

Note that it is an undecidable problem to decide whether a type-two Turing machine with no inputs produces a valid (i.e. infinite) output.

Definition 2.9 (Machine computability)**.** A partial function $\eta : (\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ is *(machine) computable* if it can be computed by a type-two Turing machine.

A sequence $p \in \Sigma^\omega$ is *(machine) computable* if there is a type-two Turing machine with no inputs which outputs p.

Since an output tape of one machine may be used as the input of another, we and that the resulting combination can be realised by a single machine, we have the following result:

Property 2.10 (Composition of machine-computable functions)**.** *Let* $\eta : (\Sigma^\omega)^l \rightharpoonup (\Sigma^\omega)^m$ *and* $\zeta : (\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ *be computable. Then* $\zeta \circ \eta : (\Sigma^\omega)^l \rightharpoonup (\Sigma^\omega)^n$*, with* $\mathrm{dom}(\zeta \circ \eta) = \eta^{-1}(\mathrm{dom}(\zeta))$*, is computable.*

Just as in the type-one case, we can combine the data on multiple input tapes into a single output tape, and later recover the original data.

Property 2.11 (Tupling)**.** *For any* $n \in \mathbb{N}$*, there is a machine-computable function* $\tau_n : (\Sigma^\omega)^n \to \Sigma^\omega$*, and machine-computable functions* $\pi_{n,i} : \Sigma^\omega \rightharpoonup \Sigma^\omega$ *such that for any* $i \leq n$*,* $\pi_{n,i}(\tau_n(p_1, \ldots, p_n)) = p_i$*.*

The natural tupling function is given by $(\tau_n(p_1, \ldots, p_n))_{in+j} = (p_j)_i$ for $i \in \mathbb{N}$ and $j = 1, \ldots, n$, and the projections by $(\pi_{n,i}(p))_j = p_{in+j}$. It is clear that these functions can be computed by a type-two Turing machine.

As well as the finitary tupling functions $(\Sigma^*)^n \to \Sigma^*$ and $(\Sigma^\omega)^n \to \Sigma^\omega$, we can also define mixed and infinite tupling functions. The infinite tupling function $(\Sigma^\omega)^\omega \to \Sigma^\omega$ can be given by

$$\tau(p_0, p_1, \ldots) = q \text{ if } q_{g(i,j)} = (p_i)_j, \text{ where } g(i,j) = (i+j)(i+j+1)/2+j \text{ for } i, j \in \mathbb{N}. \quad (2.3)$$

It is clear that g is a computable bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. If $\widehat{\Sigma} \supset \Sigma$ and contains a special *separator* character '**,**' not in Σ, then tupling $(\Sigma^*)^\omega \to \widehat{\Sigma}^\omega$ can be performed by taking $\tau(w_0, w_1, \ldots) = w_0\,\mathbf{,}\,w_1\,\mathbf{,}\,w_2\,\mathbf{,}\cdots$. To tuple $(\Sigma^*)^\omega \to \Sigma^\omega$, we can choose an "escape" character '\' as well as a separator '**,**', and replace any occurrence of '\' in w_i with the string "\\" and '**,**' with "**,**".

We shall also want to construct sequences by tupling infinitely many words from a subset W of Σ^*. We say W is *prefix-free* if v is not a prefix of w for all $v, w \in W$. For a prefix-free subset of W, if $v_1, \ldots, v_m$ and $w_1, \ldots, w_n$ are words in W with $v_1 v_2 \cdots v_m = w_1, \ldots, w_n$ then $m = n$ and $v_i = w_i$ for all i. In other words, tupling $W^\omega \to \Sigma^\omega$ can be accomplished by concatenation.

Notation 2.12. We henceforth write $\langle \cdot, \cdots, \cdot \rangle$ for any tupling function, and subscript $\langle \cdot \rangle_i$ for the projection onto the i^{th} element. We write $w \lhd p$ if $p = \langle w_0, w_1, \ldots \rangle$ and $w = w_i$ for some $i \in \mathbb{N}$.

We now consider some topological aspects of computable functions.

Definition 2.13 (Topology on Σ^ω)**.** Define a topology on Σ^ω by taking the basic open sets to be the *cylinder sets*

$$C_w = \{p \in \Sigma^\omega \mid p_i = w_i \text{ for } i = 0, \ldots, |w| - 1\} \quad (2.4)$$

where $w \in \Sigma^*$ is a word of length $|w|$.

The following theorem shows that machine-computable functions are continuous relative to the topology defined by the cylinder sets.

Theorem 2.14 (Machine computability implies continuity)**.** *Any machine-computable function* $\eta : (\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ *is continuous.*

Sketch of proof. For simplicity, consider $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$. Consider $p \in \mathrm{dom}(\eta)$. For all $n \in \mathbb{N}$, there exists $m \in \mathbb{N}$ such that n symbols of $\eta(p)$ have been written to the output tape after m computational steps, and hence after at most m digits of p have been read. Then for any $q \in \mathrm{dom}(\eta)$ such that $q|_{[0,m)} = p|_{[0,m)}$, the output after m computational steps is the same as that of p, so $\eta(q)|_{[0,n)} = \eta(p)|_{[0,n)}$. Note that setting $v = p|_{[0,m)}$ and $w = \eta(p)|_{[0,n)}$ we have shown that $C_v \subset \eta^{-1}(C_w)$. Hence η is continuous at p. □

This result provides the basis for the main results on *uncomputability*; it suffices to prove discontinuity.

Recall that a subset of a topological space is a *G_δ-set* if it is a countable intersection of open sets. In particular, any open set is a G_δ-set, and in any locally-compact Hausdorff space, so is any closed set.

Proposition 2.15 (Natural extension)**.** *Any continuous function $\eta : (\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ extends naturally to a continuous function on a G_δ-set.*

Sketch of proof. For simplicity, consider $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$. Define a partial function $\tilde{\eta} : \Sigma^* \rightharpoonup \Sigma^*$ by taking $\mathrm{dom}(\tilde{\eta}) = \{v \in \Sigma^* \mid \mathrm{dom}(\eta) \cap C_v \neq \emptyset\}$, and $w = \tilde{\eta}(v)$ of maximal length such that $\eta(C_v \cap \mathrm{dom}(\eta)) \subset C_w$. Note that if v_1 is a prefix of v_2, then $\tilde{\eta}(v_1)$ is a prefix of $\tilde{\eta}(v_2)$. Then for $p \in \Sigma^\omega$, if $C_v \cap \mathrm{dom}(\eta) \neq \emptyset$ for every prefix v of p, and $|\tilde{\eta}(v)| \to \infty$ as $|v| \to \infty$, then we can define $\hat{\eta}(p) = q$ where $\{q\} = \bigcap\{C_{\tilde{\eta}}(p|_{[0,k)}) \mid k \in \mathbb{N}\}$. Clearly, $\hat{\eta}$ is an extension of η, and $\mathrm{dom}(\hat{\eta})$ is a G_δ-set. □

The set of continuous partial functions $\eta : (\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ with G_δ-domain has continuum cardinality. This means that the continuous partial functions $(\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$ with G_δ-domain can also be represented by sequences. This is a similar closure property for continuous functions provided by the universal Turing machine result for computable functions; computable word functions can be represented by words and continuous stream functions can be represented by streams.

We can now present the main result of this section, the utm (universal Turing machine) theorem, and Kleene's smn[1] theorem. Note the interplay between the *finite* description of Turing machines by words, and the *infinite* description of continuous functions by sequences.

Theorem 2.16 (Universal Turing machines/smn)**.** *There exist machine-computable functions $\varepsilon : \Sigma^\omega \times \Sigma^\omega \rightharpoonup \Sigma^\omega$ and $\sigma : \Sigma^\omega \times \Sigma^\omega \rightharpoonup \Sigma^\omega$ with the following properties:*

1. *For any continuous function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ with G_δ-domain, there exists $a \in \Sigma^\omega$ such that $\varepsilon(a, \cdot) = \eta$.*

2. *If η is computable, then a can be taken to be a computable element of Σ^ω.*

3. *For all $a, p, q \in \Sigma^\omega$, $\varepsilon(\sigma(a, p), q) = \varepsilon(a, \tau(p, q))$.*

The function ε is the *evaluation* function. The first argument a is an encoding of the continuous partial function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ as an element of Σ^ω. Further, if the partial function η is computable, then it has a computable encoding. Note that there may be

[1]The term "smn" is standard, and refers to letters occurring in the original statement of the theorem in [37] rather than being an abbreviation.

several different a corresponding to the same function f, some of which may be uncomputable even if f is computable. The function σ is a type-two analogue of the function s in the smn-theorem. In particular, given a computable function η of two variables, there is a computable sequence a such that $\eta(p,q) = \varepsilon(a, \tau(p,q)) = \varepsilon(\sigma(a,p),q)$.

Sketch of Proof. Given a continuous function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$, we can define the set of all pairs $(v,w) \in \Sigma^* \times \Sigma^*$ such that $\eta(p)|_{|w|} = w$ whenever $p_{|v|} = v$. (In other words, any input with prefix v results in an output with prefix w.) By tupling in $(\Sigma^* \times \Sigma^*)^\omega$, we can list *all* such pairs as a sequence in Σ^ω, and reconsruct the function η from this list. If η is computable, then the list can be constructed from a description of the Turing machine. □

2.3 Computation on words and sequences

We now have two theories of computation, type-one computability, which deals with finite computations on words Σ^*, and type-two computation, which deals with infinite computations on sequences Σ^ω. In many situations, we wish to combine these two types of computation; we may, for example, wish to compute a function with mixed arguments e.g. $\Sigma^* \times \Sigma^\omega \to \Sigma^\omega$, compute an infinite sequence from a finite input $\Sigma^* \to \Sigma^\omega$, or convert an infinite computation to a finite one.

At first sight, it seems straightforward to place type-one computation in the framework of type-two computation. Both words in Σ^* and sequences in Σ^ω are naturally expressed as sequences over the alphabet $\widehat{\Sigma} = \Sigma \sqcup \{\text{␣}\}$. Where the result of a computation on an output tape is a word $w \in \Sigma^*$, the computation continues forever, writing $w_0 \cdots w_{n-1}$ on the output tape, optionally followed by an arbitrary number of '␣'. Unfortunately, this approach suffers from the subtle drawback that it is impossible to know when a word has been completed.

The simplest way of correcting this defect is to introduct a special "carriage return symbol" '↵' to signify the end of a word. Hence a word $w = w_0 w_1 \cdots w_{n-1}$ is encoded on an input tape as $w_0 w_1 \cdots w_{n-1}$↵␣$\cdots$; note that the symbols after the '↵' are taken to be blanks. Once the '↵' symbol has been encountered, the result for that tape is known. In principle, the computation could be halted once the result on all output tapes is known.

Definition 2.17 (Mixed Turing computation)**.** Computation on words Σ^* and sequences Σ^ω can be combined in a type-two Turing computation with alphabet Γ containing special symbols ␣, ↵ $\notin \Sigma$. An element p of Σ^ω is encoded on a tape by $\vec{s} \in \Gamma^{\mathbb{Z}} = \iota(p)$ with $s_i = p_i$ for $i \geq 0$ and $s_i =$ ␣ otherwise. An element w of Σ^* is encoded on a tape by $\vec{s} = \iota(p)$ with $s_i = w_i$ for $0 \leq i < |w|$, $s_{|w|} =$ ↵, and $s_i =$ ␣ otherwise.

The computation proceeds as for a standard type-two computation. A computation is *valid* if it runs forever, writing either $\iota(p)$ for some $p \in \Sigma^\omega$ or $\iota(w)$ for some $w \in \Sigma^*$. Since every output tape starts completely blank, it is not necessary to write infinitely many symbols to an output tape for which the result is an element of Σ^*; instead the computation of that element is finished once the '↵' symbol has been written. If all outputs are elements of Σ^*, the computation may halt once all output tapes have had a '↵' written on them.

Note that if we wish to restrict to purely using the binary alphabet, we can encode words $w \in \{\mathtt{0},\mathtt{1}\}^*$ as sequences in $\{\mathtt{0},\mathtt{1}\}^\omega$ by describing the length of w in some way. A

simple method is to encode the word $w = w_0, \ldots, w_{n-1}$ as $\hat{w} = \mathtt{1}w_0\mathtt{1}w_1\mathtt{1}\cdots\mathtt{1}w_{n-1}\mathtt{0}\cdots$. In other words, each element of w is preceded by a '1', and the word is followed by an infinite sequence of '0'. Alternatively, we can encode w as $\mathtt{1}^{|w|}\mathtt{0}w\mathtt{0}\cdots$.

When performing type-two computations in practise, we cannot in general wait for the infinite amount of time needed to obtain the complete answer. Instead, we may wish to terminate the computation after a fixed number of output digits. This means that a type-two machine-computable function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ would be replaced by a function $\bar{\eta} : \Sigma^* \times \Sigma^\omega \rightharpoonup \Sigma^*$, where $\bar{\eta}(v, p) = \eta(p)|_{|v|}$. Additionally, the infinite input p is itself likely to be generated from a finite input $w \in \Sigma^*$ by some function $p = \xi(w)$, e.g. p is a decimal expansion of a rational defined by the word w. We then obtain a purely finite computation of a function $\zeta : \Sigma^* \times \Sigma^* \to \Sigma^*$ defined by $\zeta(v, w) = \bar{\eta}(v, \xi(w))$.

Given that we can reduce type-two computation to type-one computation, one might wonder what is the point of studying type-two computation in its own right. The main reason is that elements of Σ^ω can be used to describe objects from mathematical analysis, such as real numbers or continuous functions, completely, whereas restricting to type-one computation either forces one to always work with approximations or to only consider computable elements. In the former case, we are often required to work with messy $\epsilon - \delta$ style considerations, while the latter is unnatural from the point of view of analysis. Even if we are ultimately only interested in finite computation, the use of sequences to represent intermediate results yields a much more elegant and simple theory than one based purely on finite computation.

2.4 Computability induced by representations

We have seen how computability theory can be developed for partial functions $(\Sigma^\omega)^m \rightharpoonup (\Sigma^\omega)^n$. However, we are really more interested in computations on more general mathematical objects, such as $\mathbb{N}$ or $\mathbb{R}$, and so we need a way to relate computation over Σ^ω to that over more general spaces.

The basic idea is that each element x of a set of interest X, should be described by one or more sequences $p \in \Sigma^\omega$. We note that the description of an element need not be unique; for example, in the decimal representation, $\mathtt{0.999}\cdots$ $\mathtt{1.000}\cdots$ denote the same real number. Further, not every sequence need correspond to an element of X; as an example, the strings $\mathtt{1000}\cdots$ and $\mathtt{1.0.00}\cdots$ are both invalid as decimal representations of a real number.

Definition 2.18 (Representations)**.** Let Σ be a fixed alphabet and X be a set. Then a *representation* of X is a partial surjective function $\delta : \Sigma^\omega \rightharpoonup X$. For $x \in X$, an element $p \in \Sigma^\omega$ such that $\delta(p) = x$ is called a δ-name of x.

Remark 2.19. In the definition of representation, there is no restriction placed on the domain. This is because there are spaces for which the domain is necessarily complicated. Ideally, we would like the domain to be an open set, so we can tell after a finite amount of data that a sequence is a valid input, or a closed set, so we can tell that a sequence is invalid. Frequently, there will be invalid sequences, but most words can be continued to both valid and invalid inputs. Where possible, it would be advantageous to have a decision procedure for whether a given word can be extended to a valid name.

Definition 2.20 (Equivalence)**.** A representation δ_1 of X is said to *reduce* to δ_2, denoted $\delta_1 \leq \delta_2$ if there is a machine 'computable function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ such that $\mathrm{dom}(\delta_2 \circ \eta) \supset$

$\mathrm{dom}(\delta_1)$ and $\delta_1 = \delta_2 \circ \eta$. In other words, given any δ_1-name p of x, $\eta(p)$ is a δ_2-name of x. Representations δ_1 and δ_2 are *equivalent* if $\delta_1 \leq \delta_2$ and $\delta_2 \leq \delta_1$.

Remark 2.21. A representation reduces to another if a name contains *more information* about the type. The use of $\delta_1 \leq \delta_2$ to denote "δ_1 reduces to δ_2" is standard.

Definition 2.22 (Computable type)**.** A *computable type* is a pair $(X, [\delta])$ where X is a set and $[\delta]$ is an equivalence class of representations of X.

We henceforth denote computable types by script $\mathcal{X}$ or calligraphic $\mathcal{X}$ letters, and sets by italic X or blackboard-bold $\mathbb{X}$ letters. Note that we always consider different sets as having different types, even if the sets are bijective and the representations respect this bijection.

Given a function $f : X \to Y$, a partial function $\eta : \Sigma^\omega \to \Sigma^\omega$ is a valid description of f if it translates any sequence p denoting x into a sequence q denoting $y = f(x)$. The function f is then *computable* if η is computable.

Definition 2.23 (Computability induced by representations)**.** A function $f : X_1 \times \cdots \times X_k \to X_0$ is $(\delta_1, \ldots, \delta_k; \delta_0)$-*computable* if there is a machine computable function $\eta : \Sigma^\omega \times \cdots \times \Sigma^\omega \rightharpoonup \Sigma^\omega$ such that

$$f(\delta_1(y_1), \ldots, \delta_k(y_k)) = \delta_0(\eta(y_1, \ldots, y_k))$$

whenever the left-hand side is defined. We say that η is a *realiser* for f.

Note that it does not make sense to say whether a *representation* itself is computable. This is because representations are used to *induce* a computability structure on another set. However, most sets encountered in practise have a canonical equivalence class of "natural" representations, and we can, of course, consider the computability of a representation with respect to a natural representation.

Remark 2.24 (Multi-representations). A multi-representation is a multivalued function $\delta : \Sigma^\omega \rightrightarrows X$. The idea is that p does not provide enough information to identify an element of X exactly, but only a set of possible values. For example, δ might describe the set of measurable functions on $\mathbb{R}$, but $\delta(p)$ is a set of functions which are equal almost everywhere. In many cases, multi-representations define a natural quotient map on X, and give a single-valued representation of elements of the quotient set. In these cases, the underlying set of the corresponding type can be taken to be the quotient set, and the representation to be the single-valued version the multi-representation.

Since elements of finite and denumerable sets can be described by a finite amount of information, and these are often important in their own right, we define a similar notion for naming systems in terms of Σ^*.

Definition 2.25 (Notations)**.** Let Σ be a fixed alphabet and X be a countable set. Then a *notation* of X is a partial surjective function $\nu : \Sigma^* \rightharpoonup X$.

Representations and notations are both *naming systems* since they relate general sets to words and sequences over some alphabet.

Remark 2.26 (Realisations). More generally, suppose we have a set R for which we already have some kind of "computational structure" defined. This could be Σ^ω, with

computations described by type-two Turing machines, or $\mathbb{N}$ with computations described by partial recursive functions. Then we can induce a computational structure on another set X using the computational structure on $\mathbb{R}$. The traditional way of doing this is by a *realisation relation* $\Vdash$ on $R \times X$. A realisation is *sound* if for all $x_1, x_2 \in R$, if $r \Vdash x_1$ and $r \Vdash x_2$, then $x_1 = x_2$, and is *complete* if $\forall x \in X,\ \exists r \in R,\ r \Vdash x$.

A sound realisation $\Vdash$ induces a partial function $\rho : R \rightharpoonup X$ by $\rho(r) = x \iff r \Vdash x$, and this function ρ is surjective if $\Vdash$ is complete. An element $r \in R$ such that $\rho(r) = x$ is called a *ρ-name* of x. Hence a representation of X can be seen as sound and complete realisation of X in Σ^ω. It is trivial that if $\delta : \Sigma^\omega \rightharpoonup R$ is a representation of R, and $\rho : R \rightharpoonup X$ is a sound and complete realisation of X, then $\rho \circ \delta$ is a realisation of X.

Chapter 3

Computable Analysis

3.1 Representations of topological spaces

We now consider representations on topological spaces. We first show that a representation of a set X induces a natural topology on X.

Definition 3.1 (Topology induced by a representation)**.** Let X be a set and δ be a representation of X. Then the topology τ induced by δ is the *final topology* of δ, namely $U \in \tau \iff \delta^{-1}(U)$ is open in $\mathrm{dom}(\delta)$.

By definition, the representation δ becomes a continuous quotient map. However, not all continuous quotient maps are "good" representations, as the following example shows.

Example 3.2. Consider the binary representation of $\mathbb{R}$. More precisely,

$$\delta(\pm a_n a_{n-1} \cdots a_0 . a_{-1} a_{-2} \cdots) = x \iff x = \pm \textstyle\sum_{k=-\infty}^{n} a_k 2^k.$$

It can be shown (see Section 3.3) that δ is a partial surjective quotient map $\Sigma^\omega \to \mathbb{R}$. Let $x_n = 1 + (-1/3)^n$ for $n \in \mathbb{N}$, and $x_\infty = 1$, so the function $f : \mathbb{N} \cup \{\infty\} \to \mathbb{R}$, $f(n) = x_n$ is continuous. Then for $n = \infty$, we have names $p_\infty = +\mathtt{0.111}\cdots$ and $p'_\infty = +\mathtt{1.000}\cdots$, whereas for x_{2n} we have a unique name $p_{2n+1} = \mathtt{1.0}\cdots$ and for x_{2n+1} we have a unique name $p_n = \mathtt{0.1}\cdots$. This means that any function $h : \mathbb{N} \cup \{\infty\} \to \Sigma^\omega$ satisfying $\delta(h(n)) = f(n)$ (there are only two) has a discontinuity at ∞. In other words, the function $h : \mathbb{N} \cup \{\infty\} \to \mathbb{R}$ does not lift through the representation δ.

In order to prevent pathological situations as described above, we impose an *admissibility* condition on representation in addition to the quotient condition.

Definition 3.3 (Admissible quotient representation)**.** A continuous representation δ of a topological space X is a *quotient* if whenever Y is a topological space and $f : X \to Y$ is such that $f \circ \delta$ is continuous, then f is continuous.

A representation δ of a topological space X is *admissible* if whenever $\phi : \Sigma^\omega \rightharpoonup X$ is a continuous partial function, there exists a continuous partial function $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ such that $\phi = \delta \circ \eta$.

An alternative definition of a quotient representation is that U is open in X whenever $\delta^{-1}(U)$ is open in Σ^ω. An alternative definition of an admissible representation is that whenever $x_i \to x_\infty$ is a convergent sequence in X, there exists a convergent sequence $\xi_n \to \xi_\infty$ in Σ^ω such that $\delta(\xi_n) = x_n$ for $n \in \mathbb{N} \cup \infty$.

The quotient representation captures the notion that δ must contain the information given by the topology of the space. Admissibility means that δ does not contain "too much" information, or that the information respects *sequential* continuity.

The following result is derived from [47]. It gives the basic properties implied by the admissibility and quotient conditions on continuous representations.

Proposition 3.4. *Let (X, τ_X) and (Y, τ_Y) be topological spaces, and δ_X, δ_Y be continuous representations of X and Y.*

1. *Suppose η is a continuous realiser for a function $f : X \to Y$ and δ_X is a quotient representation. Then f is continuous.*
2. *Suppose $f : X \to Y$ is continuous, and δ_Y is admissible. Then f has a continuous realiser.*

Proof.

1. Take $\phi = \delta_Y \circ \eta$, which is continuous. We have $f \circ \delta_X = \delta_Y \circ \eta$, so $f \circ \delta_X$ is continuous. Since δ is a quotient map, f is continuous.
2. Take $\phi = f \circ \delta_X$. Then since δ_Y is admissible, there exists a continuous η such that $\delta_Y \circ \eta = f \circ \delta_X$. □

The following result is a direct consequence of Proposition 3.4.

Corollary 3.5 (Discontinuity implies uncomputability)**.** *Let (X_i, τ_i) be topological spaces, and δ_i be admissible quotient representations for $i = 0, \ldots, k$. Then if $f : X_1 \times \cdots \times X_k \to X_0$ is $(\delta_1, \ldots, \delta_k; \delta_0)$-computable, then f is $(\tau_1, \ldots, \tau_k; \tau_0)$-continuous.*

In other words, only continuous functions can be computable, yielding a very strong link between topology and computability. The main use of this result is to show certain operations are *uncomputable*, since it is sufficient to show that the operation is discontinuous. The converse is not true, as there are continuous functions that are not computable, but in practise, most "naturally-defined" continuous functions are computable.

A common critique of computable analysis is that "only" continuous functions can be handled, and there are plenty of important functions that are discontinuous, the most important being the Heaviside function

$$H(t) = \begin{cases} 0 \text{ if } t < 0; \\ 1 \text{ if } t \geq 0. \end{cases}$$

However, the negative computability results should be interpreted as the fact that it is impossible to evaluate the function arbitrarily accurately at every point if only approximations to the argument are known (the computation fails at $t = 0$). In most practical applications, the Heaviside function occurs on the right-hand side of a differential equation, and only its integrals are involved. Interpreting the function in the space L^1 of integrable functions gives a valid name which is sufficient to compute with. Alternatively, by refining the topology on the argument space (corresponding to more information) or coarsening the topology on the result space (corresponding to less information) again allows for a valid name. In this way, the computability theory yields important information on the properties and valid uses of the function.

We now turn to the question of which topological spaces have an admissible quotient representation. An obvious condition is that the space should have at most continuum cardinality, the cardinality of Σ^ω. A second condition is that the space should at least satisfy the T_0 separation axiom, that is, for any two points x, y, there exists an open set U such that either $x \in U$ and $y \notin U$, or $y \in U$ and $x \notin U$. Such spaces are also called *Kolmogorov spaces*. Henceforth, all topological spaces considered will be Kolmogorov unless explicitly declared otherwise.

One natural class of spaces which have at most continuum cardinality are the *countably-based Kolmogorov spaces*, since every point is described by the basic open sets containing it. In Section 3.6 we give an explicit construction of an admissible quotient representation for a countably-based Kolmogorov space. However, it turns out that any topological quotient of a countably-based space also has an admissible quotient representation, and that the class of *quotients of countably-based* (qcb) Kolmogorov spaces are exactly the spaces with an admissible quotient representation. We prove this result in Section 4.2.

In general a quotient of a countably-based space need not be countably-based, but we will show that it has a countable *pseudobase* in the following sense:

Definition 3.6 (Pseudobase)**.** A collection ρ of subsets of a topological space (X, τ) is a *pseudobase* if for any $x \in X$ and $U \in \tau$, there exists $P \in \rho$ such that $x \in P$ and $P \subset U$.

It is clear that if (X, τ) is a Kolmogorov space with a countable pseudobase then X has at most continuum cardinality. Further, any pseudobase is a base if, and only if, it consists only of open sets.

The admissibility condition is strongly tied to the concept of *sequential space*, for which the topology is determined by the convergent sequences:

Definition 3.7 (Sequential space)**.** A subset S of a topological space (X, τ) is *sequentially open* if whenever $\vec{x}$ is a convergent sequence with limit $x_\infty \in S$, there exists $N \in \mathbb{N}$ such that $x_n \in S$ for all $n \geq N$. A topological space (X, τ) is *sequential* if any sequentially open set is open.

The space Σ^ω is an example of a sequential space: The property of being sequential is preserved by subspaces and quotient spaces, so any space with an quotient representation is itself a sequential space. For a sequential space to have an admissible quotient representation, we also need to bound the cardinality in some way. In Section 4.2, we show that any sequential space with a countable *sequential pseudobase* has an admissible quotient representation.

3.2 Computable type theory

We now develop a computable type theory based on the admissible quotient representations. The most important goals are to give a collection of initial concrete types, and then to build up new types from existing types. The resulting types form a *Cartesian closed category*, with objects which are computable types, and morphisms which are continuous functions (some of which are computable).

Note that there are two natural categories whose objects are computable types. In the first, all *continuous* functions are allowed as morphisms, while in the second, only

computable functions are allowed. In general, the morphisms involved in universal constructions in a category will all be computable, but the conditions apply to all continuous functions.

Definition 3.8 (Category of computable types)**.** The category of *computable types* (with continuous morphisms) is the category whose objects are computable types $\mathcal{X} = (X, [\delta])$, and morphisms $\mathcal{X} \to \mathcal{Y}$ are functions $f : X \to Y$ which are continuous with respect to the induced topologies on X and Y. A morphism is *computable* if it corresponds to a computable function. The category of *computable types computable morphisms* is the subcategory with the same objects, but whose morphisms are the computable functions.

The type of any singleton space is a *terminal object* $\mathbb{I}$ in the category. Note that there is exactly one morphism from any type to the singleton type. Further, there is a natural bijection between elements x of a space X and continuous functions from the singleton space to X. We can therefore associate any element of the object $\mathcal{X}$ with a morphism $\mathbb{I} \to \mathcal{X}$. This identification is useful in that it allows us to work purely within the category itself, and still recover element of the underlying set of $\mathcal{X}$. Further, if $f : \mathcal{X} \to \mathcal{Y}$ is a morphism in the category, and $x : \mathbb{I} \to \mathcal{X}$ represents an element of X, then $f \circ x : \mathbb{I} \to \mathcal{Y}$ represents the element $f(x)$ of Y.

The next theorem gives some closure properties of the category of computable types. These properties imply that the category of computable types is *Cartesian closed*. This is an important notion in intuitionistic/constructive type theory, since the lambda-calculus can be developed in any Cartesian closed category.

Theorem 3.9.

1. *There is a computable type $\mathbb{I}$, which is unique up to continuous/computable isomorphism, such that for any continuous/computable type $\mathcal{X}$, there is a unique computable function $\mathcal{X} \to \mathbb{I}$.*

2. *If $\mathcal{X}_1$ and $\mathcal{X}_2$ are computable types, then there is a unique computable type $\mathcal{X}_1 \times \mathcal{X}_2$ together with computable projections $p_i : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{X}_i$ such that for all continuous/computable functions $f_i : \mathcal{W} \to \mathcal{X}_i$, there exists a continuous/computable function $f : \mathcal{W} \to \mathcal{X}_1 \times \mathcal{X}_2$ such that $f_i = p_i \circ f$ for $i = 1, 2$.*

3. *If $\mathcal{X}$ and $\mathcal{Y}$ are computable types, then there is a unique computable type $\mathcal{Y}^{\mathcal{X}}$ together with a computable* evaluation *function $e : \mathcal{Y}^{\mathcal{X}} \times \mathcal{X} \to \mathcal{Y}$ such that for any continuous/computable $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$, there exists continuous/computable $\hat{f} : \mathcal{W} \to \mathcal{Y}^{\mathcal{X}}$ such that $f(w, x) = e(\hat{f}(w), x)$ for all $w \in \mathcal{W}$ and $x \in \mathcal{X}$.*

Proof.

1. Let $\mathbb{I}$ be a one-point set, with representation δ with $\mathrm{dom}(\delta) = \Sigma^\omega$. (Actually, any representation whose domain contains a computable element will do.)

2. Let δ_i be a representation of X_i, $i = 1, 2$. We say that q is a name of $(x_1, x_2) \in X_1 \times X_2$ if $q \in \mathrm{dom}(\pi_1, 2)$ and $\pi_i(p) \in \mathrm{dom}(\delta_i)$ with $\delta_i(\pi_i(q)) = x_i$ for $i = 1, 2$. We have $p_i(x_1, x_2) = p_i(\delta(q)) = \delta_i(\pi_i(q))$ for $i = 1, 2$ so the projections are computable.

   Suppose $\mathcal{P}$ and $\mathcal{P}'$ are two types satisfying the properties. Then taking $f_i = p'_i$, there exists computable $f : \mathcal{P}' \to \mathcal{P}$ such that $x_i = p'_i(x_1, x_2) = p_i(f(x_1, x_2))$ for all x_1, x_2, so id is computable. The same holds reversing the roles of $\mathcal{P}$ and $\mathcal{P}'$, so any two product types are equivalent.

3. Let δ_X, δ_Y be admissible quotient representations of X and Y. We aim to define a representation $\delta_{X\to Y}$ on $C(X;Y)$. Let $f : X \to Y$ be continuous. Then there exists continuous $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$ with G_δ-domain such that $f(\delta_X(q)) = \delta_Y(\eta(q))$ for all $q \in \mathrm{dom}(\delta_X)$. By Theorem 2.16, there exists $a \in \Sigma^\omega$ such that $\eta(\cdot) = \varepsilon(a, \cdot)$. We therefore take $\delta_{X\to Y}(a) = f$ if, and only if, $f(\delta_X(q)) = \delta_Y(\varepsilon(a, q))$ for all $q \in \mathrm{dom}(\delta_X)$.

   The evaluation function e is computable, since if $f = \delta_{X\to Y}(a)$ and $x = \delta_X(q)$, we have $e(f, x) = f(x) = f(\delta_X(q)) = \delta_Y(\varepsilon(a, q))$. Further, if $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$ is computable, then there exists computable $\xi : \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ such that $f(\delta_W(p), \delta_X(q)) = \delta_Y(\xi(p, q))$. Take $\eta : \Sigma^\omega \to \Sigma^\omega$ such that $\eta(\tau(p, q)) = \xi(p, q)$, noting that η can be taken to be computable. Then there exists computable a such that $\varepsilon(a, p) = \eta(p)$. Then for any $x \in X$, $\delta_{X\to Y}(\sigma(a, p))(x) = \delta_{X\to Y}(\sigma(a, p))(\delta_X(q)) = \delta_Y(\varepsilon(\sigma(a, p), q)) = \delta_Y(\varepsilon(a, \tau(p, q))) = \delta_Y(\eta(\tau(p, q))) = \delta_Y(\xi(p, q)) = f(\delta_W(p), \delta_X(q)) = f(w, x) = \hat{f}(w)(x)$, so $\delta_{X\to Y}(\sigma(a, p)) = \hat{f}(w)$. Hence $\hat{f}$ is computable.

   We henceforth identify $w \in \mathcal{Y}^\mathcal{X}$ with the (necessarily continuous) function $f(x) = e(w, x)$.

   To show that the representation is unique up to equivalence, suppose $\mathcal{C}$ and $\mathcal{C}'$ are types of the continuous functions satisfying the required properties, with respective evaluation functions e and e'. Since evaluation $e : \mathcal{C} \times \mathcal{X} \to \mathcal{Y}$ is computable, the function $i : \mathcal{C} \to \mathcal{C}'$ satisfying $e'(i(w), x) = e(f, x)$ is computable. Then $e'(i(f), x) = f(x)$, so $i(f)$ is equal to f i.e. the identity $\mathcal{C}$ and $\mathcal{C}'$ is computable. Similarly, the identity i' from $\mathcal{C}'$ to $\mathcal{C}$ is computable.

□

Remark 3.10. The topology on $X_1 \times X_2$ induced by the representation $\delta_{X_1\times X_2}$ is *not* necessarily the product topology. For the topology of $\mathcal{X}_1 \times \mathcal{X}_2$ must be sequential, but the product topology need not be. In fact, the topology on $\mathcal{X}_1 \times \mathcal{X}_2$ is the *sequentialisation* of the product topology.

The following equivalences hold in any Cartesian closed category.

Proposition 3.11.

1. *The product of the terminal type $\mathcal{I}$ and any type $\mathcal{X}$ is isomorphic to $\mathcal{X}$.*
2. *The morphisms $\mathcal{I} \to \mathcal{Y}^\mathcal{X}$ are in bijective correspondence with the morphisms $\mathcal{X} \to \mathcal{Y}$.*

Proof.

1. Take $\mathcal{W} = \mathcal{I}$. Since an element of any type is associated with a morphism $\mathcal{I} \to \mathcal{X}$, the elements of $\mathcal{I} \times \mathcal{X}$ are associated with pairs (i, x) where i is the unique element of $\mathcal{I}$ and $x \in \mathcal{X}$.
2. An element of $\hat{f}$ of $\mathcal{Y}^\mathcal{X}$ is defined by a morphism $\mathcal{I} \to \mathcal{Y}^\mathcal{X}$, and defines a morphism $f : \mathcal{X} \to \mathcal{Y}$ by $f(x) = \varepsilon(\hat{f}, x)$, and conversely.

□

We can additionally define countable products in the category of computable types.

Definition 3.12. Let $(\mathcal{X}_n)_{n\in\mathbb{N}}$ be computable types. Then $\prod_{n=0}^{\infty}\mathcal{X}_n$ is a computable type with representation δ given by taking

$$\delta(\tau(p_0,p_1,\ldots)) = (x_0,x_1,\ldots) \iff \delta_i(p_i) = x_i \tag{3.1}$$

where τ is the infinite tupling function given by (2.3).

There are similar initial constructions in the category of computable types.

Theorem 3.13.

1. *There is a computable type $\mathcal{E}$ which is unique up to computable isomorphism such that for any computable type $\mathcal{X}$, there is a unique computable function $\mathcal{E}\to\mathcal{X}$.*
2. *If $\mathcal{X}_1$ and $\mathcal{X}_2$ are computable types, then there is a unique computable type $\mathcal{X}_1+\mathcal{X}_2$ together with computable embeddings $j_i : \mathcal{X}_i \to \mathcal{X}_1+\mathcal{X}_2$ such that for all continuous/computable functions $f_i : \mathcal{X}_i \to \mathcal{W}$, there exists a continuous/computable function $f : \mathcal{X}_1+\mathcal{X}_2 \to \mathcal{W}$ such that $f_i = f\circ j_i$ for $i = 1, 2$.*

The proof is similar to that of Theorem 3.9, and it omitted. The initial type $\mathcal{E}$ is the type of the empty set, and the sum type is the type of the disjoint union.

3.3 Fundamental types

We now describe three fundamental logical and numerical types, and some derived product types, which form the cornerstone of computable analysis. For each type, we give a simple representation with alphabet $\{\mathtt{0},\mathtt{1}\}$, and other natural representations as appropriate. Where possible, we give representations where the domain has a particularly nice form.

3.3.1 The natural numbers

We begin with the natural numbers $\mathbb{N}$, and define the corresponding type $\mathcal{N}$. A simple representation using the alphabet $\{\mathtt{0},\mathtt{1},\hookleftarrow\}$ is given by the binary expansion using the '$\hookleftarrow$' symbol to act as a terminator:

$$\delta(p_kp_{k-1}\cdots p_1p_0\hookleftarrow\cdots) = \sum_{i=0}^{k} 2^i p_i.$$

The decimal representation uses the alphabet $\{\mathtt{0},\mathtt{1},\mathtt{2},\ldots,\mathtt{9},\hookleftarrow\}$, and is defined by

$$\delta(p_kp_{k-1}\cdots p_1p_0\hookleftarrow\cdots) = \sum_{i=0}^{k} 10^i p_i.$$

A simple representation of $\mathbb{N}$ with the alphabet $\Sigma = \{\mathtt{0},\mathtt{1}\}$ is given by a function δ with domain $\{\mathtt{0},\mathtt{1}\}^\omega \setminus \{\mathtt{1}^\omega\}$ by

$$\delta(\mathtt{1}^n\mathtt{0}\cdots) = n.$$

Alternatively, we can restrict $\mathrm{dom}(\delta)$ to $\{\mathtt{1}^n\mathtt{0}^\omega \mid n\in\mathbb{N}\}$. The above representation is very inefficient, since the number n requires $n+1$ bits to determine. In order to encode

the binary representation using only symbols $\{\mathtt{0},\mathtt{1}\}$, we need to know how to specify the end of the input. A simple way of doing this is to prefix the binary name with a unary name giving the number of binary digits; or better, the number of "machine words":

$$\delta(\mathtt{1}^m\mathtt{0}p_{km-1}p_{km-2}\cdots p_1p_0\cdots) = \textstyle\sum_{i=0}^{km-1} 2^i p_i.$$

or the logarithm of the number of digits:

$$\delta(\mathtt{1}^m\mathtt{0}p_{2^m-1}p_{2^m-2}\cdots p_1p_0\cdots) = \textstyle\sum_{i=0}^{2^m-1} 2^i p_i.$$

For example, the number 425 has binary expansion 110101001 with 9 digits, so a name of 425 would be $\mathtt{11110\,0000000110101001}\cdots$ with $m = 4$.

Under any of the above representations, it is easy to show that addition and multiplication are computable, and that comparisons are decidable predicates.

Note that there is no representation of the natural numbers with domain Σ^ω, since Σ^ω is compact but $\mathbb{N}$ is not.

The integers can be constructed by introducing a symbol $-$, or as the quotient of $\mathcal{N}\times\mathcal{N}$ under the function $z(m,n) = m - n$. The rationals can be constructed by introducing a symbol /, or as the quotient of $\mathcal{Z}\times(\mathcal{N}\setminus\{0\})$ under the function $q(m,n) = m/n$.

The space $\mathbb{N}^\infty = \mathbb{N}\cup\{\infty\}$ is particularly important, since any convergent sequence in a topological space X can be viewed as a continuous function $s : \mathbb{N}^\infty \to X$. A representation of $\mathbb{N}^\infty$ as a total function is $\delta(\mathtt{1}^n\mathtt{0}\cdots) = n;\ \delta(\mathtt{1}^\omega) = \infty$.

3.3.2 Logical types

We shall later see that almost all problems in computable analysis are undecidable. In particular, equality on Σ^ω is undecidable, since we can never tell in finite time whether two sequences are equal. For this reason, it is most natural to use a three-valued logical type $\mathcal{T}$ with values $\{\top,\bot,\uparrow\}$ representing *provable*, *disprovable* and *undecidable*. Alternative names for $\top$ and $\bot$ are, respectively, *verifiable* and *falsifiable*. An alternative name for $\uparrow$ is *indeterminate*.

The type $\mathcal{T}$ can be given a representation δ which is a total function on $\{0,1\}^\omega$. We define

$$\delta(\mathtt{0}^*\mathtt{10}\cdots) = \bot;\quad \delta(\mathtt{0}^*\mathtt{11}\cdots) = \top;\quad \delta(\mathtt{0}^\omega) = \uparrow .$$

We interpret the representation as follows. A leading 0 indicates that at the given stage of computation, there is insufficient information to determine whether the value should be true or false. The first 1 indicates that a decision has been made, and the next digit is 0 for false and 1 for true. If the result is undecidable, then the output is an infinite sequence of zeroes.

The induced topology has basic open sets $\{\top\}$ and $\{\bot\}$, so the open sets are $\big\{\{\,\},\{\top\},\{\bot\},\{\top,\bot\},\{\top,\bot,\uparrow\}\big\}$. In particular, the set $\{\uparrow\}$ is closed but not open, so although the topology is finite, it is not the discrete topology. Indeed, it is not even Hausdorff, since the only open set containing $\uparrow$ is $\{\top,\bot,\uparrow\}$.

3.3.3 The real numbers

Perhaps the most important computable type, and our first concrete example of an uncountable type, is the type of the real numbers. We first show that the binary representation is *not* an admissible representation of $\mathbb{R}$.

Example 3.14 (Binary representation). Given the real number 1, there are two possible binary expansions, $\texttt{1.00}\cdots$ and $\texttt{0.11}\cdots$. Let $x_n = 1 + (-1/2)^n$. Then for even n, $x_n > 1$ and any name of x_n begins $\texttt{1.}\cdots$, whereas for odd n, $x_n < 1$ and any name of x_n begins $\texttt{0.1}\cdots$. Hence even though $x_n \to 1$ as $n \to \infty$, we cannot choose binary expansions for x_n which converge.

Similar considerations hold for the decimal representation.

A class of representations which *are* admissible are know as *extended digit representations*. The simplest is the *binary signed-digit representation* with alphabet $\Sigma = \{\texttt{0}, \texttt{1}, \overline{\texttt{1}}, \texttt{.}\}$ The representation is defined by

$$\delta(a_n a_{n-1} \cdots a_0 \,.\, a_{-1} a_{-2} \cdots) = \sum_{i=-\infty}^{n} a_i \, 2^i, \tag{3.2}$$

where the symbol '$\overline{\texttt{1}}$' is read as -1.

It is easy to show that arithmetic is computable in with respect to the binary signed-digit representation, and that strict inequality $<$ is semidecidable. We also need to account for the topological/metric structure of the real line. The most straightforward way of doing this is by looking at limits of convergent sequences. Since a finite part of a general convergent sequence gives no information about the limit, we need to restrict to *effectively convergent sequences* for which the rate of convergence is known.

Definition 3.15 (Effective limit)**.** A limit in a metric space is *effective* if there exists a computable sequence of rationals ϵ_n such that $\epsilon_n \searrow 0$ and $d(x_m, x_n) < \epsilon_{\min(m,n)}$ for all m, n.

Without loss of generality, we can always take $\epsilon_n = 2^{-n}$ or $\epsilon_n = 1/n$.

The following theorem is due to Bauer [6]. It asserts the existence of a canonical real number type.

Theorem 3.16 (Real number type)**.** *There is a unique computable type $\mathcal{R}$ with underlying set $\mathbb{R}$ such that the constant 1 is a computable number, arithmetical operations $+, -, \times, \div$ are computable, strict comparison $<$ is semidecidable, and effective limits* lim *are computable.*

Proof. Suppose δ_1 and δ_2 are representations of $\mathbb{R}$ satisfying the required properties. Given a δ_1-name p_1 of $x \in \mathbb{R}$, we need to compute a δ_2-name of p.

Since the constant one and addition are computable, given a positive integer n, we can find a δ_1- or δ_2-name of n by computing $1 + 1 + \cdots + 1$. Since subtraction is computable, we can find a name of any integer m as $m = n_1 - n_2$. Since division is computable, we can find a name of any rational number q as $q = m/n$.

Given a δ_1-name p of $x \in \mathbb{R}$, since we can construct any rational number q and verify $q < x$ and $x < q$, we enumerate all rational numbers $l < x$ and all rational numbers $u > x$. We can therefore construct a sequence of rational numbers q_i such that $|q_i - q_j| < 2^{-\min(i,j)}$ and $\lim_{i\to\infty} q_i = x$. Since the q_i are rational, we can find a δ_2-name of each q_i. We can find a δ_2-name of x since effective limits are δ_2-computable. □

3.4 Set and function types

We now develop the theory of subsets of a topological type. We begin by introducing some convenient notation.

Notation 3.17. Write $A \bowtie U$ to denote the classical predicate $U \cap A \neq \emptyset$. If U is open, we say that A *overlaps* U. Write $U_n \nearrow U_\infty$ if $U_{n+1} \supset U_n$ for all n and $\bigcup_{n=0}^{\infty} U_n = U_\infty$.

From classical topology, we have the property that a set U is open if, and only if, its characteristic function is a continuous map from X to $\mathbb{S}$:

Proposition 3.18 (Open sets)**.** *Let X be a topological space. Then there is a canonical bijection between $\mathcal{O}(X)$ and continuous functions $X \to \mathbb{S}$ given by $\chi(x) = \top \iff x \in U$ for $U \in \mathcal{O}(X)$ and $\chi : X \to \mathbb{S}$.*

We use this as a basis for the definitions of types of subsets of a computable type.

Definition 3.19 (Open and closed set types)**.**

1. The type of open subsets of $\mathcal{X}$, denoted $\mathcal{O}(\mathcal{X})$, is defined to be the exponential object $\mathbb{S}^{\mathcal{X}}$. The interpretation of continuous $\chi : X \to \mathbb{S}$ as a point-set $U \in \mathcal{O}(X)$ is given by $U = \chi^{-1}(\{\top\})$.

2. The type of closed subsets of $\mathcal{X}$, denoted $\mathcal{A}(\mathcal{X})$, is identified with $\mathbb{S}^{\mathcal{X}}$. The interpretation of continuous $\overline{\chi} : X \to \mathbb{S}$ as a point-set $A \in \mathcal{A}(X)$ is given by $A = \overline{\chi}^{-1}(\{\uparrow\})$.

We now turn to types of *compact* sets, and the dual type of *separable* or *overt* sets, which are defined in terms of subset and overlap relations. Note that the overlap and subset relations can be defined in terms of existential and universal quantifiers as

$$S \bowtie U \iff \exists x \in S,\ x \in U$$
$$S \subset U \iff \forall x \in S,\ x \in U.$$

Further, for any set S, the overlap relation and subset relations satisfy:

$$S \bowtie (U_1 \cup U_2) \iff (S \bowtie U_1) \vee (S \bowtie U_2); \tag{3.3}$$
$$S \subset (U_1 \cap U_2) \iff (S \subset U_1) \wedge (S \subset U_2). \tag{3.4}$$

Given a fixed set S, these relations define functions $\mathcal{O}(X) \to \mathbb{S}$, or equivalently, or a set of open sets $\mathcal{O}(X)$. It can be shown that the functions defined by $\bowtie$ are always continuous, but the functions defined by $\subset$ are continuous if, and only if, S is compact The observations above motivate the definition of set types as functions defined by the overlap and subset relations.

Definition 3.20 (Separable/overt and compact set types)**.**

1. The type of *separable* or *overt* sets[1] subsets of $\mathcal{X}$, denoted $\mathcal{V}(\mathcal{X})$, is defined to be the subtype of $\mathbb{S}^{\mathcal{O}(\mathcal{X})}$ defined by functions $b : \mathcal{O}(\mathcal{X}) \to \mathbb{S}$ satisfying $b(U \cup V) = b(U) \vee b(V)$. Given any set B, we can define such a function b by $b(U) = \top \iff B \bowtie U$. The interpretation of the function b as a point-set B is given by $B = \{x \in X \mid \forall\, U \in \mathcal{O}(X),\ (x \in U \implies b(U) = \top)\}$.

2. The type of *compact* subsets of $\mathcal{X}$, denoted $\mathcal{K}(\mathcal{X})$, is defined to be the subtype of $\mathbb{S}^{\mathcal{O}(\mathcal{X})}$ defined by functions $c : \mathcal{O}(\mathcal{X}) \to \mathbb{S}$ satisfying $c(U \cap V) = c(U) \wedge c(V)$. Given any compact set C, we can define such a function c by $c(U) = \top \iff C \bowtie U$. The interpretation of the function c as a point-set C is given by $C = \{x \in X \mid \forall\, U \in \mathcal{O}(X),\ (c(U) = \top \implies x \in U)\}$.

[1]The terminology *overt* is becoming standard in the literature. We use the terminology *separable* since this is the classical property most closely related to this type.

Since $S \bowtie U \iff \mathrm{cl}(S) \bowtie U$, the function $b : \mathcal{O}(\mathcal{X}) \to \mathbb{S}$ defines a set only up to its closure; the point-set construction of B always yields a closed set. If X is a Hausdorff space, then the function $c : \mathcal{O}(\mathcal{X}) \to \mathbb{S}$ defines a compact set uniquely. However, if X is not Hausdorff, then the point-set is defined only up to its *saturation*:

Definition 3.21 (Saturation)**.** Let (X, τ) be a topological space and $S \subset X$. Then the *saturation* of S in (X, τ), denoted $\mathrm{sat}(S)$, is

$$\mathrm{sat}(S) = \bigcap \{U \in \tau \mid S \subset U\}. \tag{3.5}$$

We say S is *saturated* if $S = \mathrm{sat}(S)$.

For example, if $X = \mathbb{R}$ with topology $\tau = \{(-\infty, a) \mid a \in \mathbb{R}\}$, then any set S with $\sup(S) \in S$ is compact, and the saturated compact sets have the form $(-\infty, s]$ for $s \in \mathbb{R}$.

Remark 3.22. The interpretation of $c : \mathcal{O}(\mathcal{X}) \to \mathbb{S}$ as a point-set may fail to be compact. For if $X = (\mathbb{Q}, \tau_<)$ and $r \notin \mathbb{Q}$, then the function $c : (-\infty, a) \mapsto \top \iff a > r$ is continuous and satisfies (3.4), but the corresponding point-set C is $(-\infty, r] \cap \mathbb{Q} = (-\infty, r) \cap \mathbb{Q}$ which is not compact. Additionally, the identity $c(U) = \top \iff C \subset U$ fails for the set $U = (-\infty, r)$. In order that the interpretation of c as a point-set yields a compact set C, we require the space X to be *sober*. We shall return to this point in Section 4.4.

We now give some computability results which are valid for any topological type.

Theorem 3.23. *The following operators are computable:*

(i) Complement $\mathcal{O} \leftrightarrow \mathcal{A}$.

(ii) Finite intersection $\mathcal{O} \times \mathcal{O} \to \mathcal{O}$.

(iii) Countable union $\mathcal{O}^{\mathbb{N}} \to \mathcal{O}$.

(iv) Finite union $\mathcal{A} \times \mathcal{A} \to \mathcal{A}$.

(v) Countable intersection $\mathcal{A}^{\mathbb{N}} \to \mathcal{A}$.

(vi) (Closed) intersection $\mathcal{V} \times \mathcal{O} \to \mathcal{V}$.

(vii) Intersection $\mathcal{K} \times \mathcal{A} \to \mathcal{K}$.

(viii) Countable union $\mathcal{V}^{\mathbb{N}} \to \mathcal{V}$.

(ix) Finite union $\mathcal{K} \times \mathcal{K} \to \mathcal{K}$.

(x) Singleton $\mathcal{X} \to \mathcal{V}$ *and* $\mathcal{X} \to \mathcal{K}$.

Proof. The basic proof technique in all cases is the same; in order to prove that a function $\hat{f} : \mathcal{W} \to \mathcal{Y}^{\mathcal{X}}$ is computable, it suffices to show that the function $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$ satisfying $f(w, x) = \hat{f}(w)(x)$ is computable.

(i) $x \in U \iff x \notin X \setminus U$; $x \notin A \iff x \in X \setminus A$.

(ii) $x \in U_1 \cap U_2 \iff (x \in U_1) \wedge (x \in U_2)$.

(iii) $x \in \bigcup_{n=1}^{\infty} U_n \iff \bigvee_{n=1}^{\infty} (x \in U_n)$.

(iv) $x \notin A_1 \cup A_2 \iff (x \notin A_1) \wedge (x \notin A_2)$.

(v) $x \notin \bigcap_{n=1}^{\infty} A_n \iff \bigwedge_{n=1}^{\infty} (x \notin A_n)$.

(vi) $B \cap U \bowtie V \iff B \bowtie U \cap V$.

(vii) $C \cap A \subset U \iff C \subset U \setminus A = U \cap (X \setminus A)$.

(viii) $\bigcup_{n=1}^{\infty} B_n \bowtie U \iff \bigvee_{n=1}^{\infty} (B \bowtie U_n)$.

(ix) $C_1 \cup C_2 \subset U \iff (C_1 \subset U) \wedge (C_2 \subset U)$.

(x) $x \in U \iff \{x\} \bowtie U \iff \{x\} \subset U$.

□

We have seen that the exponential $\mathcal{Y}^{\mathcal{X}}$ (alternatively denoted $\mathcal{C}(\mathcal{X};\mathcal{Y})$) is a canonical type for the continuous functions $X \to Y$, and that evaluation $e : \mathcal{C}(\mathcal{X};\mathcal{Y}) \times \mathcal{X} \to \mathcal{Y}$ is computable. We can extend computability to images and preimages of sets.

Theorem 3.24 (Computable image/preimage)**.**

1. *Preimage* $(f, S) \mapsto f^{-1}(S)$ *is computable* $\mathcal{C}(\mathcal{X};\mathcal{Y}) \times \mathcal{O}(\mathcal{Y}) \to \mathcal{O}(\mathcal{X})$ *and* $\mathcal{C}(\mathcal{X};\mathcal{Y}) \times \mathcal{A}(\mathcal{Y}) \to \mathcal{A}(\mathcal{X})$.
2. *Image* $(f, S) \mapsto f(S)$ *is computable* $\mathcal{C}(\mathcal{X};\mathcal{Y}) \times \mathcal{V}(\mathcal{X}) \to \mathcal{V}(\mathcal{Y})$ *and* $\mathcal{C}(\mathcal{X};\mathcal{Y}) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{Y})$.

Proof.

1. $x \in f^{-1}(U) \iff f(x) \in U$ and $x \notin f^{-1}(A) \iff f(x) \notin A$.
2. $f(A) \bowtie V \iff A \bowtie f^{-1}(V)$ and $f(C) \subset V \iff C \subset f^{-1}(V)$. □

We now turn to multivalued functions $F : X \rightrightarrows Y$, which can be thought of as set-value functions $F : X \to \mathcal{P}(Y)$. A multivalued function is open- (respectively closed- or compact-) valued if $F(x)$ is open (respectively closed or compact) for all x. The *image* of a set S is defined by $F(S) = \bigcup\{F(x) \mid x \in S\} = \{y \in Y \mid \exists x \in S,\ y \in F(x)\}$. The *composition* of $F : X \rightrightarrows Y$ and $G : Y \rightrightarrows Z$ is defined by $G \circ F(x) = G(F(x)) = \{z \in Z \mid \exists y \in Y,\ y \in F(x) \wedge z \in F(y)\}$. The *weak preimage* F^{-1} of $F : X \rightrightarrows Y$ is defined by $F^{-1}(B) = \{x \in X \mid F(x) \cap B \neq \emptyset\}$, and is a multivalued function $F^{-1} : Y \rightrightarrows X$. The *strong preimage* $F^{\Leftarrow}$ of F is defined by $F^{\Leftarrow}(B) = \{x \in X \mid F(x) \subset B\}$. A multivalued function $F : X \rightrightarrows Y$ is *upper-semicontinuous* if $F^{-1}(A)$ is closed whenever A is closed; equivalently if $F^{\Leftarrow}(U)$ is open whenever U is open. F is *lower-semicontinuous* if $F^{-1}(U)$ is open whenever U is open; equivalently if $F^{\Leftarrow}(A)$ is closed whenever A is closed. A multivalued map $F : X \rightrightarrows Y$ defines a relation $R \subset X \times Y$ by $R(x, y) \iff F(x) \ni y$; conversely every relation between X and Y defines a multivalued function.

Theorem 3.25 (Action of multivalued functions)**.**

1. *Multivalued preimage* $(F, S) \mapsto F^{-1}(S)$ *is computable* $\mathcal{C}(\mathcal{X};\mathcal{V}(\mathcal{Y})) \times \mathcal{O}(\mathcal{Y}) \to \mathcal{O}(\mathcal{X})$ *and* $\mathcal{C}(\mathcal{X};\mathcal{K}(\mathcal{Y})) \times \mathcal{A}(\mathcal{Y}) \to \mathcal{A}(\mathcal{X})$.
2. *Multivalued image* $(F, S) \mapsto F(S)$ *is computable* $\mathcal{C}(\mathcal{X};\mathcal{V}(\mathcal{Y})) \times \mathcal{V}(\mathcal{X}) \to \mathcal{V}(\mathcal{Y})$ *and* $\mathcal{C}(\mathcal{X};\mathcal{K}(\mathcal{Y})) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{Y})$.

Proof.

1. $x \in F^{-1}(V) \iff F(x) \bowtie V$ and $x \notin F^{-1}(B) \iff F(x) \cap B = \emptyset \iff F(x) \subset (Y \setminus B)$.

2. $F(A) \bowtie V \iff A \bowtie F^{-1}(V)$ and $F(C) \subset V \iff C \subset F^{\Leftarrow}(V) \iff C \subset X \setminus F^{-1}(Y \setminus V)$. □

Classically, a multivalued function $X \rightrightarrows Y$ can be expressed as a subset of $X \times Y$. However, this only holds computably for open- and closed- valued functions.

Proposition 3.26.

1. *The types $\mathcal{C}(\mathcal{X}; \mathcal{O}(\mathcal{Y}))$ and $\mathcal{O}(\mathcal{X} \times \mathcal{Y})$ are computably equivalent.*
2. *The types $\mathcal{C}(\mathcal{X}; \mathcal{A}(\mathcal{Y}))$ and $\mathcal{A}(\mathcal{X} \times \mathcal{Y})$ are computably equivalent.*
3. *The types $\mathcal{C}(\mathcal{X}; \mathcal{V}(\mathcal{Y}))$ and $\mathcal{V}(\mathcal{X} \times \mathcal{Y})$ are in general not equivalent.*
4. *The types $\mathcal{C}(\mathcal{X}; \mathcal{K}(\mathcal{Y}))$ and $\mathcal{K}(\mathcal{X} \times \mathcal{Y})$ are in general not equivalent.*

Proof.

1. $y \in F(x) \iff (x, y) \in \widetilde{F}$.
2. $y \notin F(x) \iff (x, y) \notin \widetilde{F}$.
3. Consider $\mathcal{X} = \mathcal{Y} = \mathcal{R}$ and the sets $\widetilde{F}_n = \{(2^{-n}, 0)\}$ with $\widetilde{F}_\infty = \{(0, 0)\}$. Then $\widetilde{F}_n \to \widetilde{F}_\infty$ in $\mathcal{V}(\mathcal{R} \times \mathcal{R})$, but $F_n(0) = \emptyset$ for $n \in \mathbb{N}$ but $F_\infty(0) = \{0\}$.
4. If $\mathcal{X}$ is not compact, then only multivalued functions with compact domain have a graph in $\mathcal{K}(X \times Y)$. If $\mathcal{X}$ is not Hausdorff, then $F(x) \subset V \iff \{x\} \times (Y \setminus V) \cap R = \emptyset$, but $\{x\}$ need not be closed. □

The above observation has important ramifications for the notion of causality in dynamic systems. With additional *effectivity properties* of Definition 3.28, we can derive reductions between $\mathcal{C}(\mathcal{X}; \mathcal{V}(\mathcal{Y}))$ and $\mathcal{V}(\mathcal{X} \times \mathcal{Y})$, and between $\mathcal{K}(\mathcal{X} \times \mathcal{Y})$ and $\mathcal{C}(\mathcal{X}; \mathcal{K}(\mathcal{Y}))$.

Define the multivalued *restriction map* I_S for a set S by $I_S(x) = \{x\}$ if $x \in S$ and $I_S(x) = \emptyset$ otherwise. The following useful result shows that open/closed sets have overt/compact restriction maps.

Theorem 3.27 (Properties of restriction maps)**.** *The restriction map $S \mapsto I_S$ is $\mathcal{O}(\mathcal{X}) \to \mathcal{C}(\mathcal{X}; \mathcal{V}(\mathcal{X}))$-computable and $\mathcal{A}(\mathcal{X}) \to \mathcal{C}(\mathcal{X}; \mathcal{K}(\mathcal{X}))$-computable.*

Proof. If U is open, then $I_U(x) \bowtie V \iff x \in U \cap V$. If A is closed, then $I_A(x) \subset V \iff x \in V \cup (X \setminus A)$. □

3.5 Effective topological properties

We now consider some familiar classical properties of topological spaces, give "effective" versions of them, and deduce some of their properties.

Definition 3.28 (Effectivity properties)**.** Let $\mathcal{X}$ be a topological type. Then:

1. $\mathcal{X}$ is *effectively discrete* if the equality relation $=: \mathcal{X} \times \mathcal{X} \to \mathbb{S}$ is verifyable.
2. $\mathcal{X}$ is *effectively distinguishable* if the relation $\neq: \mathcal{X} \times \mathcal{X} \to \mathbb{S}$ is verifyable.
3. $\mathcal{X}$ is *effectively separable* if X is a computable element of $\mathcal{V}(\mathcal{X})$.
4. $\mathcal{X}$ is *effectively compact* if X is a computable element of $\mathcal{K}(\mathcal{X})$.

Remark 3.29. Even if inequality $\neq$ is verifyable, the space X need not be a Hausdorff, but the Hausdorff property always holds if $X \times X$ with the product topology is a sequential space; in particular if X is locally-compact.

The following proposition provides the relationship between overtness and separability; a space is computably overt if, and only if, it has a computable dense subset.

Proposition 3.30. *Let $\mathcal{X}$ be a topological type. Then:*

1. *$\mathcal{X}$ is effectively separable if there is a computable function $\xi : \mathcal{N} \to \mathcal{X}$ with dense range.*
2. *$\mathcal{X}$ is effectively compact if there is a computable surjective function $\xi : \mathcal{B} \to \mathcal{X}$.*

The following result gives a link between effectivity properties of the topological type and conversions between subsets of that type.

Theorem 3.31. *Let $\mathcal{X}$ be a topological type. Then:*

1. *The interior function $\mathcal{V}(\mathcal{X}) \to \mathcal{O}(\mathcal{X})$ is computable if, and only if, $\mathcal{X}$ is effectively discrete.*
2. *The closure function $\mathcal{K}(\mathcal{X}) \to \mathcal{A}(\mathcal{X})$ is computable if, and only if, $\mathcal{X}$ is effectively distinguishable.*
3. *The closure function $\mathcal{O}(\mathcal{X}) \to \mathcal{V}(\mathcal{X})$ is computable if $\mathcal{X}$ is effectively separable.*
4. *The saturation function $\mathcal{A}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$ is computable if $\mathcal{X}$ is effectively compact.*

Proof.

1. Let $W \in \mathcal{V}(\mathcal{X})$. Since for any $x \in X$, $\{x\} = \{y \in X \mid x = y\}$ is open, we have $x \in W \iff W \cap \{x\} \neq \emptyset$.

   Conversely, if $\mathcal{V}(\mathcal{X}) \to \mathcal{O}(\mathcal{X})$, then $x = y \iff \{x\} \cap \{y\} \neq \emptyset$. Since the singleton operator $\mathcal{X} \to \mathcal{V}(\mathcal{X})$ is always computable, we can compute a name of $\{x\}$ in $\mathcal{O}(\mathcal{X})$ and a name of $\{y\}$ in $\mathcal{V}(\mathcal{X})$. Hence $=: \mathcal{X} \times \mathcal{X} \to \mathbb{S}$ is verifyable.

2. For given x, the function $X \to \mathbb{S}$ given by $y \mapsto \top \iff x \neq y$ is computable since X is effectively distinguishable, so $\{y \mid y \neq x\}$ is computably open. Then $x \notin C \iff C \subset (X \setminus \{x\})$.

   Conversely, $x \neq y \iff \{x\} \cap \{y\} = \emptyset$. Since singleton $x \mapsto \{x\}$ is always computable $\mathcal{X} \to \mathcal{K}(\mathcal{X})$, if $\mathcal{K}(\mathcal{X}) \to \mathcal{A}(\mathcal{X})$ is computable, we can compute a $\{x\}$ in $\mathcal{K}(\mathcal{X})$ and $\{y\}$ in $\mathcal{A}(\mathcal{X})$, and hence verify $\{x\} \cap \{y\} = \emptyset$.

3. Suppose X is computable in $\mathcal{V}(\mathcal{X})$, and let $U \in \mathcal{O}(\mathcal{X})$. Then $\mathrm{cl}(U) = \mathrm{cl}(U \cap X)$ is computable in $\mathcal{V}(\mathcal{X})$. Conversely, if the closure function $\mathcal{O}(\mathcal{X}) \to \mathcal{V}(\mathcal{X})$ is computable, then since X is computable in $\mathcal{O}(\mathcal{X})$, it is also computable in $\mathcal{V}(\mathcal{X})$.

4. Suppose X is computable in $\mathcal{K}(\mathcal{X})$, and let $A \in \mathcal{A}(\mathcal{X})$. Then $A = A \cap X$ is computable in $\mathcal{K}(\mathcal{X})$. Conversely, if the saturation function $\mathcal{A}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$ is computable, then since X is computable in $\mathcal{A}(\mathcal{X})$, it is also computable in $\mathcal{K}(\mathcal{X})$.

□

Note that since the effective distinguishability/Hausdorff property and the effective separability property are relatively common for the main types used in analysis, the conversions $\mathcal{O}(\mathcal{X}) \to \mathcal{V}(\mathcal{X})$ and $\mathcal{K}(\mathcal{X}) \to \mathcal{A}(\mathcal{X})$ are usually computable, whereas the conversions $\mathcal{V}(\mathcal{X}) \to \mathcal{O}(\mathcal{X})$ and $\mathcal{A}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$ are usually not. This means that a description of an open set in $\mathcal{O}(\mathcal{X})$ provides more information than a description of its closure in $\mathcal{V}(\mathcal{X})$, and a description of a compact set in $\mathcal{K}(\mathcal{X})$ provides more information than its description in $\mathcal{A}(\mathcal{X})$.

We now prove that effectivity properties extend to countable products.

Theorem 3.32. *Let $\mathcal{X}_n$, $n \in \mathbb{N}$ be computable types.*

1. *If each $\mathcal{X}_n$ is effectively separable, then $\prod_{n=0}^{\infty} \mathcal{X}_n$ is effectively separable.*
2. *If each $\mathcal{X}_n$ is effectively compact, then $\prod_{n=0}^{\infty} \mathcal{X}_n$ is effectively compact.*

Proof.

1. Suppose $\xi_n : \mathcal{N} \to \mathcal{X}_n$ are computable functions with countable dense range. Let $h : \mathbb{N}^* \to \mathbb{N}$ be a computable bijective function. Define $\eta : \mathcal{N} \to \prod_{n=0}^{\infty} \mathcal{X}_n$ by $\eta(h(m_0, \ldots, m_j)) = (\xi_0(m_0), \xi_1(m_1), \ldots, \xi_j(m_j), \xi_{j+1}(0), \xi_{j+2}(0), \ldots)$. Then η has dense range and is computable by construction.

2. Since each $\mathcal{X}_n$ is effectively compact, there exist computable surjective functions $\xi_n : \mathcal{B} \to \mathcal{X}_n$. Define $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by $g(i,j) = (i+j)(i+j+1)/2 + j$, and note that g is a bijection. Define $\tau : \mathcal{B}^\omega \to \mathcal{B}$ by $\tau(p_0, p_1, \ldots)_k = p_{i,j}$ where $g(i,j) = k$, and note that τ is also a bijection. Define $\eta : \mathcal{B} \to \prod_{n=0}^{\infty} X_n$ by $(\eta(q))_i = \xi_i(p_i)$ for $q = \tau(p_0, p_1, \ldots)$. Then $\eta : \mathcal{B} \to \prod_{n=0}^{\infty} X_n$ is surjective since each ξ_i is surjective, and is computable by construction. □

The following result shows that projection maps are computable on open and closed subsets given suitable effectivity properties.

Proposition 3.33.

1. *Suppose Y is effectively separable. Then the set projection operator $\pi_X : \mathcal{O}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{O}(\mathcal{X})$ is computable.*
2. *Suppose Y is effectively compact. Then the set projection operator $\pi_X : \mathcal{A}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{A}(\mathcal{X})$ is computable.*

Proof.

1. $x \in \pi_X(A) \iff (\{x\} \times Y) \cap A \neq \emptyset$.
2. $x \notin \pi_X(A) \iff (\{x\} \times Y) \cap A = \emptyset$. □

We now give an effective version of local compactness.

Notation 3.34. We write $U \Subset V$ if every open cover of V has a finite subcover of U. We define $\Uparrow U = \{V \in \mathcal{O}(X) \mid U \Subset V\}$.

Definition 3.35 (Effectively locally-compact)**.** A computable type $\mathcal{X}$ is *effectively locally-compact* if there is a computable function $\mathcal{N} \to \mathcal{O}(\mathcal{X}) \times \mathcal{K}(\mathcal{X})$, $n \mapsto (V_n, K_n)$ such that for any n, $V_n \subset K_n$, and for any $x \in X$ and $W \in \mathcal{O}(X)$, there exists n such that $x \in V_n$ and $K_n \subset W$.

We say that $\mathcal{X}$ is *strongly effectively locally-compact* if we can take $K_n = \bigcap\{W \in \mathcal{O}(X) \mid V_n \Subset W\}$ for all n; if X is a Hausdorff space this is equivalent to $K_n = \mathrm{cl}(V_n)$.

Remark 3.36. Using the definition given, it is immediate that X is a countably-based space, since any U is given by $U = \bigcup\{V_n \mid K_n \subset U\}$, and is effectively separable, since $U \neq \emptyset \iff \exists n,\ U \supset K_n$. It would be interesting to find a weaker notion of effective local compactness which does not imply these properties.

Frequently, rather than find a neighbourhood of a point, we need a neighbourhood of a compact set. The next result shows that this can always be done.

Proposition 3.37. *Suppose $\mathcal{X}$ is effectively locally compact. Then there is a computable function $\mathbb{N} \to \mathcal{O}(\mathcal{X}) \times \mathcal{K}(\mathcal{X})$, $n \mapsto (W_n, L_n)$ with $W_n \subset L_n$ such that for any compact C and open U with $C \subset U$, there exists n with $C \subset W_n$ and $L_n \subset U$.*

Proof. Consider the pairs $(\bigcup_{n\in N} V_n, \bigcup_{n\in N} K_n)$ for N a finite subset of $\mathbb{N}$. Let $C \subset U$ be compact. Then $\{V_n \mid K_n \subset U\}$ is an open cover of C, so has a finite subcover $\{V_n \mid n \in N\}$. By construction $\{K_n \mid n \in N\} \subset U$. □

3.6 Standard representations of topological and metric spaces

In this section we show how to build representations of topological and metric spaces. These constructions can be used to give equivalent representations for the spaces constructed as category-theoretic products and exponentials, or to give representations of new spaces which extend the type theory. The material of this section is based on the definitions of Weihrauch and Grubba [54].

Definition 3.38 (Effective topological spaces; standard representation)**.** An *effective topological space* is a tuple (X, τ, σ, ν) where (X, τ) is a second-countable Kolmogorov (T_0) space, σ is a sub-base for τ, and ν is a notation for σ .

If σ is base for τ, we write β instead of σ, and say that (X, τ, β, ν) is a effective topological space.

The *standard representation* δ of (X, τ, σ, ν) is the function $\delta : \Sigma^\omega \rightharpoonup X$ such that

$$\delta(p) = x \iff \{\nu(w) \mid w \lhd p\} = \{J \in \sigma \mid x \in J\}. \tag{3.6}$$

In other words, a name of x in the standard representation encodes a list of *all* $J \in \sigma$ such that $x \in J$.

The following result shows that the standard representation of an effective topological space is an admissible quotient representation. The proof is similar to that of the corresponding parts of 4.5, but is considerably simpler, so we include it in full.

Theorem 3.39. *Let (X, τ, σ, ν) be an effective topological space. Then the standard representation δ of (X, τ, σ, ν) is an admissible quotient representation.*

Proof. *δ is continuous:* Let $U \subset X$ be open and $x = \delta(p) \in U$. Since σ is a sub-base, there exist $J_1, \ldots, J_k \in \sigma$ such that $x \in \bigcap_{i=1}^k J_i \subset U$. Since p is a name of x, we have $w_i \lhd p$ where $\nu(w_i) = J_i$ for $i = 1, \ldots, k$. Hence there is a prefix v of p such that $w_i \lhd v$ for all $i = 1, \ldots, k$. Hence $\delta(q) \in \bigcap_{i=1}^k J_i \subset U$ for any $q \in v\Sigma^\omega$.

δ is a quotient map: Suppose $W \subset X$ and $\delta^{-1}(W)$ is open. Let $x \in W$ and $p \in \delta^{-1}(x)$. Then there is a prefix v of p such that $q \in \delta^{-1}(W)$ whenever v is a prefix of q. From the information contained in v, we can only deduce that there are open

sets $J_1, \ldots, J_k \in \sigma$ such that $x \in J_i$ for all $i = 1, \ldots, k$. Hence $\bigcap_{i=1}^{k} J_i$ is an open neighbourhood of x which is a subset of W.

δ is admissible: Suppose $\phi : \Sigma^\omega \rightharpoonup X$ is a continuous partial function. Define a function $\tilde{\eta} : \Sigma^\omega \rightharpoonup \Sigma^\omega$ such that if v_1 is a prefix of v_2, then $\tilde{\eta}(v_1)$ is a prefix of $\tilde{\eta}(v_2)$, and that $w \lhd \eta(v)$ if, and only if, $\phi(p) \in \nu(w)$ whenever $p \in v\Sigma^\omega$. Define $\eta(p) = \lim_{n\to\infty} \tilde{\eta}(p|_n)$. By padding $\tilde{\eta}(v)$ with names of the empty set if necessary, we can ensure that $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$. Then by definition, $w \lhd \eta(p) \iff \phi(p) \in \nu(W)$, so $\delta(\eta(p)) = \phi(p)$ for all $p \in \mathrm{dom}(\phi)$. □

We would like to define concrete representations of subsets of an effective topological space. Since any open set is a union of basic open sets, we can define a representation of open sets by taking a name of U to be an encoding of a list of basic open sets J whose union is U. For technical convenience, we henceforth assume that $\emptyset \in \beta$.

Definition 3.40 (Standard representations of open and closed sets)**.** Let (X, τ, β, ν) be an effective topological space.

1. The standard representation $\theta_<$ of open sets $\mathcal{O}(X)$ is given by
$$\theta_<(p) = U \iff \bigcup\{\nu(w) \mid w \lhd p\} = U.$$
2. The standard upper representation $\psi_>$ of closed sets $\mathcal{A}(X)$ is given by
$$\psi_>(p) = A \iff \bigcup\{\nu(w) \mid w \lhd p\} = X \setminus A.$$

Unfortunately, the representation $\theta_<$ is in general too strong. The following result shows that it is always possible to verify $x \in U$ given a $\theta_<$-name of U.

Lemma 3.41. *Let (X, τ, β, ν) be an effective topological space. Then inclusion $\in: X \times \mathcal{O}(X) \to \mathbb{S}$ is computable.*

Proof. If $x \in U$ and $U = \bigcup_{i=0}^{\infty} J_i$, then $x \in U \iff \exists m \in \mathbb{N},\ x \in J_m$. □

Without additional assumptions it is not possible to compute intersections, in general.

Definition 3.42 (Computable intersection property)**.** An effective topological space (X, τ, β, ν) has the *computable intersection property* if there is a recursively-enumerable subset $\mathcal{I}$ of $\mathrm{dom}(\nu) \times \mathrm{dom}(\nu) \times \mathrm{dom}(\nu)$ such that for all $w_1, w_2 \in \mathrm{dom}(\nu)$,

$$\nu(v_1) \cap \nu(v_2) = \bigcup\{\nu(w) \mid (v_1, v_2, w) \in \mathcal{I}\}.$$

For convenience, we may sometimes say that there is a recursively enumerable subset $\mathcal{I}$ of $\beta \times \beta \times \beta$ such that $I_1 \cap I_2 = \bigcup\{J \in \beta \mid (I_1, I_2, J) \in \mathcal{I}\}$.

Proposition 3.43. *Let (X, τ, β, ν) be an effective topological space. Then intersection $\mathcal{O}(X) \times \mathcal{O}(X) \to \mathcal{O}(X)$ is $(\theta_<, \theta_<; \theta_<)$-computable if, and only if, (X, τ, β, ν) has the computable intersection property.*

Proof. If (X, τ, β, ν) has the computable intersection property, then we have $(\bigcup_{i=0}^{\infty} \nu(u_i)) \cap (\bigcup_{j=0}^{\infty} \nu(v_j)) = \bigcup_{i,j=0}^{\infty} (\nu(u_i) \cap \nu(v_j)) = \bigcup_{i,j=0}^{\infty} \bigcup\{\nu(w) \mid (u_i, v_j, w) \in \mathcal{I}\}$, so intersection is $(\theta_<, \theta_<; \theta_<)$-computable. Conversely, if intersection is $(\theta_<, \theta_<; \theta_<)$-computable, then $\nu(u) \cap \nu(v) = (\bigcup_{i=0}^{\infty} \nu(u)) \cap (\bigcup_{j=0}^{\infty} \nu(v)) = \bigcup_{k=0}^{\infty} \nu(w_k)$, so (X, τ, β, ν) has the effective intersection property. □

We now show that the computable intersection property is precisely what is needed that $(\mathcal{O}(X), \theta_<)$ has the type of $\mathcal{O}(\mathcal{X})$.

Theorem 3.44. *Let (X, τ, β, ν) be an effective topological space, and $\mathcal{X} = (X, [\delta])$ where δ is the standard representation of X. Then $(\mathcal{O}(X), \theta_<) \equiv (\mathcal{O}(X), \delta_{\mathcal{X}\to\mathbb{S}})$ i.e. has type $\mathcal{O}(\mathcal{X})$ if, and only if, (X, τ, β, ν) has the computable intersection property.*

Proof. If $\theta_<$ is effectively equivalent to $\delta_{\mathcal{X}\to\mathbb{S}}$, then intersection is $(\theta_<, \theta_<; \theta_<)$-computable, so (X, τ, β, ν) has the computable intersection property by Proposition .

Conversely, suppose (X, τ, β, ν) has the computable intersection property, and $p : X \to \mathbb{S}$ is continuous. Let $w_1, \ldots, w_k \in \mathrm{dom}(\nu)$, and simulate a computation of p on input $\langle w_1, w_2, \ldots, w_k, \ldots\rangle$. If this computation outputs `1` after reading at most the words $w_1, \ldots, w_k$, then we can deduce that $\bigcap_{i=1}^k \nu(w_i) \subset p^{-1}(\top)$. For either $\bigcap_{i=1}^k \nu(w_i) = \emptyset$, in which case $\bigcap_{i=1}^k \nu(w_i) \subset p^{-1}(\top)$ is vacuous, or $x \in \bigcap_{i=1}^k \nu(w_i) = \emptyset$ for some $x \in X$, in which case $\langle w_1, w_2, \ldots, w_k\rangle$ is a prefix of a name of x, and so the computation is valid.

Now since any $x \in p^{-1}(\top)$ has a δ-name, and the computation π of $p(x)$ is completed in a finite time, there exists $(w_1, \ldots, w_k)$ such that $q = \langle w_1, \ldots, w_k, \ldots\rangle$ is a δ-name of x, and computation of p on input q outputs `1` after reading at most the words $w_1, \ldots, w_k$, proving that $x \in \bigcap_{i=1}^k J_i \subset U$ with $J_i = \nu(w_i)$. Hence $p^{-1}(\top)$ is equal to $\bigcup\{\bigcap_{i=1}^k \nu(w_i) \mid \pi$ outputs `1` after reading at most $w_1, \ldots, w_k\}$. The result follows since we can write $\bigcap_{i=1}^k \nu(v_i) = \bigcup_{j=0}^\infty \nu(w_j)$ for some w_j computable from $v_1, \ldots, v_k$. □

Following [54], we make the following definition:

Definition 3.45 (Computable topological space)**.** An effective topological space (X, τ, β, ν) is a *computable topological space* if $\mathrm{dom}(\nu)$ is recursively-enumerable and (X, τ, β, ν) has the computable intersection property.

We now give concrete standard representations of overt and compact subsets of computable topological spaces. For the representation of open sets, a name p is a list of words $w \in \mathrm{dom}(\nu)$ encoding a list of basic open sets $J \in \beta$.

Definition 3.46 (Standard representations of overt and compact sets)**.** Let (X, τ, β, ν) be an effective topological space.

1. The standard lower representation $\psi_<$ of the closed sets $A \in \mathcal{A}(X)$ is given by
$$\psi_<(p) = A \iff \{\nu(w) \mid w \lhd p\} = \{J \in \beta \mid A \bowtie J\}.$$
2. The standard representation $\kappa_>$ of compact sets $\mathcal{K}(X)$ is given by
$$\kappa_>(p) = C \iff \{(\nu(w_1), \ldots, \nu(w_k)) \mid \langle w_1, \ldots, w_k\rangle \lhd p\} = \{(J_1, \ldots, J_k) \in \beta^* \mid C \subset J_1 \cup \cdots \cup J_k\}.$$

We can also define a standard representation for the space of continuous functions.

Definition 3.47 (Standard representation of continuous functions)**.** Let $(X, \tau_X, \beta_X, \nu_X)$ and $(Y, \tau_Y, \sigma_Y, \nu_Y)$ be effective topological spaces. The standard representation γ of $C(X; Y)$ is given by
$$\gamma(p) = f \iff \forall w_Y \in \mathrm{dom}(\nu_Y),\ f^{-1}(\nu(w_Y)) = \bigcup\{\nu_X(w_X) \mid \langle w_X, w_Y\rangle \lhd p\}.$$
Standard representations for spaces of multifunctions can be defined analogously.

Intuitively, a name of f in the standard representation γ of $C(X;Y)$ encodes a list $\mathcal{F}$ of pairs $(I,J) \in \sigma_X \times \sigma_Y$ such that $f^{-1}(J) = \bigcup\{I \mid (I,J) \in \mathcal{F}\}$. It is easy to verify that the standard representations of spaces of overt and compact subsets and of continuous functions are equivalent to those in Section 3.4 for computable topological spaces.

In order to relate open and overt sets, or compact and closed sets, we need some extra effectivity properties (see [11]) of the notation ν of the basic open sets $J \in \beta$.

Definition 3.48. Let (X,τ,β,ν) be an effective topological space. Then (X,τ,β,ν) has the:

1. *effective overlap property* if $\{w_1,w_2 \in \Sigma^* \times \Sigma^* \mid \nu(w_1) \cap \nu(w_2) \neq \emptyset\}$ is recursively-enumerable.
2. *effective disjointness property* if there is a recursively-enumerable set $\mathcal{D} \subset \mathrm{dom}(\nu) \times \mathrm{dom}(\nu)$ such that $\bigcup\{\nu(w_1) \times \nu(w_2) \mid (w_1,w_2) \in \mathcal{D}\} = \{(x,y) \in X \times X \mid x \neq y\}$.

The following theorem relates the effectivity properties for the standard representations with

Theorem 3.49. *Let (X,τ,β,ν) be an effective topological space. Then*

1. *the effective overlap property is equivalent to every open set being effectively overt.*
2. *the effective disjointness property is equivalent to every saturated compact set being effectively closed.*

Proof.

1. If (X,τ,β,ν) has the effective intersection property, then if $U = \bigcup_{j=1}^{\infty} J_i$ is open and $L \in \beta$, then $U \bowtie L \iff \exists i \in \mathbb{N},\ J_i \cap L \neq \emptyset$. Conversely, we have $J \bowtie L \iff (\bigcup_{i=0}^{\infty} J) \bowtie L$.
2. Effective disjointness is equivalent to the set $\{(x_1,x_2) \in X \times X \mid x_1 \neq x_2\}$ being computable as an open set. □

The definition of the effective disjointness predicate used here is different from that of [11]. We return to this point in Remark 3.56.

If (X,τ,β,ν) is a computable locally-compact topological space such that, then we have an alternative representation for the open sets.

Definition 3.50. Let (X,τ,σ,ν) be an effective topological space. The Scott representation $\theta'_<$ of open sets $\mathcal{O}(X)$ is given by

$$\theta'_<(p) = U \iff \{\nu(w) \mid w \lhd p\} = \{I \in \beta \mid I \Subset U\}.$$

Remark 3.51. If (X,τ) is a Hausdorff space, then $I \Subset \bigcup_{i=1}^{k} J_i$ is equivalent to $\bar{I} \subset \bigcup_{i=1}^{k} J_i$.

In order to relate the representations θ and θ' we need an extra *effective covering property.*

Definition 3.52 (Effective covering property)**.** Let (X,τ,β,ν) be an effective topological space. Then (X,τ,β,ν) has the *effective covering property* if $\{v,w_1,\ldots,w_k \in \Sigma^* \times \Sigma^* \times \cdots \times \Sigma^* \mid \nu(v) \Subset \nu(w_1) \cup \cdots \cup \nu(w_k)\}$ is recursively-enumerable.

The following theorem shows that the effective covering property is a necessary and sufficient condition for equivalence of the representations $\theta_<$ and $\theta'_<$.

Theorem 3.53. *Let (X, τ, β, μ) be an effective topological space space which is locally-compact. Then the representations $\theta_<$ and $\theta'_<$ are equivalent if, and only if, (X, τ, β, μ) has the effective covering property.*

Proof. Suppose (X, τ, β, μ) has the effective covering property. For any locally-compact space (X, τ) and any $U \in \mathcal{O}(X)$, we have $U = \bigcup\{I \in \beta \mid I \Subset U\}$. Hence any $\theta'_<$-name is also a $\theta_<$-name. If $U = \bigcup_{i=0}^{\infty} J_i$ and $I \Subset U$, there exists k such that $I \subset \bigcup_{i=1}^{k} J_i$, so we can prove that $I \Subset U$. Hence any $\theta_<$-name is also a $\theta'_<$-name

Conversely, if $\theta_<$ and $\theta'_<$ are equivalent, then we can compute a $\theta'_<$-name of $\bigcup_{i=1}^{k} J_i$, which amounts to enumerating $\{w \in \mathrm{dom}(\nu) \mid \nu(w) \Subset \bigcup_{i=1}^{k} J_i\}$. Hence (X, τ, β, μ) has the effective covering property. □

We now give standard representations of spaces of functions and multifunctions in an effectively locally-compact space.

Definition 3.54. If $(X, \tau_X, \beta_X, \nu_X)$ is a computable topological space and $(Y, \tau_Y, \sigma_Y, \nu_Y)$ be a computable topological space, then

1'. The Isbell representation γ' of $C(X; Y)$ is given by
$$\gamma'(p) = f \iff \{(\nu_X(v), \nu_Y(w)) \mid \langle v, w\rangle \lhd p\} = \{(I, J) \in \beta_X \times \beta_Y \mid I \Subset f^{-1}(J)\}.$$

In other words, a γ'-name of f encodes a list $\mathcal{F}'$ of *all* pairs $(I, J) \in \sigma_X \times \sigma_Y$ such that $I \Subset f^{-1}(J)$.

2'. The Isbell representation $\mu'_<$ of $C(X; \mathcal{V}(Y))$ is given by
$$\mu'_<(p) = F \iff \{(\nu_X(v), \nu_Y(w)) \mid \langle v, w\rangle \lhd p\} = \{(I, J) \in \beta_X \times \beta_Y \mid I \Subset F^{-1}(J)\}.$$

3'. The Isbell representation $\mu'_>$ of $C(X; \mathcal{K}(Y))$ is given by
$$\begin{aligned}\mu'_<(p) = F \iff \{(\nu_X(v), \nu_Y(w_1), \ldots, \nu_Y(w_k)) \mid \langle v, w_1, \ldots, w_k\rangle \lhd p\}\\ = \{(I, J_1, \ldots, J_k) \in \beta_X \times \beta_Y^* \mid I \Subset F^{\Leftarrow}(J_1 \cup \cdots \cup J_k)\}.\end{aligned}$$

The following result is a direct corollary of Theorem 3.53.

Corollary 3.55. *Let $(X, \tau_X, \beta_X, \nu_X)$ be an effectively locally-compact space, and $(Y, \tau_Y, \sigma_Y, \nu_Y)$ be an effective topological space. Then the standard representation γ of $C(X; Y)$ is equivalent to the Isbell representation γ'.*

Remark 3.56. If (X, τ, β, ν) is a locally-compact Hausdorff space, then we can choose a basis β such that $\bar{J}$ is compact for all $J \in \beta$. Then the effective covering property can be written as $\{I, J_1, \ldots, J_k \mid \bar{I} \subset \bigcup_{i=1}^{k} J_i\}$ is recursively-enumerable. Further, the effective disjointness property can be written as $\{I_1, I_2 \in \beta \times \beta \mid \bar{I}_1 \times \bar{I}_2 = \emptyset\}$ is recursively-enumerable. This recovers the definitions of [11].

We now turn to the metric spaces.

Definition 3.57 (Computable metric space)**.** A computable metric space is a tuple (X, d, ξ, α), where (X, d) is a metric space, $\xi : \Sigma^* \rightharpoonup X$ encodes a countable dense subset of X, and $\alpha : \Sigma^* \times \Sigma^* \times \mathbb{N} \rightharpoonup \mathbb{Q}$ is such that $|d(\xi(w_1), \xi(w_2)) - \alpha(w_1, w_2, \epsilon)| < \epsilon$.

The standard representation of a computable metric space is the *Cauchy representation*, that is,

$$\delta(\langle w_1, w_2, \ldots\rangle) = x \iff \lim_{n\to\infty} \xi(w_n) = x \text{ and } \alpha(w_m, w_n, 2^{-(\min(m,n)+1)}) < 2^{-(\min(m,n)+1)}.$$

Theorem 3.58. *The standard representation δ of a computable metric space (X, d, ξ, α) is an admissible quotient representation, and the metric d is a computable function $\mathcal{X} \times \mathcal{X} \to \mathcal{R}$.*

Proof. The proof that δ is an admissible quotient representation is similar to that of Theorem 3.39 and is omitted. To show that the metric is computable, let $\delta(\langle v_1, v_2, \ldots\rangle) = x$ and $\delta(\langle w_1, w_2, \ldots\rangle) = y$. Observe that $d(\xi(v_n), x) \leq 2^{-n}$ and $d(\xi(w_n), y) \leq 2^{-n}$ If $\delta(\langle v_1, v_2, \ldots\rangle) = x$ we have

$$\begin{aligned} |d(x,y) - \alpha(v_{n+2}, w_{n+2}, 2^{-(n+1)})| &\leq d(x, \xi(v_{n+2})) + d(\xi(w_{n+2}), y) \\ &\quad + \left|d(\xi(v_{n+2}), \xi(w_{n+2})) - \alpha(v_{n+2}, w_{n+2}, 2^{-(n+1)})\right| \\ &\leq 2^{-(n+2)} + 2^{-(n+2)} + 2^{-(n+1)} \leq 2^{-n}. \end{aligned}$$

Hence $\alpha(v_{n+2}, w_{n+2}, 2^{-(n+1)})$ is a strongly convergent Cauchy sequence with limit $d(x, y)$. □

We can construct a complete computable metric space from a countable metric space by defining a representation on equivalence classes of strict Cauchy sequences.

3.A Summary of computable types and operations

Predicates:

Element $\chi_S(x) \iff x \in S$.
Overlap $S \bowtie U \iff \exists\, x \in S,\, x \in U \iff A \cap U \neq \emptyset$.
Subset $S \subset U \iff \forall\, x \in S,\, x \in U$.

General constructions:

Terminal $\mathcal{I}$.
Element $\mathcal{X} \equiv \mathcal{I} \to \mathcal{X}$.
Subtype $\{x \in \mathcal{X} \mid p(x)\}$.
Sum $\mathcal{X}_1 + \mathcal{X}_2$ with inclusions $i_{1,2} : \mathcal{X}_{1,2} \to \mathcal{X}_1 + \mathcal{X}_2$.
Product $\mathcal{X}_1 \times \mathcal{X}_2$ with projections $p_{1,2} : \mathcal{X}_1 \times \mathcal{X}_2 \to X_{1,2}$.
Exponential $\mathcal{Y}^{\mathcal{X}} \equiv (\mathcal{X} \to \mathcal{Y})$; $\mathcal{Y}^{\mathcal{A}\times\mathcal{X}} \equiv (\mathcal{Y}^{\mathcal{X}})^{\mathcal{A}}$ (or $\mathcal{A} \times \mathcal{X} \to \mathcal{Y} \equiv \mathcal{A} \to \mathcal{Y}^{\mathcal{X}}$) with evaluation $\varepsilon : \mathcal{Y}^{\mathcal{X}} \times \mathcal{X} \to \mathcal{Y}$.

Countable types:

Binary Words $\{0,1\}^*$.
Boolean $\mathcal{B} := \{\top, \bot\}$.
Sierpinski $\mathcal{S} := \{\top, \uparrow\}$.
Trilogic $\mathcal{T} := \{\top, \bot, \uparrow\}$.
Naturals $\mathcal{N} := \{0, 1, 2, \ldots\}$.
Integers $\mathcal{Z} := \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
Rationals $\mathcal{Q} := \{m/n \mid m \in \mathbb{Z},\ n \in \mathbb{N},\ \text{and } n \neq 0\}$.
Rational step functions
Rational polynomials

Uncountable types:

Cantor set $\{0,1\}^\omega$.
Real numbers $\mathcal{R} := \{[x] \in \mathcal{Q}^{\mathcal{N}} \mid \mathrm{abs}(x_m - x_n) \leq 2^{-\min(m,n)},\ x \sim y \Longleftrightarrow \mathrm{abs}(x_n - y_n) \leq 2^{1-n}\}$.
Continuous functions $\mathcal{C}(\mathcal{X};\mathcal{Y}) := \mathcal{Y}^{\mathcal{X}}$.
Linear functions $\mathcal{L}(\mathcal{V};\mathcal{W}) := \{f \in \mathcal{C}(\mathcal{V};\mathcal{W}) \mid f(x + sy) = f(x) + sf(y)\}$.
Measures (weak) $\mathcal{M}(\mathcal{X}) := \{\mu \in \mathcal{L}(\mathcal{C}(\mathcal{X};\mathcal{R});\mathcal{R}) \mid f \geq 0 \implies \mu(f) \geq 0\}$.
Open sets $\mathcal{O}(\mathcal{X}) :\simeq \mathcal{S}^{\mathcal{X}}$; $\quad U \simeq \chi_U$.
Closed sets $\mathcal{A}(\mathcal{X}) :\simeq \mathcal{S}^{\mathcal{X}}$; $\quad A \simeq \chi_{X\setminus A}$.
Separable sets $\mathcal{V}(\mathcal{X}) := \{(A \bowtie \cdot) \in \mathcal{S}^{\mathcal{O}(\mathcal{X})} \mid A \bowtie (U \cup V) \iff A \bowtie U \vee A \bowtie V\}$.
Compact sets $\mathcal{K}(\mathcal{X}) := \{(C \subset \cdot) \in \mathcal{S}^{\mathcal{O}(\mathcal{X})} \mid C \subset (U \cap V) \iff C \subset U \wedge C \subset V\}$.
Flow $\{\phi \in \mathcal{C}(\mathcal{X}; \mathcal{C}(\mathcal{R};\mathcal{X})) \equiv \mathcal{C}(\mathcal{X} \times \mathcal{R};\mathcal{X}) \mid \phi(x, s+t) = \phi(\phi(x,s),t)\}$.

Logic:

Unprovability $\mathcal{B} \to \mathcal{S}$, $\bot \mapsto \uparrow$.
Countable disjunction $\mathcal{S}^{\mathcal{N}} \to \mathcal{S}$, $(a_n)_{n\in\mathbb{N}} \mapsto \bigvee_{n\in\mathbb{N}} a_n$.
Finite disjunction $\mathcal{S} \times \mathcal{S} \to \mathcal{S}$, $(a_1, a_2) \mapsto a_1 \wedge a_2$.

Arithmetic:

Unit $1 \in \mathcal{R}$.
Addition $(x, y) \mapsto x + y : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$.
Subtraction $(x, y) \mapsto x - y : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$.
Multiplication $(x, y) \mapsto x \times y : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$.
Division $(x, y) \mapsto x \div y : \mathcal{R} \times (\mathcal{R} \setminus 0) \to \mathcal{R}$.
Comparison $x \mapsto \operatorname{sgn}(x) : \mathcal{R} \to \mathcal{T}$.
Limit $(x_n)_{n\in\mathbb{N}} \mapsto \lim_{n\to\infty} x_n : \mathcal{R}^{\mathcal{N}} \rightharpoonup \mathcal{R}$ if $\exists\, \epsilon : \mathbb{N} \to \mathbb{Q}^+$ such that $\lim_{n\to\infty} \epsilon_n = 0$ and $|x_m - x_n| \leq \epsilon_{\min(m,n)}$.

Sets:

Complement $\mathcal{O} \leftrightarrow \mathcal{A}$.
Finite union $\mathcal{O} \times \mathcal{O} \to \mathcal{O}$, $\mathcal{A} \times \mathcal{A} \to \mathcal{A}$, $\mathcal{V} \times \mathcal{V} \to \mathcal{V}$, $\mathcal{K} \times \mathcal{K} \to \mathcal{K}$.
Countable union $\mathcal{O}^{\mathcal{N}} \to \mathcal{O}$, $\mathcal{V}^{\mathcal{N}} \to \mathcal{V}$.
Finite intersection $\mathcal{O} \times \mathcal{O} \to \mathcal{O}$, $\mathcal{A} \times \mathcal{A} \to \mathcal{A}$, $\mathcal{V} \times \mathcal{O} \to \mathcal{V}$, $\mathcal{K} \times \mathcal{A} \to \mathcal{K}$.
Countable intersection $\mathcal{A}^{\mathcal{N}} \to \mathcal{A}$. ($\mathcal{K}^{\mathcal{N}} \to \mathcal{K}$ can be derived if $\mathcal{X}$ is Hausdorff.)
Singleton $\mathcal{X} \to \mathcal{V}(\mathcal{X})$, $\mathcal{X} \to \mathcal{K}(\mathcal{X})$.
Closure $\mathcal{O}(\mathcal{X}) \to \mathcal{V}(\mathcal{X})$ if $\mathcal{X}$ is effectively separable.
Identity $\mathcal{K}(\mathcal{X}) \to \mathcal{A}(\mathcal{X})$ if $\mathcal{X}$ is effectively distinguishable.
Evaluation $\mathcal{C}(\mathcal{X}; \mathcal{Y}) \times \mathcal{X} \to \mathcal{Y}$.
Preimage $\mathcal{C}(\mathcal{X}; \mathcal{Y}) \times \mathcal{O}(\mathcal{Y}) \to \mathcal{O}(\mathcal{X})$, $\mathcal{C}(\mathcal{X}; \mathcal{Y}) \times \mathcal{A}(\mathcal{Y}) \to \mathcal{A}(\mathcal{X})$.
Image $\mathcal{C}(\mathcal{X}; \mathcal{Y}) \times \mathcal{V}(\mathcal{X}) \to \mathcal{V}(\mathcal{Y})$, $\mathcal{C}(\mathcal{X}; \mathcal{Y}) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{Y})$.
Element $\mathcal{X} \times \mathcal{O}(\mathcal{X}) \to \mathcal{S}$.
Overlap $\mathcal{V}(\mathcal{X}) \times \mathcal{O}(\mathcal{X}) \to \mathcal{S}$.
Subset $\mathcal{K}(\mathcal{X}) \times \mathcal{O}(\mathcal{X}) \to \mathcal{S}$.

Chapter 4

Classical Topology

In this chapter, we relate the computability theory developed in Chapter 3 to concepts from classical topology. The material in this chapter is mostly quite technical, and is not needed to actually use the type theory.

4.1 Sequential spaces

Since all spaces with an admissible quotient representation are sequential spaces, we begin with a brief overview of sequential spaces.

Lemma 4.1. *The space Σ^ω is a countably-based sequential space.*

Proof. A countable base for Σ^ω is given by the *cylinder sets* $C_w = \{p \in \Sigma^\omega \mid \forall i < |w|,\ p_i = w_i\}$ where $w \in \Sigma^*$ and $|w|$ denotes the length of w.

Suppose U is sequentially open and $p \in U$, but $p \notin \mathrm{int}(U)$. Then for all n, the set $p|_n\Sigma^\omega \setminus U = \{q \notin U \mid p|_n = q|_n\}$ is nonempty. Therefore there exists a sequence p_n such that $p_n|_n = p|_n$ and $p_n \notin U$, contradicting U being sequentially open. Hence $p \in \mathrm{int}(U)$, and since $p \in U$ is arbitrary, U is open. □

Recall that a subset C of X is *compact* if any open cover of C has a finite subcover, *countably compact* if any countable open cover of C has a finite subcover, and *sequentially compact* if any sequence in C has a convergent subsequence.

Theorem 4.2. *Let X be a topological space. Then any compact subset of X is countably compact, and any countably compact subset is sequentially compact. If X is a sequential space, then any sequentially compact set is countably compact. If X has a countable pseudobase, then any countably compact set is compact.*

Proof. It is immediate from the definition that any compact subset is countably compact. If C is countably compact and $\vec{x}$ is a sequence in C, define $A_n = \mathrm{cl}\{x_n, x_{n+1}, \ldots\}$ and $U_n = X \setminus A_n$. Then $\{U_n \mid n \in \mathbb{N}\}$ is a countable collection of open increasing open sets such that no U_n contains C. Hence $U_\infty = \bigcup_{n=0}^\infty U_n$ is not a cover of C, so $A_\infty \cap C \neq \emptyset$ where $A_\infty = X \setminus U_\infty$. However, $A_\infty = \bigcap_{n=0}^\infty \mathrm{cl}\{x_n, x_{n+1}, \ldots\}$ is the set of limit points of $\vec{x}$. So some subsequence of $\vec{x}$ has limit in C. This shows that C is sequentially compact.

Conversely, let X is a sequential space and C is sequentially compact. Suppose $\mathcal{U} = \{U_0, U_1, \ldots\}$ is a countable collection of open sets with no finite subset covering C. Then there exists a sequence of points x_n such that $x_n \in C \setminus \bigcup_{k=0}^n U_k$. Since C

is sequentially compact, the sequence $\vec{x}$ has a convergent subsequence $x_{k(n)}$ with limit $x_\infty \in C$. By taking $y_n = x_{k_m}$ with $k_m \geq m$, we can construct a convergent sequence $y_n \to y_\infty$ with $y_n \in C \setminus \bigcup_{k=0}^{n} U_k$ and limit $y_\infty \in C$. Then $y_\infty \notin \bigcup_{n=0}^{\infty} U_n$, so $\mathcal{U}$ is not an open cover of C. Taking the contrapositive, we see that any countable open cover has a finite subcover.

To prove the final part, we show the following result.

Lemma 4.3. *Suppose X is has a countable pseudobase. Then for any $\mathcal{U} \subset \mathcal{O}(X)$, there exists countable $\mathcal{V} \subset \mathcal{U}$ such that $\bigcup \mathcal{V} = \bigcup \mathcal{U}$.*

Finally, suppose X has a countable pseudobase $\mathcal{B}$ and C is countably compact. Let $\mathcal{U}$ be an open cover of C. For $x \in C$, there exists $U \in \mathcal{U}$ such that $x \in U$. Since $\mathcal{B}$ is a pseudobase, there exists $B \in \mathcal{B}$ such that $x \in B \subset U$. By the axiom of choice, there is a countable subset $\mathcal{C}$ of $\mathcal{B}$ such that $C \subset \bigcup \mathcal{C}$ and for all $B \in \mathcal{C}$, there exists $U \in \mathcal{U}$ such that $B \subset U$. By the axiom of countable choice, there is a countable subset $\mathcal{V}$ of $\mathcal{U}$ such that for all $B \in \mathcal{C}$, there exists $U \in \mathcal{V}$ with $B \subset U$. By construction $C \subset \bigcup \mathcal{C} \subset \bigcup \mathcal{V}$, so $\mathcal{V}$ is a countable subset of $\mathcal{U}$ which covers C. □

4.2 Spaces with an admissible quotient representation

We now prove one of the most important results of computable type theory, namely a characterisation of spaces with an admissible quotient representation. We will need the following definition:

Definition 4.4 (Sequential pseudobase)**.** A collection ρ of subsets of a topological space X is called a *sequential pseudobase* if for any convergent sequence $x_n \to x_\infty$ and any open set $U \ni x_\infty$, there exists $V \in \rho$ such that $V \subset U$ and $x_n \in V$ for all sufficiently large n.

Theorem 4.5 (Schröder [46, 5])**.** *Let X be a Kolmogorov topological space. Then the following are equivalent:*

1. *X has an admissible quotient representation.*
2. *X is a quotient of a countably-based space.*
3. *X is a sequential space with a countable sequential pseudobase.*

Proof.

(1 $\implies$ 2) Σ^ω is a countably-based space, so any subspace R is also countably-based. Let $\delta : \Sigma^\omega \rightharpoonup X$ be a quotient representation, and take $R = \mathrm{dom}(\delta)$, so $\delta : R \to X$ is a total quotient map. Thus X is a quotient of the countably-based space R.

(2 $\implies$ 3) Suppose X is a topological space, β is a countable base for X, and $q : X \to Y$ is a quotient map.

Y is a sequential space: Suppose $V \subset Y$ is sequentially open. We need to show that V is open. Since q is a quotient map, V is open if, and only if, $q^{-1}(V)$ is open. Let $x_\infty \in X$ but $x_\infty \notin \mathrm{int}(q^{-1}(V))$. Let $\{I_0, I_1, \ldots\} = \{I \in \beta \mid x \in I\}$. Since for any $n \in \mathbb{N}$, $\bigcap_{j=0}^{n} J_i$ is open and $x_\infty \notin \mathrm{int}(q^{-1}(V))$, the set $\bigcap_{j=0}^{n} J_i \setminus q^{-1}(V)$ is

nonempty. Hence we can choose a sequence x_n with $x_n \in \bigcap_{j=0}^{n} J_i \setminus q^{-1}(V)$. Since β is a base for X, for any open $U \ni x$, there exists n such that $\bigcap_{j=0}^{n} J_i \subset U$. Hence $x_n \to x_\infty$. Then $f(x_n) \to x_\infty$, so $f(x_n)$ is a convergent sequence in $Y \setminus V$. Since V is sequentially open, $x_\infty = \lim_{n\to\infty} f(x_n) \notin V$, so $x_\infty \notin f^{-1}(V)$. Hence $\mathrm{int}(q^{-1}(V)) = q^{-1}(V)$, so $q^{-1}(V)$ is open.

Y has a countable pseudobase: Let β_X be a countable base of X. Define

$$\mathcal{B}_Y = \{\mathrm{sat}(q(I_1 \cup \ldots \cup I_k)) \mid (I_1, \ldots, I_k) \in \beta_X^*\}. \tag{4.1}$$

We claim that $\mathcal{B}$ is a countable pseudobase of Y.

We first prove the following: Suppose K is a compact subset of Y, V open and $K \subset V$. Let U_n be a sequence of increasing open subsets of X such that $\bigcup_{n=0}^{\infty} U_n = q^{-1}(V)$. Then there exists m such that $\mathrm{sat}(q(U_m)) \supset K$. To this end, suppose that for all n, $\mathrm{sat}(q(U_m)) \not\supset K$. Then there exists a sequence of open sets $(W_n)_{n\in\mathbb{N}}$ such that for all n, $q(U_n) \subset W_n$ but $W_n \not\supset K$. Then $U_n \subset q^{-1}(W_n)$, which implies $\bigcap_{n\geq m} q^{-1}(W_n)$. Since q is a quotient map and $q^{-1}(\bigcap_{n\geq m} W_n) = \bigcap_{n\geq m} q^{-1}(W_n)$, we see that $V_m = \bigcap_{n\geq m} W_n$ is open for any m. Since $q^{-1}(V_m) \supset U_m$, we have $\bigcup_{m=0}^{\infty} q^{-1}(V_m) = q^{-1}(V)$, from which $\bigcup_{m=0}^{\infty} V_m = V$. Since K is compact, there exists m such that $K \subset V_m$, which means $K \subset \bigcap_{n=m}^{\infty} W_n$ and hence $K \subset W_m$, a contradiction. Hence there exists m such that $\mathrm{sat}(q(U_m)) \supset K$.

To complete the proof, let $y_n \to y_\infty$ with $y_\infty \in V$. Then there exists m such that $y_n \in V$ for all $n \geq m$. Take $K = \{y_m, y_{m+1}, \ldots, y_\infty\}$. Let $(I_0, I_1, \ldots)$ be a list of all $I \in \beta$ such that $I \subset q^{-1}(V)$, and let $U_j = \bigcup_{i=0}^{j} I_j$. Hence there exists k such that $y_n \in \mathrm{sat}(q(I_0 \cup \cdots \cup I_k))$ for all $n \geq m$.

(3 $\implies$ 1) Let X be a sequential topological space, and $\mathcal{B} = \{B_0, B_1, \ldots\}$ be a countable pseudobase with prefix-free notation $\nu : \Sigma^* \twoheadrightarrow \mathcal{B}$. Define a function $\delta : \Sigma^\omega \rightharpoonup X$ by

$$\delta(p) = x \iff \forall w \lhd p,\ x \in \nu(w) \text{ and } \forall \text{ open } U \ni x, \exists w \lhd p,\ \nu(w) \subset U. \tag{4.2}$$

Here, by $w \lhd p$ we mean that $p = w_0 w_1 w_2 \cdots$ with $w_i \in \mathrm{dom}(\nu)$ for all i and $w = w_i$ for some i. We claim that δ is an admissible quotient representation.

δ is single-valued: If $x \neq y$, there exists open U containing exactly one of x, y. Without loss of generality, suppose $x \in U$ and $y \notin U$. Then for any name p of x, there exists $w \lhd p$ such that $x \in \nu(w) \subset U$, but for any name q of y and any w such that $\nu(w) \subset U$, we have $y \notin \nu(w)$, so $w \not\lhd q$.

δ is surjective: For any $x \in X$, there exists $B \in \beta$ such that $x \in B$.

δ is continuous: Let $U \in \mathcal{O}(X)$ and $p \in \delta^{-1}(U)$ with $x = \delta(p)$. Then there exists $B = \nu(w) \in \beta$ such that $x \in B \subset U$, and then $w \lhd p$. There is an open neighbourhood W of p in Σ^ω such that $w \lhd q$ for all $q \in W$, so for any $q \in W \cap \mathrm{dom}(\delta)$, we have $\delta(q) \in B \subset U$. Hence $p \in W \subset \delta^{-1}(U)$, so δ is continuous.

δ is a quotient: Suppose that $f : X \to Y$ and $\phi = f \circ \delta$ is continuous. Let V be open in Y and $U = f^{-1}(V)$ with $x_\infty \in U$. Let p_∞ be a δ-name of x_∞ so $y_\infty = \phi(p_\infty) \in V$. Let p_n be a sequence in $\mathrm{dom}(\delta)$ such that $p_n \to p_\infty$, and let $x_n = \delta(p_n)$ Since δ is continuous and X is a sequential space, $x_n \to x_\infty$ as $n \to \infty$.

Since ϕ is continuous, there exists N such that $\phi(p_n) \in V$ for all $n \geq N$. Then for $n \geq N$ we have $f(x_n) = f(\delta(p_n)) = \phi(p_n) \in V$, so $x_n \in f^{-1}(V)$. Hence $f^{-1}(V)$ is sequentially open, so is open.

δ is admissible: Suppose $\phi : \Sigma^\omega \rightharpoonup X$ is a continuous partial function. Define a function $\tilde{\eta} : \Sigma^\omega \rightharpoonup \Sigma^\omega$ such that if v_1 is a prefix of v_2, then $\tilde{\eta}(v_1)$ is a prefix of $\tilde{\eta}(v_2)$, and that $w \lhd \eta(v)$ if, and only if, $\phi(p) \in B$ whenever $p \in v\Sigma^\omega$. Define $\eta(p) = \lim_{n\to\infty} \tilde{\eta}(p|_n)$. By padding $\tilde{\eta}(v)$ with names of the empty set if necessary, we can ensure that $\eta : \Sigma^\omega \rightharpoonup \Sigma^\omega$.

It remains to show that for all $p \in \mathrm{dom}(\phi)$, $\eta(p)$ is a δ-name of $\phi(p)$. First, if $w \lhd \eta(p)$, then $\phi(p) \in \nu(w)$. It remains to show that if $\phi(p) \in U$, there exists $w \lhd \eta(p)$ such that $x \in \nu(w) \subset U$. If $\phi(p_\infty) \in U$, there exists $B = \nu(w) \in \mathcal{B}$ such that $\phi(p_\infty) \in B$. Then for any sequence $p_n \to p_\infty$, the sequence $\phi(p_n)$ converges to $\phi(p_\infty)$. Since B is pseudobasic, there exists $N \in \mathbb{N}$ such that $\phi(p_n) \in B$ for all $n \geq N$. The $p_n \in \phi^{-1}(B)$ for all $n \geq N$. Since the sequence p_n is arbitrary, there is a neighbourhood of p mapping into B, so $w \lhd p$.

□

Remark 4.6. It *not* true that convergent sequences lift under topological quotients. For example, the disjoint union of $[-1, 0]$ and $[0, +1]$ quotients onto $[-1, +1]$, but any sequence in $[-1, +1]$ with infinitely many positive and negative elements does not lift to the base space.

The construction of the pseudobase for the topological quotient is [46, Lemma 3.1.12].

Remark 4.7. It is not true in general that if $q : Y \to X$ is a quotient map and δ_Y is an admissible quotient representation of Y, then $q \circ \delta_Y$ is an admissible quotient representation of X. See [18].

Remark 4.8. To the best of our knowledge, it is an open question as to whether every separable sequential space satisfies the equivalent conditions of Theorem 4.5.

4.3 The Scott topology on open sets

We now give a more detailed study of the open set type introduced by Definition 3.19. We first study the induced topology on $\mathcal{O}(X)$.

Definition 4.9 (Scott topology)**.** Let X be a topological space. A collection $\mathcal{U}$ of open subsets of X is *Scott open* if:

1. $U \in \mathcal{U}$, $V \in \mathcal{O}(X)$ and $U \subset V$ implies $V \in \mathcal{U}$, and
2. whenever $\mathcal{V} \subset \mathcal{O}(X)$ and $\bigcup \mathcal{V} \in \mathcal{U}$, there is a finite subset $\{V_1, \ldots, V_n\}$ of $\mathcal{V}$ such that $\bigcup_{k=1}^n V_k \in \mathcal{U}$.

A collection $\mathcal{U}$ of open sets is *ω-Scott open* if 1. above and

2'. whenever $\mathcal{V} \subset \mathcal{O}(X)$ is countable and $\bigcup \mathcal{V} \in \mathcal{U}$, there is a finite subset $\{V_1, \ldots, V_n\}$ of $\mathcal{V}$ such that $\bigcup_{k=1}^n V_k \in \mathcal{U}$, or equivalently

2". whenever $U_n \nearrow U_\infty$ with $U_\infty \in \mathcal{U}$, then there exists n such that $U_n \in \mathcal{U}$.

It is straightforward to show that the collection of Scott open sets and ω-Scott open sets are topologies on $\mathcal{O}(X)$, called the *Scott topology* and *ω-Scott topology* respectively.

In general, the ω-Scott topology is finer than the Scott topology. However, the following result shows that the ω-Scott topology coincides with the usual Scott topology if X has a countable pseudobase; in particular, if X is a quotient of a countably-based space.

Proposition 4.10. *Let X be a topological space with a countable pseudobase. Then a set is ω-Scott open if, and only if, it is Scott open.*

The proof is immediate from Theorem 4.2. we henceforth use 2. and 2'. interchangeably for defining Scott open sets when working in a space with a countable pseudobase, in particular, in a quotient of a countably-based space.

The following properties of the ω-Scott topology are elementary.

Lemma 4.11. *Let X be a topological space.*

1. *If $U_n \nearrow U_\infty$ with $U_\infty \supset V$, then $U_n \to V$ in the ω-Scott topology.*
2. *If C is countably compact, then $\{U \mid C \subset U\}$ is open in the ω-Scott topology.*
3. *The set of open sets with the ω-Scott topology is a sequential space.*

Proof.

1. Immediate from the definition of the ω-Scott topology.
2. Suppose $U_n \nearrow U_\infty$ with $C \subset U_\infty$. Then $\{U_n \mid n \in \mathbb{N}\}$ is a countable open cover of C so has a finite subcover $\{U_1, \ldots, U_n\}$. Then $C \subset U_n$.
3. Suppose $\mathcal{W} \subset \mathcal{O}(X)$, and $\mathcal{W}$ is sequentially open in the ω-Scott topology. Then if $U \in \mathcal{W}$ and $V \supset U$, then since $V \to U$ in the ω-Scott topology, we have $\mathcal{W} \ni V$. Further, if $U_n \nearrow U_\infty$ with $U_\infty \in \mathcal{W}$, then $U_n \to U_\infty$ in the ω-Scott topology, so there exists N such that $U_n \in \mathcal{W}$ for $n \geq N$ since $\mathcal{W}$ is sequentially open. Hence the ω-Scott topology is sequential.

□

The following theorem shows that the Scott topology (and by Proposition 4.10 the ω-Scott topology) is an explicit description of the topology on $\mathcal{O}(X)$ induced by the representation of the topological type $\mathcal{S}^{\mathcal{X}}$.

Theorem 4.12. *Let $\mathcal{X} = (X, [\delta])$ be a topological type. A subset of $\mathcal{O}(X)$ is open in the topology induced by the representation $\delta_{\mathcal{X}\to\mathbb{S}}$ if, and only if, it is Scott open.*

Proof. It suffices to show that the Scott topology is the coarsest topology such that the inclusion map $X \times \mathcal{O}(X) \to \mathbb{S}$, $(x, U) \mapsto \top \iff x \in U$ is sequentially continuous.

Suppose $x_n \to x_\infty$ in X, and $U_n \to U_\infty$ in the Scott topology with $x_\infty \in U_\infty$. Since $x_\infty \in U_\infty$, there exists N such that $x_n \in U_\infty$ for all $n \geq N$. Consider $\mathcal{V} = \{V \in \mathcal{O}(X) \mid \{x_N, x_{N+1}, \ldots, x_\infty\} \subset V\}$. The set $\mathcal{V}$ is Scott open, since $\{x_n, x_{n+1}, \ldots, x_\infty\}$ is compact. Hence there exists M such that $U_m \in \mathcal{V}$ for all $m \geq M$. Then for $k \geq \max(M, N)$ we have $\{x_k, \ldots, x_\infty\} \subset U_k$, so $x_k \in U_k$. Thus inclusion is sequentially continuous using the Scott topology on $\mathcal{O}(X)$.

Suppose $\mathcal{T}$ is a sequential topology on $\mathcal{O}(X)$ such that inclusion $X \times \mathcal{O}(X) \to \mathbb{S}$ is sequentially continuous. Suppose $\mathcal{U}$ is Scott-open but not $\mathcal{T}$-open. Since $\mathcal{T}$ is a sequential

topology, $\mathcal{U}$ is not $\mathfrak{T}$-sequentially-open. Hence there exists a sequence $U_n \to_{\mathfrak{T}} U_\infty$ with $U_\infty \in \mathcal{U}$ but $U_n \notin \mathcal{U}$ for any $n \in \mathbb{N}$. Since $\mathcal{U}$ is Scott-open, we have $\bigcup_{n=0}^{\infty} U_n \not\supset U_\infty$. Let $x \in U_\infty \setminus \bigcup_{n=0}^{\infty} U_n$. Then $x \in U_\infty$ but $x \notin U_n$ for any n, contradicting inclusion being sequentially-continuous with respect to $\mathfrak{T}$. Hence the Scott topology is coarser than any other topology making inclusion sequentially-continuous. □

For any quotient of a countably-based space X, we henceforth assume the Scott topology on $\mathcal{O}(X)$, so $\mathcal{O}(\mathcal{O}(X))$ consists of Scott-open subsets of $\mathcal{O}(X)$.

4.4 Sober spaces

We now consider the closed and compact subsets of X, and their relationships with the types $\mathcal{V}(\mathcal{X})$ and $\mathcal{K}(\mathcal{X})$ given by Definition 3.20 as subtypes of $\mathcal{O}(\mathcal{O}(\mathcal{X}))$. In particular, we are interested in spaces for which any function $\mathcal{O}(X) \to \mathbb{S}$ satisfying both (3.3) and (3.4) defines a singleton, and the consequences for $\mathcal{V}(\mathcal{X})$ and $\mathcal{K}(\mathcal{X})$.

Definition 4.13 (Filters and cofilters)**.** A subset $\mathcal{D}$ of $\mathcal{P}(X)$ is *directed* if $S \in \mathcal{D}$ and $T \supset S$ implies $T \in \mathcal{D}$. A directed subset $\mathcal{F}$ of $\mathcal{P}(X)$ is a *filter* if $S_1 \in \mathcal{F} \wedge S_2 \in \mathcal{F} \implies S_1 \cap S_2 \in \mathcal{F}$. A directed subset $\mathcal{F}$ of $\mathcal{P}(X)$ is a *cofilter* if $\emptyset \notin \mathcal{F}$ and $S_1 \cup S_2 \in \mathcal{F} \implies S_1 \in \mathcal{F} \vee S_2 \in \mathcal{F}$. A directed subset of $\mathcal{P}(X)$ is an *ultrafilter* if it is both a filter and a cofilter. A (co)filter which is a Scott-open subset of $\mathcal{O}(X)$ is a *Scott-open (co)filter*.

Remark 4.14. The condition $\emptyset \notin \mathcal{F}$ in the definition of a cofilter could have been omitted, but since for any set $S \subset X$, we have $S \not\bowtie \emptyset$, such a cofilter could never arise as the collection of sets intersecting some set S.

If S is any set, then $\{U \in \mathcal{O}(X) \mid S \subset U\}$ is a filter, and $\{U \in \mathcal{O}(X) \mid S \bowtie U\}$ is a cofilter. If $x \in X$ is a point, then we have $\{x\} \subset U \iff \{x\} \bowtie U \iff x \in U$, so the set of neighbourhoods of x is an ultrafilter, and is easily shown to be open in the Scott topology. Conversely, if X is a T_0 space, and S is a set such that $S \subset U \iff S \bowtie U$, then S is a singleton. Hence the set of Scott-open ultrafilters are "point-like" in the sense that they can only arise from a subset of X as the set of neighbourhoods of a point. However, as the following example shows, not all Scott-open ultrafilters are the set of neighbourhoods of a point.

Example 4.15. Let $X = (\mathbb{Q}, \tau_<)$, where $\tau_< = \{(-\infty, a) \cap \mathbb{Q} \mid a \in \mathbb{R}\}$. In other words, X is the restriction of the reals with the topology of lower convergence to the subspace of the rationals. Take $r \notin \mathbb{Q}$, and let $\mathcal{U} = \{(-\infty, a) \mid a > r\}$. Then $\mathcal{U}$ is Scott-open, since if $U_n = (-\infty, a_n)$ with $U_n \nearrow U_\infty$ with $U_\infty = (-\infty, a_\infty) \in \mathcal{U}$, then $a_n \nearrow a_\infty > r$, so $a_n > r$ for some n. Further, it is clear that $\mathcal{U}$ is an ultrafilter. However, there is no $q \in \mathbb{Q}$ such that $\mathcal{U} = \{(-\infty, a) \mid a > q\}$, so $\mathcal{U}$ is not the set of neighbourhoods of a point.

The topology $\tau_<$ on $\mathbb{Q}$ is equivalent to the topology $\tau_<$ on $\mathbb{R}$, in the sense that there is a bijection between open sets preserving unions and intersections. This shows that a set of points of a topological space cannot be recovered from the lattice of open sets. However, if we view the Scott-open ultrafilters as providing a canonical set of points, we do obtain a unique space. We call a space *sober* if every Scott-open ultrafilter is the closure of a (unique) point. Thus a sober space "has enough points" in the sense that any "point-like" collection of open sets corresponds to a real point.

Definition 4.16 (Sober space)**.** A topological space X is *sober* if any Scott-open ultrafilter of X is the set of neighbourhoods of a point.

In the literature (see [31, 28]), an alternative definition of *sober* space is used. A closed set A is *irreducible* if whenever $A = A_1 \cup A_2$ where A_1, A_2 are closed sets, either $A_1 = A$ or $A_2 = A$. A topological space X is *(classically) sober* if every non-empty irreducible closed subset is the closure of a point.

We now show that the definition of sober space used here coincides with the classical definition.

Theorem 4.17. *Let X be a topological space. Then the following are equivalent:*

1. *Any Scott-open ultrafilter of X is the set of neighbourhoods of a point.*
2. *Any irreducible closed subset of X is the closure of a point.*

Proof. Let A be an irreducible closed set, and $\mathcal{U} = \{U \in \mathcal{O}(X) \mid A \bowtie U\}$. If A is disjoint from $U_1 \cap U_2$, then $A = A \setminus (U_1 \cap U_2) = (A \setminus U_1) \cup (A \setminus U_2)$, so either $A \setminus U_1 = A$ or $A \setminus U_2 = A$ since A is irreducible, and hence A is disjoint from either U_1 or U_2. Hence if $A \bowtie U_1$ and $A \bowtie U_2$, then $A \bowtie (U_1 \cap U_2)$. Hence $\mathcal{U}$ is an ultrafilter. Further, if $A \bowtie \bigcup \mathcal{V}$ with $\bigcup \mathcal{V} \in \mathcal{U}$, there exists $x \in A$ such that $x \in \bigcup \mathcal{V}$, and then $x \in V$ for some $V \in \mathcal{V}$. Hence $\mathcal{U}$ is Scott-open. Therefore, if any open ultrafilter is the neighbourhood filter of a point, we have $\mathcal{U} = \{U \in \mathcal{O}(X) \mid x \in U\}$. Then $\mathrm{cl}\{x\} \bowtie U \iff \{x\} \bowtie U \iff x \in U \iff U \in \mathcal{U} \iff A \bowtie U$, so $\mathrm{cl}\{x\} = A$.

Let $\mathcal{U}$ be a Scott-open ultrafilter, and let $V = \bigcup\{U \mid U \notin \mathcal{U}\}$. Suppose $V \in \mathcal{U}$. Then since $\mathcal{U}$ is Scott open, we have $V_1 \cup \cdots \cup V_k \in \mathcal{U}$ with $V_i \notin \mathcal{U}$ for all i. Taking k minimal, we have $V_1 \cup \cdots \cup V_{k-1} \notin \mathcal{U}$ and $V_k \notin \mathcal{U}$, contradicting $\mathcal{U}$ being an cofilter. Hence $V \notin \mathcal{U}$. Suppose $V = V_1 \cap V_2$ with $V_1, V_2 \neq V$. Then each V_i is a strict superset of V, so $V_1, V_2 \in \mathcal{U}$ by definition of V. But then $V = V_1 \cap V_2 \in \mathcal{U}$ since $\mathcal{U}$ is a filter, again a contradiction. Hence $A = X \setminus V$ is an irreducible closed set. Therefore, if every irreducible closed set is the closure of a point, we have $X \setminus \bigcup\{U \mid U \notin \mathcal{U}\} = \mathrm{cl}\{x\}$. Then $U \in \mathcal{U} \iff U \not\subset V \iff \mathrm{cl}\{x\} \bowtie U \iff x \in U$. □

Remark 4.18. A space is *supersober* if the set of limit points of each ultrafilter on X is either empty or a singleton closure. Theorem 4.17 does not show that any supersober space is sober, since it refers only to Scott-open ultrafilters, whereas the definition of a supersober space refers to arbitrary ultrafilters.

We now give two results which relate the classical closed and compact subsets of X with $\mathcal{V}(X)$ and $\mathcal{K}(X)$ considered as subsets of $\mathcal{O}(\mathcal{O}(X))$.

Theorem 4.19. *Let X be a T_0 topological space. Then there is a bijection between closed subsets of X and Scott-open cofilters in $\mathcal{O}(X)$.*

Proof. Given $A \in \mathcal{A}(X)$, define $\mathcal{F} = \{U \in \mathcal{O}(X) \mid A \bowtie U\}$. Clearly $\mathcal{F}$ is a Scott-open cofilter. Conversely, given a Scott-open cofilter $\mathcal{F}$, define $A = \{x \in X \mid x \in U \in \mathcal{O}(X) \implies U \in \mathcal{F}\}$. If $x \notin A$, there exists $U \in \mathcal{O}(X)$ such that $U \notin \mathcal{F}$. But then $A \cap U = \emptyset$. Hence A is closed. Finally, if $A \in \mathcal{A}(X)$ and $\mathcal{F} = \{U \in \mathcal{O}(X) \mid A \bowtie U\}$, define $B = \{x \in X \mid x \in U \in \mathcal{O}(X) \implies U \in \mathcal{F}\}$. If $x \in A$, then for any $U \in \mathcal{O}(X)$ with $x \in U$, we have $x \in A \cap U$, so $A \bowtie U$ and $U \in \mathcal{F}$, hence $x \in B$. If $x \notin A$, then taking $V = X \setminus A$ we have $x \in V$ but $x \notin \mathcal{F}$, so $x \notin B$. □

Recall that the *saturation* of a set S is $\mathrm{sat}(S) = \bigcap\{U \in \mathcal{O}(X) \mid S \subset U\}$, and a set S is *saturated* if $S = \mathrm{sat}(S)$. The next result is due to Hofmann and Mislove [31], which establishes an isomorphism between compact saturated sets and open filters. We give a direct proof due to Keimel and Paseka [36].

Theorem 4.20 (Hofmann-Mislove)**.** *Let X be a sober space. Then there is a bijection between saturated compact subsets of X and Scott-open filters in $\mathcal{O}(X)$.*

Proof. Let C be a compact subset of X and $\mathcal{U} = \{U \in \mathcal{O}(X) \mid C \subset U\}$. Then $\mathcal{U}$ is open since C is compact, and is clearly a filter.

Let $\mathcal{U}$ be an open filter and $C = \bigcap\mathcal{U}$. Suppose $C \subset V$ but $V \notin \mathcal{U}$. Then by Zorn's lemma, there is an open set W containing V which is maximal among all open sets not in $\mathcal{U}$. If $W = W_1 \cap W_2$ with $W_1, W_2 \neq W$, then we would have $W_1, W_2 \in \mathcal{U}$ and so $W_1 \cap W_2 \in \mathcal{W}$, a contradiction. Hence $X \setminus W$ is irreducible, so $X \setminus W = \mathrm{cl}\{x\}$ for some $x \in X$. Any open set not containing x is therefore a subset of W. Hence $x \in U$ for all $U \in \mathcal{U}$ so $x \in C$, but $x \notin W$ and $C \subset W$, a contradiction. Thus if $C \subset U$ then $U \in \mathcal{U}$.

Now let $\mathcal{V}$ be an open cover of C, so $C \subset \bigcup\mathcal{V}$. Then $\bigcup\mathcal{V} \in \mathcal{U}$, and since $\mathcal{U}$ is open, there exists $V_1, \ldots, V_k \in \mathcal{V}$ such that $\bigcup_{i=1}^k V_i \in \mathcal{U}$, so $C \subset \bigcup_{i=1}^k V_i$. Thus C is compact. □

4.5 Core compact spaces

A weaker version of local compactness is that of *core compactness*. We shall see that any sober core-compact space is locally-compact, which justifies restricting to local compactness in the computability theory. Much of the material in this sections is based on work Escardo and Heckmann [23] and Escardo, Lawson and Simpson [24].

For core-compact spaces we have an alternative representation of open sets; instead of denoting an open set as a countable union of basic open sets, we denote it by sets which are "compactly contained" in U. Recall from Notation 3.34 that $U \Subset V$ if every open cover of V has a finite subcover of U. Note that if X has a countable pseudobase, then $U \Subset V$ if any monotone sequence $V_n \nearrow V$ has $V_n \supset U$ for some n. We first give some elementary properties of the relation $\Subset$.

Lemma 4.21. *Let X be a topological space and $U, V, W \in \mathcal{O}(X)$. Then:*

1. $U \Subset V \implies U \subset V$.
2. $U \subset U' \Subset V' \subset V \implies U \Subset V$.
3. $U \Subset W \wedge V \Subset W \implies U \cup V \Subset W$

Note that it is not true in general that $U \Subset V$ and $U \Subset W$ implies $U \Subset V \cap W$ (though we shall see that this does hold in a sober core-compact space).

Definition 4.22 (Core compact)**.** A topological space X is core-compact if, for every open set V and every $x \in V$, there exists an open set U such that $x \in U$ and $U \Subset V$.

If X is core-compact, then for any $W \in \mathcal{O}(X)$, we have $W = \bigcup\{V \in \mathcal{O}(X) \mid V \Subset W\}$.

The following result is [23, Lemma 5.2]

Lemma 4.23. *Let X be a core-compact space.*

1. *For any $U \in \mathcal{O}(X)$, the set $\uparrow U = \{V \in \mathcal{O}(X) \mid U \Subset V\}$ is Scott open.*

2. *If $\mathcal{Q} \subset \mathcal{O}(X)$ is Scott open and $V \in \mathcal{Q}$ then $U \Subset V$ for some $U \in \mathcal{Q}$.*

3. *The sets $\uparrow U$ for $U \in \mathcal{O}(X)$ form a base of the Scott topology of $\mathcal{O}(X)$.*

Proof.

1. If $V \in \uparrow U$ and $V \subset W$, then $U \Subset V \subset W$ so $U \Subset W$ and hence $W \in \uparrow U$. If $W \in \uparrow U$ then there exists $V \in \mathcal{O}(X)$ with $U \Subset V \Subset W$, so $V \in \uparrow U$. Hence every open cover of a member W of $\uparrow U$ has a finite subcover of a member V of $\uparrow U$.

2. The open set V is the union of the open sets $U \Subset V$, and such open sets are closed under the formation of finite unions.

3. An immediate consequence of (2) and (3).

□

The following lemma shows that the relation $U \Subset W$ can be interpolated.

Lemma 4.24 (Interpolation lemma)**.** *Suppose X is core-compact and $U \Subset W$. Then there exists $V \in \mathcal{O}(X)$ with $U \Subset V \Subset W$.*

Proof. The open set W is the union of the open sets $V \Subset W$, and each open set $V \Subset W$ is the union of the open sets $T \Subset V$. Hence W is the union of the collection $\mathcal{T} = \{T \in \mathcal{O}(X) \mid \exists V \in \mathcal{O}(X),\ T \Subset V \Subset X\}$. For $T_1, T_2 \in \mathcal{T}$, we have $T_1 \cup T_2 \Subset V_1 \cup V_2 \Subset W$, so $\mathcal{T}$ is closed under finite union. Hence there exists $T \in \mathcal{T}$ with $U \subset T$. Taking the corresponding V, we have $U \subset T \Subset V \Subset W$ and hence $U \Subset V \Subset W$. □

The following result shows that we can find a Scott-open filter $\mathcal{V}$ interpolating $U \Subset W$.

Lemma 4.25. *Suppose X is a core-compact space and $U \Subset W$. Then there exists a Scott open filter $\mathcal{V}$ such that $U \subset \bigcap \mathcal{V}$ and $W \in \mathcal{V}$.*

Proof. Recursively construct a sequence of open sets V_n such that $V_0 = V$ and for all n, $U \Subset V_{n+1} \Subset V_n$. Define $\mathcal{V} = \{W \in \mathcal{O}(X) \mid \exists n,\ V_n \subset W\}$. Then if $W \in \mathcal{V}$ and $W' \supset W$, we have $V_n \subset W \subset W'$ for some n, so $W' \in \mathcal{V}$. Further, if $W_1, W_2 \in \mathcal{V}$, then there exists n_1, n_2 such that $V_{n_1} \subset W_1$ and $V_{n_2} \subset W_2$. Let $n = \max(n_1, n_2)$, so $V_n \Subset V_{n_1} \subset W_1$ and $V_n \Subset V_{n_2} \subset W_2$, so $V_n \subset W_1 \cap W_2$, so $W_1 \cap W_2 \in K$. Finally, if $\mathcal{W} \subset \mathcal{O}(X)$ such that $\bigcup \mathcal{W} \in \mathcal{V}$, then there exists n such that $\bigcup \mathcal{W} \supset V_n$, so $V_{n+1} \Subset \bigcup \mathcal{W}$ and hence there is a finite subset $\{W_1, \ldots, W_m\}$ of $\mathcal{W}$ such that $\bigcup_{j=1}^{m} W_j \supset V_{n+1}$. Hence $\mathcal{V}$ is a Scott-open filter. □

The following result shows that core-compactness generalises the classical notion of local compactness, and that for sober space, local compactness is equivalent to core-compactness. The proof that any sober core-compact space is locally-compact is due to Hofmann and Lawson [32].

Theorem 4.26. *Any locally-compact space X is core-compact, and any sober core-compact space is locally-compact.*

Proof. If X is locally-compact, then for any open set V and any $x \in V$, there exists open U and compact K such that $x \in U$, $U \subset K$ and $K \subset V$. Then any open cover of V has a finite subcover of K and hence of U, so $U \Subset V$.

Conversely, suppose X is a sober core-compact space, that W is open and $x \in W$. Since X is core-compact, there exists an open set U such that $x \in U$ and $U \Subset V$. By Lemma 4.25 there exists a Scott open filter $\mathcal{V}$ such that $U \subset \bigcap \mathcal{V}$ and $W \in \mathcal{V}$. By Theorem 4.20, the set $K = \bigcap \mathcal{V}$ is compact. Hence $x \in U \subset K \subset W$. □

The following result is [24, Corollary 6.11]. We give a direct proof.

Theorem 4.27. *If a core-compact space is a quotient of a countably-based space, then it is itself countably-based.*

Proof. It suffices to show the result for a sober core-compact space X, which is therefore locally-compact. By Theorem 4.5, the space X has an admissible quotient representation $\delta : R \to X$ where $R \subset \Sigma^\omega$. Let $x \in X$ and $U \in \mathcal{O}(X)$ be such that $x \in U$. Since X is locally-compact, there exists $V \in \mathcal{O}(X)$ and $K \in \mathcal{K}(X)$ such that $x \in V \subset K \subset U$. Let $(I_0, I_1, \ldots)$ be a list of sets with each I_j in a countable basis or R such that $\delta^{-1}(U) = \bigcup_{j=0}^{\infty} I_j$. Suppose $\delta(\bigcup_{j=0}^{k-1} I_j)$ is not a cover of K for any j. Then there is a sequence $(x_k)_{k\in\mathbb{N}}$ with $x_k \in K \setminus n\bigcup_{j=0}^{k-1} I_j$ for all k. Since K is compact and X is a sequential space, K is sequentially-compact, so (x_k) has a convergent subsequence. Without loss of generality, we can assume that (x_k) is itself convergent and $\lim_{k\to\infty} x_k = x_\infty$.

Since δ is admissible, there is a convergent sequence $(r_k)_{k\in\mathbb{N}}$ with limit r_∞ such that $\delta(r_k) = x_k$ for all $k \in 0, 1, \ldots, \infty$. Since $r_\infty \in I_n$ for some n, there is a finite subset $I_{m_1}, \ldots, I_{m_k}$ of $(I_0, I_1, \ldots)$ such that $V \subset K \subset \bigcup_{j=1}^{k} I_{m_j} \subset U$.

We have shown that if β is a countable base for R, $x \in X$ and $U \in \mathcal{O}(X)$, then there is a finite subset $\{J_1, \ldots, J_k\}$ of β such that $x \in \mathrm{int}(\bigcup_{i=1}^{k} J_i) \subset U$. Hence $\{\mathrm{int}(\bigcup_{i=1}^{k} J_i) \mid J_1, \ldots, J_k \in \beta^*\}$ is a countable base for X. □

Note that the proof given relies crucially on the admissibility of the admissible quotient representation δ, which may be different from the original quotient map q.

The following theorem can be derived from [23, Theorem 5.3]. Its main significance is to show that core-compactness of X implies local-compactness of $\mathcal{O}(X)$.

Theorem 4.28. *Let X be core-compact. Then $\mathcal{O}(X)$ with the Scott topology is locally-compact.*

Proof. Let $U \in \mathcal{O}(X)$ and $\mathcal{W} \subset \mathcal{O}(X)$ be Scott open. By Lemma 4.23, there exists $W \in \mathcal{O}(X)$ such that $U \in \Uparrow W$ and $\Uparrow W \in \mathcal{W}$. Since $U \in \Uparrow W$ means $W \Subset U$, and X is core-compact, by Lemma 4.24 there exists $V \in \mathcal{O}(X)$ such that $W \Subset V \Subset U$. Note that $\uparrow V = \{S \in \mathcal{O}(X) \mid V \subset S\}$ is compact in the Scott topology, since any open cover has a singleton subcover. By Lemma 4.23 the set $\Uparrow V$ is open in the Scott topology. Then $U \in \Uparrow V \subset \uparrow V \Subset \mathcal{W}$ as required. □

Chapter 5

Applications to Dynamic Systems

We now use the computable type theory developed in the previous section to give some results on computable properties of dynamic systems. When considering solutions of nondeterministic systems, we are often interested in function spaces with set-valued types. In this section we now give some applications of the computability type theory to problems in control and systems theory. We focus on three problems, namely the evolution of hybrid systems, computation of reachable and viable sets, and control synthesis. Note that using the computable types developed earlier, many of the results are almost trivial to prove.

5.1 System behaviour

The set of solutions of a dynamic system is the space of continuous functions $\xi : T \to X$, where T is the time domain, and X is the state space. Throughout this section, we will assume that X is effectively Hausdorff and effectively separable. We require the *property of state*, that if ξ and η are solutions with $\xi(s) = \eta(s)$, then there is a solution ζ with $\zeta(t) = \xi(t)$ for $t \leq s$, and $\zeta(t) = \eta(t)$ for $t \geq s$. For an autonomous system, we also require *time-invariance*, that if ξ is a solution and $s \in T$, then the function defined by $\eta(t) = \xi(t + s)$ is also a solution.

For a deterministic system, there is only one trajectory through a given initial state. The solution operator may be represented either as a function $\tilde{\phi} : X \times T \to X$, or as a function $\hat{\phi} : X \to C(T; X)$ taking an initial point x to the trajectory $\xi : T \to X$ with $\xi(0) = x$. By the exponentiation property, the types $\mathcal{X} \times \mathcal{T} \to \mathcal{X}$ and $\mathcal{X} \to \mathcal{C}(\mathcal{T}; \mathcal{X})$ are equivalent.

In a non-deterministic system there may be may different trajectories with the same initial state. In a stochastic system, the behaviour can be described by a function $X \times T \to Pr(X)$, where $Pr(X)$ is the set of probability measures on X. Since we have not considered computable measure theory in this article, we will not consider stochastic systems further.

In general, we do not merely wish to compute the evolution only for a system described by computable data, but for all systems within a class, even if the system data is uncomputable. We therefore express the computability results in terms of both the system description and the initial state.

The simplest class of system to consider is that of a deterministic discrete-time system defined by a continuous function $f : X \to X$ with the update law given by

$x_{n+1} = f(x_n)$. The evolution function $\tilde{\phi}_f : X \times \mathbb{N} \to X$ is given by $\tilde{\phi}_f(x, 0) = x$ and $\tilde{\phi}(x, n+1) = f(\tilde{\phi}(x, n))$. Unsurprisingly, the evolution is computable from f; this is direct from the definition.

Proposition 5.1. *The evolution of a discrete-time system defined by the update rule $x' = f(x)$ where $f \in C(X; X)$, is computable as a function $\mathcal{C}(\mathcal{X}; \mathcal{X}) \times \mathcal{X} \to \mathcal{C}(\mathcal{N}; \mathcal{X})$ (equivalently as a function $\mathcal{C}(\mathcal{X}; \mathcal{X}) \times \mathcal{X} \times \mathcal{N} \to \mathcal{X}$).*

5.2 Nondeterministic systems

Nondeterministic systems frequently arise in control and systems theory as models of systems with control or disturbance inputs. A system with state space X and input space U is described by a function $f : X \times U \to X$ with $x_{n+1} = f(x_n, u_n)$. If the inputs u_n are under the control of the user, we are interested in determining whether there *exists* a trajectory with some given property. Hence we should compute *overt* sets of trajectories. If instead the inputs u_n are disturbances from the external environment, then we are interested in properties which hold for *all* possible trajectories. Hence we should try to obtain *compact* sets of trajectories.

In both cases, we can define a multivalued map $F : X \rightrightarrows X$ describing the evolution by $F(x) = f(x, U)$, and obtain an update law $x_{n+1} \in F(x_n)$. If U is overt, then $F(x)$ is computable from f and U as an overt set, and if U is compact, then $F(x)$ is computable as a compact set. Hence it suffices to consider systems defined by multimaps $F : X \to \mathcal{V}(X)$ and $F : X \to \mathcal{K}(X)$

Note that a *set* of possible solutions is described, with no distinction between what is likely or unlikely; merely between what is possible and impossible.

There are many different ways of representing the solution space. The simplest way of representing the solution space is as the *behaviour* of the system, which is simply the set of all solutions, $\bar{\Phi} \in \mathcal{P}(C(T; X))$. The canonical *solution trajectory* operator is a function $\hat{\Phi} : X \to \mathcal{P}(C(T; X))$ such that $\hat{\Phi}(x) = \{\xi : T \to X \mid \xi \in \bar{\Phi} \text{ and } \xi(0) = x\}$. Another useful representation is in terms of the finite *reachability* operator, $\tilde{\Phi} : X \times T \to \mathcal{P}(X)$ defined as $\tilde{\Phi}(x, t) = \{\xi(t) \mid \xi \in \bar{\Phi} \wedge \xi(0) = x\}$.

The following result shows that while the reachable sets and the set of all solutions can be recovered from the solution to the initial-value problem. Further, unless X is compact, the set of all solutions cannot be represented as an element of $\mathcal{K}(\mathcal{C}(\mathcal{N}; \mathcal{X}))$, whereas $\hat{\Phi} : \mathcal{X} \to \mathcal{V}(\mathcal{C}(\mathcal{N}; \mathcal{X}))$ cannot be recovered from $\bar{\Phi} \in \mathcal{V}(\mathcal{C}(\mathcal{N}; \mathcal{X}))$ This means that in order to study properties of the system, we should compute $\hat{\Phi}$ and not $\bar{\Phi}$ or $\tilde{\Phi}$.

Proposition 5.2.

1. *The type $\mathcal{K}(\mathcal{C}(\mathcal{N}; \mathcal{X}))$ reduces to $\mathcal{C}(\mathcal{X}; \mathcal{K}(\mathcal{C}(\mathcal{N}; \mathcal{X})))$, which reduces to $\mathcal{C}(\mathcal{X}; \mathcal{C}(\mathcal{N}; \mathcal{K}(\mathcal{X})))$.*

2. *The type $\mathcal{C}(\mathcal{X}; \mathcal{V}(\mathcal{C}(\mathcal{N}; \mathcal{X})))$ reduces to both $\mathcal{C}(\mathcal{X}; \mathcal{C}(\mathcal{N}; \mathcal{V}(\mathcal{X})))$ and to $\mathcal{V}(\mathcal{C}(\mathcal{N}; \mathcal{X}))$.*

Proof.

1. Given $\bar{\Phi} \in \mathcal{K}(C(\mathbb{N}; X))$, define $\hat{\Phi} : X \to \mathcal{K}(C(\mathbb{N}; X))$ by $\hat{\Phi}(x) = \bar{\Phi} \cap \{\xi \in C(\mathbb{N}; X) \mid 0 \in \xi^{-1}(\{x\})\}$. The Hausdorff property is required that $\{x\}$ is effectively closed. Given $\hat{\Phi} : X \to \mathcal{K}(C(\mathbb{N}; X))$, define $\tilde{\Phi} : X \times \mathbb{N} \to \mathcal{K}(X)$ by $\tilde{\Phi}(x, n) = \{\xi(n) \mid \xi \in \hat{\Phi}(x)) = (\hat{\Phi}(x))(n)$, which is computable.

2. Given $\hat{\Phi} : X \to \mathcal{V}(C(\mathbb{N}; X))$, define $\bar{\Phi} = \hat{\Phi}(X)$, which is computable since X is assumed to be effectively separable, and hence a computable element of $\mathcal{V}(\mathcal{X})$. Define $\tilde{\Phi} : X \times \mathbb{N} \to \mathcal{V}(X)$ by $\tilde{\Phi}(x, n) = \{\xi(n) \mid \xi \in \hat{\Phi}(x)) = (\hat{\Phi}(x))(n)$. □

We now consider the computability of the solution operator $\hat{\Phi}$. We begin with a generally useful result on multivalued maps.

Theorem 5.3. *The following types are computably equivalent:*

1. $F : \mathcal{X} \to \mathcal{V}(\mathcal{Y})$, $F^{-1} : \mathcal{O}(\mathcal{Y}) \to \mathcal{O}(\mathcal{X})$ *and* $F : \mathcal{V}(\mathcal{X}) \to \mathcal{V}(\mathcal{Y})$.
2. $F : \mathcal{X} \to \mathcal{K}(\mathcal{Y})$, $F^{-1} : \mathcal{A}(\mathcal{Y}) \to \mathcal{A}(\mathcal{X})$ *and* $F : \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{Y})$.

Proof.

1. $x \in F^{-1}(U) \iff F(x) \bowtie U$; $F(B) \bowtie U \iff B \bowtie F^{-1}(U)$.
2. $x \notin F^{-1}(A) \iff F(x) \subset (X \setminus A)$; $F(C) \subset U \iff C \subset X \setminus F^{-1}(X \setminus U)$. □

We can use these properties to compute the forward-time evolution of discrete-time nondeterministic systems.

Corollary 5.4. *The behaviour of a nondeterministic discrete-time system F is computable in the following cases:*

1. *If $F : \mathcal{X} \to \mathcal{V}(\mathcal{X})$, then $\hat{\Phi} : \mathcal{X} \to \mathcal{V}(\mathcal{C}(\mathcal{N}, \mathcal{X}))$ is computable from F.*
2. *If $F : \mathcal{X} \to \mathcal{K}(\mathcal{X})$, then $\hat{\Phi} : \mathcal{X} \to \mathcal{K}(\mathcal{C}(\mathcal{N}, \mathcal{X}))$ is computable from F.*

5.3 Differential systems

We now consider the computability of systems defined by differential equations or differential inclusions. For simplicity, we assume that X is a Euclidean space $\mathbb{R}^n$, though these results also extend to differential manifolds and locally-compact Banach spaces. To prove the results of this section we need to go back to first principles to solve the differential systems; in particular, we need to resort to the classical Arzela-Ascoli and Michael theorems to assert the existence of solutions.

Theorem 5.5. *Let $f : X \to X$ be locally-Lipschitz continuous. Then the solution operator $\hat{\phi}_f$ of $\dot{x} = f(x)$ is computable $\mathcal{C}(\mathcal{X}; \mathcal{X}) \times \mathcal{X} \to \mathcal{C}(\mathcal{R}, \mathcal{X})$.*

The proof is essentially standard [19], though is too long to include here. A simple proof can be found in [16]. Note that we can weaken the locally-Lipschitz condition to simply requiring uniqueness of solutions [44].

We now turn to nondeterministic differential systems as defined by differential inclusions $\dot{x} \in F(x)$. For an introduction to differential inclusions, see [2]. Following the well-known solution concept of Filippov [25], we may first need to compute the convex hull of the right-hand side. The continuous case was first proved in [43], but easily splits into the lower- and upper-semicontinuous cases. The lower-semicontinuity with the one-sided Lipschitz condition was proved in [26]. Full proofs can be found in [17].

Theorem 5.6.

1. *Let F be one-sided locally-Lipschitz lower-semicontinuous with closed convex values. Then the solution operator $(F, x) \mapsto \hat{\Phi}_F(x)$ of $\dot{x} \in F(x)$ is computable $\mathcal{C}(\mathcal{X}; \mathcal{V}(\mathcal{X})) \times \mathcal{X} \to \mathcal{V}(\mathcal{C}(\mathcal{R}, \mathcal{X}))$.*

2. *Let F be upper-semicontinuous with compact convex values. Then the solution operator of $\dot{x} \in F(x)$ is computable $\mathcal{C}(\mathcal{X}; \mathcal{K}(\mathcal{X})) \times \mathcal{X} \to \mathcal{K}(\mathcal{C}(\mathcal{R}, \mathcal{X}))$.*

5.4 Evolution of hybrid systems

A hybrid system comprises continuous evolution interspersed with discrete jumps. Since from the results of Sections 5.2 and 5.3, the evolution of continuous-time and discrete-time systems are computable, we focus on the problem of *event detection*. An event occurs whenever the state of the system enters a *guard set* G. Consider the case of a hybrid system defined as the tuple (X, F, G, R), where $\dot{x} \in F(x)$ defines the continuous dynamics, the guard set G is entered when $g(x) \geq 0$, and the reset is given by $x' \in R(x)$. Let $D = \{x \mid g(x) \leq 0\}$.

Suppose $\xi(t)$ is a continuous trajectory with $g(\xi(0)) < 0$, and $g(\xi(t)) > 0$ for some $t > 0$. Then clearly the trajectory ξ crosses the guard set at some time. We define the *hitting time* τ_h by $\tau_h(\xi) = \min\{t \in \mathbb{R} \mid g(\xi(t)) = 0\}$ and the *crossing time* τ_c as $\tau_c(\xi) = \inf\{t \in \mathbb{R} \mid g(\xi(t)) > 0\}$. Clearly $\tau_h(\xi) \leq \tau_c(\xi)$, but the two need not be equal, in general. If $\tau_h(\xi) = \tau_c(\xi)$, then we say that ξ crosses g *transversely* at $\tau = \tau_h(\xi)$. Otherwise, it may be the case that $\xi(t)$ slides along the guard set G between τ_h and τ_c, or touches G and re-enters D before later crossing G. We define the *touching time set* as $\tau(\xi) = \{t \in \mathbb{R} \mid g(\xi(t)) = 0 \wedge \forall s \leq t,\ g(\xi(s)) \leq 0\}$.

Lemma 5.7. *The set of trajectories ξ with $g(\xi(0)) < 0$ and $g(\xi(t)) > 0$ for some $t > 0$ is computable in $\mathcal{O}(\mathcal{C}(\mathcal{R}; \mathcal{X}))$.*

Theorem 5.8. *The touching time set $\tau(g, \xi)$ is computable as a function $\mathcal{C}(\mathcal{X}, \mathcal{R}) \times \mathcal{C}(\mathcal{R}, \mathcal{X}) \to \mathcal{A}(\mathcal{R})$. Further, if $g(\xi(t)) > 0$ for some $t > 0$, then $\tau(g, \xi)$ can be computed in $\mathcal{K}(\mathcal{R})$.*

Proof. Define $\gamma_{\xi,g}(t) = g(\xi(t))$ and $\mu_{\xi,g}(t) = \sup\{g(\xi(s)) \mid s \in [0, t]\}$ which is a computable function (see [53]). Then $\tau(g, \xi) = \gamma_{\xi,g}^{-1}(\{0\}) \cap \mu_{\xi,g}^{-1}((-\infty, 0])$ so is computable. If $g(\xi(t)) > 0$, then $\tau(g, \xi) = \tau(g, \xi) \cap [0, t]$, so is effectively compact. □

Theorem 5.9. *Consider a hybrid system where Φ_F is a compact-valued multiflow, and R is compact-valued. Then set of points $\Psi(X_0)$ reachable after the first event is computable as a compact set.*

Proof. The set of points reachable after the first event of a continuous solution ξ is $R(\xi(\tau(g, \xi))$, which is computable in $\mathcal{C}(\mathcal{X})$ from $\xi \in \mathcal{C}(\mathcal{R}; \mathcal{X})$. The set of trajectories with initial condition X_0 is $\Phi_F(X_0)$, so is computable in $\mathcal{C}(\mathcal{C}(\mathcal{R}; \mathcal{X}))$. Then $\Psi(X_0)$ is the union of $R(\xi(\tau(g, \xi))$ for ξ in the compact set $\Phi_F(X_0)$, so is computable in $\mathcal{C}(\mathcal{X})$. □

Unfortunately, the set of points reachable after the first event is not computable as an overt set, since the crossing time is not computable as an overt set. It turns out that event detection is easier in the context of backwards reachability

Theorem 5.10. *Consider a hybrid system where Φ_F is an overt multiflow and R is overt-valued. Let V be an open set. Define $\Psi^{-1}(V)$ to be the set of points for which there is a solution for which the state is in V immediately after the first jump. Then $\Psi^{-1}(V)$ is computable as an open set.*

Proof. The trajectory ξ crosses G in $R^{-1}(V)$ if $\xi(\tau(g,\xi)) \subset V$. Since $R : \mathcal{X} \to \mathcal{V}(\mathcal{X})$, $U = R^{-1}(V)$ is computable in $\mathcal{O}(X)$. Since $\tau(g,\xi)$ is compact and ξ is continuous, $W = \{\xi \mid \tau(g,\xi) \subset U\}$ is computable in $\mathcal{O}(\mathcal{C}(\mathcal{R};\mathcal{X}))$. Since $\Phi : \mathcal{X} \to \mathcal{V}(\mathcal{C}(\mathcal{R};\mathcal{X}))$, $\Phi^{-1}(W)$ is computable in $\mathcal{O}(\mathcal{X})$. We have $\Psi^{-1}(V) = \{x \mid \exists \xi \in \Phi_F(x) \text{ s.t. } \xi(\tau(g,\xi)) \subset R^{-1}\}) = \Phi_F^{-1}(\{\xi(\tau(g,\xi)) \subset R^{-1}\})$, so $\Psi^{-1}(V)$ is computable in $\mathcal{O}(\mathcal{X})$. □

5.5 Reachable and viable sets

We now apply the results of Section 5.2 to prove computability of some infinite-time operators in discrete-time dynamical systems. Computability of reachable sets was considered in [13]. Computability of the viability kernel was considered in [45]. Similar results for upper-semicontinuous hybrid systems have been obtained in [3, 29].

In this section, we will assume that X is effectively locally compact, so there is a recursively enumerable set $\mathcal{D}$ of pairs $(U_n, C_n) \in \mathcal{O} \times \mathcal{K}$ such that $U_n \subset C_n$ for all n, and for any compact K and open V with $K \subset V$ there exists n such that $K \subset U_n$ and $C_n \subset V$.

We define the *reachable set* of a system $F : X \to \mathcal{P}(X)$ with initial state set X_0 as

$$\text{reach}(F, X_0) = \{x \in X \mid \exists \text{ solution } \xi \text{ and } t \in T \text{ with } \xi(0) \in X_0 \text{ and } \xi(t) = x\}.$$

Theorem 5.11. *The reachable set operator* reach *is computable as a function $\mathcal{C}(\mathcal{X};\mathcal{V}(\mathcal{X})) \times \mathcal{V}(\mathcal{X}) \to \mathcal{V}(\mathcal{X})$, but not as a function $\mathcal{C}(\mathcal{X};\mathcal{K}(\mathcal{X})) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$.*

Proof. We can write $\text{reach}(F, X_0) = \bigcup_{i=0}^{\infty} R_i$, where $R_0 = X_0$ and $R_{i+1} = R_i \cup F(R_i)$. Then $\text{reach} : C(X;\mathcal{V}(X)) \times \mathcal{V}(X) \to \mathcal{V}(X)$ is computable since all operations are computable. However, reach fails to be computable from $C(\mathcal{X};\mathcal{K}(\mathcal{X})) \times \mathcal{K}(\mathcal{X})$ to $\mathcal{K}(\mathcal{X})$ even if X is compact since it is easy to show that reach is not upper-semicontinuous in parameters, as in Example 5.12. □

Example 5.12. Consider the system $f : \mathbb{R} \to \mathbb{R}$ defined by $f_\epsilon(x) = \epsilon + x + x^2 - x^4$. Then $\text{reach}(f_0, \{-1/2\}) \subset [-1, 0]$, but $\text{reach}(f_\epsilon, \{-1/2\}) \not\subset [-1, 1/2]$ for any $\epsilon > 0$.

We define the *chain-reachable set* of F as limit of all ϵ-orbits, or equivalently as

$$\text{chainreach}(F, X_0) = \bigcap \{U \in \mathcal{O}(X) | \, \text{cl}(U) \text{ is compact, and } X_0 \cup F(\text{cl}(U)) \subset U\}.$$

Theorem 5.13. *If* $\text{chainreach}(F, X_0)$ *is bounded, then* $\text{chainreach} : \mathcal{C}(\mathcal{X};\mathcal{K}(\mathcal{X})) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$ *is computable, and is the optimal $\mathcal{K}(\mathcal{X})$-computable over-approximation to* reach.

Proof. It is clear that $\text{chainreach}(F, X_0) = \bigcap \{K \mid (K, V) \in \mathcal{D} \text{ and } X_0 \cup F(K) \subset V\}$, proving computability. The proof of optimality involves considering perturbations, and can be found in [14]. □

The *viability kernel* of a multivalued map F and a set S is given by

$$\operatorname{viab}(F, A) = \{x \in X \mid \exists \text{ solution } \xi \text{ s.t. } x = \xi(0) \text{ and } \forall t \in T,\ x(t) \in S\}.$$

Theorem 5.14. *The viability kernel operator* $\operatorname{viab}(F, S)$ *is computable as a function* $\mathcal{C}(\mathcal{X}, \mathcal{K}(\mathcal{X})) \times \mathcal{K}(\mathcal{X}) \to \mathcal{K}(\mathcal{X})$.

Proof. Write $\operatorname{viab}(F, A) = \bigcap_{i=0}^{\infty} S_i$, where $S_0 = S$ and $S_{i+1} = S_i \cap F^{-1}(S_i)$. □

The viability kernel is not computable as an open or overt set. However, we can define a *robust viability kernel*

$$\operatorname{rob}\operatorname{viab}(F, S) = \{C \in \mathcal{K}(X) \mid C \subset S \cap F^{-1}(\operatorname{int}(C))\}.$$

Theorem 5.15. *The robust viability kernel operator* $\operatorname{rob}\operatorname{viab}(F, S)$ *is computable as a function* $\mathcal{C}(\mathcal{X}, \mathcal{V}(\mathcal{X})) \times \mathcal{O}(\mathcal{X}) \to \mathcal{O}(\mathcal{X})$.

Proof. Write $\operatorname{rob}\operatorname{viab}(F, S) = \bigcup\{V \mid (K, V) \in \mathcal{D} \text{ and } K \subset S \cap F^{-1}(V)\}$. □

5.6 Control synthesis

A *noisy control system* with state space X, input space U and noise space V is a function $f : X \times U \times V \to X$. We assume that U is an overt space and V a compact space, and define $F_U : X \to \mathcal{P}(X \times U)$, $F_U(x) = \{(x, u) \mid u \in U\}$, and $F_V : X \times U \to \mathcal{P}(X)$ by $F_V(x, u) = \{f(x, u, v) \mid v \in V\}$.

The *controllable set* of $\operatorname{ctrl}(f, T, S)$ with target set T and safe set S is determined recursively by $T_0 = T \cap S$ and $T_{i+1} = T_i \cup \{x \in X \mid \exists u \in U,\ \forall v \in V,\ f(x, u, v) \in T_i\} \cap S$.

Theorem 5.16. *The controllable set operator* $\operatorname{ctrl} : \mathcal{C}(\mathcal{X}, \mathcal{U}, \mathcal{V}; \mathcal{X}) \times \mathcal{O}(\mathcal{X}) \times \mathcal{O}(\mathcal{X}) \to \mathcal{O}(\mathcal{X})$ *is computable.*

Proof. The multivalued functions $F_U : X \to \mathcal{V}(X \times U)$ and $F_V : (X \times U) \to \mathcal{K}(X)$ can be computed from f, U and V. Write $T_{i+1} = T_i \cup (F_U^{-1}(F_V^{\Leftarrow}(T_i)) \cap S)$ and $C = \bigcup_{i=0}^{\infty} T_i$. □

Classically, a *state feedback control law* is a function $g : X \to U$. There are systems which are controllable by a discontinuous state feedback, but not a continuous feedback, so we cannot hope to solve a general control problem by computing a continuous state feedback. One solution to this difficulty is to first compute a *supervisor*, which is a multivalued function $G : X \rightrightarrows U$ such that taking inputs $u_n \in G(x_n)$ always gives a solution. If G is open-valued, we can then construct a deterministic feedback law by taking $g(x) \in G(x)$.

Theorem 5.17. *If* $\operatorname{ctrl}(f, T, S) \supset X_0$*, then there is an supervisor* $G : X \to O(U)$ *which can be computed from* f, T, S.

Proof. From $T_{i+1} = T_i \cup F_U^{-1}(F_V^{\Leftarrow}(T_i))$ and $X_0 \subset T_n$, we can find open B_i, C_i such that $\overline{C}_0 \subset T$, $X_0 \subset \bigcup_{i=0}^{n} B_i$, and for all i, $\overline{C}_i \subset B_i$ and $B_{i+1} \subset F_U^{-1}(F_V^{\Leftarrow}(C_i)$. For $x \in B_i \setminus \bigcup_{j=0}^{i-1} \overline{C}_j$, define $G_i(x) = \{u \in U \mid (x, u) \in F_V^{\Leftarrow}(C_{i-1})\}$, and define $G(x) = \bigcup\{G_i(x) \mid x \in B_i \setminus \bigcup_{j=0}^{i-1} \overline{C}_j\}$. □

A noisy control system with *partial observations* with output space Y and measurement noise space W is defined by functions $f : X \times U \times V \to X$ and $h : X \times W \to Y$. We assume W is compact, and define $H : X \to \mathcal{K}(Y)$ by $H(x) = \{h(x, w) \mid w \in W\}$. An *observer* for the system takes values $\hat{X} \in \mathcal{K}(X)$, with with initialisation $\hat{X}_0 = X_0 \cap H^{-1}(y_0)$ and update rule $\hat{X}_{n+1} = \hat{F}(\hat{X}_n, u_n, y_{n+1}) = F_V(\hat{X}_n, u_n) \cap H^{-1}(y_{n+1})$.

In order to guarantee control to the target set within the safe set, we require $\hat{X}_n \subset T$ for some n, and $\hat{X}_i \subset S$ for all $i \leq n$. We can therefore define sets $T_i \subset \mathcal{K}(X)$ by $T_0 = \{C \in \mathcal{K}(S) \mid C \subset T\}$ and $T_{i+1} = \{C \in \mathcal{K}(S) \mid \exists u \in U,\ \forall y \in Y,\ \hat{F}(C, u, y) \in T_i\}$. Note that $\{C \in \mathcal{K}(X) \mid C \subset U\}$ is open in $\mathcal{K}(X)$ for U open in X. Then T_{i+1} is the set of all state estimates for which we can choose an input such that the next state estimate is guaranteed to be in T_i. The problem is solvable if, and only if, there exists n such that $\bigcup_{i=1}^{n} T_i \supset \{C \in \mathcal{K}(X) \mid \exists y \in Y \text{ s.t. } C = X_0 \cap H^{-1}(y)\}$.

Define the controllable set operator ctrl by $X_0 \in \mathrm{ctrl}(f, h, S, T)$ if X_0 is controllable into T inside S under the system (f, h).

Theorem 5.18. *The controllable set operator is computable* $\mathrm{ctrl} : \mathcal{C}(\mathcal{X}, \mathcal{U}, \mathcal{V}; \mathcal{X}) \times \mathcal{C}(\mathcal{X}, \mathcal{W}; \mathcal{Y}) \times \mathcal{O}(\mathcal{X}) \times \mathcal{O}(\mathcal{X}) \to \mathcal{O}(\mathcal{K})$.

Proof. Define a function $\hat{H} : \mathcal{K}(X) \times Y \to \mathcal{K}(X)$ by $\hat{H}(C, y) = C \cap H^{-1}(Y)$. Then $\hat{H}$ is computable. Further, either Y is compact or $\hat{H}(C, Y) = \hat{H}(C, H(C)) \cup \emptyset$, so $\hat{H}(C, Y) = \{\hat{H}(C, y) \mid y \in Y\} \in \mathcal{K}(\mathcal{K}(X))$. For $X_0 \in \mathcal{K}(X)$, define $\hat{X}_0 = \{C \in \mathcal{K}(X) \mid \exists y \in Y \text{ s.t. } C = X_0 \cap H^{-1}(y)\}$. Then $\hat{X}_0 = \hat{H}(X_0)$ is computable in $\mathcal{K}(\mathcal{K}(\mathcal{X}))$.

The set $T_0 = \{C \in \mathcal{K}(S) \mid C \subset T\} = \{C \in \mathcal{K}(X) \mid C \subset S \cap T\}$ is computable in $\mathcal{O}(\mathcal{K}(X))$ directly from S and T. We now show that T_{i+1} is computable in $\mathcal{O}(\mathcal{K}(X))$. Let $V_i = \{C \in \mathcal{K}(X) \mid \forall y \in Y,\ C \cap H^{-1}(y) \in T_i\}$ and $U_{i+1} = \{(C, u) \in \mathcal{K}(X) \times U \mid F_V(C, u) \in V_i\}$, so $T_{i+1} = \{C \in \mathcal{K}(S) \mid \exists u \in U,\ (C, u) \in U_{i+1}\}$. Then $V_i = \{C \in \mathcal{K}(X) \mid \hat{H}(C, Y) \subset T_i\}$, and since $\hat{H}(C, Y) \in \mathcal{K}(\mathcal{K}(X))$, we have V_i computable in $\mathcal{O}(\mathcal{K}(\mathcal{X}))$. Then $U_{i+1} = F_V^{-1}(V_i)$, so is computable in $\mathcal{O}(\mathcal{K}(\mathcal{X}) \times \mathcal{U})$. Defining $\tilde{F}_U(C) = C \times U \in \mathcal{V}(\mathcal{K}(X) \times U)$, we obtain $\{C \in \mathcal{K}(X) \mid \exists u \in U,\ (C, u) \in U_{i+1}\} = \tilde{F}_U^{-1}(U_{i+1})$, so $T_{i+1} = \tilde{F}_U^{-1}(U_{i+1} \cap \{C \in \mathcal{K}(X) \mid C \subset S\}$ is computable in $\mathcal{O}(\mathcal{K}(\mathcal{X}))$. □

In a similar way to the case of state feedback, we can construct a supervisor $G : \mathcal{K}(\mathcal{X}) \to \mathcal{O}(\mathcal{U})$ solving the control problem. However, in order to realise the control strategy, we need to replace the state estimator update $\hat{F}$ by a finite automaton approximation, and the supervisor G by a function on this automaton. This process is not as straightforward as designing a state feedback control law since the space $\mathcal{K}(X)$ is not Hausdorff. Details are given in [15].

Bibliography

[1] Oliver Aberth. *Introduction to precise numerical methods*. Elsevier/Academic Press, Amsterdam, second edition, 2007. With 1 CD-ROM (Windows).

[2] Jean-Pierre Aubin and Arrigo Cellina. *Differential inclusions*, volume 264 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1984.

[3] Jean-Pierre Aubin, John Lygeros, Marc Quincampoix, and Shankar Sastry. Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Trans. Automatic Control*, 47(1):2–20, 2002.

[4] Ingo Battenfeld. *Topological Domain Theory*. PhD thesis, University of Edinburgh, 2008.

[5] Ingo Battenfeld, Matthias Schröder, and Alex Simpson. A convenient category of domains. In *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin*, volume 172 of *Electron. Notes Theor. Comput. Sci.*, pages 69–99. Elsevier, Amsterdam, 2007.

[6] Andrej Bauer. *The Realizability Approach to Computable Analysis and Topology*. PhD thesis, Carnegie Mellon University, 2000.

[7] E. J. Beggs and J. V. Tucker. Computations via Newtonian and relativistic kinematic systems. *Appl. Math. Comput.*, 215(4):1311–1322, 2009.

[8] Errett Bishop and Douglas Bridges. *Constructive analysis*, volume 279 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1985.

[9] Jens Blanck. Domain representations of topological spaces. *Theoret. Comput. Sci.*, 247(1-2):229–255, 2000.

[10] Vasco Brattka. *Recursive and Computable Operations over Topological Structures*. PhD thesis, FernUniversität Hagen, 1998.

[11] Vasco Brattka and Gero Presser. Computability on subsets of metric spaces. *Theoretical Comp. Sci.*, 305:43–76, 2003.

[12] Maria Manuel Clementino, Eraldo Giuli, and Walter Tholen. A functional approach to general topology. In *Categorical foundations*, volume 97 of *Encyclopedia Math. Appl.*, pages 103–163. Cambridge Univ. Press, Cambridge, 2004.

[13] Pieter Collins. Continuity and computability of reachable sets. *Theoret. Comput. Sci.*, 341(1-3):162–195, 2005.

[14] Pieter Collins. Optimal semicomputable approximations to reachable and invariant sets. *Theory Comput. Syst.*, 41(1):33–48, 2007.

[15] Pieter Collins. Computability of controllers for discrete-time semicontinuous systems. In *Proceedings of the 18th International Symposium on the Mathematical Theory of Networks and Systems, Blacksburg, Virginia*, 2008.

[16] Pieter Collins and Daniel Graça. Effective computability of solutions of ordinary differential equations — the thousand monkeys approach. In *Proceedings of the 5th International Conference on Computability and Complexity in Analysis (CCA'08)*, Electronic Notes in Theoretical Computer Science, pages 53–64. Elsevier, Amsterdam, The Netherlands, 2008.

[17] Pieter Collins and Daniel S. Graça. Effective computability of solutions of differential inclusions: the ten thousand monkeys approach. *J.UCS*, 15(6):1162–1185, 2009.

[18] Fredrik Dahlgren. Partial continuous functions and admissible domain representations. *J. Logic Comput.*, 17(6):1063–1081, 2007.

[19] J. W. Daniel and R. E. Moore. *Computation and Theory in Ordinary Differential Equations*. W. H. Freeman & Co Ltd, 1970.

[20] Abbas Edalat. A computable approach to measure and integration theory. *Inform. and Comput.*, 207(5):642–659, 2009.

[21] Abbas Edalat and Philipp Sünderhauf. A domain-theoretic approach to computability on the real line. *Theoret. Comput. Sci.*, 210(1):73–98, 1999.

[22] Martín Escardó. Synthetic topology of data types and classical spaces. *Electronic Notes in Theoretical Computer Science*, 87:21–156, 2004.

[23] Martín Escardó and Reinhold Heckmann. Topologies on spaces of continuous functions. In *Proceedings of the 16th Summer Conference on General Topology and its Applications (New York)*, volume 26, pages 545–564, 2001/02.

[24] Martín Escardó, Jimmie Lawson, and Alex Simpson. Comparing Cartesian closed categories of (core) compactly generated spaces. *Topology Appl.*, 143(1-3):105–145, 2004.

[25] Alekseĭ F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Mathematics and Its Applications. Kluwer Academic Publishers, Dordrecht, 1988.

[26] Grzegorz Gabor. Continuous selection of the solution map for one-sided Lipschitz differential inclusions. *Nonlinear Anal.*, 66(5):1185–1197, 2007.

[27] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *Continuous lattices and domains*, volume 93 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2003.

[28] Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. *A compendium of continuous lattices*. Springer-Verlag, Berlin, 1980.

[29] R. Goebel and A. R. Teel. Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica J. IFAC*, 42(4):573–587, 2006.

[30] Eldon Hansen and G. William Walster. *Global optimization using interval analysis*, volume 264 of *Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker Inc., New York, 2004. Second edition, revised and expanded, With a foreword by Ramon Moore.

[31] K. Hofmann and M. Mislove. Local compactness and continuous lattices. In *Continuous Lattices*, volume 871 of *Lect. Notes Math.*, pages 209–248, 1981.

[32] Karl H. Hofmann and Jimmie D. Lawson. The spectral theory of distributive continuous lattices. *Trans. Amer. Math. Soc.*, 246:285–310, 1978.

[33] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Éric Walter. *Applied interval analysis*. Springer-Verlag London Ltd., London, 2001. With examples in parameter and state estimation, robust control and robotics, With 1 CD-ROM (UNIX, Sun Solaris).

[34] Peter T. Johnstone. *Sketches of an elephant: a topos theory compendium. Vol. 1*, volume 43 of *Oxford Logic Guides*. The Clarendon Press Oxford University Press, New York, 2002.

[35] Peter T. Johnstone. *Sketches of an elephant: a topos theory compendium. Vol. 2*, volume 44 of *Oxford Logic Guides*. The Clarendon Press Oxford University Press, Oxford, 2002.

[36] Klaus Keimel and Jan Paseka. A direct proof of the Hofmann-Mislove theorem. *Proc. Amer. Math. Soc.*, 120(1):301–303, 1994.

[37] S. C. Kleene. General recursive functions of natural numbers. *Math. Ann.*, 112(1):727–742, 1936.

[38] Ker-I Ko. *Complexity theory of real functions*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1991.

[39] Boris A. Kushner. Markov's constructive analysis; a participant's view. *Theoret. Comput. Sci.*, 219(1-2):267–285, 1999. Computability and complexity in analysis (Castle Dagstuhl, 1997).

[40] Per Martin-Löf. *Intuitionistic type theory*. Bibliopolis, Napoli, 1984.

[41] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to interval analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[42] Marian B. Pour-El and J. Ian Richards. *Computability in analysis and physics*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1989.

[43] Anuj Puri, Pravin Varaiya, and Vivek Borkar. Epsilon-approximation of differential inclusions. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *LNCS*, pages 362–376, Berlin, 1996. Springer.

[44] Keijo Ruohonen. An effective Cauchy-Peano existence theorem for unique solutions. *Int. J. Found. Comput. Sci.*, 7(2):151–160, 1996.

[45] Patrick Saint-Pierre. Approximation of the viability kernel. *Appl. Math. Optim.*, 29(2):187–209, 1994.

[46] Matthias Schröder. *Admissible Representations for Continuous Computations*. PhD thesis, Fachbereich Informatik, FernUniversität Hagen, 2002.

[47] Matthias Schröder. Extended admissibility. *Theoret. Comput. Sci.*, 284(2):519–538, 2002. Computability and complexity in analysis (Castle Dagstuhl, 1999).

[48] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, 1996.

[49] Paul Taylor. A lambda calculus for real analysis. http://www.monad.me.uk/, 2008.

[50] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc. (2)*, 43(1):230–265, 1937.

[51] Walter P. van Stigt. *Brouwer's intuitionism*, volume 2 of *Studies in the History and Philosophy of Mathematics*. North-Holland Publishing Co., Amsterdam, 1990.

[52] Steven Vickers. *Topology via logic*, volume 5 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1989.

[53] Klaus Weihrauch. *Computable analysis*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2000. An introduction.

[54] Klaus Weihrauch and Tanja Grubba. Elementary computable topology. *J.UCS*, 15(6):1381–1422, 2009.

[55] Martin Ziegler. Physically-relativized Church-Turing Hypotheses: Physical foundations of computing and complexity theory of computational physics. *Appl. Math. Comput.*, 215(4):1431–1447, 2009.

Centrum Wiskunde & Informatica (CWI) is the national research institute for mathematics and computer science in the Netherlands. The institute's strategy is to concentrate research on four broad, societally relevant themes: earth and life sciences, the data explosion, societal logistics and software as service.

Centrum Wiskunde & Informatica (CWI) is het nationale onderzoeksinstituut op het gebied van wiskunde en informatica. De strategie van het instituut concentreert zich op vier maatschappelijk relevante onderzoeksthema's: aard- en levenswetenschappen, de data-explosie, maatschappelijke logistiek en software als service.

Bezoekadres:
Science Park 123
Amsterdam

Postadres:
Postbus 94079, 1090 GB Amsterdam
Telefoon 020 592 93 33
Fax 020 592 41 99
info@cwi.nl
www.cwi.nl

CWI

Centrum Wiskunde & Informatica