Computability of Operators on Continuous and Discrete Time Streams

J.V. Tucker

Department of Computer Science,
Swansea University, Singleton Park, Swansea SA2 8PP, Wales
J.V.Tucker@swansea.ac.uk

J.I. Zucker

Department of Computing and Software,
McMaster University, Hamilton, Ontario L8S 4K1, Canada
zucker@mcmaster.ca

Abstract

A stream is a sequence of data indexed by time. The behaviour of natural and artificial systems can be modelled by streams and stream transformations. There are two distinct types of data stream: streams based on continuous time and streams based on discrete time. Having investigated case studies of both kinds separately, we have begun to combine their study in a unified theory of stream transformers, specified by equations. Using only the standard mathematical techniques of topology, we have proved continuity properties of stream transformers. Here, in this sequel, we analyse their computability. We use the theory of computable functions on algebras to design two distinct methods for defining computability on continuous and discrete time streams of data from a complete metric space. One is based upon low-level concrete representations, specifically enumerations, and the other is based upon high-level programming, specifically **while** programs, over abstract data types. We analyse when these methods are equivalent. We demonstrate the use of the methods by showing the computability of an analog computing system. We discuss the idea that continuity and computability are important for models of physical systems to be "well-posed".

Key words and phrases: abstract computability, analog computing, concrete computability, continuous streams, discrete streams, enumerations, fixed points, many-sorted algebras, topological algebras, complete metric spaces, stream operators, synchronous concurrent algorithms, 'while' programs

1 Introduction

1.1 On generalising computability theory and streams

The generalisation of computability theory to arbitrary data types aims at models analysing the computability of functions $f\colon A \to B$ on any sets A and B. At the origins of the theory are countable sets of discrete data that can be faithfully coded by strings and natural numbers, since our understanding of algorithms and computability is founded upon the data types of strings and natural numbers.

However, many applications require us to compute on uncountable sets of continuous data, including: real and complex numbers; bit streams, signals and waveforms; spatial objects, scenes and animations; scalar and vector fields; and probability distributions. The importance of such data types motivates generalisations of computability theory. In our view,

> *Each data type requires its own computability theory, one that is capable of (i) analysing the special nature of computation with the data and (ii) illuminating the use of the data in applications.*

A generalisation of computability theory to arbitrary data types should expand and deepen our understanding of finite computation, and be a practical tool for developing useful computability theories for particular data types.

Data distributed in space and time are to be found everywhere. For the examples of continuous data mentioned, we can expect the custom-made computability theories to have much in common mathematically. For example, they would involve the approximation of functions on topological, metric, normed and ordered spaces of various kinds. In many contexts, one finds that continuous data are represented by functions of the form $u\colon X \to A$, where X is a set of points in time or space, and A is a set of data. More specifically, in some cases the sets X and A have a topology (possibly discrete) and the functions of interest are those in the set

$$\mathcal{C}[X, A] \;=\; \{u\colon X \to A \mid u \text{ is a continuous total function}\}.$$

Thus, mathematically, we are interested in models of computability for functions of the form

$$\Phi\colon A^r \times \mathcal{C}[X, A]^m \;\to\; \mathcal{C}[X, A]^n.$$

Now, a stream is a sequence of data indexed by time. There are two distinct types of data stream: streams based on continuous time and streams based on discrete time. The behaviour of systems in time can be modelled by streams and stream transformations. Continuous time is used in models based on ordinary and partial differential equations (ODEs and PDEs), and in networks of analog devices. Discrete time is used in cellular automata, neural nets and other concurrent algorithms, and in approximations to ODEs and PDEs. Thus, we will concentrate on the special case that X represents time. Although there are many ways of modelling time, we choose $X = \mathbb{T}$, where $\mathbb{T}$ is either the non-negative reals $\mathbb{R}^{\geq 0}$ or the non-negative integers $\mathbb{N}$, to represent continuous and discrete time respectively.

In these two cases, the functions $u \in \mathcal{C}[\mathbb{T}, A]$ will be called *streams*, and $\mathcal{C}[\mathbb{T}, A]$ will be called a *stream space*. We call the streams *continuous* or *discrete*, according as $\mathbb{T} = \mathbb{R}^{\geq 0}$ or $\mathbb{T} = \mathbb{N}$. In [TZ11] we established a basic theory of the specification of functions Φ on streams as fixed points of operators with contracting properties; the existence, uniqueness and continuity of the Φ were proved.

Computability theories for $\mathcal{C}[\mathbb{T}, A]$ should be useful for developing particular computability theories for all sorts of streams. In general, we have found [TZ04, TZ05, TZ06] that computability models fall into one of two classes: concrete models, which involve building a specific representation of the data type; and abstract models, which involve programming directly with the primitive operations of the data type in ways that are independent of specific representations. We will consider concrete and abstract computability models for functions Φ on the data type $\mathcal{C}[\mathbb{T}, A]$, and compare them. Our choice of concrete model is based on enumeration theory. Our choice of abstract model is based upon high-level programming with '`while`' programs over algebras.

Let us examine these ideas in more detail.

1.2 Stream transformers as fixed points

Each data type of the form $\mathcal{C}[\mathbb{T}, A]$ arises typically in some practical situation, and has its own special features. The algorithmic models that are characteristic of that situation determine, or at least suggest, a corresponding computability theory. For example, we will motivate and illustrate our computability theories using these examples:

(i) *Analog streams*: X is continuous time $\mathbb{T} = \mathbb{R}^{\geq 0}$, the non-negative reals, and the data are reals $A = [0, 1]$ or $A = \mathbb{R}$ or spectra modelled by continuous mappings from a compact space to the reals.

(ii) *Digital streams*: X is discrete time $\mathbb{T} = \{0, 1, 2, \dots\}$, and the data are bits $A = \{0, 1\}$.

We have encountered these data types before: *continuous streams* processed by analog networks [TZ07]; *discrete streams* processed by digital networks [TZ94, TTZ09].

Here, we investigate stream transformers of the form

$$\Phi \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \to \ \mathcal{C}[\mathbb{T}, A]^m \tag{1.1}$$

where for tuples of system parameters $\boldsymbol{c} \in A^r$, initial values $\boldsymbol{a} \in A^s$ and input streams $\boldsymbol{x} \in \mathcal{C}[\mathbb{T}, A]^p$, the value $\Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in \mathcal{C}[\mathbb{T}, A]^m$ is obtained as the *fixed point* of a *contracting operator*

$$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \colon \ \mathcal{C}[\mathbb{T}, A]^m \ \to \mathcal{C}[\mathbb{T}, A]^m \tag{1.2}$$

where

$$F \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \to \ (\mathcal{C}[\mathbb{T}, A]^m \ \to \ \mathcal{C}[\mathbb{T}, A]^m)$$

is represented more conveniently in the uncurried form:

$$F \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \times \mathcal{C}[\mathbb{T}, A]^m \ \to \ \mathcal{C}[\mathbb{T}, A]^m,$$

so that $F(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, \ \cdot\ ) = F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ in (1.2). We assume that F satisfies a *causality* condition which is natural in the context of stream processing, because of time, and turns out to be crucial in the proofs of the theorems.

1.3 Concrete and abstract computability

For a huge range of spaces $\mathbb{T}$ and A, we can equip $\mathcal{C}[\mathbb{T}, A]$ with the compact-open topology and consider the partial functions on $\mathcal{C}[\mathbb{T}, A]$ that are computable or approximably computable with respect to this topology. In this paper we will develop two computability models for $\mathcal{C}[\mathbb{T}, A]$ in the case that A is a *complete metric space*:

(1) a *concrete computability* model, in which computations are based on concrete representations of the space $\mathcal{C}[\mathbb{T}, A]$ constructed from $\mathbb{N}$, and

(2) an *abstract computability* model, independent of representations of the space $\mathcal{C}[\mathbb{T}, A]$, in which computations are based on programs from a high level imperative language, using a suitable set of algebraic operations on $\mathcal{C}[\mathbb{T}, A]$.

Thus, this paper plays a role in advancing our general theory of computability [TZ00]. In general, abstract computability implies concrete computability but *not* conversely [TZ06]. Indeed, the converse is quite subtle even in the case of simple algebras. We have studied the problem for computable algebras in [TZ06] and, more significantly, for metric algebras A in [TZ04], and shown that equivalence theorems are possible. One aim of this paper is to show how to use these general cases to design solutions for the stream space $\mathcal{C}[\mathbb{T}, A]$.

Designing abstract models capable of capturing concrete models is an important general problem, one that is surprisingly tricky. It has the general form:

> *Given a class $\boldsymbol{Conc}_R(A)$ of concrete computable functions on a set A via representation R, find algebraic operations on A to make an algebra $\underline{A}$ and an abstract programming language L such that*
>
> $$\boldsymbol{Conc}_R(A) \ = \ \boldsymbol{Abs}_L(\underline{A}),$$
>
> *where $\boldsymbol{Abs}_L(\underline{A})$ is the class of L-computable functions over $\underline{A}$, or L-approximable functions over $\underline{A}$, assuming $\underline{A}$ is a metric algebra.*

The concrete model of computability for the space $\mathcal{C}[\mathbb{T}, A]$ is constructed using the theory of enumerations of countable sets, started by Malcev [Mal71]. The abstract model of computability for the space $\mathcal{C}[\mathbb{T}, A]$ is constructed using the theory of 'while' programs over many-sorted algebras. There are dozens of different choices of computability models for algebras, many of which we have classified [TZ00, TZ99, SHT99].

The high level language we actually used in (2) is a significant expansion of the ***While*** language. It must be equipped with a "countable choice" operation and finite arrays, and is denoted ***WhileCC**** [TZ04]; and we consider *approximability* of functions by such programs. In [TZ04] we proved the equivalence of these imperative and enumeration models, i.e., the *soundness* and *completeness* of the abstract model with respect to the concrete model over certain algebras.

We will first prove concrete computability of Φ, assuming concrete computability of F, and then, by applying this completeness result to the space $\mathcal{C}[\mathbb{T}, A]$, prove abstract computability of Φ, assuming abstract computability of F.

1.4 Outline of paper and results

In Section 2 we review the stream space $\mathcal{C}[\mathbb{T}, A]$, and summarise the theory and results from [TZ11], which proves, firstly, the existence and uniqueness, and secondly, continuity of the fixed point function Φ for the contracting operator F, under some reasonable assumptions, such as continuity of F.

In Section 3 we develop our concrete model of computability on $\mathcal{C}[\mathbb{T}, A]$, and prove:

Theorem A (Concrete computability): *Under some reasonable assumptions, if F is concretely computable, then so is Φ.*

Finally, in Section 4, we develop the abstract model of computability (***WhileCC**** approximability) on $\mathcal{C}[\mathbb{T}, A]$. By means of Theorem A, together with the completeness theorem for abstract vs concrete computability [TZ04], applied to the space $\mathcal{C}[\mathbb{T}, A]$, we prove

Theorem B (Abstract computability): *Under some further reasonable assumptions, notably effective uniform local continuity of F, if F is **WhileCC*** approximable, then so is Φ.*

One of the aims of this paper is to develop computability theories that can analyse applications. We wish to consider networks processing continuous and discrete streams — introduced in [TZ07] and [TTZ09], respectively – from a common standpoint. In Section 5, these applications are illustrated by two examples:

(1) *analog networks*, with continuous time $\mathbb{T} = \mathbb{R}^{\geq 0}$, using the theory developed in [TZ07], and specifically the case study of a mass/spring/damper system investigated there;

(2) *synchronous concurrent algorithms* (SCAs), with discrete time $\mathbb{T}{=}\mathbb{N}$, using the theory developed in [TTZ09].

These are both archetypal discrete space models, which formally include a huge range of mathematical models of natural and artificial systems.

This paper is a sequel to [TZ11]. We have tried to minimise its dependence on that paper with a short review of the latter in §2.1, and also to make it independent of [TZ07, TTZ09]. However, the motivation and technicalities are best apprehended in the light of our entire work.

A remark about notation: on the whole it is fairly standard. The symbol '$\rightharpoonup$' will denote a partial function.

Acknowledgments. We thank Jens Blanck (Swansea), Nick James (McMaster University) and Ken Johnson (INRIA, France) and three anonymous referees for many useful comments and suggestions on earlier drafts of this paper.

The research of the second author was supported by a grant from the Natural Sciences and Engineering Research Council (Canada). The second author also appreciates the generosity of the Computer Science Department of Swansea University, which has hosted a number of his visits for the purpose of collaborating with the first author.

2 Background

2.1 Review of the stream space $\mathcal{C}[\mathbb{T}, A]$

As in [TZ11], we will investigate the space $\mathcal{C}[\mathbb{T}, A]$ of *continuous* or *discrete A-valued streams*, i.e., continuous functions from $\mathbb{T}$ to A, where $\mathbb{T}$ is either $\mathbb{R}^{\geq 0}$ or $\mathbb{N}$, representing continuous and discrete time respectively, and (A, d_A) is a complete metric space. (Note that in the case $\mathbb{T} = \mathbb{N}$, all functions from $\mathbb{T}$ to A are trivially continuous.)

We will also assume that A is *separable*, which will be relevant for concrete computability in Section 3.

The stream space $\mathcal{C}[\mathbb{T}, A]$ is given the *local uniform* (or *compact-open*) topology, generated by the pseudometrics

$$\mathsf{d}_T(u, v) \;=\; \sup_{0 \leq t \leq T} \mathsf{d}_A(u(t), v(t))$$

for all $T \geq 0$, where in fact T can be restricted to ranging over a sequence $T_0 < T_1 < T_2 < \dots$ of increasing unbounded values, defining a *compact exhaustion* (K_k) of $\mathbb{T}$, with $K_k = [0, T_k]$ $(k = 0, 1, 2, \dots)$ [TZ11]. In this paper we will take $T_k = k\tau$, for some fixed $\tau > 0$, to give the "standard exhaustion" $K_k = [0, k\tau]$, and perforce taking $\tau = 1$ when $\mathbb{T} = \mathbb{N}$. (In practice τ will be chosen as the "contraction increment" for a contracting operator: see Definition 2.2.2 below.)

This topology is metrisable [TZ11, §2.3], with the metric

$$\mathsf{d}_{\mathcal{C}[\mathbb{T},A]}(u, v) \;=_{df}\; \sum_{k=0}^{\infty} \min\big(\, \mathsf{d}_{k\tau}(u, v),\; 2^{-k} \,\big). \tag{2.1}$$

However it turns out to be easier to take $\mathcal{C}[\mathbb{T}, A]$ as a "pseudometric algebra" with global pseudometric function

$$\mathsf{D} \colon \mathbb{N} \times \mathbb{T} \times \mathcal{C}[\mathbb{T}, A]^2 \;\to\; \mathbb{R}$$

defined by

$$\mathsf{D}(k, \tau, u, v) \;=\; \mathsf{d}_{k\tau}(u, v) \tag{2.2}$$

as basic, rather than the metric $\mathsf{d}_{\mathcal{C}[\mathbb{T},A]}$. In any case, $\mathsf{d}_{\mathcal{C}[\mathbb{T},A]}$ is easily computable from D by (2.1) and (2.2). We also have an evaluation function for streams

$$\mathsf{eval} \colon \mathcal{C}[\mathbb{T}, A] \times \mathbb{T} \;\to\; A$$

where
$$\mathsf{eval}(u, t) \;=\; u(t).$$

So we gather these functions into a *many-sorted topological algebra*[1] :

$$\begin{array}{|ll|}\hline \mathsf{algebra} & \mathcal{C}[\mathbb{T},A] \\ \mathsf{carriers} & A,\ \mathbb{T},\ \mathcal{C}[\mathbb{T},A],\ \mathbb{R},\ \mathbb{N} \\ \mathsf{functions} & \mathsf{d}_A\colon A^2 \to \mathbb{R}, \\ & \mathsf{d}_{\mathbb{T}}\colon \mathbb{T}^2 \to \mathbb{R}, \\ & \mathsf{D}\colon \mathbb{N}\times\mathbb{T}\times\mathcal{C}[\mathbb{T},A]^2 \to \mathbb{R}, \\ & \mathsf{eval}\colon \mathcal{C}[\mathbb{T},A]\times\mathbb{T} \to A \\ \mathsf{end} & \\ \hline \end{array} \tag{2.3}$$

where $\mathsf{d}_{\mathbb{T}}$ and d_A are the metrics on $\mathbb{T}$ and A respectively.

$\mathcal{C}[\mathbb{T},A]$ is a topological algebra, because each of the five carriers has an associated topology (i.e., the compact open topology for the function space, together with the usual one for $\mathbb{R}$, and the discrete one for $\mathbb{N}$) with respect to which the basic functions are continuous. The carrier $\mathbb{R}$ is needed for the metric operations on A and $\mathbb{T}$. The carrier $\mathbb{N}$ is needed for (i) the domain of the global pseudometric function D, and (ii) the "N-standardness" of the algebra [TZ04, TZ05, TTZ09] so as to facilitate the completeness theorem for abstract vs concrete computability on $\mathcal{C}[\mathbb{T},A]$ (Theorem 4.4.6).

The *signature*[2] Σ of $\mathcal{C}[\mathbb{T},A]$ can be inferred from (2.3). First, corresponding to the five carriers of $\mathcal{C}[\mathbb{T},A]$, there are five sorts in the set $\boldsymbol{Sort}(\Sigma)$ of Σ-*sorts*:

- A of data, i.e., points in the metric space A,
- T of instants of time in $\mathbb{T}$,
- C of streams, i.e., elements of $\mathcal{C}[\mathbb{T},A]$,
- R of reals $\mathbb{R}$, for the metrics
- N of naturals $\mathbb{N}$, for use in computation.

Next, for each function shown in (2.3), the signature Σ has a function symbol of the corresponding type. Hence Σ can be displayed as follows:

$$\begin{array}{|ll|}\hline \mathsf{signature} & \Sigma \\ \mathsf{sorts} & \mathsf{A},\ \mathsf{T},\ \mathsf{C},\ \mathsf{R},\ \mathsf{N} \\ \mathsf{functions} & \mathsf{d_A}\colon \mathsf{A}^2 \to \mathsf{R}, \\ & \mathsf{d_T}\colon \mathsf{T}^2 \to \mathsf{R}, \\ & \mathsf{D}\colon \mathsf{N}\times\mathsf{T}\times\mathsf{C}^2 \to \mathsf{R}, \\ & \mathsf{eval}\colon \mathsf{C}\times\mathsf{T} \to \mathsf{A} \\ \mathsf{end} & \\ \hline \end{array} \tag{2.4}$$

This is quite rudimentary — it ignores the basic functions in the algebraic structures of A, $\mathbb{T}$, $\mathbb{R}$ and $\mathbb{N}$. This is irrelevant for concrete computation on $\mathcal{C}[\mathbb{T},A]$, which derives

[1] That is, a many-sorted algebra in which the carriers have topologies, relative to which each of the basic functions is continuous

[2] A signature Σ for a many-sorted algebra A consists of a set of Σ-sorts, one for each carrier of A, and (typed) Σ-function symbols, one for each basic function of A.

computational power from tracking functions on $\mathbb{N}$ (as we will see). It *is* relevant for abstract computation, as studied in the next section, where we will find it necessary to expand the algebraic structure of $\mathcal{C}[\mathbb{T}, A]$ (§4.1).

For ease of notation, we will sometimes refer to the five carriers of $\mathcal{C}[\mathbb{T}, A]$ as $\mathcal{C}_s$ for $s \in \boldsymbol{Sort}(\Sigma)$.

2.2 Causality, contraction and continuity: A review

We briefly review the main definitions and theorems of [TZ11]. First, we define the concepts of *causality* and *contraction* for stream operators [TZ11, §§ 3.2, 3.3] which are crucial assumptions in all the following theorems. We use the following notation: for $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}[\mathbb{T}, A]^m$ and $0 \leq a \leq b$:

$$\mathsf{d}_{a,b}(\boldsymbol{u}, \boldsymbol{v}) \;=_{df}\; \sup_{a \leq t \leq b} \mathsf{d}_A(\boldsymbol{u}(t), \boldsymbol{v}(t)).$$

Now consider an operator $F \colon \mathcal{C}[\mathbb{T}, A]^m \to \mathcal{C}[\mathbb{T}, A]^m$.

Definition 2.2.1 (Causality). F is *causal* if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}[\mathbb{T}, A]^m$:

$$\forall\, T \geq 0, \qquad \boldsymbol{u}\restriction_{[0,T)} \;=\; \boldsymbol{v}\restriction_{[0,T)} \;\;\Longrightarrow\;\; F(\boldsymbol{u})(T) \;=\; F(\boldsymbol{v})(T).$$

Definition 2.2.2 (Contracting operator). Let $0 < \lambda < 1$ and $\tau > 0$.
F is *contracting w.r.t.* (λ, τ), or $F \in \boldsymbol{Contr}(\lambda, \tau)$, if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}[\mathbb{T}, A]^m$:

$$\text{for all } T \geq 0 \qquad \boldsymbol{u}\restriction_T = \boldsymbol{v}\restriction_T \;\;\Longrightarrow\;\; \mathsf{d}_{T,\,T+\tau}(F(\boldsymbol{u}), F(\boldsymbol{v})) \;\leq\; \lambda \cdot \mathsf{d}_{T,\,T+\tau}(\boldsymbol{u}, \boldsymbol{v}) \qquad (2.5)$$

where we write $\boldsymbol{u}\restriction_T$ for $\boldsymbol{u}\restriction_{[0,T]}$.

We call λ and τ the ***contraction modulus*** and ***contraction increment*** respectively.

Note that if F is causal, (2.5) can be rewritten as

$$\text{for all } T \geq 0 \qquad \boldsymbol{u}\restriction_T = \boldsymbol{v}\restriction_T \;\;\Longrightarrow\;\; \mathsf{d}_{T+\tau}(F(\boldsymbol{u}), F(\boldsymbol{v})) \;\leq\; \lambda \cdot \mathsf{d}_{T+\tau}(\boldsymbol{u}, \boldsymbol{v}).$$

Also, if F is causal and $F \in \boldsymbol{Contr}(\lambda, \tau)$ and $0 < \tau' < \tau$, then $F \in \boldsymbol{Contr}(\lambda, \tau')$.

Theorem 1 (Fixed point of contracting and causal operator).
Given a stream transformer $F \colon \mathcal{C}[\mathbb{T}, A]^m \to \mathcal{C}[\mathbb{T}, A]^m$, suppose $F \in \boldsymbol{Contr}(\lambda, \tau)$ for $0 < \lambda < 1$ and $\tau > 0$, and F is causal. Then F has a unique fixed point, i.e., there is a unique $\boldsymbol{u} = \mathsf{FP}(F) \in \mathcal{C}[\mathbb{T}, A]^m$ such that $F(\boldsymbol{u}) = \boldsymbol{u}$.

This is proved in [TZ11, §3].

Now, with applications in mind, we consider operators of the form

$$F \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \;\to (\mathcal{C}[\mathbb{T}, A]^m \;\to\; \mathcal{C}[\mathbb{T}, A]^m) \qquad (2.6)$$

$(0 \le r,\ 0 \le s \le m,\ p > 0,\ m > 0)$ or, in uncurried form,

$$F\colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \times \mathcal{C}[\mathbb{T}, A]^m \ \to\ \mathcal{C}[\mathbb{T}, A]^m \tag{2.7}$$

where A^r contains the r system parameters $\boldsymbol{c} = (c_1, \ldots, c_r)$, A^s contains the s initial constants $\boldsymbol{a} = (a_1, \ldots, a_s)$, and $\mathcal{C}[\mathbb{T}, A]^p$ contains the p input streams $\boldsymbol{x} = (x_1, \ldots, x_s)$. Think of $\boldsymbol{a}$ as the *initial values* of the first s of the m non-input stream variables $\boldsymbol{u}$. Then for $\boldsymbol{c} \in A^r$, $\boldsymbol{a} \in A^s$ and $\boldsymbol{x} \in \mathcal{C}[\mathbb{T}, A]^p$, $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is the operator

$$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} = F(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, \ \cdot\ )\colon\ \mathcal{C}[\mathbb{T}, A]^m \ \to\ \mathcal{C}[\mathbb{T}, A]^m. \tag{2.8}$$

Next we restrict the parameters $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$ to an open subset $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$, which we call the "contraction domain" for F, and assume that we have U-indexed *families* of contraction moduli $\langle \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ and increments $\langle \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ such that $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is contracting with respect to $(\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$, for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$. We write (boldface) '$\boldsymbol{\lambda}$' and '$\boldsymbol{\tau}$' for the functions corresponding to these two families, thus:

$$\begin{aligned} &\boldsymbol{\lambda}\colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \rightharpoonup\ \mathbb{R} \\ \text{and} \qquad &\boldsymbol{\tau}\colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \rightharpoonup\ \mathbb{T} \end{aligned} \tag{2.9a}$$

which are defined (at least) on U, such that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$:

$$\begin{aligned} \boldsymbol{\lambda}(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \ &=\ \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \\ \boldsymbol{\tau}(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \ &=\ \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}. \end{aligned} \tag{2.9b}$$

Then, by Theorem 1, for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$, $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ has a unique fixed point $\boldsymbol{u} = \mathsf{FP}(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$, depending on the parameters and inputs $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$. We now want to consider this fixed point as a function of $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$. Hence we define the *fixed point function*

$$\Phi\colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \rightharpoonup\ \mathcal{C}[\mathbb{T}, A]^m \tag{2.10a}$$

by: $\boldsymbol{dom}(\Phi) = U$, and for $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$,

$$\Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \ =\ \mathsf{FP}(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}). \tag{2.10b}$$

We must also define the concepts of *shift invariance of operators* and *closure of domains under shifts* [TZ11, §§4.1, 4.2] which are crucial assumptions in Theorem B. Briefly, a stream operator F is shift invariant if its behaviour is invariant under time shifts, with *suitable changes made to the initial constants*.

More precisely, we first define, for any $\boldsymbol{u} \in \mathcal{C}[\mathbb{T}, A]^m$ and $T \ge 0$, the *shifted stream tuple*

$$\boldsymbol{shift}_T(\boldsymbol{u})(t) \ =_{df}\ \boldsymbol{u}(T + t).$$

We also use the following notation: for $\boldsymbol{u} \in \mathcal{C}[\mathbb{T}, A]^m$, $s \le m$ and $T \ge 0$, we write $\boldsymbol{u}^s(T)$ for the s-tuple $(u_1(T), \ldots, u_s(T))$.

Definition 2.2.3 (Shift invariance with updated initial values). An operator F as in (2.7) is *shift invariant* if for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{u}) \in A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \times \mathcal{C}[\mathbb{T}, A]^m$ and $T \geq 0$, if $F(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{u})\restriction_T = \boldsymbol{u}\restriction_T$ (i.e., $\boldsymbol{u}$ is a T-approximate fixed point of $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$), then

(i) $\boldsymbol{u}^s(0) = \boldsymbol{a}$ (i.e., the inputs are "compatible"), and

(ii) $F(\boldsymbol{c}, \boldsymbol{u}^s(T), \boldsymbol{shift}_T(\boldsymbol{x}), \boldsymbol{shift}_T(\boldsymbol{u})) = \boldsymbol{shift}_T(F(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{u}))$.

This is the "invariance property" of F, subject to the "updating" of the initial values from $\boldsymbol{u}^s(0)$ $(= \boldsymbol{a})$ to $\boldsymbol{u}^s(T)$.

We conclude this survey of [TZ11] with a theorem on the continuity of the fixed point function Φ. First we need one more definition.

Definition 2.2.4 (Closure of domain under shifts). Given Φ as in (2.10), with $\boldsymbol{dom}(\Phi) = U$, we say that U is *closed under shifts w.r.t.* Φ if for all $T > 0$ and all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$:

$$(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \quad \Longrightarrow \quad (\boldsymbol{c},\ \Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})^s(T),\ \boldsymbol{shift}_T(\boldsymbol{x})) \in U.$$

Theorem 2 (Continuity of FP). *Given stream operators F and $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ as in* (2.7) *and* (2.8)*, an open set $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$, and U-indexed families of contraction moduli $\boldsymbol{\lambda} = \langle \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ and increments $\boldsymbol{\tau} = \langle \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$, suppose*

(i) $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \in \boldsymbol{Contr}(\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$ *for all* $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$,

(ii) $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ *is causal,*

(iii) F *is shift invariant,*

(iv) F *is continuous*[3] *on* U,

(v) $\boldsymbol{\lambda}$ *and* $\boldsymbol{\tau}$ *are locally bounded on* U, *and*

(vi) U *is closed under shifts w.r.t.* Φ,

where Φ *is the fixed point function for* F *as in* (2.10)*, given by Theorem 1.*
Then Φ *is continuous on* U.

[3] This continuity assumption is made with respect to the first 3 arguments of F only, i.e., $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$. No assumption of continuity need be made for the fourth (stream) argument.

3 Concrete computability in $\mathcal{C}[\mathbb{T}, A]$

3.1 Enumerations α; α-tracking functions

The simplest way to make a concrete model is to start with the theory of enumerated sets begun by Malcev [Mal71]. We have applied this theory to complete separable metric spaces [TZ04, TZ05], and will now construct an enumeration-based model for the space $\mathcal{C}[\mathbb{T}, A]$, and prove concrete computability of the fixed point of a contracting operator.[4]

We begin with enumerations of elements of the algebra $\mathcal{C}[\mathbb{T}, A]$. We fix a family

$$\alpha \;=\; \langle \alpha_s \mid s \in \boldsymbol{Sort}(\Sigma)\rangle$$

of *enumerations* of certain subsets Z_s of the carriers $\mathcal{C}_s$ of $\mathcal{C}[\mathbb{T}, A]$, i.e., surjections

$$\alpha_s \colon \Omega_{\alpha,s} \;\twoheadrightarrow\; Z_s \subseteq \mathcal{C}_s \qquad (s \in \boldsymbol{Sort}(\Sigma))$$

of some subset $\Omega_{\alpha,s}$ of $\mathbb{N}$ with Z_s. The set Z_s is called the (α_s-)*enumerated subset* of $\mathcal{C}_s$. The elements of $\Omega_{\alpha,s}$ can be thought of as *codes* (via the coding given by the inverse of α) for the elements of Z_s, and $\Omega_{\alpha,s}$ is an (α-)*code set* for Z_s.

The family of enumerations α is said to *represent* A.

Our concrete model (§3.2) will be built from such a family of enumerations α.

Let us consider the nature of these enumerations. To summarise their construction: the carriers of $\mathcal{C}[\mathbb{T}, A]$ are the sets

$$A, \quad \mathbb{T}, \quad \mathcal{C}[\mathbb{T}, A], \quad \mathbb{R}, \quad \mathbb{N},$$

with (respectively) enumerated subsets

$$Z_{\mathsf{A}}, \quad Z_{\mathsf{T}} = \mathbb{Q}^{\geq 0} \text{ or } \mathbb{N}, \quad Z_{\mathsf{C}}, \quad Z_{\mathsf{R}} = \mathbb{Q}, \quad Z_{\mathsf{N}} = \mathbb{N}.$$

The enumeration α_{A} will be, typically, a "standard" enumeration of $Z_{\mathsf{A}} \subset A$, with code set $\Omega_{\alpha,\mathsf{A}}$ a decidable subset of $\mathbb{N}$. For example, if $A = \mathbb{R}$, we can take $Z_{\mathsf{A}} = \mathbb{Q}$ and α_{A} to be a standard coding of $\mathbb{Q}$. The mapping α_{T} is a standard enumeration of the non-negative rationals (if $\mathbb{T} = \mathbb{R}^{\geq 0}$) or the identity (if $\mathbb{T} = \mathbb{N}$). The mapping α_{R} is a standard enumeration of the rationals, and α_{N} is the identity on $\mathbb{N}$.

Suitable enumerations α_{C} of subsets Z_s of $\mathcal{C}[\mathbb{T}, A]$ are not quite so trivial to construct from α_{A} and α_{T}, especially if density of Z_{C} in $\mathcal{C}[\mathbb{T}, A]$ is required (see the following definition).

Definition 3.1.1 (Density of $\boldsymbol{\alpha}$). The family α of enmerations is *dense in* $\mathcal{C}[\mathbb{T}, A]$ if for all Σ-sorts s, Z_s is a dense subset of A_s.

[4] An alternative treatment of concrete computation on $\mathcal{C}[\mathbb{T}, A]$ in the case that $\mathbb{T} \subseteq \mathbb{R}^m$ and $A = \mathbb{R}^n$, also using the compact-open topology, can be found in [Wei00, Ch. 6].

The assumption of density is needed to obtain an interesting concrete computability theory (as described in §3.2). In fact, this assumption is non-trivial only for two sorts: the data sort A, and the stream sort C.

So A must be chosen to be separable. What about $\mathcal{C}[\mathbb{T}, A]$? It can be shown that separability of $\mathcal{C}[\mathbb{T}, A]$ follows from separability of $\mathbb{T}$ and A. However this is of no use in constructing an enumerated subset of $\mathcal{C}[\mathbb{T}, A]$, since we cannot simply use the enumeration of $\mathcal{C}[\mathbb{T}, A]$ given by such a proof to construct an enumerated subset of $\mathcal{C}[\mathbb{T}, A]$, because the family α of enumerations must satisfy certain effectivity conditions, which will be needed in Theorem A: effective local uniform continuity of α-streams, and Σ-effectivity of $\overline{\alpha}$ (Definitions 3.1.3 and 3.2.8).

Definition 3.1.2 (α-streams). Streams in Z_{C}, i.e., in the range of α_{C}, will be called *α-streams*.

Another desirable property of the enumerations α (specifically α_{C}) is *effective locally uniform continuity* of α-streams:

Definition 3.1.3 (Effective locally uniform continuity of α-streams). The α-streams are said to be is *effectively locally uniformly continuous* if there is a recursive function $\mu\colon \mathbb{N}^3 \to \mathbb{N}$ (an *effective locally uniform continuity modulus*) such that for all k, ℓ and $n \in \Omega_{\alpha,\mathsf{C}}$, writing $z_n = \alpha_{\mathsf{C}}(n)$:

$$\forall t_1, t_2 \in [0, k\tau] : |t_1 - t_2| < 2^{-\mu(n,k,\ell)} \implies \mathsf{d}_A(z_n(t_1),\, z_n(t_2)) < 2^{-\ell},$$

or, more simply but equivalently: There is a recursive function $\mu'\colon \mathbb{N}^2 \to \mathbb{N}$ such that for all n, k, writing $z_n = \alpha_{\mathsf{C}}(n)$:

$$\forall t_1, t_2 \in [0, k\tau] : |t_1 - t_2| < 2^{-\mu'(n,k)} \implies \mathsf{d}_A(z_n(t_1),\, z_n(t_2)) < 2^{-k}.$$

Remark 3.1.4. When $\mathbb{T} = \mathbb{N}$, this condition is trivially satisfied, since (by the discreteness of $\mathbb{N}$) *all* streams are continuous — in fact, effectively globally uniformly continuous.

The value of this assumption of *effective locally uniform continuity* of α-streams may not be immediately obvious. Note that the streams are, in any case, continuous by definition, and hence locally uniformly continuous. The extra assumption here, namely *effectivity* of local uniform continuity for the α-streams, is conceptually reasonable[5]. Its value will become clearer when we consider $\overline{\alpha}$-computability (§3.2), as it will permit proofs of useful results, such as $\overline{\alpha}$-computability of integration in our analog network example (see §5.1).

For convenience, we combine the above two useful conditions on α (density and effective local uniform continuity of α-streams) into a "regularity" condition for α, which will be assumed in the rest of the paper; in particular, in Theorems A and B, the Completeness Theorem 4.4.6, and the discussion in §6.2.

[5] This can be compared to clause (ii) in Definition A of computability of functions in [PER89, p. 25], which is actually an assumption of effective global uniform continuity, since the domain is compact.

Definition 3.1.5 (Regularity of α). The family α of enumerations of $\mathcal{C}[\mathbb{T}, A]$ is *regular* if it satisfies the two conditions:

(a) α is dense in $\mathcal{C}[\mathbb{T}, A]$ (Definition 3.1.1),

(b) α-streams are effectively locally uniformly continuous (Definition 3.1.3),

We introduce some terminology and notation. For a *Σ-product type* $u = s_1 \times \cdots \times s_m$, we have the product space

$$\mathcal{C}^u \;=_{df}\; \mathcal{C}_{s_1} \times \cdots \times \mathcal{C}_{s_m},$$

and the product α^u-enumeration

$$\alpha^u =_{df} \; (\alpha_{s_1}, \ldots, \alpha_{s_m}) \colon \Omega^u_\alpha \;\twoheadrightarrow\; Z^u$$

(defined in the obvious way) of the subset

$$Z^u =_{df} \; Z_{s_1} \times \cdots \times Z_{s_m} \;\subseteq\; \mathcal{C}^u$$

with the product domain

$$\Omega^u_\alpha \;=_{df}\; \Omega_{\alpha,s_1} \times \cdots \times \Omega_{\alpha,s_m} \;\subseteq\; \mathbb{N}^m.$$

Definition 3.1.6 (α-tracking functions[6]). Let $f \colon \mathcal{C}^u \rightharpoonup \mathcal{C}_s$ and $\varphi : \mathbb{N}^m \rightharpoonup \mathbb{N}$.

(a) φ is a *tracking function with respect to* α, or *α-tracking function*, for f, if the following diagram commutes:

$$\begin{array}{ccc} Z^u & \xrightarrow{\;f \restriction Z^u\;} & Z_s \\ {\scriptstyle \alpha^u}\uparrow & & \uparrow{\scriptstyle \alpha_s} \\ \Omega^u_\alpha & \xrightarrow[\;\varphi \restriction \Omega^u_\alpha\;]{} & \Omega_{\alpha,s} \end{array} \tag{3.1}$$

in the sense that for all $k \in \Omega^u_\alpha$

$$f(\alpha^u(k)) \downarrow \;\Longrightarrow\; \varphi(k) \downarrow \wedge \varphi(k) \in \Omega_{\alpha,s} \wedge f(\alpha^u(k)) = \alpha_s(\varphi(k)). \tag{3.2a}$$

(b) φ is a *strict α-tracking function* for f if in addition, for all $k \in \Omega^u_\alpha$

$$f(\alpha^u(k)) \uparrow \;\Longrightarrow\; \varphi(k) \uparrow . \tag{3.2b}$$

Here we use the notation $\alpha^u(k) = (\alpha_{s_1}(k_1), \ldots, \alpha_{s_m}(k_m))$, where $k = (k_1, \ldots, k_m)$. (We will sometimes drop the type super- and subscripts.)

[6] These are called *realizations* in [Wei00]

Note that we are not concerned with the behaviour – or the (un)definedness – of f off Z^u, or of φ off Ω^u_α.

We are looking for sufficient conditions for a function on $\mathcal{C}[\mathbb{T}, A]$ to have an α-tracking function, and a function on $\mathbb{N}$ to be an α-tracking function.

Let $f\colon \mathcal{C}^u \rightharpoonup \mathcal{C}_s$, where u is an m-ary product type, and let $\varphi\colon \mathbb{N}^m \rightharpoonup \mathbb{N}$.

Definition 3.1.7 (α-closedness of functions on $\mathcal{C}[\mathbb{T}, A]$ and tracking functions).

(a) f is *α-closed* iff for any $x \in Z^u$, if $f(x)\downarrow$ then $f(x) \in Z_s$, i.e., $f \restriction Z^u\colon Z^u \rightharpoonup Z_s$.

(b) φ is *α-closed* iff for any $k \in \Omega^u_\alpha$, if $\varphi(k)\downarrow$ then $\varphi(k) \in \Omega_{\alpha,s}$. i.e., $\varphi \restriction \Omega^u_\alpha\colon \Omega^u_\alpha \rightharpoonup \Omega_{\alpha,s}$.

Definition 3.1.8 (α-equivalence).

(a) For $k_1, k_2 \in \Omega_{\alpha,s}$, $k_1 \approx^s_\alpha k_2$ iff $\alpha_s(k_1) = \alpha_s(k_2)$.

(b) For $k_1, k_2 \in \Omega^u_\alpha$, $k_1 \approx^u_\alpha k_2$ iff $\alpha^u(k_1) = \alpha^u(k_2)$.

Note that $\approx^s_\alpha$ and $\approx^u_\alpha$ are equivalence relations on $\Omega_{\alpha,s}$ and Ω^u_α respectively.

We will often drop the type symbols 's' and 'u' from this notation.

Definition 3.1.9 (α-compatibility[7]). A function $\varphi\colon \Omega^u_\alpha \rightharpoonup \Omega_{\alpha,s}$ is *compatible with* α, or *α-compatible*, iff for all $k_1, k_2 \in \Omega^u_\alpha$,

$$\begin{aligned} k_1 \approx_\alpha k_2 \implies \text{either} \quad & \varphi(k_1)\downarrow \wedge \varphi(k_2)\downarrow \wedge \varphi(k_1) \approx_\alpha \varphi(k_2) \\ \text{or} \quad & \varphi(k_1)\uparrow \wedge \varphi(k_2)\uparrow . \end{aligned}$$

Lemma 3.1.10. *Suppose φ is an α-tracking function for f. Then*

(a) *f is α-closed,*

(b) *φ is α-closed,*

(c) *φ is α-compatible.*

In the other direction:

Lemma 3.1.11.

(a) *If f is α-closed, then f has an α-tracking function.*

(b) *If φ is α-compatible and α-closed, then φ is an α-tracking function for some function.*

[7] This is called *extensionality* in [Wei00]

Remark 3.1.12 (Σ-subalgebra generated by α).
Suppose every basic function of $\mathcal{C}[\mathbb{T}, A]$ is α-closed. (This is the case, for example, with Example 3.2.12 below.) Then α generates a Σ-subalgebra of $\mathcal{C}[\mathbb{T}, A]$, based on the family of α-enumerated sets $Z = \langle Z_s \mid s \in \boldsymbol{Sort}(\Sigma)\rangle$ and the restrictions of these functions to Z:

algebra	$\mathcal{Z}$
carriers	Z_{A}, Z_{T}, Z_{C}, Z_{R}, Z_{N}
functions	$\mathsf{d}_A \restriction_Z : {Z_{\mathsf{A}}}^2 \to Z_{\mathsf{R}}$,
	$\mathsf{d}_{\mathbb{T}} \restriction_Z : {Z_{\mathsf{T}}}^2 \to Z_{\mathsf{R}}$,
	$\mathsf{D} \restriction_Z : Z_{\mathsf{N}} \times Z_{\mathsf{T}} \times {Z_{\mathsf{C}}}^2 \to Z_{\mathsf{R}}$,
	$\mathsf{eval} \restriction_Z : Z_{\mathsf{C}} \times Z_{\mathsf{T}} \to Z_{\mathsf{A}}$
end	

3.2 $\overline{\alpha}$-computability on $\mathcal{C}[\mathbb{T}, A]$

We now turn to considerations of *computability* with respect to an enumeration α.

First note that typically,

$$\Omega_\alpha \text{ is a decidable subset of } \mathbb{N}, \text{ and } \approx_\alpha \text{ is a decidable relation,} \tag{3.3}$$

in which case, α can be effectively modified in a standard way so as to further satisfy

$$\Omega_\alpha = \mathbb{N} \text{ and } \alpha \text{ is 1-1.}$$

Definition 3.2.1 (α-computability). Suppose φ is a (strict) α-tracking function for f, and φ is a computable (i.e., recursive) partial function. Then f is said to be (*strictly*) α-*computable*.

The enumerations α satisfying (3.3) are insufficient, in general, to study computability theory in structures such as $\mathbb{R}$, and stream spaces. For example, a standard enumeration α_{R} of the rationals $\mathbb{Q} \subset \mathbb{R}$ does not by itself produce a satisfactory model of computation on the reals. We have to go beyond such enumerated sets to their *computational closures*, as we now describe.

We assume Z_s is dense in $\mathcal{C}_s$. We can use the elements of Z_s to computably approximate elements of $\mathcal{C}_s$. Those elements of $\mathcal{C}_s$ that are approximated in this way are called the α-*computable elements* of $\mathcal{C}_s$.

For each Σ-sort s, the α-*computational closure of* Z_s is constructed as the set $\mathcal{C}\ell_\alpha(Z_s)$ of α-*computable elements* of $\mathcal{C}_s$, where $Z_s \subseteq \mathcal{C}\ell_\alpha(Z_s) \subseteq \mathcal{C}_s$. This will be a countable set with its own enumeration

$$\overline{\alpha} : \Omega_{\overline{\alpha},s} \twoheadrightarrow \mathcal{C}\ell_\alpha(Z_s). \tag{3.4}$$

where the code set $\Omega_{\overline{\alpha},s} \subseteq \mathbb{N}$ is the domain of the enumeration $\overline{\alpha}_s$, to be defined below. This gives the enumerated space $(\mathcal{C}\ell_\alpha(Z_s), \overline{\alpha}_s)$ and the picture:

$$\begin{array}{ccccc} Z_s & \subseteq & \mathcal{C}\ell_\alpha(Z_s) & \subseteq & \mathcal{C}_s \\ \uparrow{\scriptstyle \alpha_s} & & \uparrow{\scriptstyle \overline{\alpha}_s} & & \\ \mathbb{N} & & \Omega_{\overline{\alpha},s} & & \end{array} \tag{3.5}$$

The elements of $\mathcal{C}\ell_\alpha(Z_s)$ in (3.4) are the *α-computable elements of $\mathcal{C}_s$*, i.e., limits in $\mathcal{C}_s$ of α-effective Cauchy sequences (to be defined below) of elements of Z_s. Then $\Omega_{\overline{\alpha},s}$ is the set of $\overline{\alpha}$-codes c of the α-computable elements $\overline{\alpha}_s(c) \in \mathcal{C}\ell_\alpha(Z_s)$.

We now describe the construction of the family $\overline{\alpha} = \langle \overline{\alpha}_s \mid s \in \boldsymbol{Sort}(\Sigma) \rangle$ of enumerations (3.4) of the sets $\mathcal{C}\ell_\alpha(Z_s)$.

First, the metric space A: details of the construction of $\mathcal{C}\ell_\alpha(Z_\mathsf{A})$ and $\overline{\alpha}_\mathsf{A}$ can be found in [TZ04, TZ05]. We repeat them here for the reader's convenience.

The set $\Omega_{\overline{\alpha},\mathsf{A}} \subseteq \mathbb{N}$ consists of *codes* for $\mathcal{C}\ell_\alpha(Z_\mathsf{A})$ (w.r.t. α), i.e., pairs of numbers $c = \langle e, m \rangle$ where

(i) e is an index for a total computable function $\{e\}$ defining a Cauchy sequence $\alpha_\mathsf{A} \circ \{e\} \colon \mathbb{N} \to Z_\mathsf{A}$, i.e., the sequence

$$\alpha_\mathsf{A}(\{e\}(0)),\ \alpha_\mathsf{A}(\{e\}(1)),\ \alpha_\mathsf{A}(\{e\}(2)),\ \ldots\ , \tag{3.6}$$

of elements of Z_s,

(ii) m is an index for a computable modulus of convergence for this sequence:

$$\forall k, l \geq \{m\}(n) :\ \mathsf{d}(\alpha_\mathsf{A}(\{e\}(k)), \alpha_\mathsf{A}(\{e\}(l))) < 2^{-n}. \tag{3.7}$$

For any such code $c = \langle e, m \rangle \in \Omega_{\overline{\alpha}}$, $\overline{\alpha}(c)$ is defined as the limit in A of the Cauchy sequence (3.6), and $\mathcal{C}\ell_\alpha(Z_\mathsf{A})$ is the range of $\overline{\alpha}_\mathsf{A}$, as shown in diagram (3.5).

Remark 3.2.2 (Fast Cauchy sequences). We may assume, when convenient, that the modulus of convergence for a given code is the *identity*, i.e., replace (3.7) by the simpler

$$\forall k, l \geq n :\ \mathsf{d}_s(\alpha_\mathsf{A}(\{e\}(k)), \alpha_\mathsf{A}(\{e\}(l))) < 2^{-n}$$

or, equivalently,

$$\forall k > n :\ \mathsf{d}_s(\alpha_\mathsf{A}(\{e\}(k)), \alpha_\mathsf{A}(\{e\}(n))) < 2^{-n} \tag{3.8}$$

because any code $c = \langle e, m \rangle$ satisfying (3.7) can be effectively replaced by a code for the same element of $\mathcal{C}\ell_\alpha(Z_\mathsf{A})$ satisfying (3.8), namely $c' = \langle e', m_1 \rangle$, where m_1 is a standard code for the identity function on $\mathbb{N}$, and $e' = \mathrm{comp}(e, m)$, where $\mathrm{comp}(x, y)$ is a primitive recursive function for "composition" of (indices of) computable functions, i.e., $\{\,\mathrm{comp}(e, m)\,\}(x) \simeq \{\,e\,\}(\{\,m\,\}(x))$. In the case of a code $c = \langle e, m_1 \rangle$ satisfying (3.8), the

sequence (3.6) is called a *fast* (α-*effective*) *Cauchy sequence*. We may then, for simplicity, call e itself the "code", and the argument of $\overline{\alpha}_{\mathsf{A}}$. So we can shift between "c-codes" and "e-codes" as convenient.

Lemma 3.2.3 (Closure of α-computability operation). *The enumerated subset $(\mathcal{C}\ell_\alpha(Z), \overline{\alpha})$ is "computationally closed in A", in the sense that the limit of a (fast) $\overline{\alpha}$-effective Cauchy sequence of elements of $\mathcal{C}\ell_\alpha(Z_s)$ is again in $\mathcal{C}\ell_\alpha(Z_s)$, i.e.,*

$$\mathcal{C}\ell_{\overline{\alpha}}(\mathcal{C}\ell_\alpha(Z_s)) = \mathcal{C}\ell_\alpha(Z_s).$$

The proof uses the well-known technique of taking "diagonal approximating sequences" from double sequences.[8]

We turn to the definitions of α_s and $\overline{\alpha}_s$ for the other sorts s in Σ, i.e., $s \in \{\,\mathsf{R},\, \mathsf{N},\, \mathsf{T},\, \mathsf{C}\,\}$.

For the reals $\mathbb{R}$: α_{R} is a standard enumeration of the rationals $Z_{\mathsf{R}} = \mathbb{Q}$, and the computational closure of $\mathbb{Q}$ in $\mathbb{R}$ (defined as for A) is the set $\mathcal{C}_\alpha(\mathbb{Q})$ of *computable reals*.

For the naturals $\mathbb{N}$: the enumerations α_{N} and $\overline{\alpha}_{\mathsf{N}}$ are both taken to be simply the identity enumeration. Hence any function from $\mathbb{N}^m$ to $\mathbb{N}$ has itself as a tracking function.

For the time line $\mathbb{T}$: The enumerations α_{T} and $\overline{\alpha}_{\mathsf{T}}$ resemble one of the previous two cases, depending on whether $\mathbb{T} = \mathbb{R}^{\geq 0}$ or $\mathbb{T} = \mathbb{N}$.

Finally, for the stream space $\mathcal{C}[\mathbb{T}, A]$ with its enumerated subset $(Z_{\mathsf{C}}, \alpha_{\mathsf{C}})$, we first define[9] a sequence (u_n) of elements of $\mathcal{C}[\mathbb{T}, A]$ to be *locally uniformly Cauchy* if

$$\forall T\, \forall \epsilon > 0\, \exists N\, \forall m, n \geq N : \mathsf{d}_T(u_m,\, u_n) \leq \epsilon.$$

Now let

$$\mathcal{C}_\alpha(\mathbb{T}, A) \;=_{df}\; \mathcal{C}\ell_\alpha(Z_{\mathsf{C}}) \;\subset\; \mathcal{C}[\mathbb{T}, A]$$

be the set of all limits in $\mathcal{C}[\mathbb{T}, A]$ of α-*effectively* locally uniform Cauchy sequences of elements of Z_{C} (i.e., effective in their α_{C}-codes) — such limits always existing by the (local uniform) completeness of $\mathcal{C}[\mathbb{T}, A]$[10] — and let $\Omega_{\overline{\alpha},\mathsf{C}} \subset \mathbb{N}$ be the set of $\overline{\alpha}$-*codes* for elements of $\mathcal{C}_\alpha(\mathbb{T}, A)$. More precisely, $\Omega_{\overline{\alpha},\mathsf{C}}$ consists of pairs of numbers $c = \langle e, m\rangle$ where

(i) e is an index for a total recursive function defining a sequence

$$z_0,\, z_1,\, z_2,\, \ldots \tag{3.9}$$

of elements of Z_{C}, where $z_n = \alpha(\{e\}(n))$, having a limit $z \in \mathcal{C}[\mathbb{T}, A]$, and

(ii) m is an index for a modulus of local uniform convergence for this sequence; i.e., for all k:

$$\forall n, p \geq \{m\}(k),\; \mathsf{d}_{k\tau}(z_n, z_p) \leq 2^{-k}. \tag{3.10}$$

[8] See, e.g., [PER89, p. 20, Prop. 1]

[9] [TZ11, Definition 2.2.4]

[10] [TZ11, Lemma 2.2.5]

For any such code c, $\overline{\alpha}_{\mathsf{C}}(c)$ is defined as the limit $z \in \mathcal{C}[\mathbb{T}, A]$ of the sequence (3.9). This defines (cf. (3.4), with $s = \mathsf{C}$) the function

$$\overline{\alpha}_{\mathsf{C}} \colon \Omega_{\overline{\alpha},\mathsf{C}} \;\twoheadrightarrow\; \mathcal{C}_\alpha(\mathbb{T}, A).$$

Definition 3.2.4 ($\overline{\alpha}$-streams). The elements of $\mathcal{C}_\alpha(\mathbb{T}, A)$, i.e. streams in the range of $\overline{\alpha}_{\mathsf{C}}$, are called *$\alpha$-computable streams* or *$\overline{\alpha}$-streams* (cf. Definition 3.1.2: α-streams).

Note that since $\overline{\alpha}$, like α, is an enumeration of a countable set, the basic properties of α in §3.1 apply also to $\overline{\alpha}$. However, the code sets $\Omega_{\overline{\alpha}}$ are not generally decidable, in contrast with Ω_α (cf. (3.3)).

Definitions 3.2.5 ($\overline{\alpha}$-tracking functions and $\overline{\alpha}$-computability). The concepts of (strict) *$\overline{\alpha}$-tracking function* and (strict) *$\overline{\alpha}$-computability* are defined analogously to the corresponding concepts for α (Definitions 3.1.6 and 3.2.1), by replacing 'α' by '$\overline{\alpha}$' in (3.1) and (3.2).

Remark 3.2.6 (Value of τ). Recall that τ (occurring in (3.10)) is a fixed positive real, defining a standard exhaustion of $\mathbb{T}$ (cf. §2.1). We will assume from now on that if $\mathbb{T} = \mathbb{N}$ then $\tau = 1$, and if $\mathbb{T} = \mathbb{R}^{\geq 0}$ then τ is some α-computable real, which we can take to be (in Theorem A) the value of the contraction increment $\tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$, or (in Theorem 2 and Theorem B) a local minimum for that value. Note that notwithstanding the appearance of 'τ' in (3.10), the concept of "$\overline{\alpha}$-stream" is independent of the choice of value for τ, as can easily be checked.[11]

Remark 3.2.7 (Locally fast Cauchy sequences). We may assume, when convenient, that the modulus of convergence for a given code is the *identity*, i.e., replace (3.10) by the simpler

$$\forall n, p \geq k : \; \mathsf{d}_{k\tau}(z_n, z_p) < 2^{-k},$$

or equivalently,

$$\forall n > k : \; \mathsf{d}_{k\tau}(z_n, z_k) < 2^{-k},$$

by an argument similar to the one in Remark 3.2.2.

Another desirable property of a family of enumerations $\overline{\alpha}$ is:

Definition 3.2.8 (Σ-effectivity of $\overline{\alpha}$). $\overline{\alpha}$ is (*strictly*) *Σ-effective* if the basic functions of the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$, namely $\mathsf{d}_{\mathbb{T}}$, d_A, D, and eval, are (strictly) $\overline{\alpha}$-computable.

Notation 3.2.9 ($\overline{\alpha}$-enumerated subsets of carriers). We use the notation

- A_α for $\mathcal{C}\ell_\alpha(Z_{\mathsf{A}})$, the set of α-computable points in A,
- $\mathbb{T}_\alpha$ for $\mathcal{C}\ell_\alpha(Z_{\mathsf{T}})$, the set of α-computable time instants,
- $\mathcal{C}_\alpha(\mathbb{T}, A)$ for $\mathcal{C}\ell_\alpha(Z_{\mathsf{C}})$, the set of α-computable streams, and

[11] Cf. Lemmas 3.1.3 and 3.2.11 in [TZ11], which hold for *any* compact exhaustion (K_k).

- $\mathbb{R}_\alpha$ for $\mathcal{C}\ell_\alpha(Z_\mathsf{R})$, the set of α-computable reals.

(Recall that $\mathcal{C}\ell_\alpha(Z_\mathsf{N}) = Z_\mathsf{N} = \mathbb{N}$.)

Remark 3.2.10 (Computable subalgebra generated by $\overline{\alpha}$). (Cf. Remark 3.1.12.) The family of enumerations $\overline{\alpha}$ is said to *generate a computable subalgebra* of the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$ if the family of $\overline{\alpha}$-enumerated subsets of the carriers of $\mathcal{C}[\mathbb{T}, A]$ (at all Σ-sorts) forms a Σ-subalgebra of $\mathcal{C}[\mathbb{T}, A]$:

algebra	$\mathcal{C}_\alpha(\mathbb{T}, A)$
carriers	$A_\alpha,\ \mathbb{T}_\alpha,\ \mathcal{C}_\alpha(\mathbb{T}, A),\ \mathbb{R}_\alpha,\ \mathbb{N}$
functions	$\mathsf{d}_{A,\alpha}\colon A_\alpha^2 \to \mathbb{R}_\alpha,$
	$\mathsf{d}_{\mathbb{T},\alpha}\colon \mathbb{T}_\alpha^2 \to \mathbb{R}_\alpha,$
	$\mathsf{D}_\alpha\colon \mathbb{N} \times \mathbb{T}_\alpha \times \mathcal{C}_\alpha(\mathbb{T}, A)^2 \to \mathbb{R}_\alpha,$
	$\mathsf{eval}_\alpha\colon \mathcal{C}_\alpha(\mathbb{T}, A) \times \mathbb{T}_\alpha \to A_\alpha$
end	

A sufficent condition for $\overline{\alpha}$ to generate a computable subalgebra is given by

Lemma 3.2.11. *If $\overline{\alpha}$ is Σ-effective (Definition 3.2.8), then $\overline{\alpha}$ generates a computable subalgebra of $\mathcal{C}[\mathbb{T}, A]$.*

This follows immediately from Lemma 3.1.10(a) applied to $\overline{\alpha}$.

Example 3.2.12 (Computable streams). As a simple example of a computable subalgebra of a stream algebra: take $\mathbb{T} = \mathbb{R}^{\geq 0}$ and $A = \mathbb{R}$, and let α_A be a standard enumeration of $\mathbb{Q} \subset \mathbb{R}$. For a countable, dense and effectively locally uniformly continuous subset of $\mathcal{C}[\mathbb{R}^{\geq 0}, \mathbb{R}]$, we can take $Z_\mathsf{C} = \mathsf{ZZ}$, the set of all *continuous rational "zigzag functions" from $\mathbb{R}^{\geq 0}$ to $\mathbb{R}$ with finite support*, a typical example of which is shown in Figure 1, where we require that the starting and turning points ($p_1, \ldots, p_7$ in the figure) have rational coordinates, and which are zero from some point on (p_7 in the figure).

We let α_C be some standard enumeration of ZZ. This enumeration is easily seen to be regular (Definition 3.1.5).

We could also have used, as our starting point Z_C, the set of polynomial functions of t with rational coefficients, which is dense in $\mathcal{C}[\mathbb{R}^{\geq 0}, \mathbb{R}]$, by Weierstrass's theorem [Sim63]. This would produce the same set $\mathcal{C}_\alpha[\mathbb{R}^{\geq 0}, \mathbb{R}]$ of computable elements of $\mathcal{C}[\mathbb{R}^{\geq 0}, \mathbb{R}]$. (This is easily proved by showing that the basic functions in each of these two systems — rational zigzag functions and rational polynomial functions — are effective local uniform limits of basic functions in the other system.)

This example will be used again in §5.2, Example 1(i), dealing with analog networks.

The following lemma is used in the proof of Theorem A.

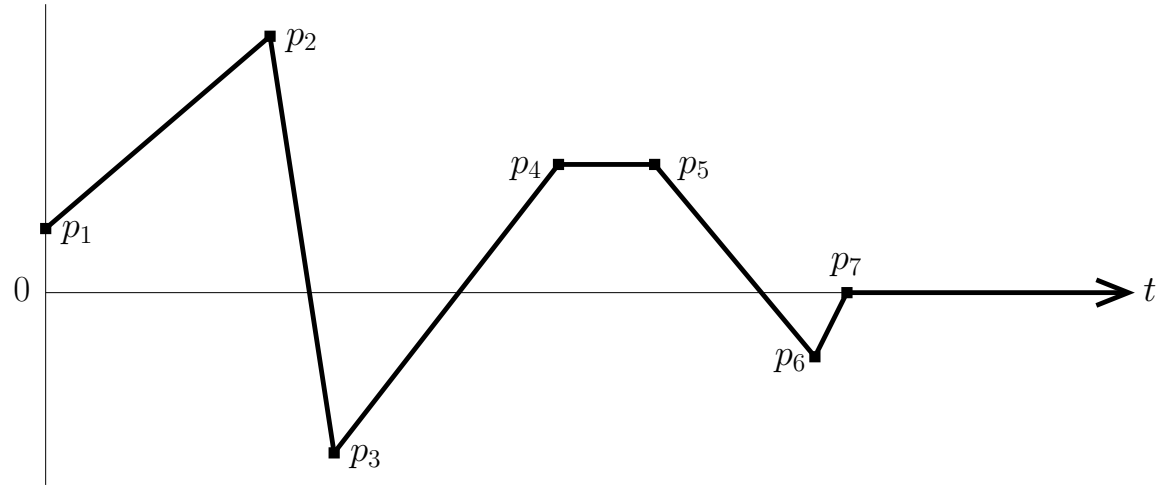

FIGURE 1: Zigzag function (points $p_1, \ldots, p_7$ are rational)

Lemma 3.2.13. *If effective locally uniform continuity* (*Definition* 3.1.3) *holds for α-streams, then it also holds for $\overline{\alpha}$-streams.*

This is proved by adapting the proof for preservation of effective (globally) uniform continuity under effective (globally) uniform convergence in [PER89, Ch. 0, Thm 4].

We are ready for the first of the two main theorems of this paper.

Theorem A ($\overline{\alpha}$-computability of FP).
Suppose the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$ is represented by an enumeration α such that

(i) *α is regular* (*Definition* 3.1.5), *and*

(ii) *$\overline{\alpha}$ is Σ-effective* (*Definition* 3.2.8).

Given stream operators F and $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ as in (2.7) *and* (2.8), *open $U \subseteq A^{r+s} \times \mathcal{C}[\mathbb{T}, A]^p$, and* (*as in* (2.9)) *families of contraction moduli $\boldsymbol{\lambda} = \langle \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ and increments $\boldsymbol{\tau} = \langle \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$, suppose for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$*

(iii) $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \in \boldsymbol{Contr}(\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$,

(iv) *$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is causal,*

(v) *F is $\overline{\alpha}$-computable,*

(vi) *$\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are $\overline{\alpha}$-computable.*

Let

$$\Phi \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \rightharpoonup \mathcal{C}[\mathbb{T}, A]^m \tag{3.11a}$$

be the unique fixed point function for F given by Theorem 1 *with $\boldsymbol{dom}(\Phi) = U$, so that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$,*

$$\Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) = \mathsf{FP}(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}). \tag{3.11b}$$

Then Φ is $\overline{\alpha}$-computable.

Proof (outline): It is a lengthy but straightforward exercise to show that the fixed point of F, constructed according to the proof of Theorem 1 [TZ11, §3], is computable in $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$, under the given assumptions on the $\overline{\alpha}$-computability of F, $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$, and assuming we begin with an α-computable stream tuple $\boldsymbol{u}_0$ such that $\boldsymbol{v}_1^{(0)} = F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}(\boldsymbol{u}_0)$. The important thing to show is that the double sequence of stream tuples $\boldsymbol{v}_k^{(n)}$ is *computable in* k and n, as well as $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$. At $k\tau$-approximate limits ($\tau = \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$, $k = 1, 2, \dots$) we take effective "diagonal approximating sequences" as in the proof of Lemma 3.2.3, to show that Φ is $\overline{\alpha}$-computable. Some details are given in [TZ07] (for a stronger definition of contracting operators).

One point worth noting is that we have to check that $\overline{\alpha}$-*computability*, as well as *effective local uniform continuity*, are preserved by these iterated sequences and their limits. For this we must use a property of sequences of streams stronger than local uniform convergence, namely *effective local uniform convergence* of α-computable streams. □

Remark 3.2.14 (Type of F for $\overline{\alpha}$-computability). The assumption of $\overline{\alpha}$-computability of F in Theorem A and elsewhere is with respect to the uncurried typing in (2.7). (Cf. [TZ11, Remark 4.1.1].)

Corollary 3.2.15. *If, in Theorem A, we add the assumption:*

(vii) *U is $\overline{\alpha}$-semicomputable,*

then the conclusion can be strengthened to:

Then Φ is strictly $\overline{\alpha}$-computable.

Proof: The distinction between $\overline{\alpha}$-computability and *strict* $\overline{\alpha}$-computability of f vanishes in the case that $\boldsymbol{dom}(f)$ is $\overline{\alpha}$-semicomputable, i.e., the domain of a strictly $\overline{\alpha}$-computable function [TZ04, Lemma 10.2.4]. □

3.3 Relative $\overline{\alpha}$-computability

We want to develop a concept of "relative $\overline{\alpha}$-computability", so that Theorem A could have a more general form, in which assumption (v) is *deleted*, (vi) is *replaced* by (something like)

(v′) *$\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are $\overline{\alpha}$-computable relative to F,*

and the conclusion is changed to (something like)

Then Φ is $\overline{\alpha}$-computable relative to F.

In order to do this, we must re-define the objects of our computation theory so that the $\overline{\alpha}$-tracking functions are made explicit. (The reason for this will emerge below — see Remark 3.3.4.) So we define:

Definition 3.3.1 ($\overline{\alpha}$-tracked function).

(a) An ($\overline{\alpha}$-)*tracked function* on $\mathcal{C}[\mathbb{T}, A]$ is a pair (f, φ) where f is a function on $\mathcal{C}[\mathbb{T}, A]$ and φ is a tracking function for f.

(b) A *strictly* ($\overline{\alpha}$-)*tracked function* is defined similarly, with the added condition that φ is a *strict* tracking function for f.

Suppose (f, φ) is an $\overline{\alpha}$-tracked function on $\mathcal{C}[\mathbb{T}, A]$. We can think of φ as a *concrete implementation* of f (whether computable or not). Hence the objects of our study here are not simply functions f on $\mathcal{C}[\mathbb{T}, A]$, but rather functions-together-with-implementations (f, φ).

Note also that although f is uniquely determined by φ, φ is not uniquely determined by f. In fact (assuming $\boldsymbol{dom}(f)$ is infinite and ignoring considerations of computability of φ) if f has one tracking function then it has uncountably many (even strict) tracking functions.

Definition 3.3.2 (Relative $\overline{\alpha}$-computability). Given two $\overline{\alpha}$-tracked functions (f, φ) and (g, ψ) on $\mathcal{C}[\mathbb{T}, A]$, we say that (f, φ) is $\overline{\alpha}$-*computable in* (or *relative to*) (g, ψ) if φ is computable in ψ.

Lemma 3.3.3 (Transitivity of relative $\overline{\alpha}$-computability).
Suppose (f, φ) and (g, ψ) are $\overline{\alpha}$-tracked functions on $\mathcal{C}[\mathbb{T}, A]$.

(a) *If (f, φ) is $\overline{\alpha}$-computable in (g, ψ), and (g, ψ) is $\overline{\alpha}$-computable in (h, θ), then (f, φ) is $\overline{\alpha}$-computable in (h, θ).*

(b) *If (f, φ) is $\overline{\alpha}$-computable in (g, ψ), and (g, ψ) is $\overline{\alpha}$-computable, (f, φ) is $\overline{\alpha}$-computable.*

Remark 3.3.4. The need to prove this transitivity lemma is the reason we formulated our $\overline{\alpha}$-computability theory in terms of $\overline{\alpha}$-tracked functions.

Theorem A can then be re-formulated in terms of relative computability:

Theorem $\mathbf{A}^{\mathsf{rel}}$ (Relative $\overline{\alpha}$-computability of FP).
Suppose the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$ is represented by an enumeration α such that

(i) *α is regular (Definition 3.1.5), and*

(ii) *$\overline{\alpha}$ is Σ-effective (Definition 3.2.8).*

Given stream operators F as in (2.7) with $\overline{\alpha}$-tracking function φ, and $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ as in (2.8), open $U \subseteq A^{r+s} \times \mathcal{C}[\mathbb{T}, A]^p$, and (as in (2.9)) families of contraction moduli $\boldsymbol{\lambda} = \langle \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ and increments $\boldsymbol{\tau} = \langle \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$, with $\overline{\alpha}$-tracking functions ψ and θ respectively, suppose for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$

(iii) *$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \in \boldsymbol{Contr}(\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$,*

(iv) *$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is causal,*

(v) *$(\boldsymbol{\lambda}, \psi)$ and $(\boldsymbol{\tau}, \theta)$ are $\overline{\alpha}$-computable in (F, φ).*

Let

$$\Phi \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \rightharpoonup \mathcal{C}[\mathbb{T}, A]^m$$

be the unique fixed point function for F given by Theorem 1, with $\boldsymbol{dom}(\Phi) = U$, so that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$,

$$\Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \;=\; \mathsf{FP}(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}).$$

Then Φ (together with a suitable $\overline{\alpha}$-tracking function) is $\overline{\alpha}$-computable in (F, φ).

We omit the proof, which is a "relativised" version of the proof of Theorem A.

Theorem A follows immediately from this, by Lemma 3.3.3(b).

Similarly, there is a relativised version of the "strictly $\overline{\alpha}$-computable" version of Theorem A (Corollary 3.2.15).

4 Abstract computability in $\mathcal{C}[\mathbb{T}, A]$

In this section we investigate abstract computability in $\mathcal{C}[\mathbb{T}, A]$, using the imperative programming language ***WhileCC**** over the algebra $\mathcal{C}[\mathbb{T}, A]$.

At the heart of all our abstract models of computability are algebraic structures. The abstract models esentially schedule the basic operations and relations of these algebraic structures in ways that are independent of the representation of the data and the implementation of these operations and relations. In the case of an imperative abstract model, algebraic terms are evaluated by assignments and control structures such as 'while'. In topological algebras, abstract models typically do not compute directly all the concretely computable functions; rather they effectively approximate them to arbitrary precision [TZ99].

In [TZ04, TZ05] we investigated, and compared, abstract and concrete models of computability on metric algebras. The concrete model considered was $\overline{\alpha}$-computability, as described in the previous section, and the abstract model was approximable ***WhileCC****(Σ) computability (for the appropriate signature Σ), to be described below. A completeness theorem was proved, asserting their equivalence under quite general conditions.

In the previous section we proved (Theorem A) *concrete computability* of the fixed point $\mathsf{FP}(F)$ of a stream transformer F, assuming concrete computability of F. From this, and a version of the completeness theorem in [TZ04], we will in turn infer (Theorem B) *abstract computability* of $\mathsf{FP}(F)$, assuming abstract computability of F.

4.1 Expanding the stream algebra $\mathcal{C}[\mathbb{T}, A]$ to $\mathcal{C}[\mathbb{T}, A](\boldsymbol{\alpha}_{\mathsf{c}})$

Recall the definitions of the stream algebra $\mathcal{C}[\mathbb{T}, A]$ (2.3) and its signature Σ (2.4).

In order to have a satisfactory abstract model of computing on $\mathcal{C}[\mathbb{T}, A]$, which will satisfy the completeness theorem for abstract vs concrete computability, we must expand its algebraic structure and signature. This is done in three ways:

(1) Since the boolean datatype B is needed for abstract computation models, specifically boolean tests in high level programming languages, we adjoin to Σ the boolean sort B and the standard boolean operations $(\wedge,\ \vee,\ \neg)$ as well as equality on some of the sorts, such as N. Correspondingly, we adjoin to $\mathcal{C}[\mathbb{T}, A]$ the set $\mathbb{B}$ of booleans and the boolean operations.

(2) We add operations and constants at each sort. This will be explained in detail in Discussion 4.4.8.

(3) We add the enumeration function $\alpha_{\mathsf{c}} \colon \mathbb{N} \to \mathcal{C}[\mathbb{T}, A]$ as a basic algebraic operation.

The motivations for these expansions to $\mathcal{C}[\mathbb{T}, A]$, with an example, will be given in Discussion 4.4.8.

We will denote the expanded algebra by $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$, with signature $\Sigma(\alpha_{\mathsf{c}})$.

4.2 *WhileCC**(α_{c}) computability

The syntax and semantics of the ***WhileCC**** programming language are discussed in detail in [TZ04, TZ05]. To give a brief review: it extends the ***While**** language (i.e., the 'while' programming language with arrays) with a new 'choose' assignment construct to model *nondeterministic countable choice*.

The syntax of the ***WhileCC**** statements is defined essentially (following the version in [TZ05]) by extending the *assignment statement* with a new case:

$$\mathtt{x} \;:=\; \mathsf{choose}\ \mathtt{z} : P(\mathtt{z}, \dots)$$

where $\mathtt{x}$ and $\mathtt{z}$ are variables of sort nat, and $P(\mathtt{z}, \dots)$ is a *semicomputable predicate* of $\mathtt{z}$ (and other variables), i.e., the halting set of a ***WhileCC**** boolean-valued procedure with $\mathtt{z}$ among its input variables.

Intuitively, '$\mathsf{choose}\ \mathtt{z} : P$' selects *some* value n such that $P(n, \dots)$ is true, if any such n exists (and is undefined otherwise). Any concrete model will select a particular such n, according to the implementation. In our abstract semantics, the meaning is given as *the set of all possible such n's* (hence "countable choice"), together (possibly) with the divergence symbol '$\uparrow$'.

We then write ***WhileCC****(α_{c}) computability to mean ***WhileCC**** computability on $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$.

4.3 *WhileCC**(α_{c}) approximability

The basic notion of abstract computability that we will be working with, for the sake of comparison with concrete computability on $\mathcal{C}[\mathbb{T}, A]$, is ***WhileCC**** *approximable computability*, or ***WhileCC**** *approximability*, on $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$, which we now define.

Here we write C for any Σ-algebra, with carriers C_s for the Σ-sorts s, and product spaces $C^{\,u} = C_{s_1} \times \cdots \times C_{s_m}$ for product types $u = s_1 \times \cdots \times s_m$. Thereafter we will apply these definitions to the special case $C = \mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$.

Let u be a Σ-product type and s a Σ-sort. Let $P : \mathsf{nat} \times u \;\to\; s$ be a ***WhileCC**** procedure. Then the semantics of P is given by the multivalued function

$$P_n^C \;=_{df}\; P^C(n, \cdot) : \; C^{\,u} \;\rightrightarrows\; C_s^{\uparrow}$$

where $C_s^{\uparrow} =_{df} C_s \cup \{\uparrow\}$ and '$\rightrightarrows$' means that P_n^C is multivalued, i.e., for all $x \in C^{\,u}$, $P_n^C(x)$ is a (non-empty) countable subset of $C_s^{\uparrow}$. Now let $f : C^{\,u} \rightharpoonup C_s$ be a single-valued partial function on $\mathcal{C}$. Then we define:

(a) f is ***WhileCC**** *approximable* by P on C iff for all $n \in \mathbb{N}$ and all $x \in C^{\,u}$:

$$x \in \boldsymbol{dom}(f) \;\implies\; \uparrow \notin P_n^C(x) \;\subseteq\; \mathbf{B}(f(x), 2^{-n})$$

where the open ball $\mathbf{B}(\,\cdot\,,\,\cdot\,)$ is defined with respect to the metric on C_s.

(b) f is *strictly* ***WhileCC**** *approximable* by P on C iff in addition to (a),

$$x \notin \boldsymbol{dom}(f) \implies P_n^C(x) = \{\uparrow\}.$$

Now, in the particular case that $C = \mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$, we write ***WhileCC****(α_{c}) approximability to mean ***WhileCC**** approximability on $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$.

Remark 4.3.1 (Equivalence of ordinary and strict *WhileCC approximability).** The distinction between ***WhileCC****(α_{c}) approximability and *strict* ***WhileCC****(α_{c}) approximability of f vanishes in the case that $\boldsymbol{dom}(f)$ is ***WhileCC****(α_{c}) semicomputable [TZ04, §9.3]. This is again the case in our analog network example (cf. Corollary 3.2.15).

4.4 Concrete and abstract computability compared; Completeness theorem

We are ready to make the connection between concrete and abstract computability on stream algebras.

We will re-state the completeness theorem of [TZ04] as it applies to the stream algebra $\mathcal{C}[\mathbb{T}, A]$. For this we need an important concept.

Definition 4.4.1 (Open exhaustion). Let U be an open subset of $A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$. An *open exhaustion of* U is a sequence (U_ℓ) of open subsets of $A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$ such that

$$U_1 \subseteq U_2 \subseteq U_3 \subseteq \ \ldots \qquad \text{and} \qquad \bigcup_{\ell=1}^{\infty} U_\ell \ = \ U.$$

We also need an *effective* notion of open exhaustion.

Definition 4.4.2 (*WhileCC(α_{c})-effective open exhaustions).** An open exhaustion (U_ℓ) of $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$ is ***WhileCC****(α_{c})-*effective* in $\mathcal{C}[\mathbb{T}, A]$ if it satisfies the following two conditions:

(a) (***WhileCC****(α_{c})-*effective Archimedean property of* (U_ℓ) *in* U.)
There is a ***WhileCC****(α_{c}) procedure

$$P_{\mathsf{loc}} \colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \ \rightrightarrows \ \mathbb{N}^{\uparrow}$$

which, given $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$, produces some ℓ which "locates" $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U_\ell$; more precisely:

$$P_{\mathsf{loc}}(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \ = \ \begin{cases} \{\, \ell \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U_\ell \,\} & \text{if } \ (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \\ \{\uparrow\} & \text{otherwise.} \end{cases}$$

Typically, the procedure $P_{\mathsf{loc}}(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$ is realised in the form of a construct

$$\mathsf{choose}\ \ell : \text{“}(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U_\ell\text{”}.$$

(b) (***WhileCC****(α_{c})*-effective openness of* (U_ℓ).)
There is a ***WhileCC****(α_{c}) computable function

$$\gamma\colon \mathbb{N} \times \mathbb{T} \times A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \rightharpoonup \mathbb{N}$$

such that for all $\ell \in \mathbb{N}$, $\tau \in \mathbb{T}$ and $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U_\ell$,

$$\gamma(\ell, \tau, \boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \downarrow k \text{ for some } k \text{ such that } \mathbf{N}_{k\tau}((\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}), 2^{-k}) \subseteq U_\ell$$

where $\mathbf{N}_{k\tau}(\dots)$ is the open neighbourhood of $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$ determined by the pseudometric $\mathsf{d}_{k\tau}$ in $\mathcal{C}[\mathbb{T}, A]$.

Remark 4.4.3. If U has a ***WhileCC****(α_{c})-effective open exhaustion, then U is ***WhileCC****(α_{c}) semicomputable [TZ04].

Now we want to define the concept of *local uniform continuity* with respect to some open exhaustion, as well as an *effective version* of this. The most suitable form of these definitions for our purposes is in terms of the pseudometrics $\mathsf{d}_{k\tau}$ $(k = 0, 1, 2, \dots)$. (See Remark 3.2.6 concerning the value of τ.)

For the rest of this subsection, assume

$$f\colon A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p \rightharpoonup \mathcal{C}[\mathbb{T}, A]^m$$

and let (U_ℓ) be a ***WhileCC****(α_{c})-effective open exhaustion of $U =$ ***dom***(f). It follows from Remarks 4.3.1 and 4.4.3 that ***WhileCC****(α_{c})-approximability of f is equivalent to *strict* ***WhileCC****(α_{c})-approximability of f.

Definition 4.4.4 (Local uniform continuity). f is *locally uniformly continuous* with respect to (U_ℓ) iff for all ℓ, n, τ there exists j such that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$, $(\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}') \in U_\ell$,

$$\mathsf{d}_{j\tau}((\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}), (\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}')) < 2^{-j} \implies \mathsf{d}_{n\tau}(f(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}), f(\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}')) < 2^{-n}.$$

This concept is *made effective* by taking j to be recursive in ℓ, n.

Next, we use a modified version of [TZ11, Lemma 3.2.10] as a test for local uniform continuity:

Lemma 4.4.5 (Test for local uniform continuity). *Suppose f is causal, and for all $\boldsymbol{u} \in \mathcal{C}[\mathbb{T}, A]^m$ and all k, ℓ, n, τ there exists j such that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})$, $(\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}') \in U_\ell$*

$$\mathsf{d}_{k\tau}((\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}), (\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}')) < 2^{-j} \implies \mathsf{d}_{k\tau}(f(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}), f(\boldsymbol{c}', \boldsymbol{a}', \boldsymbol{x}')) < 2^{-n}.$$

Then f is locally uniformly continuous w.r.t. (U_ℓ).

This test is made effective *by taking j to be recursive in k, ℓ, n.*

We are ready for the completeness theorem.

Theorem 4.4.6 (Completeness for abstract vs concrete computability on $\mathcal{C}[\mathbb{T}, A]$). *Suppose the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$ is represented by an enumeration α such that*

(i) *α is regular* (*Definition* 3.1.5),

(ii) *$\overline{\alpha}$ is $\Sigma(\alpha_{\mathsf{c}})$-effective* (*Definition* 3.2.8), *and*

(iii) *for all*[12] *sorts s, α_s is **WhileCC***(α_{c}) approximable on $\mathcal{C}[\mathbb{T}, A]$.*

Let (U_ℓ) be an open exhaustion of $U = \mathbf{dom}(f)$ such that

(iv) *(U_ℓ) is **WhileCC***(α_{c})-effective* (*Definition* 4.4.2), *and*

(v) *f is effectively locally uniformly continuous w.r.t. (U_ℓ)* (*Definition* 4.4.4).

Then

$$f \text{ is } \mathbf{WhileCC^*}(\alpha_{\mathsf{c}}) \text{ approximable on } \mathcal{C}[\mathbb{T}, A] \iff f \text{ is } \overline{\alpha}\text{-computable on } \mathcal{C}[\mathbb{T}, A].$$

Proof: From Theorem C in [TZ04, §10]. □

Remark 4.4.7 (Use of assumptions (ii) and (iii) in the completeness theorem). Assumptions (ii) and (iii) are each crucial in proving one of the two directions of the completeness theorem. Assumption (ii) ($\Sigma(\alpha_{\mathsf{c}})$-*effectivity of* $\overline{\alpha}$) means, roughly, that the enumerations $\overline{\alpha}$ are "strong" enough to compute the basic functions of $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$, and hence also functions **WhileCC***(α_{c}) approximable in them:

$$\mathbf{WhileCC^*}(\alpha_{\mathsf{c}}) \text{ approx.} \implies \overline{\alpha}\text{-comp.}$$

Assumption (iii) (**WhileCC***(α_{c}) *approximability of* α) means, conversely, that **WhileCC*** approximability w.r.t. the basic functions of $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$ is "strong" enough to compute the enumerations $\alpha_s \colon \mathbb{N} \to C_s$, and hence also $\overline{\alpha}$-computable functions:

$$\mathbf{WhileCC^*}(\alpha_{\mathsf{c}}) \text{ approx.} \impliedby \overline{\alpha}\text{-comp.}$$

Discussion 4.4.8 (Expanding $\mathcal{C}[\mathbb{T}, A]$ to $\mathcal{C}[\mathbb{T}, A](\boldsymbol{\alpha_{\mathsf{c}}})$: Explanation and example). In connection with the completeness theorem above, let us return to the three points listed in §4.1, and consider each of them in turn.

(1) As stated above, the boolean datatype B is needed in order to include boolean tests in **WhileCC*** programs.

(2) The point here is to expand the algebraic structures at the various sorts s by adding enough basic functions to ensure that the enumerations α_s of C_s are **WhileCC***(α_{c}) approximable (assumption (iii)). In more detail, we add the following:

- For A: this depends on the set A. Let us take the most important case for our examples, namely $A = \mathbb{R}$. We then adjoin the following constants and operations over the reals: 0, 1, +, −, ×, as well as the (continuous, partial) inverse $\mathsf{inv}^{\mathsf{R}} \colon \mathbb{R} \to \mathbb{R}$, where

$$\mathsf{inv}^{\mathsf{R}}(x) \;=\; \begin{cases} 1/x & \text{if } \; x \neq 0 \\ \uparrow & \text{if } \; x = 0, \end{cases}$$

[12] Actually here we only need sorts $s \neq \mathsf{C}$, since α_{c} is trivially a basic operation of $\mathcal{C}[\mathbb{T}, A](\alpha_{\mathsf{c}})$.

and the (continuous, partial) equality and order operations $\mathsf{eq}_{\mathsf{R}}, \mathsf{less}_{\mathsf{R}} \colon \mathbb{R}^2 \rightharpoonup \mathbb{B}$ where

$$\mathsf{eq}_{\mathsf{R}}(x,y) \;=\; \begin{cases} \uparrow & \text{if } \; x = y \\ \mathbb{f} & \text{if } \; x \neq y, \end{cases} \qquad \text{and} \qquad \mathsf{less}_{\mathsf{R}}(x,y) \;=\; \begin{cases} \mathbb{t} & \text{if } \; x < y \\ \mathbb{f} & \text{if } \; x > y \\ \uparrow & \text{if } \; x = y. \end{cases}$$

The significance of these partial inverse, equality and order operations, in connection with computability and continuity, is discussed in [TZ04].

Note also that in connection with the use of this structure on the reals in the completeness theorem and Theorem B, we could just as well replace the (partial) real inverse operation $\mathsf{inv}^{\mathsf{R}}$ by the (total) natural inverse operation $\mathsf{inv}^{\mathsf{N}} \colon \mathbb{N} \to \mathbb{R}$, where

$$\mathsf{inv}^{\mathsf{N}}(n) \;=\; \begin{cases} 1/n & \text{if } \; n \neq 0 \\ 0 & \text{if } \; n = 0. \end{cases}$$

which is continuous, like *any* function on the discrete space $\mathbb{N}$.

- For R: this datatype is expanded in exactly the way described above for A (assuming $A = \mathbb{R}$).
- For N: 0 and successor, and (continuous, total) equality and order on $\mathbb{N}$:

$$\mathsf{eq}_{\mathsf{N}}(m,n) \;=\; \begin{cases} \mathbb{t} & \text{if } \; m = n \\ \mathbb{f} & \text{otherwise}, \end{cases} \qquad \text{and} \qquad \mathsf{less}_{\mathsf{N}}(m,n) \;=\; \begin{cases} \mathbb{t} & \text{if } \; m < n \\ \mathbb{f} & \text{otherwise}. \end{cases}$$

- For T: this is expanded either like R (assuming $\mathbb{T} = \mathbb{R}^{\geq 0}$)[13] or like N (assuming $\mathbb{T} = \mathbb{N}$).

(3) In our first example (Example 3.2.12 above, which we will revisit in §5.2), with $\mathbb{T} = \mathbb{R}^{\geq 0}$ and $A = \mathbb{R}$, the enumeration α_{c} (see Figure 1) is easily seen to satisfy the regularity condition (i) in the completeness theorem. However, α_{c} itself is not ***WhileCC**** computable (or even approximable).[14] In fact, *none* of the streams in this algebra is ***WhileCC**** computable, not even the zigzag streams, nor even the stream with constant value 0! That is why we have to include α_{c} as one of the primitive operators on $\mathcal{C}[\mathbb{T}, A]$, i.e., work with ***WhileCC****(α_{c}) instead of ***WhileCC**** computability on $\mathcal{C}[\mathbb{T}, A]$ here and in Theorem B below.

4.5 Abstract computability of FP of contracting operator

We can now prove a theorem on abstract computability of the FP of a contracting stream space operator by combining the corresponding theorem for concrete computability (Theorem A) with the above completeness theorem.

---

[13] modified suitably for $\mathbb{R}^{\geq 0}$, e.g., redefining $x - y$ as $\max(x - y, 0)$

[14] Note that the *cartesian* form $\boldsymbol{cart}(\alpha) \colon \mathbb{N} \times \mathbb{T} \to \mathbb{R}$ of $\alpha \colon \mathbb{N} \to \mathcal{C}[\mathbb{T}, \mathbb{R}]$ *is* ***While**** computable on the field $\mathbb{R}$. This is discussed further in Section 6.

Theorem B (Approximable *WhileCC(α_{c}) computability of FP).**
Suppose the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$ is represented by an enumeration α such that

(i) *α is regular (Definition 3.1.5),*

(ii) *$\overline{\alpha}$ is $\Sigma(\alpha_{\mathsf{c}})$-effective (Definition 3.2.8), and*

(iii) *for all sorts s, α_s is **WhileCC***(α_{c}) approximable on $\mathcal{C}[\mathbb{T}, A]$.*

Given stream operators F and $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ as in (2.7) and (2.8), $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$, and families of contraction moduli $\boldsymbol{\lambda} = \langle \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ and increments $\boldsymbol{\tau} = \langle \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle$ as in (2.9), such that for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$

(iv) *$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \in \boldsymbol{Contr}(\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}, \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}})$,*

(v) *$F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is causal,*

(vi) *F is **WhileCC***(α_{c}) approximable,*

(vii) *F is shift invariant (Definition 2.2.3),*

(viii) *$\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are **WhileCC***(α_{c}) approximable, and*

(ix) *$\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are continuous, with **WhileCC***(α_{c})-computable moduli of continuity.*

Let (U_ℓ) be an open exhaustion of U such that

(x) *(U_ℓ) is **WhileCC***(α_{c}) effective (Definition 4.4.2), and*

(xi) *F is effectively locally uniformly continuous[15] w.r.t. (U_ℓ) (Definition 4.4.4).*

Let Φ be the fixed point function for F as in (2.10), given by Theorem 2, and suppose also

(xii) *U is closed under shifts w.r.t. Φ (Definition 2.2.4).*

*Then Φ is strictly **WhileCC***(α_{c}) approximable.*

Proof: The main idea is to apply the $\overline{\alpha}$-computability theorem for the FP (Theorem A), together with the completeness theorem for stream spaces (Theorem 4.4.6) in both directions, to obtain the result. The main problem here is in applying the completeness theorem to Φ in the direction

$$\Phi \textit{ is } \overline{\alpha}\textit{-computable} \quad \Longrightarrow \quad \Phi \textit{ is } \boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}}) \textit{ approximable,}$$

where we need the condition that Φ is (not just continuous, but) *effectively locally uniformly continuous* w.r.t. a suitable **WhileCC***(α_{c})-effective open exhaustion of U.

We prove this by assuming that F is effectively locally uniformly continuous w.r.t. an open exhaustion (U_ℓ), and using this to show that Φ is effectively locally uniformly continuous w.r.t. (not necessarily (U_ℓ), but) some *refinement* of (U_ℓ). This is done by a careful analysis, and adaptation, of the proof of continuity of Φ in Theorem 2 [TZ11, §4].

The main point is to control the variation of the contraction modulus $\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ and incre-

[15] This assumption of effective local uniform continuity of F is made w.r.t. the first 3 arguments of F only, i.e., $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$. No continuity assumption need be made for the fourth (stream) argument. (Cf. the footnote for condition (iv) in Theorem 2 in §2.2.)

ment $\tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ for $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$, as well as the value of

$$\begin{aligned} D_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} =_{df} \ D_1 \ &= \ \mathsf{d}_\tau(\boldsymbol{v}_1^{(0)}, \boldsymbol{v}_1^{(1)}) \\ &= \mathsf{d}_\tau(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}(\boldsymbol{u}_0), F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}(F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}(\boldsymbol{u}_0))) \end{aligned} \tag{4.1}$$

(for a fixed $\boldsymbol{u}_0$) [TZ11, (4.16)]. So, defining the function

$$\boldsymbol{D} =_{df} \ \langle D_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \mid (\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \rangle,$$

we note that by assumptions (viii) and (ix) on $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$, and (vi) and (xi) on F,

$$\begin{gathered} \textit{the functions } \boldsymbol{\lambda}, \boldsymbol{\tau} \textit{ and } \boldsymbol{D} \textit{ are all } \boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}}) \textit{ approximable and continuous,} \\ \textit{with a } \boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}}) \textit{ computable modulus of continuity.} \end{gathered} \tag{4.2}$$

Further, for each $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U$ there is some $\ell \in \mathbb{N}$ such that

$$\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \ < \ \lambda_\ell \ =_{df} \ 1 - 1/\ell \tag{4.3a}$$

and also

$$\tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \ > \ \tau_\ell \ =_{df} \ 1/\ell \tag{4.3b}$$

and also

$$D_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} \ < \ D_\ell \ =_{df} \ \ell. \tag{4.3c}$$

So we define the exhaustion

$$(V_1, \ V_2, \ V_3, \ \dots) \tag{4.4a}$$

of U, where

$$V_\ell \ =_{df} \ \{(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in U \ \mid \lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} < \lambda_\ell \ \text{ and } \ \tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} > \tau_\ell \ \text{ and } \ D_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}} < D_\ell\}. \tag{4.4b}$$

From (4.2) it follows that this exhaustion of U is open, and moreover, $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}})$ effective (Definition 4.4.2).

Now in applying the proof of continuity of Φ in Theorem 2 in [TZ11] to the present proof of effective local uniform continuity of Φ (as stated above), note the following:

(1) In the inductive proof of in [TZ11, (4.14)], in the base case, $\lambda_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$, $\tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ and $D_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ $(=D_1)$ can be replaced, respectively, by λ_ℓ, τ_ℓ and D_ℓ (4.3), which are constant over V_ℓ. Note we can also take τ_ℓ for the value of τ in 'd_τ' in (4.1).

(2) The choice of N in the inequality [TZ11, (4.17)] depends on the values of λ and D_1, which (as we have seen) can be taken as constant over V_ℓ, and hence N can also be taken as constant over V_ℓ.

(3) In the inductive step [TZ11, (4.25), (4.26)] δ_1 and δ_2 can also be taken as constant over V_ℓ (the latter by induction hypothesis).

Next, define the open exhaustion

$$(W_1,\ W_2,\ W_3,\ \dots\ )$$

of U as the *common refinement* of the open exhaustions (U_ℓ) and (V_ℓ) of U (4.4), i.e.,

$$W_\ell\ =_{df}\ U_\ell \cap V_\ell$$

for $\ell = 1, 2, \dots$. Then by assumption (xi), F is effectively locally uniformly continuous w.r.t. (U_ℓ), and hence w.r.t. (W_ℓ). Hence (in the notation in the proof of Theorem 2 in [TZ11]) we can show, by induction on k, that Φ_k is effectively locally uniformly continuous w.r.t. (W_ℓ), with modulus of continuity computable in k.

Next we apply the test in Lemma 4.4.5 to show that Φ is effectively locally uniformly continuous w.r.t. (W_ℓ). Now we apply the completeness theorem to infer that Φ is $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}})$ approximable. Finally, with the help of Remarks 4.3.1 and 4.4.3, we conclude that Φ is *strictly* $\boldsymbol{WhileCC^*}$ approximable. $\square$

Remark 4.5.1 (Type of F for $\boldsymbol{WhileCC^*}(\boldsymbol{\alpha_{\mathsf{c}}})$ approximability). The assumption of $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}})$ approximability of F in Theorem B and elsewhere is with respect to the uncurried typing in (2.7). (Cf. Remark 3.2.14.)

Remark 4.5.2 (Continuity assumption for $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$). The list of assumptions in Theorem B includes a *continuity* assumption (vii) for $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$. (Cf. [TZ11], Remark 4.2.9 and Theorem 2, where the weaker assumption of *local boundedness* was made for $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$.) This continuity assumption is used to show that the exhaustion (V_ℓ) is open (see (4.4)). In fact we could replace (ix) here by a boundedness assumption:

(vii′) $\boldsymbol{\lambda}$ *and* $\boldsymbol{\tau}$ *are effectively locally uniformly bounded w.r.t.* (U_ℓ)

where this concept is defined in the obvious way. Such an assumption would actually simplify the proof of Theorem B, by obviating the need for a refinement of the given exhaustion (U_ℓ) of U. The problem would, however, then be shifted to finding, in any particular application, an open exhaustion (U_ℓ) of U which would have all the required properties at the outset. (See also the discussion in Example 1(ii) in §5.2 below.)

Remark 4.5.3 (Relative abstract computability: Conjecture). A stronger formulation of Theorem B results from *deleting* assumption (vi), *replacing* (viii) by

(viii′) $\boldsymbol{\lambda}$ *and* $\boldsymbol{\tau}$ *are* $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}})$ *approximable relative to* F,

and changing the conclusion to:

Then Φ *is* $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{c}})$ *approximable relative to* F.

(Compare Theorem $\mathrm{A}^{\mathsf{rel}}$ in §3.3.) We conjecture that this formulation is true. One possible proof would depend on a relativised version of the abstract/concrete Completeness Theorem 4.4.6.

5 Examples

Recall that in [TZ11] we used two running examples to illustrate the theory: analog networks and synchronous concurrent algorithms (SCAs).[16] We continue with these two examples. We first note that both these examples involve networks N, with modules and channels, and make a few general remarks about module functions, and their relationship to the network state functions F^N.

5.1 Networks and modules: Some remarks

We will assume that the module functions, and (hence) the network state function F, are *total*, so as to ensure the ***Network Determinacy Assumption***:

> *For a certain domain of system parameters, initial stream values and input streams, there is a well-determined value for the stream on each channel at all times.*

This implies the *totality of the network state function* and hence of the *module functions*, and hence the *totality of the streams*. This is the case (for now) with all our examples below.

It would be interesting to consider the consequences of dropping thes totality assumptions. We will revisit this point in §6.3(3).

For convenience, we repeat Remark 4.1.6 in [TZ11].

Remark 5.1.1 (Network state function just vectorisation of module functions). In the case of an analog network N, the fact that the network state function F^N is formed from the *module functions* of N by simple vectorisation means that many interesting properties of the module functions, such as continuity and computability, are easily seen to be inherited by F^N.

We now give two applications of this remark to Theorems A and B.[17]

Remark 5.1.2 (Theorems A and B in terms of module functions). Suppose F is the network state function for a network N. Then

(a) Theorem A holds if assumption (v) is *replaced* by:
(v′) *the module functions of N are $\overline{\alpha}$-computable.*

(b) Theorem B holds if in any of assumptions (v), (vi) and (vii), "F" is replaced by "the module functions of N".

Similarly, the "relativised" versions of Theorems A and B (cf. Theorem A^{rel} in §3.3 and Remark 4.5.3) could (presumably) be re-formulated in terms of module functions.

---

[16] See §§2.1.1, 4.4.4 and 5.2.13 in [TZ11].

[17] See Remark 5.2.11 in [TZ11] for an application to Theorem 2.

5.2 Two examples

• ***Example 1: Analog networks***

In this example, F represents a state function for a network with r parameters $\boldsymbol{c} \in A^r$, s initial values $\boldsymbol{a} \in A^s$, p input channels with input streams $\boldsymbol{x} \in \mathcal{C}[\mathbb{T}, A]^p$, and m modules. The output channels of the network will form a subset of the m module output channels. For simplicity, we can assume that *all* the module output channels are also network output channels. The input/output function for the network, or *network function*, will then be the fixed point function Φ (2.10).

(i) *Concrete computability; Applying Theorem A*

Here $\mathbb{T} = \mathbb{R}^{\geq 0}$, with the standard exhaustion $K_k = [0, k\tau]$, where τ is the relevant value of the contraction increment $\tau_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$. Suppose also $A = \mathbb{R}$ and α_{A} is a standard enumeration of $\mathbb{Q} \subset \mathbb{R}$. The construction of a countable, dense and effectively locally uniformly continuous subset Z_{C} of $\mathcal{C}[\mathbb{R}^{\geq 0}, \mathbb{R}]$ was described above in Example 3.2.12.

As a particular case of this example, consider (Figure 2) the mass/spring/damper case study described in [TZ07], with the "improved" network N_2 constructed in [JZ12, §3.1.2] (cf. also [TZ11, Example 3.3.5(1)][18]).

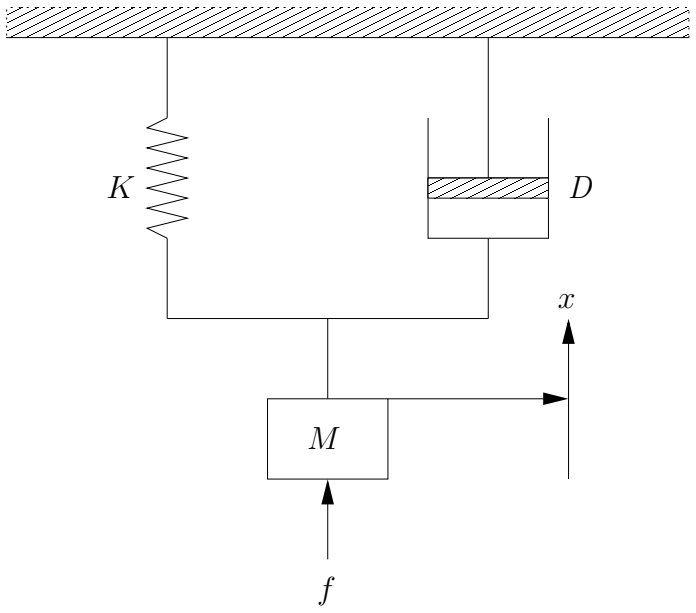

FIGURE 2: Case Study: Mass/spring/damper system

Regarding the assumptions of Theorem A: assumption (i) (regularity of α) was noted in Example 3.2.12. Assumption (ii) (Σ-effectivity of $\overline{\alpha}$) amounts to $\overline{\alpha}$-computability of the basic functions ($\mathsf{d}_{\mathbb{T}}$, d_A, D, eval) of the Σ-algebra $\mathcal{C}[\mathbb{T}, A]$, which can easily be checked.

[18] Note that on p. 3390, line 10 of [TZ11], the reference should be to eqn (3.9c), not (3.9a).

Assumption (iii) (the contracting property) was proved for this case in [TZ11, Example 4.2.13(1)]. Regarding assumptions (iv) and (vi): causality is discussed in [TZ11, Example 3.4.9(1)], and $\overline{\alpha}$-computability of $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ is obvious from their definitions [TZ11, (4.27)].

For assumption (v): The $\overline{\alpha}$-computability of the module functions used in this case study (pointwise addition, scalar multiplication and integration) is proved in [TZ07, Sec. 5]. Note that *effective locally uniform continuity* of $(Z_{\mathsf{C}}, \alpha_{\mathsf{C}})$ (part of assumption (ii)) is used to prove $\overline{\alpha}$-computability of integration[19]. This gives assumption (v) — or, more simply, assumption (v′) (see Remark 5.1.2).

Hence Theorem A, as well as Theorem 2 (as discussed in [TZ11, Example 4.2.13(1)]) can be applied to the system shown in Figure 2, to give:

Proposition 1. *In the system in Figure 2, the displacement x is continuous and $\overline{\alpha}$-computable as a function of the mass M, spring constant K, damping constant D, initial displacement x_0, initial velocity v_0, external force f and time t.*

Note that the "function" referred to in the statement of this proposition is essentially the *cartesian form* of the fixed point function Φ (2.10, 3.11) constructed in Theorems 2 and A, where if

$$\Phi\colon U \;\to\; \mathcal{C}[\mathbb{T}, A]^m$$

then

$$\boldsymbol{cart}(\Phi)\colon U \;\times \mathbb{T} \;\to\; A^m \tag{5.1a}$$

is defined by

$$\boldsymbol{cart}(\Phi)(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}, t) \;=\; \Phi(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x})(t). \tag{5.1b}$$

This is discussed further in §6.2 below.

We return to the subject of cartesian forms in considering abstract computability for this example (below), as well as in Example 2 (SCAs), and in Section 6.

The second case study from [TZ07], for an iterated mass/spring/damper system, can be handled similarly.

(ii) *Abstract computability: Applying Theorem B*

Most of the conditions in Theorem B have already been checked in part (i) above for Theorem A. The "new" conditions (ii) ($\Sigma(\alpha_{\mathsf{C}})$-effectivity of $\overline{\alpha}$), (iii) ($\boldsymbol{WhileCC^*}(\alpha_{\mathsf{C}})$ approximability of α), (vii) (shift invariance), and (ix) (closure of U under shifts) can easily be checked.

The important thing to check in our list of assumptions in Theorem B turns out to be (xi): the *effective local uniform continuity* of the network state function F with respect to a suitable open exhaustion (U_ℓ) of $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$, since we must find an open exhaustion (U_ℓ) which works correctly for the particular network under consideration. We return to this below.

---

[19] Cf. the proof of Theorem 5 in [PER89, Ch. 0]

Now if all the module functions are effectively *globally* uniformly continuous, then so is F by Remark 5.1.1, and there is no problem.

However the assumption of global uniform continuity does not always hold. Consider the functions associated with the three classical examples of modules in our two case studies : *addition*, *scalar multiplication* and *integration*. Of these, the first and third are (effectively) globally uniformly continuous. However *scalar multiplication* of a function:

$$\mathsf{F}\colon \mathbb{R}^{>0} \times \mathcal{C}[\mathbb{T}, \mathbb{R}] \;\to\; \mathcal{C}[\mathbb{T}, \mathbb{R}]$$

with $\mathsf{F}(c, u)(t) = c \cdot u(t)$, is (just like multiplication of reals on $\mathbb{R}$) *not* globally uniformly continuous on its domain. It is *locally* uniformly continuous with respect, e.g., to the exhaustion (U_ℓ) of $\mathbb{R}^{>0} \times \mathcal{C}[\mathbb{T}, \mathbb{R}]$, where $U_\ell =_{df} \{\, x \in \mathbb{R}^{>0} \mid x < \ell \,\} \times \mathcal{C}[\mathbb{T}, \mathbb{R}]$.

These considerations motivate the following, in the general case for Example 1. Let $U \subseteq A^r \times A^s \times \mathcal{C}[\mathbb{T}, A]^p$, and let F and $F_{\boldsymbol{c,a,x}}$ be as in equations (2.7) and (2.8).

To ensure local uniform continuity of F with respect to a suitable open exhaustion, we try to construct, by inspection, such an exhaustion

$$(U_1,\; U_2,\; U_3,\; \dots\,)$$

of $A^{r+s} \times \mathcal{C}[\mathbb{T}, A]^p$, where

$$U_\ell \;=_{df}\; A_{\ell,1} \times \cdots \times A_{\ell,r+s} \times \mathcal{C}[\mathbb{T}, A]^p \;\subseteq\; A^{r+s} \times \mathcal{C}[\mathbb{T}, A]^p$$

for suitable $A_{\ell,i} \subseteq A$.

Returning to the special case of the mass/spring/damper problem (Figure 2) with network N_2 [TZ11, Example 3.3.5(1)]: we have $A = \mathbb{R}$ and the network state function has the form

$$F = F^{N_2} :\; (\mathbb{R}^{>0})^3 \times \mathbb{R}^2 \times \mathcal{C}[\mathbb{T}, \mathbb{R}] \times \mathcal{C}[\mathbb{T}, \mathbb{R}]^2 \;\to\; \mathcal{C}[\mathbb{T}, \mathbb{R}]^2$$

where we will use the notation

$$F(M,\, K,\, D,\, x_0,\, v_0,\, f,\, x,\, v) \;=\; (y,\, w)$$

with the system parameters M (mass), K (spring constant) and D (damping constant); initial values x_0 (initial displacement) and v_0 (initial velocity); input stream f (external force), and remaining streams x (displacement) and v (velocity), with corresponding outputs y and w respectively. Each of the last two is produced [TZ11, eqns (3.11)] by one of the module functions:

$$y(t) \;=\; \mathsf{F}_x(x_0, v)(t) \;=\; \int_0^t v(s)ds \,+\, x_0 \tag{5.2a}$$

$$w(t) \;=\; \mathsf{F}_v(M, K, D, v_0, f, x, v)(t) \;=\; \frac{1}{M}\int_0^t (f(s) - Kx(s) - Dv(s))ds \,+\, v_0. \tag{5.2b}$$

For convenience, we will use, for now, the word "parameters" to refer to the system parameters M, K, D, initial values x_0, v_0 and input stream f.

In calculating the contraction domain U in this example, the system of equations originally used for this example in [TZ07] produced a contraction domain

$$U \;=\; \{\, (M,\, K,\, D,\, x_0,\, v_0,\, f) \in (\mathbb{R}^{>0})^3 \times \mathbb{R}^3 \mid M > \max(K, 2D) \,\}.$$

In [Jam12, JZ12], it was shown how, by changing the system of equations to the form shown in (5.2) above (by eliminating the acceleration parameter), the result could be improved to produce a contraction domain of

$$U \;=\; (\mathbb{R}^{>0})^3 \times \mathbb{R}^3.$$

(Some details are given in [TZ11, Example 3.3.5(1)].)

We must still show that F is *effectively locally uniformly continuous* as a function from the parameters (only!) $(M,\, K,\, D,\, x_0,\, v_0,\, f)$, to the outputs $(y,\, w)$.[20] In the following computation, therefore, we must consider other values of the parameters $(M',\, K',\, D',\, x_0',\, v_0',\, f')$, with corresponding outputs $(y',\, w')$, and changes in their values $\delta M = |M' - M|$, $\delta x_0 = |x_0' - x_0|$, $\delta f = |f' - f|$, etc. (meaning, in the last case, pointwise subtraction).

However we hold x and v fixed, i.e., $\delta x = \delta v = 0$.

Hence, for any $T > 0$, writing $\|u\|_T = \sup_{t \le T} |u(t)|$ for the pseudonorm [Roy63] corresponding to the pseudometric d_T, we have from (5.2):

$$\|\delta y\|_T \;=\; \delta x_0 \tag{5.3a}$$

$$\begin{aligned} \|\delta w\|_T \;\lesssim\; & \frac{T}{M}\Big(\|\delta f\|_T \,+\, \|x\|_T\, \delta K \,+\, \|v\|_T\, \delta D\Big) \,+ \\ & +\; T \cdot \frac{\delta M}{M^2}\Big(\|f\|_T \,+\, K\|x\|_T \,+\, D\|v\|_T\Big) \end{aligned} \tag{5.3b}$$

since $\delta x = \delta v = 0$, and $\delta(\dfrac{1}{M}) \approx \dfrac{\delta M}{M^2}$, and using the general formula $\delta(a \cdot b) \;\approx\; a \cdot \delta b + b \cdot \delta a$.

It follows immediately from (5.3a) that F_x is (globally) uniformly continuous.

What about F_v? Since the tuple of parameters and input stream

$$(M,\, K,\, D,\, x_0,\, v_0,\, f)$$

is in the region

$$U \;=\; (\mathbb{R}^{>0})^3 \times \mathbb{R}^2 \times \mathcal{C}[\mathbb{T}, \mathbb{R}]$$

it follows from (5.3b) that F_v is not uniformly continuous in the arguments M, K, D and f, since $\frac{1}{M}$, K, D and f can be made arbitrarily large.

---

[20] See footnote to condition (xi) in Theorem B

We therefore construct an exhaustion $(U_1, U_2, \dots)$ of $(\mathbb{R}^{>0})^3 \times \mathbb{R}^2 \times \mathcal{C}[\mathbb{T}, \mathbb{R}]$, the individual members of which constrain M from below, and K, D and f from above:

$$U_\ell \ =_{df}\ \mathbb{R}_\ell \times \mathbb{R}^2 \times \mathcal{C}[\mathbb{T}, \mathbb{R}]_\ell \ \subseteq\ (\mathbb{R}^{>0})^3 \times \mathbb{R}^2 \times \mathcal{C}[\mathbb{T}, \mathbb{R}]$$

$(\ell = 1, 2, \dots)$, where

$$\begin{aligned} \mathbb{R}_\ell \ &=_{df}\ \{\, (M, K, D) \in (\mathbb{R}^{>0})^3 \mid M > 1/\ell,\ K < \ell,\ D < \ell \,\}, \\ \mathcal{C}[\mathbb{T}, \mathbb{R}]_\ell \ &=_{df}\ \{\, f \in \mathcal{C}[\mathbb{T}, \mathbb{R}] \mid \|f\|_T < \ell \,\}. \end{aligned}$$

Then, by Lemma 4.4.5, the module functions F_v and (of course) F_x are both effectively locally uniformly continuous on each U_ℓ. Hence so is the network state function F, by the reasoning in Remark 5.1.1.

Hence Theorem B, applied to the system in Figure 2, gives us:

Proposition 2. *In the system in Figure 2, the displacement x is **WhileCC***(α_{C}) approximable as a function of the mass M, spring constant K, damping constant D, initial displacement x_0, initial velocity v_0, external force f and time t.*

Again, we are using the cartesian form of the function Φ in Theorem B.

The reasoning for the second case study (iterated mass/spring/damper system) is similar.

Remark 5.2.1. The rather formidable list of assumptions in Theorem B is satisfied by both our examples (this one, and Example 2 below), and in particular these two mass/spring/damper case studies, which have acted as a "reality check" for much of the research described in this and related papers [TZ07, TZ11].[21]

• ***Example 2: Synchronous concurrent algorithms (SCAs)***

These were investigated in [TTZ09], where it was shown how the network stream transformer Φ could be obtained by a simultaneous primitive recursion, and again (from the present viewpoint) in [TZ11], where it was shown [TZ11, Examples 3.3.5(2) and 3.4.9(2)] how Φ could also be obtained as the fixed point of a contracting stream transformer. Continuity of Φ followed [TZ11, Example 4.2.13(2)] as a simple special case of Theorem 2.

(i) *Concrete computability: Applying Theorem A*

We turn to the question of concrete computability of Φ.

Now $\mathbb{T} = \mathbb{N}$, with the standard exhaustion $K_k = \{0, 1, \dots, k\}$. For Z_{C} we can take, for example, the set of functions u such that for all i, $u(i) \in Z_{\mathsf{A}}$, and further, for some fixed $z_0 \in Z_{\mathsf{A}}$, $u(i) = z_0$ for i sufficiently large. Then Z_{C} is countable and dense in $\mathcal{C}[\mathbb{N}, A]$ [TTZ09]. We let α_{C} be some "standard" enumeration of Z_{C}, based on the enumeration α_{A} of Z_{A}. This enumeration is easily seen to be regular: *density* and *Σ-effectivity* of $\overline{\alpha}$ are clear, while *effective local uniform continuity* of $(Z_{\mathsf{C}}, \alpha_{\mathsf{C}})$ is not an issue (Remark 3.1.4).

[21] The book [Hyn70] by D.E. Hyndman was very helpful as the source of these two case studies.

Again we have a very simple special case of Theorem A:

Theorem A′. *Suppose a Σ-algebra $\mathcal{C}[\mathbb{T}, A]$, based on an SCA network N, is represented by a regular enumeration α. If the module functions of N are $\overline{\alpha}$-computable, then so is the network function Φ.*

This is because Φ is defined from the module functions by PR (primitive recursion), which preserves $\overline{\alpha}$-computability. The latter fact follows, for example, from the sequence of results

$$\begin{aligned} \textit{PR computability} &\implies \textbf{\textit{While}}^* \textit{ computability} \\ &\implies \textbf{\textit{WhileCC}}^*(\alpha_{\mathsf{c}}) \textit{ computability} \\ &\iff \overline{\alpha}\text{-computability.} \end{aligned}$$

Caution! It is actually the cartesian form of Φ, $\textbf{\textit{cart}}(\Phi)$, that is defined by PR, and hence $\overline{\alpha}$-computable (cf. [TZ11, Example 4.2.13(2)].) However, as we will see in §6.2 (Lemma 6.2.5),

$$\textbf{\textit{cart}}(\Phi) \ \overline{\alpha}\textit{-computable} \implies \Phi \ \overline{\alpha}\textit{-computable}, \tag{5.5}$$

at least in the case that $\mathbb{T} = \mathbb{N}$.

(ii) *Abstract computability: Applying Theorem B*

We turn to the question of abstract computability of SCAs.

Here we can simplify the statement of Theorem B slightly, by recalling (from [TZ11, Example 3.3.5(2)]) that $F_{\boldsymbol{c},\boldsymbol{a},\boldsymbol{x}}$ is automatically causal and contracting for all $(\boldsymbol{c}, \boldsymbol{a}, \boldsymbol{x}) \in \textbf{\textit{dom}}(F)$, with constant contraction modulus and increment. Hence assumptions (iv), (v), (viii) and (ix) are redundant.

The attentive reader may ask: why can't we drastically simplify Theorem B in this case, along the lines of Theorem A′ above in the case of $\overline{\alpha}$-computability, by noting simply that Φ is defined from the module functions by primitive recursion, which preserves $\textbf{\textit{WhileCC}}^*$ (and, for that matter, $\textbf{\textit{While}}^*$) computability? The answer (as pointed out in the cautionary comment above) is that actually it is not Φ, but $\textbf{\textit{cart}}(\Phi)$, that is defined by primitive recursion, and we do not have a result analogous to (5.5) for abstract computability. In fact, the analogous assertion is known to be false for $\textbf{\textit{While}}^*$ computability [TZ94]. What about $\textbf{\textit{WhileCC}}^*$ computability? This is discussed further in Section 6 (Lemma 6.2.6).

Hence the only proof of Theorem B that we have at present, even in the case of SCA networks, proceeds via $\overline{\alpha}$-computability and the completeness theorem for abstract/concrete computability.

6 Concluding remarks

Stream processing occurs everywhere, often without being recognised as such. There are many occasions where a theoretical analysis of a computation has led to a model of stream processing [Ste97]. We have used basic topological notions to model stream processing in continuous and discrete time, in a uniform way, and applied our general models of concrete and abstract computability [TZ04] to establish the computability of stream processing. We have used, as examples, two simple, commonly found paradigms of stream processing: analog networks and synchronous concurrent algorithms (SCAs) which we previously studied independently [TZ07, TTZ09].

There are many further examples of discrete space stream processing (such as dataflow networks, networks of sensors, coupled map lattices, finite element modelling) and many other relevant computability models (such as TTE, higher type computability) waiting to be combined and investigated. First (§6.1) we will elaborate on the idea that discrete space models of natural and artificial systems lead directly to the study of continuous and computable discrete and continuous time stream processing. Secondly (§6.2) we will comment on some technical issues in our formulation of stream transformers and their computability, and then (§§ 6.3, 6.4) note some topics for future research.

6.1 Modelling systems by continuous and computable stream transformers

The source of much computation lies in mathematical models of natural and artificial systems. The architectures of these systems are described in terms of components distributed in space. The properties of these systems are measured by data assigned to the components. The dynamic behaviours of these systems are described in terms of processes in which the data measuring the state of the components change in time. In general, models of systems can be classified by choosing each of space, time and data to be either discrete or continuous (8 possibilities). In the case of systems made from components, the existence of a system architecture requires discrete space only. Thus the models can use either discrete or continuous time $\mathbb{T}$ and discrete or continuous data A (4 possibilities). Specifically, the points of the discrete space are assigned components — such as units, cells, neighbourhoods, etc. — which are observed or measured by data that changes in time. Thus, the local behaviour of a model at a point is given by a stream of data from A timed by $\mathbb{T}$, i.e., an element of $\mathcal{C}[\mathbb{T}, A]$. These observations lead us to propose the following thesis:

> *Any discrete space model of system behaviour in time can be represented by a family of stream transformers indexed by an architecture.*

A consequence of this thesis is that whenever we find a discrete space mathematical model based on equations, we can reformulate it as a system of equations for defining stream transformers.

In [TZ11] we showed (using the notation of §1.2):

> *If F is contracting and causal, then Φ exists and is unique. If, in addition, F is continuous, then so is Φ.*

As discussed in [TZ11], continuity has long been recognised as a fundamental property of

models because continuity implies the *physical stability* of the fixed point solution Φ to the specification given by F. For example, repeating an experiment to observe a physical process requires this stability since initial conditions and other parameters can never be exactly determined or, therefore, reproduced. Hadamard [Had52, Had64], Courant and Hilbert [CH53] and others required that for a scientific problem to be "well posed", the solution should depend *continuously* on the parameters; or, in another formulation suitable for our present purposes:

> *For a model of a physical system to be well posed, its behaviour should depend continuously on the data.*

This can be viewed as motivating our investigation in [TZ11], by requiring that Φ be *continuous* in the data.

Another necessary condition for a model to serve a useful purpose is that system behaviour can be simulated. This leads to a further, related, desideratum:

> *For a model of a physical system to be well posed, its behaviour should be computable in the data.* (6.1)

Kreisel has explored this idea in [Kre74]. Interestingly, he uses Hadamard's principle there in order to reject certain proposed physical experiments aimed at refuting (6.1). It can, likewise, be viewed as motivating the present investigation. In the present context, it amounts to requiring that Φ be *computable* in the data.

Although for some models computability implies continuity [KLS59, Tse59, Tse62] the properties of computability and continuity are conceptually quite distinct, and should be considered separately.

6.2 Computability of cartesian forms of stream-valued functions

An essential technique throughout this paper and [TZ11] has been to reduce higher type definitions to lower type, by an "uncurrying" process (see Remarks 3.2.14 and 4.5.1, and [TZ11, Remark 4.1.1]). This allows us to use (relatively) elementary technical concepts from topology and computability theory.

Another form of type reduction (or uncurrying) that we find ourselves working with repeatedly is in the construction of the *cartesian forms* of stream operators: cf. the cautionary comment in §5.2, Example 2(i).

In order to investigate this situation more systematically, we first consider the general phenomenon of a "mismatch" between concrete and abstract computability on stream algebras. As an example, take a simple stream-valued function

$$f\colon A^q \times \mathcal{C}[\mathbb{T}, A]^p \to \mathcal{C}[\mathbb{T}, A] \tag{6.2}$$

where for any input data $\boldsymbol{a} \in A^q$ and $\boldsymbol{u} = (u_1, \ldots, u_p) \in \mathcal{C}[\mathbb{T}, A]^p$, $f(\boldsymbol{a}, \boldsymbol{u})$ is (to take some simple examples) the pointwise doubling of u_1, or the pointwise sum of u_1 and u_2, or the constantly zero stream. Now any of these operations is easily seen to be concretely

computable; however, none of them is ***While**** computable. The only ***While**** computable function of the form (6.2) is the *input dependent stream projection*

$$f(\boldsymbol{a}, \boldsymbol{u}) \;=\; u_{f_0(\boldsymbol{a},\boldsymbol{u})}$$

for some (computable) $f_0 \colon A^q \times \mathcal{C}[\mathbb{T}, A]^p \to \{1, \ldots, p\}$. This is proved by the *subalgebra property* [TZ00]: the value $f(\boldsymbol{a}, \boldsymbol{u})$ must lie in the subalgebra generated by the inputs, which consists only of $\boldsymbol{u}$. Further discussion of this, in the case $\mathbb{T} = \mathbb{N}$, can be found in [TZ94, TTZ09], where we worked mainly with primitive recursive computability over $\mathcal{C}[\mathbb{T}, A]$.

Now let f be a (partial) stream-valued function on $\mathcal{C}[\mathbb{T}, A]$, say

$$f \colon D \;\rightharpoonup\; \mathcal{C}[\mathbb{T}, A]^m \tag{6.3}$$

where D is some product of carriers in the stream algebra $\mathcal{C}[\mathbb{T}, A]$. Note that although f in (6.3) may be partial, all streams, i.e., elements of $\mathcal{C}[\mathbb{T}, A]$, are still total (and continuous, by definition). Hence, for any $\boldsymbol{x} \in D$, if $f(\boldsymbol{x})\!\downarrow$, then for all $t \in \mathbb{T}$, $f(\boldsymbol{x})(t)\!\downarrow$.

Recall the definition of the *cartesian form* of f (cf. (5.1)):

$$\boldsymbol{cart}\,(f) \colon D \times \mathbb{T} \;\rightharpoonup\; A^m \tag{6.4}$$

where for all $\boldsymbol{x} \in D$ and $t \in \mathbb{T}$,

$$\boldsymbol{cart}\,(f)(\boldsymbol{x}, t) \;\simeq\; f(\boldsymbol{x})(t)$$

where '$\simeq$' is "Kleene equality": the two sides are either both defined and equal, or both undefined.

Unlike the type transformation (2.6) $\Rightarrow$ (2.7) (cf. Remarks 3.2.14 and 4.5.1), the '***cart***' operation is not, strictly speaking, an uncurrying operation, for two reasons:

(1) The type of the curried form of (6.4) is not $D \rightharpoonup \mathcal{C}[\mathbb{T}, A]^m$ as in (6.3), but $D \rightharpoonup \mathcal{C}[\mathbb{T}, A^m]$. However, these two function spaces are homeomorphic [TZ11, Cor. 2.5.2].

(2) More interestingly, the type shown in (6.3) is actually a *subtype* of the curried form of (6.4), since it consists only of functions of that type which are *continuous* in the second argument (by continuity of streams).

In any case, like the transformation (2.6) $\Rightarrow$ (2.7), '***cart***' is a very useful "type lowering" technique.

Now, given a model of computability $\mathcal{M}(\mathcal{C}[\mathbb{T}, A])$ on the stream algebra $\mathcal{C}[\mathbb{T}, A]$, we define the model $\mathcal{M}_{\mathsf{cart}}(\mathcal{C}[\mathbb{T}, A])$ by

$$f \in \mathcal{M}_{\mathsf{cart}}(\mathcal{C}[\mathbb{T}, A]) \quad \Longleftrightarrow_{df} \quad \boldsymbol{cart}\,(f) \in \mathcal{M}(\mathcal{C}[\mathbb{T}, A]).$$

It is easy to see that the stream operations listed above (pointwise doubling etc.) are all in $\boldsymbol{While}_{\mathsf{cart}}(\mathcal{C}[\mathbb{T}, A])$. This suggests that $\boldsymbol{While}_{\mathsf{cart}}$, $\boldsymbol{While}^*_{\mathsf{cart}}$, etc., are better models of computation on $\mathcal{C}[\mathbb{T}, A]$ than (respectively) ***While***, ***While****, etc.

In general we have $\mathcal{M}(\mathcal{C}[\mathbb{T}, A]) \subseteq \mathcal{M}_{\mathsf{cart}}(\mathcal{C}[\mathbb{T}, A])$, i.e.,

$$f \text{ is } \mathcal{M}\text{-computable} \implies \boldsymbol{cart}(f) \text{ is } \mathcal{M}\text{-computable}.$$

We have seen that the reverse implication

$$\boldsymbol{cart}(f) \text{ is } \mathcal{M}\text{-computable} \implies f \text{ is } \mathcal{M}\text{-computable} \tag{6.5}$$

fails for deterministic abstract models like ***While**** , because of the *subalgebra property* of these models. What about concrete computability? In other words, under what conditions does the implication

$$\boldsymbol{cart}(f) \text{ is } \overline{\alpha}\text{-computable} \implies f \text{ is } \overline{\alpha}\text{-computable} \tag{6.6}$$

(effectively in $\overline{\alpha}$-codes) hold for all stream transformers f on $\mathcal{C}[\mathbb{T}, A]$?

This turns out to be linked to the following problem. Note first that a stream can be "computable" in two senses: as a *function* from $\mathbb{T}$ to A, or as a *point* in the stream space, i.e., in the range of $\overline{\alpha}_{\mathsf{c}}$. What is the relation between these two notions of computability for streams? This is problem is formalised by the concept of *functional adequacy*, which says that every computable function from $\mathbb{T}$ to A can be "represented" by a computable stream:

Notation 6.2.1. Given a stream $u \in \mathcal{C}[\mathbb{T}, A]$, let

$$\mathsf{fun}(u)\colon \mathbb{T} \to A$$

be the "corresponding" function, i.e., for all $t \in \mathbb{T}$,

$$\mathsf{fun}(u)(t) = \mathsf{eval}(u, t).$$

Definition 6.2.2 (Functional adequacy of $\overline{\alpha}$).

(a) The family of enumerations $\overline{\alpha}$ of $\mathcal{C}[\mathbb{T}, A]$ is *functionally adequate* if for any stream $u \in \mathcal{C}[\mathbb{T}, A]$, if $\mathsf{fun}(u)$ is $\overline{\alpha}$-computable (as a function), then u is an $\overline{\alpha}$-stream (i.e., a stream in the range of $\overline{\alpha}_{\mathsf{c}}$ (Def. 3.2.4)).

(b) $\overline{\alpha}$ is *effectively functionally adequate* if (a) holds, effectively in $\overline{\alpha}$-codes.

Note that the converse of functional adequacy, i.e., "*for every $\overline{\alpha}$-stream u, $\mathsf{fun}(u)$ is $\overline{\alpha}$-computable*", is trivially true.

For the rest of §6.2, we assume α is a regular enumeration of $\mathcal{C}[\mathbb{T}, A]$ (Definition 3.1.5), and so α-streams, and hence also $\overline{\alpha}$-streams, are effectively locally uniformly continuous (by Lemma 3.2.13).

Lemma 6.2.3.

(a) *If $\overline{\alpha}$ is effectively functionally adequate then the implication* (6.6) *holds for every stream transformer f on $\mathcal{C}[\mathbb{T}, A]$.*

(b) *Conversely, if* (6.6) *holds for every f, then $\overline{\alpha}$ is functionally adequate.*

The proof of (a) uses the S-m-n theorem [Kle52, Rog67] on the tracking function for $\boldsymbol{cart}(f)$ to construct a tracking function for f. The proof of (b) follows easily from the definitions.

Examples 6.2.4 (Functional adequacy). We continue with our two examples from Section 5.

• ***Example 1: Analog networks***. Here $\mathbb{T} = \mathbb{R}^{\geq 0}$. Assume (continuing with Example 1 in §5.2) that $A = \mathbb{R}$. Now the $\overline{\alpha}$-computable functions from $\mathbb{T}$ to $\mathbb{R}$ are continuous by the Kreisel-Lacombe-Shoenfield-Tseitin theorem [KLS59, Tse59, Tse62][22], and hence (classically) locally uniformly continuous. However, this does not imply that they are *effectively* locally uniformly continuous. In fact a counterexample is given by M. Beeson in [Bee85, p. 71, Exc. 2]. This function[23] $u_{\mathsf{B}} \colon \mathbb{T} \to \mathbb{R}$ is $\overline{\alpha}$-computable as a function, but not effectively locally uniformly continuous, and hence not an $\overline{\alpha}$-stream. This function u_{B} then also provides a counterexample to functional adequacy for $\overline{\alpha}$. Hence, by Lemma 6.2.3(b), (6.6) fails.

• ***Example 2: SCAs***. Here $\mathbb{T} = \mathbb{N}$, and so (by Remark 3.1.4) all functions on $\mathbb{T}$ are automatically continuous — in fact effectively globally uniformly continuous. From this it is easy to prove effective functional adequacy of $\overline{\alpha}$: Given an $\overline{\alpha}$-computable $f \colon \mathbb{N} \to A$, we can define an effective sequence of elements of Z_{C} (see Example 2(i) in §5.2) which approaches f effectively locally uniformly, and hence forms an effectively locally uniform Cauchy sequence, which has a limit u in $\mathcal{C}_\alpha(\mathbb{T}, A)$ (see §3.2) (all effective in codes). Clearly, u is extensionally equivalent to f. Hence by Lemma 6.2.3(a) we have:

Lemma 6.2.5. (6.6) *holds for SCAs.*

This result was used in Section 5 (in the form (5.5)) to give a simple proof of $\overline{\alpha}$-computability of the network function Φ of an SCA, assuming $\overline{\alpha}$-computability of the network module functions (Theorem A′ in §5.2, Example 2).

Next, we can ask whether (6.5) holds in the case that $\mathcal{M}$ is $\boldsymbol{WhileCC^*}(\alpha_{\mathsf{C}})$ approximability, i.e., whether (or under what circumstances)

$$\boldsymbol{cart}(f) \text{ is } \boldsymbol{WhileCC^*}(\alpha_{\mathsf{C}}) \text{ approx.} \implies f \text{ is } \boldsymbol{WhileCC^*}(\alpha_{\mathsf{C}}) \text{ approx.} \tag{6.7}$$

But this follows from Lemma 6.2.5 and the Completeness Theorem:

Lemma 6.2.6. *Under the assumptions of the Completeness Theorem* 4.4.6, (6.7) *holds for SCAs.*

---

[22] For expositions of this theorem, see [Bee85, pp. 61–62] and [TvD88, Vol. 1, pp. 321–322]

[23] Actually Beeson's counterexample has domain $[0, 1]$, but that can be easily modified.

More on the '***cart***' operation on stream-valued functions, and its relation to models of (deterministic, abstract) computation, can be found in [TZ94].

6.3 Future research on stream spaces

The study of computation on stream algebras provides a rich source of topics for future research. We mention four such topics here:

(1) ***Relative computability of fixed point.*** In §3.3 we proved a relativised version (Theorem A^{rel}) of Theorem A, on concrete ($\overline{\alpha}$-)computability of the fixed point Φ of the contracting stream operator F. However we were unable to prove an analogous relativised version of Theorem B, on abstract computability of the FP (see Remark 4.5.3).

Two possible approaches to such a proof would be (i) proving a relativised version of the completeness theorem (4.4.6), and then using Theorem A^{rel}, by analogy with our proof of Theorem B; or (ii) finding a direct proof of the result, not relying on the completeness theorem or Theorem A. In the latter case, we would not need to work with ***WhileCC**** approximability as our model of abstract computation. This brings us to another point:

(2) ***Other models of abstract computability.*** Our use of ***WhileCC**** approximability in Section 4 was motivated by the proof method of Theorem B, which used the completeness theorem (4.4.6). Now ***WhileCC**** computability (or approximability) is clearly nondeterministic, which is a valuable property in investigating computation on topological models in general [TZ04, TZ05], but may be less appropriate with the analog networks studied in this paper (cf. the Network Determinacy Assumption in §5.1).[24]

Since the FP is computed by a system of (determinisitic) approximations, a more appropriate abstract computability model (in the present context) might be ***While**** approximability.

(3) ***Partial and nondeterministic module functions.*** As noted above, from considerations of *continuity*, we are nevertheless led to consider module functions that are nondeterministic (or many-valued) and partial [TZ04, TZ05].

These features (which imply discarding our Network Determinacy Assumption in §5.1) will complicate the theory considerably — for example, in the case of SCAs, they would require replacing a single global clock by a system of local clocks [TTZ09, §8.2(1)]. However, they constitute an important generalisation, because of the desirability of continuity by Hadamard's principle [TZ11, Discussion 4.2.14].

Continuity considerations are especially significant with *hybrid systems*, at analog/digital interfaces [NK93].

(4) ***Generalisation of stream concept.*** The considerations in (3) will lead to the investigation of streams which are also partial and nondeterministic. Note again that the Network Determinacy Assumption will no longer hold.

The use of *piecewise continuous* streams (in the case $\mathbb{T} = \mathbb{R}^{\geq 0}$) forms another important

[24] But see (3) and (4) below!

generalisation of the stream concept. For example, points of discontinuity in such streams could correspond to boundaries in phase space.

6.4 Future research on computability of spatial objects

Returning to the general ideas of the Introduction, each data type of the form $\mathcal{C}[X, A]$ arises typically in some practical situation, and has its own special features. The algorithmic models that are characteristic of that situation determine, or at least suggest, a corresponding computability theory. We have considered the case that X is *time*, but alternatively, the case of *space* is also fundamentally important:

(i) *Graphic scenes*: In 3-dimensional volume graphics, X can be continuous space, $X = \mathbb{R}^3$, and data are attributes of spatial objects, such as colour or opacity, measured by $A = \{0,1\}^k$ or $A = [0,1]$.

(ii) *Machine states*: In machine states, X could be a 2-dimensional discrete address space, $X = \mathbb{Z}^2$, and data are k-bit words $A = \{0,1\}^k$.

(iii) *Analog fields*: Quite generally, X can be a continuous space modelled by a manifold, and data can be measurements from a normed vector space.

We have encountered some of these data types before. In particular, we have considered *spatial objects* in volume graphics and there arise several interesting open graphics questions involving computability [CT00, BSHT98, Joh06, JT11].

Do the mathematical methods we have used to study streams apply to spatial objects? On the face of it, much of our mathematics ought to apply to arbitrary function spaces $\mathcal{C}[X, A]$. Interestingly, this does not seem to be the case. For example, in the data type of spatial objects, the important notion of causality does not seem to have a natural counterpart.

Even in simple geometric spaces such as $X = \mathbb{R}^n$, the theory of spatial objects and their specifications by operations and equations raises several questions where an appropriate computability theory is necessary for the answer. One such question, raised in [JT11, Sec. 8], is the following: *For a given set of basic spatial objects, and high level operations on them, is every computable spatial object effectively approximable by objects built up from these basic objects by these operations?*

Some interesting work in generalising the concept of causality for spaces $\mathcal{C}[X, A]$ where X is an arbitrary σ-compact space, has been done in [Jam12, Ch. 4].

References

[Bee85] M. Beeson. *Foundations of Constructive Mathematics.* Springer-Verlag, 1985.

[BSHT98] J. Blanck, V. Stoltenberg-Hansen, and J.V. Tucker. Streams, stream transformers and domain representations. In B. Möller and J.V. Tucker, editors, *Prospects for hardware foundations*, volume 1546 of *Lecture Notes in Computer Science.* Springer-Verlag, 1998.

[CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics, Vol. II.* Interscience, 1953. Translated and revised from the German edition [1937].

[CT00] M. Chen and J.V. Tucker. Constructive volume geometry. *Computer Graphics Forum*, 19:281–293, 2000.

[Had52] Jacques Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Dover, 1952. Translated from the French edition [1922].

[Had64] J. Hadamard. *La Théorie des Équations aux Dérivées Partielles*. Éditions Scientifiques, 1964.

[Hyn70] D.E. Hyndman. *Analog and Hybrid Computing*. Pergamon Press, 1970.

[Jam12] N. James. *Existence, Continuity and Computability of Unique Fixed Points in Analog Network Models*. Ph.D. Thesis, Department of Computing & Software, McMaster University, 2012.

[Joh06] K. Johnson. *Algebraic Specifications of Spatial Data Types with Applications to Constructive Volume Geometry*. PhD thesis, Department of Computer Science, Swansea University, Swansea, Wales, 2006.

[JT11] K. Johnson and J.V. Tucker. The data type of spatial objects. *Formal Aspects of Computing*, 2011. DOI: 10.1007/s00165-011-0182-7.

[JZ12] Nick D. James and Jeffery Zucker. A class of contracting stream operators. *The Computer Journal*, 2012. DOI: 10.1093/comjnl/bxs054.

[Kle52] S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

[KLS59] G. Kreisel, D. Lacombe, and J. Shoenfield. Partial recursive functions and effective operations. In A. Heyting, editor, *Constructivity in Mathematics: Proceedings of the Colloqium in Amsterdam, 1957*, pages 290–297. North Holland, 1959.

[Kre74] G. Kreisel. A notion of mechanistic theory. *Synthese*, 29:11–26, 1974.

[Mal71] A.I. Mal'cev. Constructive algebras I. In *The metamathematics of algebraic systems. A.I. Malcev, Collected papers: 1936–1967*, pages 148–212. North Holland, 1971.

[NK93] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 317–356. Springer-Verlag, 1993.

[PER89] M.B. Pour-El and J.I. Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1989.

[Rog67] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[Roy63] H.L. Royden. *Real Analysis*. Macmillan, 1963.

[SHT99] V. Stoltenberg-Hansen and J.V. Tucker. Concrete models of computation for topological algebras. *Theoretical Computer Science*, 219:347–378, 1999.

[Sim63] G.F. Simmons. *Introduction to Topology and Modern Analysis*. McGraw-Hill, 1963.

[Ste97] R. Stephens. A survey of stream processing. *Acta Informatica*, 34:491–541, 1997.

[Tse59] G.S. Tseitin. Algebraic operators in constructive complete separable metric spaces. *Doklady Akademii Nauk SSSR*, 128:49–52, 1959. In Russian.

[Tse62] G.S. Tseitin. Algebraic operators in constructive metric spaces. *Tr. Mat. Inst. Steklov*, 67:295–361, 1962. In Russian. Translated in AMS Translations (2) 64:1–80. MR 27#2406.

[TTZ09] B.C. Thompson, J.V. Tucker, and J.I. Zucker. Unifying computers and dynamical systems using the theory of synchronous concurrent algorithms. *Applied Mathematics and Computation*, 215:1386–1403, 2009.

[TvD88] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction, Vols I and II.* North Holland, 1988.

[TZ94] J.V. Tucker and J.I. Zucker. Computable functions on stream algebras. In H. Schwichtenberg, editor, *Proof and Computation: NATO Advanced Study Institute International Summer School at Marktoberdorf, 1993*, pages 341–382. Springer-Verlag, 1994.

[TZ99] J.V. Tucker and J.I. Zucker. Computation by 'while' programs on topological partial algebras. *Theoretical Computer Science*, 219:379–420, 1999.

[TZ00] J.V. Tucker and J.I. Zucker. Computable functions and semicomputable sets on many-sorted algebras. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 5, pages 317–523. Oxford University Press, 2000.

[TZ04] J.V. Tucker and J.I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, 5:611–668, 2004.

[TZ05] J.V. Tucker and J.I. Zucker. Computable total functions, algebraic specifications and dynamical systems. *Journal of Logic and Algebraic Programming*, 62:71–108, 2005.

[TZ06] J.V. Tucker and J.I. Zucker. Abstract versus concrete computability: The case of countable algebras. In V. Stoltenberg-Hansen and J. Väänänen, editors, *Logic Colloquium '03, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, held in Helsinki, Finland, August 14–20, 2003*, volume 24 of *Lecture Notes in Logic*, pages 377–408. Association for Symbolic Logic, 2006.

[TZ07] J.V. Tucker and J.I. Zucker. Computability of analog networks. *Theoretical Computer Science*, 371:115–146, 2007.

[TZ11] J.V. Tucker and J.I. Zucker. Continuity of operators on continuous and discrete time streams. *Theoretical Computer Science*, 412:3378–3403, 2011.

[Wei00] K. Weihrauch. *Computable Analysis: An Introduction.* Springer-Verlag, 2000.