Continuous Functions on Final Coalgebras

Neil Ghani\textsuperscript{1} Peter Hancock\textsuperscript{2}

\textit{University of Nottingham, UK}

Dirk Pattinson\textsuperscript{3}

\textit{University of Leicester, UK}

Abstract

It can be traced back to Brouwer that continuous functions of type $\text{Str}A \rightarrow B$, where $\text{Str}A$ is the type of infinite streams over elements of $A$, can be represented by well founded, $A$-branching trees whose leafs are elements of $B$. This paper generalises the above correspondence to functions defined on final coalgebras for power-series functors on the category of sets and functions.

While our main technical contribution is the characterisation of all continuous functions, defined on a final coalgebra and taking values in a discrete space by means of inductive types, a methodological point is that these inductive types are most conveniently formulated in a framework of dependent type theory.

\textit{Key words:} Continuous Functions, Final Coalgebras, Constructive Mathematics

1 Introduction

Brouwer’s ‘choice sequences’ are at the heart of his intuitionistic reconstruction of the continuum. A choice sequence is a stream of discrete objects (for example natural numbers), where there need be no algorithm deriving or generating the successive terms of the streams. Choice sequences are paradigm examples of ‘infinite objects’, known to us only through finite information observed of them, as it were externally. For a thorough account see [5,8,9].

According to Brouwer, all functions on choice sequences are continuous. Continuity of a function means that if you want it to provide you some finite amount of information about the value of the function, then you need only supply it some finite amount of information about its argument. Boiled down to cinders, Brouwer’s

\textsuperscript{1} Email: nxg@cs.nott.ac.uk
\textsuperscript{2} Email: hancock@spamcop.net
\textsuperscript{3} Email: dirk@mcs.le.ac.uk

This paper is electronically published in
\textit{Electronic Notes in Theoretical Computer Science}
\textit{URL:} www.elsevier.nl/locate/entcs
reasoning was that the definition of such a function has to have a certain inductive structure, starting with constant functions, and built up by a form of patching together. On account of this inductive structure, the function has to be continuous.

In concrete terms, one defines a binary operator on sets

\[ R_A B = \mu X. B + (A \to X) \]

where \( \mu X. \ldots \) denotes formation of the initial algebra (\( R_A \) is in fact the free monad generated by \( (A \to _) \)), and then by well-founded structural recursion a function

\[ \text{eat} : R_A B \to \text{Str} A \to B \]

\[ \text{eat}(\text{inl} \, y) \alpha = y \]

\[ \text{eat}(\text{inr} \, f) \alpha = \text{eat}(f \alpha_0)\alpha' \]

Here \( \alpha \) has type \( \text{Str} A \), \( \alpha_0 \) denotes the head of \( \alpha \), and \( \alpha' \) denotes its tail.

The function \( \text{eat} \) defines a representation \( |t| = \text{eat} \, t \) of functions of type \( \text{Str} A \to B \) by elements \( t \) of \( R_A B \). These objects \( t \) are interpreted as terminating programs that read some finite number of \( A \)'s, then eventually return a \( B \).

Graphically, we can represent a continuous function of type \( \text{Str}\{0, 1\} \to B \) by a finite tree as exemplified in Figure 1. In general the representative of a continuous function of type \( \text{Str} A \to B \) will be well-founded, and finite if \( A \) is finite. To compute the value of the function at an infinite stream of bits \( (a_n)_{n \in \omega} \), we start at the root of the tree and take the branch according to the first symbol \( a_0 \) of the tree. This process is then repeated using the stream \( (a_{n+1})_{n \in \omega} \), starting from the node we have just reached. Once we hit a leaf, we output the corresponding element of \( B \). For example, the function represented by the tree depicted in Figure 1 assigns the value \( b_2 \) to every stream with initial segment \( (1, 0, 1) \).

Different well-founded objects of type \( R_A B \) may represent the same function of type \( \text{Str} A \to B \). The representation is intensional. Nevertheless, there is a clear computational difference to be drawn between intensionally different implementations of the same function: they deliver output with greater or less demand for input.

Brouwer’s thesis was that this representation is complete, at least as far as he was concerned. Classically, completeness is witnessed by the fact that \( \text{eat} \) is sur-
jective, giving us a many-to-one correspondence between

$$\mu X.B + (A \to X) \xrightarrow{\text{eat}} (\text{Str} A \xrightarrow{\text{cts}} B)$$

in the sense that every element of $\mu X.B + (A \to X)$ represents a continuous function of type $\text{Str}A \to B$, but each such function can have multiple representations.

In today’s modern terminology, streams of $A$’s are inhabitants of the maximal fixed point or final co-algebra $(\nu X).A \times X$, alias $A^{\omega}$. So Brouwer’s thesis says that all functions from this final co-algebra have an inductive structure, on account of which they are continuous. A question motivating this paper is: to what kinds of functors besides $(A \times \_)$ can Brouwer’s analysis be extended? Our answer is: at least to functors with a discrete codomain that are expressible as power-series: $\sum_{n\in\omega}A_n \times X^n$.

In more detail, we construct a family $(R_n)_{n\in\omega}$ of inductive types and show, that all continuous functions of type $(\nu F)^n \to B$, where $B$ is discrete, can be represented by an element of type $R_n$. As an immediate corollary, we obtain an inductive characterisation of the set of all continuous functions of type $(\nu F)^n \to B$ as the least set of all functions that contains constant functions and is closed under a patching operation, that is discussed in detail in Section 5.

Our motivation is more than curiosity. An easy extension of the stream-based representation above is to the representation of continuous functions of type $\text{Str}A \to \text{Str}B$ by elements of type $\nu X.R_A(B \times X)$. This gives us an analysis of stream-processing components, with a composition operator definable directly on the representations. If this analysis can be generalised to functors that ‘branch’ (such as $\nu X.A \times X^2$, or $\nu X.A \times X^2 + B \times X^3$), then we would be in a position to model communication through the medium of simple state machines, with some powerful technology at our disposal. The present paper sets out in the direction of such a generalisation.

## 2 Preliminaries and Notation

We write $\omega = \{0, 1, 2, \ldots\}$ for the set of natural numbers and identify $n \in \omega$ with the set of its predecessors, i.e. $n = \{0, 1, \ldots, n-1\}$. The real numbers are denoted by $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$ are the non-negative reals. We use $\text{Set}$ to refer to the category of sets and functions, and $\text{Set}^{\omega}$ for the category of functors from $\omega$ to $\text{Set}$, where we take $\omega$ to be discrete. Thus, a typical element of $\text{Set}^{\omega}$ is an $\omega$-indexed family $(A_n)_{n\in\omega}$ of sets $A_n$, which we denote briefly by $(A_n)$. A typical morphism from $(A_n)$ to $(B_n)$ is an $\omega$-indexed family $(f_n : A_n \to B_n)_{n\in\omega}$, written $(f_n)$.

The coproduct of two sets $A$ and $B$ is denoted by $A + B$, and we write $\text{inl} : A \to A + B$ and $\text{inr} : B \to A + B$ for the canonical injections. Similarly, the product of $A$ and $B$ is denoted by $A \times B$ and $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ for the canonical projections.

If $(f_i)_{i \in I}$ is a family of functions $A_i \to B$, we write $\text{in} : A_i \to \sum_{i \in I} A_i$ for the canonical injection and $[f_i \mid i \in I]$ for the unique function $u : (\sum_{i \in I} A_i) \to B$ that
satisfies \( u \circ \text{in}_i = f_i \).

If \( F \) is an endofunctor on \( \text{Set} \), we write \( \mu F \) (resp. \( \nu F \)) for the initial algebra (resp. final coalgebra) of \( F \), whenever it exists. The structure map of the initial algebra (resp. final coalgebra) is denoted by \( \text{in} : F(\mu F) \to \mu F \) (resp. \( \text{out} : \nu F \to F(\nu F) \)). Sometimes, we write \( \mu X.F X \) (resp. \( \nu X.F X \)) instead of \( \mu F \) or \( \nu F \) to make the argument of \( F \) explicit. The scope of these and other variable binding operations is ‘as far as possible’.

We write \( \text{Str} A \) for \( \nu X.A \times X \) and adopt the convention that if \( \alpha : \text{Str} A \), then \( \alpha = (\alpha_0, \alpha') \), where \( \alpha_0 \) is its head, and \( \alpha' \) is its tail. Similarly, \( \text{Tree} A = \nu X.X \times X \times A \times X \).

We use standard notation and write \( A \to B \) for the set of all functions from \( A \) to \( B \); if \( A \) and \( B \) are topological spaces, \( A \to B \) denotes the continuous functions from \( A \) to \( B \).

### 3 Continuous Functions

For the whole section suppose \( F : \text{Set} \to \text{Set} \) is an \( \omega \)-continuous endofunctor and \( B \in \text{Set} \). \( \omega \)-continuity means that \( F \) commutes with limits of projective chains, and so has a final coalgebra \( \text{Lim}_n F^n 1 \). We use \( \nu F = \nu X.F(X) \) as shorthand for this projective limit, and denote the canonical projections by \( p_n : \nu F \to F^n 1 \). The connecting morphism of the sequence \( 1 \leftarrow F^1 \leftarrow F^2 \leftarrow \cdots \) are denoted by \( p_{nm} : F^m 1 \to F^n 1 \). Where need arises, we will consider \( B \) to be equipped with the discrete topology. It is well known [2] that the representation of the final coalgebra as projective limit gives rise to an ultrametric, that we now introduce.

**Definition 3.1 (Baire Topology)** Let’s define an ultrametric on \( \nu F \) by stipulating

\[
d(x, y) = 2^{-n} \text{ where } n = \max\{k \mid p_k(x) = p_k(y)\}
\]

for \( x, y \in \nu F \). We call the induced topology the Baire topology, and always assume that \( \nu F \) is topologised in this way. If \( n \geq 0 \), we define the metric \( d_n \) on \((\nu F)^n\) for \( x = (x_1, \ldots, x_n) \) and \( y = (y_1, \ldots, y_n) \in (\nu F)^n \) by \( d(x, y) = \max\{d(x_i, y_i) \mid i = 1, \ldots, n\} \).

It is easy to see that also \( d_n \) defines an ultrametric on \((\nu F)^n\). The fact that the topologies we are dealing with are induced by ultrametrics is of no consequence in this paper.

Note: the definition of the ultrametric is rather non-constructive. We could also describe the Baire topology as generated by the neighbourhood basis \( \{N_p\}_p \) of all objects of \( \nu F \) that share a common prefix \( p \). However, this would involve a little more machinery that we prefer to do without.

The intuitive idea of a continuous function (to a discrete space) is that its value \( f(a) \) depends on only some finite ‘amount’ of information about its argument. Building on the fact that \( B \) has the discrete topology, we have

**Lemma 3.2** A function \( f : A \to B \), where \( A \) is a metric space and \( B \) has the
discrete topology is continuous iff

\[ a \in f^{-1}(b) \implies \exists \epsilon > 0. B_\epsilon(a) \subseteq f^{-1}(b) \]

for all \( a \in A \) and all \( b \in B \), where \( B_\epsilon(a) \) is the closed ball with centre \( a \) and radius \( \epsilon \).

Thus, the function value \( f(a) \) is constant throughout some neighbourhood of \( a \). In case of the Baire topology on streams, an \( \epsilon \)-neighbourhood of \( a \) is the set of streams that share a (fixed and finite) prefix with \( a \). Thus the value of the function depends on only a prefix of \( a \).

In the case of uniform continuity, this amount does not depend on the argument. This is illustrated by the following classical examples.

**Example 3.3** Let \( FX = \omega \times X \); as usual we write \( \text{Str} \omega \) for \( \nu F \) and \( (a_n)_{n \in \omega} \) for a typical element of \( \text{Str} \omega \).

(i) Every constant function \( f : \text{Str} \omega \rightarrow \omega \) is continuous.

(ii) For every \( k \in \omega \) the function \( (a_n)_{n \in \omega} \mapsto a_k \) is uniformly continuous.

(iii) If \( (f_n) \) is a sequence of continuous functions of type \( \text{Str} \omega \rightarrow \omega \), then \( \lambda(a_n). f_{a_0}(a_{n+1}) : \text{Str} \omega \rightarrow \omega \) is continuous.

(iv) The function \( (a_n)_{n \in \omega} \mapsto a_{a_0} \) is continuous, but not uniformly continuous.

(v) The assignment \( (a_n)_{n \in \omega} \mapsto \begin{cases} 0 & \text{if } \forall k \in \omega. a_k = 0 \\ 1 & \text{otherwise} \end{cases} \) does not define a continuous function. To define the function requires decidability of a \( \Pi^0_1 \) statement, which corresponds to Bishop’s limited principle of omniscience [4]. This is of course constructively invalid.

Computationally speaking, we are interested in how much information about the argument \( a \) a function \( f \) might need to access in order to compute the value \( f(a) \). A quantitative measure is given by the modulus of continuity.

In general topology, a modulus of continuity is a function that produces, for every point \( x \) of a metric space and every \( \epsilon > 0 \), a \( \delta > 0 \) satisfying the usual \( \epsilon - \delta \) definition of continuity at \( x \). As we are only interested in functions that take values in a discrete space, the following definition adapts this to our setting.

**Definition 3.4** Suppose \( f : (\nu F)^n \rightarrow B \) is a (not necessarily continuous) function. A function \( m : (\nu F)^n \rightarrow \mathbb{R}_{\geq 0} \) is a modulus of continuity for \( f \), sometimes abbreviated to just modulus, if

\[ d_n(x, y) \leq m(x) \implies f(x) = f(y) \]

for all \( x, y \in (\nu F)^n \).

As a trivial consequence of this definition, we have

**Lemma 3.5** A function \( f : (\nu F)^n \rightarrow B \) is continuous iff it has a modulus.
Note that a modulus of continuity is by no means unique: if \( m \) is a modulus for \( f \), then so is every \( m' \) with \( m'(x) \leq m(x) \).

Frequently a modulus of continuity takes the form \( m(x) = 2^{-k(x)} \) for some \( k : (\nu F)^{\omega} \rightarrow \omega \). By virtue of \( \omega \) being well ordered, in this case there is (classically) a largest modulus of continuity, which corresponds to the minimal lookahead that is needed to compute the value of a specific function.

### 4 Representation of Continuous Functions

In this section we show how continuous functions of type \( \nu X. F X \rightarrow B \), where \( F \) is an endofunctor on \( \mathsf{Set} \) representable as a power-series, are induced by an inductive structure. We invite the reader to check that a simple minded generalisation of the definition of \( \mathsf{eat} \) given in the introduction is bound to fail e.g. for the case of infinite trees, or the final coalgebra for the functor \( F X = X \times A \times X \). Consuming the root \( a \) of the tree will leave us with \( \text{two} \) subtrees, to which we cannot apply \( \mathsf{eat}(a) \). Informally speaking, one would represent continuous functions of type \( \text{Tree}A \rightarrow B \) by trees whose branching degree is raised to the power of two every time an arc is traversed, as depicted in Figure 2. Given a tree \( t \), we start at the root of the tree, and choose the first branch according to the root element of the tree. Removing this root element, we are left with \( \text{two} \) trees, \( (t_1, t_2) \), namely the left and the right subtree of \( t \), and accordingly with a pair \( (a_1, a_2) \in A \times A \) of root elements. We choose our next branch according to \( (a_1, a_2) \), which leaves us with \( \text{four} \) subtrees; we continue this process until we reach a leaf of the tree, which we take as the function value. For example, the function represented in Figure 2 assigns \( b_1 \) to every tree that extends the left hand tree in Figure 3, and \( b_0 \) to every (infinite) extension of the right hand tree.
Trees of this type are conveniently defined using indexed families of sets or dependent types, where the dependency is exploited to record the branching degree. Technically, this will involve moving from the initial algebra \( \mu X.B + (A \to X) \), defined in the category of sets, to an initial algebra for an endofunctor \( M \) on the category \( \text{Set}^\omega \). We will recover representations of continuous functions, say of type \((\text{Tree} A)^n \to B\) as the elements of \( R_n \), where \( (R_n)_{n \in \omega} \) is the initial algebra of the functor \( M \), defined by

\[
M : \text{Set}^\omega \to \text{Set}^\omega \text{ defined by } M((X_n)_{n \in \omega}) = (B + (A^n \to X_{2n}))_{n \in \omega}
\]

It will follow from our general analysis that \( R_1 \) represents all continuous functions of type \( \text{Tree} A \to B \), and therefore generalises the correspondence (1) to infinite binary trees. The price of this generalisation is that we need to work with set valued functions, that is to say indexed families of sets, rather than just sets. Our generalisation then becomes

\[
\mu(X_n)_{n \in \omega}.(B + (A^n \to X_{2n}))_{n \in \omega} \xrightarrow{\text{est}} ((\text{Tree} A)^n \xrightarrow{\text{cts}} B)
\]

For the whole paper, we fix a power-series endofunctor \( F(X) = \sum_{i \in \omega} A_i \times X^i \), where \((A_i)_{i \in \omega} : \text{Set}^\omega \). Note that a power-series endofunctor is automatically \( \omega \)-continuous.

In this section, we describe in detail a representation of continuous functions of type \((\nu F)^n \to B\), which makes use of inductively defined dependent types. To make the exposition more readable, we make the following notational conventions.

**Convention 4.1** If \( n \in \omega \) and \( \sigma : n \to \omega \), we let \( A_{\sigma} = \prod_{i < n} A_{\sigma(i)} \) and \( \bar{\sigma} = \sum_{i < n} \sigma(i) \).

With this notation, the family of inductively defined types that we use to represent continuous functions of type \( \nu F \to B \) is an initial algebra for the following functor.

**Definition 4.2** The functor \( M : \text{Set}^\omega \to \text{Set}^\omega \), defined by

\[
M(X_n)_{n \in \omega} = (B + \prod_{\sigma : n \to \omega} (A_{\sigma} \to X_{\bar{\sigma}}))_{n \in \omega}
\]

is the representation functor associated with \( F \).

If \((R_n)_{n \in \omega}\) is an initial algebra of \( M \), then the elements of \( R_n \) will represent the continuous functions of type \((\nu F)^n \to B\). Before we proceed to the representation theorem, we need to establish the following fact:

**Lemma 4.3** The functor \( M \) has an initial algebra.

**Proof** Because limits in functor categories are computed pointwise. \( \square \)

We write \((R, \rho)\) for the initial algebra of \( M \) throughout, where \( R = (R_n)_{n \in \omega} \) and \( \rho = (\rho_n)_{n \in \omega} \).
In order to make the sequel of the paper more readable, we introduce names for the components of the structure maps \( \rho_n \) of the initial \( M \)-algebra:

**Convention 4.4** For the remainder of the paper, we fix the constructors

\[
\text{const}_n : B \to R_n \quad \text{and} \quad \text{fork}_n : \left( \prod_{\sigma : n \to \omega} (A_\sigma \to R_\bar{\sigma}) \right) \to R_n
\]

with the property that \( \rho_n = [\text{const}_n, \text{fork}_n] \). In other words, \( \text{const}_n = \rho_n \circ \text{inl} \) and \( \text{fork}_n = \rho_n \circ \text{inr} \).

In order to draw the analogy between the definition of \( \text{eat} \) for \( F X = A \times X \), we fix the following canonical isomorphism:

**Convention 4.5** For the remainder of the paper, we fix a canonical isomorphism

\[
d_n : (\nu F)^n \xrightarrow{\text{out} \times \cdots \times \text{out}} \left( \sum_{i \in \omega} A_i \times (\nu F)^i \right)^n \xrightarrow{\alpha} \sum_{\sigma : n \to \omega} A_\sigma \times (\nu F)^\bar{\sigma}
\]

where the right hand arrow is the iterated distributive law\(^4\). Moreover, we fix the following family of constructors, that are jointly inverse to \( d_n \):

\[
c_\sigma : A_\sigma \times (\nu F)^\bar{\sigma} \to (\nu F)^n \quad \text{for} \quad \sigma : n \to \omega
\]

with the property

\[
[c_\sigma \mid \sigma : n \to \omega] = d_n^{-1}
\]

where \([\ldots]\) is the cotuple.

Using these conventions, the equations for \( \text{eat}_n : R_n \to (\nu F)^n \to B \) now become:

\[
\text{eat}_n \quad (\text{const} \ b) \ (c_\sigma(a, t)) = b
\]
\[
\text{eat}_n \quad (\text{fork} \ (f_\sigma)) \ (c_\sigma(a, t)) = \text{eat}_\bar{\sigma} f_\sigma(a)(t)
\]

(2)

Note that in contrast to the case with the functor for \( F(X) = A \times X \), we may have to invoke the \( \text{eat} \) function of some higher (or lower) arity in the recursive call.

Our next goal is to show that the equation above uniquely determine a continuous function of type \((\nu F)^n \to B\).

**Lemma 4.6** There exists a unique family of functions \( (\text{eat}_n)_{n \in \omega} \), where \( \text{eat}_n : R_n \to (\nu F)^n \to B \), that satisfies Equation (2).

**Proof** We fix \( Z = (Z_n) \in \text{Set}^\omega \) where \( Z_n = (\nu F)^n \to B \). Note that \( Z \) carries an \( M \)-algebra structure \( \zeta = (\zeta_n) \), defined by

\[
\zeta_n \ (\text{inl} \ b) = \lambda x. b
\]
\[
\zeta_n \ (\text{inr} \ (f_\sigma)) = [uc \ f_\sigma \mid \sigma : n \to \omega] \circ d_n
\]

\(^4\) The iterated distributive law is proved by the axiom of (finite) choice and some shuffling of quantifiers.
where, for a function \( f : A \to B \to C \), the mapping \( ucf : A \times B \to C \) is the un-curried version of \( f \), defined by \((a,b) \mapsto f(a)(b)\), as usual.

It is routine to check that the induced universal arrow \( u : R \to Z \), satisfies Equation 2. Conversely, every family of functions \( u_n \), with \( u_n : R_n \to Z_n \) that satisfies Equation 2 is an \( M \)-algebra morphism; hence \( eat_n \) is uniquely defined. \( \square \)

This is the first half of the theorem: we still need to ensure that \( eat_n \) is continuous. In view of Lemma 3.5, this can be ensured by explicitly providing a modulus of continuity for \( eat_n \). If we represent continuous functions of type \( Str A \to B \) by means of well founded, \( A \)-branching trees, the modulus of continuity is simply the length of the path through the tree that we need to traverse in order to reach a \( B \)-labelled leaf. For example, a modulus of continuity for the function represented in Figure 1 assigns 1 to each stream starting with 0, and 3 to streams with initial segment \((1,0,0)\). The next definition translates this to our dependently typed setting.

**Lemma 4.7** There exists a unique family of functions \((m_n)\), with \( m_n : R_n \to (\nu F)^n \to \omega \), that satisfies the equations

\[
\begin{align*}
m_n &\ (\text{const } b) \ (c\sigma(a,t)) = 0 \\
m_n &\ (\text{fork } (f\sigma)) \ (c\sigma(a,t)) = 1 + m_\sigma(f\sigma(a))(t)
\end{align*}
\]

for all \( \sigma : n \to \omega \), all \( a \in A_\sigma \) and all \( t \in (\nu F)^\sigma \).

**Proof** As in the proof of the preceeding lemma, we put \( Z = (Z_n) \) with \( Z_n = (\nu F)^n \to \omega \) and define an \( M \)-algebra structure on \( Z \) by

\[
\begin{align*}
\zeta_n &\ (\text{inl } b) = \lambda x.0 \\
\zeta_n &\ (\text{inr } (f\sigma)) = 1 + [uc f\sigma | \sigma : n \to \omega] \circ d_n
\end{align*}
\]

where, for a function \( g : X \to \omega \) we put \( 1 + g = \lambda x.1 + g(x) \). One then shows that \((m_n)\) arises as the unique algebra morphism of type \( R \to Z \), and that every family of functions satisfying Equation (3) qualifies as an algebra morphism. \( \square \)

We need one little technical lemma before we can show that \( m_n \) produces a modulus of continuity for \( eat_n \):  

**Lemma 4.8** Suppose \( x = c_\sigma(a,t) \) and \( y = c_\tau(b,s) \in (\nu F)^n \), where \( \sigma, \tau : n \to \omega \) and \( (a,b) \in A_\sigma \times A_\tau \) and \((t,s) \in (\nu F)^\sigma \times (\nu F)^\tau \). Assume \( k \geq 0 \).

If \( d_n(x,y) \leq 2^{-(k+1)} \), then \( \sigma = \tau \), \( a = b \) and \( d_\sigma(s,t) \leq 2^{-k} \).

**Proof** Note that, for \( x = (x_1, \ldots, x_n) \) and \( y = (y_1, \ldots, y_n) \) we have

\[
d_n(x,y) \leq 2^{-i} \iff \forall 1 \leq i \leq n. p_i(x_i) = p_i(y_i).
\]

9
The claim now follows from considering the diagram

\[
\begin{array}{c}
p_k \ \mu F \\
F^k \hookrightarrow F^{k+1} \sum_{i \in \omega} A_i \times (F^k)^i = F^{k+1}
\end{array}
\]

that defines the projection \( p_{k+1} \), where \( ! : F^1 \to 1 \) is the only such arrow. \qed

**Lemma 4.9** If \( m_n : R_n \to (\nu F)^n \to \omega \) is as in Lemma 4.7, then \( 2^{-m_n(a)} : (\nu F)^n \to \mathbb{R}_{\geq 0} \) is a modulus of continuity for \( \text{eat}_n(a) \).

**Proof** We prove the following statement by induction on \( k \):

For all \( n \in \omega \), all \( x, y \in (\nu F)^n \) and all \( r \in R_n \): if \( m_n(r)(x) = k \) and \( d_n(x, y) \leq 2^{-k} \), then \( \text{eat}_n(r)(a) = \text{eat}_n(r)(b) \).

For \( k = 0 \) there is nothing to show, as in this case we have \( r = \text{const} \) for some \( b \in B \). We now establish the claim for \( k + 1 \), where we assume that \( x = c_\sigma(a, t) \) and \( y = c_\tau(b, s) \) for \( \sigma, \tau : n \to k \), \( (a, b) \in A_\sigma \times A_\tau \) and \( (s, t) \in (\nu F)^\sigma \times (\nu F)^\tau \).

So assume that \( m_n(r)(x) = k + 1 \) and \( d_{k+1}(x, y) \leq 2^{-(k+1)} \). By Lemma 4.8 we have \( \sigma = \tau, a = b \) and \( d_\sigma(s, t) \leq 2^{-k} \). We abbreviate \( z = \bar{\sigma} = \bar{\tau} \).

By definition of \( m_n \), we have that \( r = \text{fork}(f_\sigma) \). Let \( r' = f_\sigma(a) \), thus \( r' = f_\tau(b) \). As \( d_\sigma(s, t) \leq 2^{-k} \), we have by induction hypothesis, that

\[ \text{eat}_z r's = \text{eat}_z r't \]

and \( \text{eat}_n(r)(c_\sigma(a, t)) = \text{eat}_n(r)(c_\tau(b, s)) \) follows by equational reasoning. \qed

In particular, this shows that all functions \( \text{eat}_n(r) \), with \( r \in R_n \), are continuous.

**Corollary 4.10** The functions \( \text{eat}_n(a) : (\nu F)^n \to B \) are continuous for all \( n \in \omega \) and all \( a \in R_n \).

## 5 Completeness

Recall that \((R_n)_{n \in \omega}\) is the carrier of the initial algebra of the representation functor associated with a power-series \( F : \text{Set} \to \text{Set} \). We have seen in the previous section that every element \( a \in R_n \), determines a continuous function \( \text{eat}_n(a) \) of type \((\nu F)^n \to B \). In this section we establish the converse, i.e. that every continuous function \( f : (\nu F)^n \to B \) can be represented by an element of \( R_n \). We begin by examining the special case of streams.

**Lemma 5.1** Every continuous function has a representative: \( \forall f : \text{Str} A \xrightarrow{\text{cts}} B. \exists r : \mu X.B + (A \to X). \text{eat} r = f \).

**Proof** The proof is non-constructive, and in no sense exhibits the representative. Suppose that \( f \) is continuous, but has no representative. Then it cannot be constant, and for some \( a : A \), \( f_a \) has no representative, where \( f_a(\alpha) = f(a, \alpha_0, \alpha_1, \ldots) \).

Continuing in this way, and applying the axiom of dependent choice, there is a
stream \((a_0, a_1, \ldots) : \text{Str}A\) such that for all \(n\), \((\ldots (f_{a_0})_{a_1} \ldots)_{a_{n-1}}\) is not constant. It follows that \(f\) cannot be continuous at this argument, in contradiction with our assumption. \(\square\)

We can immediately draw a corollary

**Lemma 5.2** A function \(f : \text{Str}A \to B\) is continuous if and only if it belongs to the least class \(K \subseteq (\text{Str}A \to B)\) containing all constant functions, and closed under the operation \(p_k\).

\[
p_k : (A \to (\text{Str}A \to B)) \to (\text{Str}A \to B)
\]

\[
pk(f) = \lambda \alpha.f(\alpha_0, \alpha')
\]

**Proof** The continuous functions contain the constant functions and are closed under \(p_k\), so all functions in \(K\) are continuous. The ‘only if’ direction is direct from the lemma. \(\square\)

Consider now the equations

\[
\text{rep} : (\text{Str}A \xrightarrow{\text{cts}} B) \to \mu X.B + (A \to X),
\]

\[
f \mapsto \begin{cases} \text{inl}(b) & \forall \alpha.f(\alpha) = b \\ \text{inr}(\lambda a.\text{rep}(f_a)) & \text{otherwise} \end{cases}
\]

where, for a function \(f : \text{Str}A \to B\), the mapping \(f_a : \text{Str}A \to B\) is its formal derivative, given by \(f_a(a_0, a_1, \ldots) = f(a, a_0, a_1, \ldots)\). Note that \(p_k\) is inverse to the derivative operation \(f \mapsto (\lambda a.f_a)\).

The inductive characterisation of continuity of \(f\) ensures that the set of representatives of an arbitrary continuous function is non-empty. It does not of itself produce a canonical representative.

It seems almost certain (but surprisingly tricky to prove) that the equations 4 have a unique solution, and so the function \(\text{rep}\) is classically well defined. Were that to be so, we could strengthen the lemma above to say that the function \(\text{eat}\) has a right-sided inverse: \(\text{eat} \circ \text{rep} = id : \text{Str}A \xrightarrow{\text{cts}} B\)

Next we turn to the general case.

**Theorem 5.3** Every continuous function has a representative:

\[
\forall n \in \omega. \forall f : (\nu F)^n \xrightarrow{\text{cts}} B. \exists r : R_n. \text{eat}_n r = f.
\]

**Proof** The proof is essentially the same as for the previous lemma. Suppose that \(f : (\nu F)^n \xrightarrow{\text{cts}} B\) is continuous, but has no representative in \(R_n\). Then \(f\) cannot be constant, and for some \(\sigma : n \to \omega\) and \(a : A_\sigma\), \(f^n\) has no representative, where the
function $f^a$ (the derivative of $f$ by $a$) is given by

$$f^a : (\nu F)^\sigma \to B$$

$$t \mapsto f(c_\sigma(a, t)).$$

As in the case of streams, continuing in this way we get a shrinking sequence of neighbourhoods of the argument throughout each of which $f$ is not constant. It follows that $f$ cannot be continuous at this argument, in contradiction with our assumption.

As in the case of streams, we can quickly deduce

**Corollary 5.4** A function of type $\sum_{n \in \omega} ((\nu F)^n \to B)$ is continuous if and only if it belongs to the least class $K$ containing all constant functions, and for each $n \in \omega$, $\sigma : n \to \omega$ closed under

$$pk_{n,\sigma} : (A_{\sigma} \to (\nu F)^\sigma \to B) \to (\nu F)^n \to B$$

$$pk_{n,\sigma}(f, c_\sigma(a, t)) = f(a, t)$$

This is the limit of what we have established.

It seems rather likely that these results can be strengthened to give an explicit description of a right-inverse to $\text{eat}_n$, as follows. Guided by the isomorphisms

$$(\nu F)^n \overset{d_n}{\underset{[c_\sigma]_{\sigma.n \to \omega}}{\longrightarrow}} \sum_{\sigma,n \to \omega} A_{\sigma} \times (\nu F)^\sigma$$

introduced in Section 4, we might stipulate the following equations for a family of function

$$\text{rep} : \prod_{n \in \omega} ((\nu F)^n \overset{\text{cts}}{\longrightarrow} B) \to R_n),$$

to be right inverse to $(\text{eat}_n)$:

$$\text{rep}_n : ((\nu F)^n \overset{\text{cts}}{\longrightarrow} B) \to R_n$$

$$f \mapsto \begin{cases} \text{const}\bar{b} & f = \lambda x.b \\ \text{fork}(\lambda a.\text{rep}_a f^a)_{\sigma,n \to \omega} & \text{otherwise} \end{cases} \quad (5)$$

where, for $a \in A_{\sigma}$, the function $f^a$ (the derivative of $f$ by $a$) is given by

$$f^a : (\nu F)^\sigma \to B$$

$$t \mapsto f(c_\sigma(a, t)).$$

Note that $c_\sigma : A_{\sigma} \times (\nu F)^\sigma \to (\nu F)^n$ and $t$ in the above has type $(\nu F)^\sigma$.

It is reasonable to conjecture that the equation (5) has a unique solution $\text{rep}$, such that
6 Conclusion

We have given a detailed intensional analysis of continuous functions from $\nu F$ to $B$, where $F$ is an $\omega$-continuous functor, $\nu F$ is the final coalgebra of $F$ equipped with the natural topology, and $B$ is discrete. More precisely, we have defined representations of such functions by elements of an inductively defined family $(R_n) : \text{Set}^{\omega}$, and established that this representation is complete, though not unique. It has an intensional character, reflecting differences in greediness of different implementations of the same function.

In the case of streams, the situation is ‘de luxe’: we have not shown it here, but we can extend the representation to continuous functions of type $\text{Str}A \to \text{Str}B$, using elements of a coinductively defined type $\nu X.R_A(B \times X)$. This gives us a useful model of stream-processing components, with in particular a composition operator definable directly on the representations. It seems very likely that something analogous can be carried out for final coalgebras of $\omega$-continuous functors beyond $(A \times)$, by approximately the same method.

The difficulty is largely a matter of getting all the ingredients under control in a simple notation and in the right form. It may be that a switch from defining our functors by power-series $\sum_n A_n \times X^n$ to defining them using a container type $\sum_{s : S} X^{P(s)}$ (where the sets $\{P(s) \mid s : S\}$ are finite) would simplify the notation and tame some of the details.

The interest of such an extension is in part that with more flexible functors, we can use types of potentially infinite data other than streams to model other forms of communication than by streams or unbounded buffers. For example, a Moore machine is describable as an element of $\nu X.O \times S^I$, where $I$ and $O$ are respectively the input and output alphabets. It may be possible to model communication by shared memory, or communication ports of some kind in this way, and so bring some powerful techniques to bear on these tricky situations.

Related work

On a superficial level, our results appear to be dual to the insights reported in [1], where it is shown that function types with certain domain types can be represented as final coalgebras, whereas we show that continuous functions are represented by initial algebras, at least when the codomain is discrete. Both ideas have at their core the adjunction $(A \to) \dashv (A \times)$ between exponentiation and multiplication.

There is some similarity between the work reported in [3] and our results. This work concerns methods for extracting constructive content from classical proofs, focusing on theorems involving infinite sequences and non-constructive choice principles. One method studied removes any reference to infinite sequences and transforms the theorem into a system of inductive definitions. The theorems studied are concerned with the concept of well-quasi orderings rather than, as in our
case, with continuity. However the (naive) definitions of both these have the $\Pi_1^1$ form "$\forall f : \text{Str}A, \exists n : \text{Nhd}(f), \ldots$. In both cases we give an inductive analysis of a $\Pi_1^1$ concept.

As pointed out in the introduction, our results can be traced back to Brouwer’s argument for the bar-theorem. Some light on, if not evidence for Brouwer’s analysis was provided by Kreisel and Troelstra who showed that a formal system for choice sequences divined from Brouwer’s ideas was interpretable in another formal system without infinite objects, but featuring an inductive definition. The name of the inductively defined set was ‘K’. This set was the inspiration for the function ‘eat’. Extensions of Gödel’s T by a generalised inductive definition like that for K were studied by Howard [7,6].

Acknowledgements

Thanks to Conor McBride for discussions on the topic, and suggestion of other approaches.

References

[1] T. Altenkirch. Representations of first order function types as terminal coalgebras. In S. Abramsky, editor, Proc. TLCA 2001, volume 2044 of Lecture Notes in Computer Science, pages 8–21. Springer, 2001.

[2] M. Barr. Terminal coalgebras in well-founded set theory. Theor. Comput. Sci., 114(2):299–315, 1993.

[3] U. Berger and M. Seisenberger. Applications of inductive definitions and choice principles to program synthesis. In In From Sets and Types to Topology and Analysis Towards practicable foundations for constructive mathematics, number 48 in Oxford Logic Guides, pages 137–148. Oxford University Press, 2005.

[4] E. Bishop. Foundations of Constructive Analysis. McGraw-Hill, 1967.

[5] M. Dummett. Elements of Intuitionism. Oxford Logic Guides. Clarendon Press, 2000. 2nd edition.

[6] W. Howard. Functional interpretation of bar induction by bar recursion. Compositio Mathematica, 20:107–124, 1968.

[7] W. Howard. Assignment of ordinals to terms for type zero bar recursive functionals. Journal of Symbolic Logic, 35:345, 1970.

[8] P. Martin-Löf. Mathematics of infinity. In P. Martin-Löf and G. Mints, editors, Conference on Computer Logic, volume 417 of Lecture Notes in Computer Science, pages 146–197. Springer, 1990.

[9] A. Troelstra. Choice Sequences: a chapter of intuitionistic mathematics. Oxford Logic Guides. Clarendon Press, 1977.