Resource-bounded Continuity and Sequentiality for Type-two Functionals (Extended Abstract)

Samuel R. Buss
Department of Mathematics
University of California, San Diego
La Jolla, CA
92093-0112
sbuss@ucsd.edu

Bruce M. Kapron
Computer Science Department
University of Victoria
Victoria BC
CANADA V8W 3P6
bmkapron@csr.uvic.ca

Abstract

We define notions of resource-bounded continuity and sequentiality for type-two functionals with total inputs, and prove that there are continuous functionals which cannot be efficiently simulated by sequential functionals. We also show that for some naturally-defined classes of continuous functionals, an efficient simulation is possible.

1. Introduction

The notion of *continuity* has long played an important role in higher-type computability theory, as well as in the theory of programming languages. A central theme in this area is the relationship between continuity and *sequentiality*. Continuity can be seen as a proper generalization of sequentiality, although there are certain special cases for which the two notions coincide for example when considering type-two functionals with *total* function inputs. The primary purpose of this paper it to point out that, even in this special case, continuity is in fact more powerful then sequentiality when the quantitative consideration of how much an input function must be queried, is admitted.

Early models of continuous higher-type functionals were proposed by Kleene [10] and Kreisel [13]. Kleene used the term *countable* rather than continuous, reflecting the fact that continuous functionals have countable representations. At this point in history, sequential higher-type computability was primarily understood in terms of Kleene's schemas S1-S9 [11],[12]. All of these models defined classes of functionals with total input. It was readily apparent that at type level two, the sequential and continuous functionals on total inputs coincided. Later results by Tait [20] and by Gandy and Hyland [4] showed that this was not the case at type level three (see, e.g., Normann [16], § 4.3-4.4.)

More recently, the relative power of continuous and sequential functionals on total inputs has been considered by Berger [1], Plotkin [18] and Normann [17]. Berger conjectured in [1] that all effective elements in the hierarchy of total continuous functionals were in fact definable in (sequential) PCF. In [18], Plotkin proved a special case of the conjecture, which is proved in full generality by Normann in [17]. Recently, new models of sequential higher-order functionals have been proposed by Longley [14] and van Oosten [23]. van Oosten's model provides a simple combinatorial structure which makes obvious the connection between sequential functionals and familiar notions of decision trees from computational complexity theory.

While the results in this paper pertain to models for computation in higher types, the techniques used are closely related to work in Boolean decision tree complexity. Namely, a technique known as "Blum's trick" ([2], [6], [21]), which is used to show that Boolean functions with small nondeterministic and co-nondeterministic complexity have small-depth decision trees is generalized to show that in certain cases, sequential functionals can efficiently simulate continuous functionals. A lower bound on Boolean decision trees for *search* problems [7] is the basis for our main result, showing that in general, not all type-two continuous functionals with total inputs can be efficiently simulated by sequential functionals. Note that the results presented here are about functionals in the full type hierarchy (at level two.)

2. Preliminaries

Definition 2.1 We denote by $\mathbb{N} \rightharpoonup \mathbb{N}$ the set of all partial functions from $\mathbb{N}$ to $\mathbb{N}$ and by $\mathbb{N} \to \mathbb{N}$ the set of all total functions. If $f : \mathbb{N} \rightharpoonup \mathbb{N}$, then $\mathrm{dom}(f)$, the *domain* of f is the set of all $z \in \mathbb{N}$ for which $f(z)$ is defined. A *type-two functional* is a mapping $F : (\mathbb{N} \rightharpoonup \mathbb{N}) \times \mathbb{N} \rightharpoonup \mathbb{N}$. A *finite function* is a partial function $c : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain.

A type-two functional F is *continuous* if for every $f : \mathbb{N} \rightharpoonup \mathbb{N}$ and $x \in \mathbb{N}$ for which $F(f,x)$ is defined, there is a finite set $Q \subseteq \mathrm{dom}(f)$ such that for all functions $g : \mathbb{N} \rightharpoonup \mathbb{N}$, if $f(z) = g(z)$ for all $z \in Q$, then $F(f,x) = F(g,x)$.

In [23], van Oosten introduces a class of sequential functionals over all finite types, based on a combinatory algebra of partial functions from $\mathbb{N}$ to $\mathbb{N}$. We will follow his approach in defining sequential type-two functionals. This definition is based on the idea of a "dialogue" between $g, f : \mathbb{N} \rightharpoonup \mathbb{N}$. g queries f at a sequence of inputs, with each query being determined only by answers to previous queries. After a sufficient number of successful queries, g may produce a final answer. Our definitions here are almost identical to those of [23], although our final definition of sequentiality is for the more general case of functionals which may take functions as well as numbers as input.

Definition 2.2 Let $\langle \cdot \rangle : \mathbb{N}^* \to \mathbb{N}$ be an efficient sequence encoding function. An encoding $u = \langle u_0, \ldots, u_{n-1} \rangle$ of a sequence is a *dialogue* between $g, f : \mathbb{N} \rightharpoonup \mathbb{N}$ if for all i, $0 \leq i < n-1$, there is a j such that $g(u^{<i}) = 2j$ and $f(j) = u_i$, where $u^{<i} = \langle u_0, \ldots, u_{i-1} \rangle$. The *application* $g|f$ is defined with value y (written $g|f = y$,) if there is a dialogue u between g and f such that $g(u) = 2y+1$. If $g(u^{<i}) = 2j$ we will say that g *queries* f *at* j. If $g(u) = 2y+1$, we will say that g *answers* y. For a given g and f, we do not rule out the possibility that there is no dialogue between g and f (e.g., g may never answer, or may reach a point where it does not have a query.) Moreover, there can be no more than one possible dialogue between f and g.

If $g : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ and $x \in \mathbb{N}$, let g_x denote the function $\lambda z.g(x,z)$. Every such g determines a type-two functional F_g defined by

$$F_g(f,x) = y \text{ iff } g_x|f = y.$$

A type two functional F is *sequential* if $F = F_g$ for some $g : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$

Suppose $F = F_g$ is a sequential functional. Then for any f and x, let

$$\mathrm{Queries}(g,f,x) = \{\tfrac{1}{2} g_x(u^{<i}) \mid 0 \leq i < n-1\},$$

where $\langle u_1, \ldots, u_{n-1} \rangle$ is the dialogue between g_x and f.

Note: The reader might wonder why we are giving first-class status to natural numbers, rather than working with pure types (representing $x \in \mathbb{N}$ as $\lambda y.x$.) While the latter approach would be possible, in our opinion the benefit of working directly with numbers outweighs the drawback of a mixed type structure.

We note the following alternate characterization of sequentiality:

Proposition 2.1 *A type-two functional F is sequential iff there is an oracle Turing machine M and a function g such that $F = M^g$.*

PROOF: Suppose $F = F_g$. It is easy to construct an oracle Turing machine M so that M^g on inputs f and x just follows the dialogue between g_x and f to compute $F(f,x)$. The converse follows directly from the sequential nature of Turing machine computation. ∎

It is well-known that there are continuous type-two functionals which are not sequential, e.g.,

$$F(f,x) = \begin{cases} 1 & \text{if } f(x) = 1 \text{ or } f(x+1) = 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

On the other hand, if we consider type-two functionals restricted to total inputs, it has long been known that continuous and sequential type-two functionals coincide, at least since the time of Kleene's work on the countable functionals [10]. A proof of this result is given in the next section. Note that early formulations of this result did not refer explicitly to sequential functionals as presented here, but to the equivalent notion of computability in an oracle.

In the sequel, we will not be concerned with partial functions or functionals, so we will use the term function to refer to total functions, and the term functional to refer only to *total type-two functionals with total function inputs.*

3. Resource-bounded continuity and sequentiality

The efficiency of continuous and sequential functionals will be defined in terms of the amount of information they must obtain about function inputs in order to produce an answer. While this obviously must account for the number of input values at which the function input must be queried, it is also reasonable to also account for the actual size of input values as well. This is formalized as follows.

Definition 3.1 Suppose F is a functional, $x, y \in \mathbb{N}$, and c is a finite function. Then c is a *y-certificate* for F, x if $F(f,x) = y$ for every $f \supseteq c$. When the actual value y is not important, we will just say that c is a *certificate* for F, x. The *size* of a certificate c, denoted $|c|$, is $\Sigma_{z \in \mathrm{dom}(c)} |z|$ where for $x \in \mathbb{N}$, $|x|$ denotes the length of the dyadic notation of x (so, in particular $|0| = 0$.)

Note: The definition of the size of a certificate is intended to capture the cost of accessing a function input f in a black-box manner, i.e., "how much" of f must be examined in order to determine $F(f,x)$. The given definition accounts not only for the number of queries that must be made to f, but also the size of the queries. This reflects our intuition that it should be more costly to construct larger inputs at which to query f. Note that our definition does not count the cost of answers from f as part of the overall cost. This choice is somewhat arbitrary. However, in the sequel we will normally bound the size of certificate uniformly in f, in such a way that the cost of the size of answers from f may always be accounted for in the bound.

The definition of continuity may now be rephrased using certificates:

Proposition 3.1 *A functional F is continuous iff for all functions f and $x, y \in \mathbb{N}$, $F(f,x) = y$ iff there is a y-certificate c for F, x such that $c \subseteq f$.*

By using certificate size, it is possible to define a more refined version of continuity.

Definition 3.2 Suppose F, B are functionals. F is B-*continuous* if for all functions f and $x, y \in \mathbb{N}$, $F(f,x) = y$ iff there is a y-certificate g for F, x such that $g \subseteq f$, and $|g| \leq B(f,x)$. In this case, we will also say that B is a *modulus of continuity for* F. If $\mathcal{F}$ is a collection of functionals, then F is $\mathcal{F}$-*continuous* if it is B-continuous for some $B \in \mathcal{F}$.

We can also introduce a resource-bounded version of sequentiality, based on the notion of the length of a dialogue.

Definition 3.3 The *length* of a dialogue $u = \langle u_0, \ldots, u_{n-1} \rangle$, between g and f, denoted $|u|$, is $\sum_{i=0}^{n-1} \left| \frac{1}{2} g(u^{<i}) \right|$

Definition 3.4 Suppose F, B are functionals. F is B-*sequential* if there is a function g such that for all functions f, $F(f,x) = y$ iff $g_x|f = y$ via a dialogue u between g_x and f, such that $|u| \leq B(f,x)$. If $\mathcal{F}$ is a collection of functionals, then F is $\mathcal{F}$-*sequential* if it is B-sequential for some $B \in \mathcal{F}$.

Note: The motivation behind our definition of the length of a dialogue is essentially the same as the for the definition of the size of a certificate.

We now prove a general result relating resource-bounded continuity and resource-bounded sequentiality.

Theorem 3.1 *Any functional which is B-continuous is $\lambda f \lambda x.(B(f,x) \cdot 2^{B(f,x)})$-sequential.*

PROOF: Suppose F is B-continuous. Given f and x, for $0 \leq i \leq 2^{B(f,x)} - 1$, let c_i be defined as follows:

$$c_i(u) = \begin{cases} f(u) & \text{if } u < i; \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Since F is B-continuous, it must be the case that some c_i is a certificate for F, x. Now we can design a g such that for inputs f and x, g_x successively queries f at $0, 1, \ldots$ until it has found $c_i \subseteq f$ which is a y-certificate for F, x, and then returns y. Note that this results in a dialogue consisting of at most $2^{B(f,x)}$ queries, with each query having size at most $B(f,x)$. ■

The method for constructing g in this proof is essentially Kleene's method of *associates* [10]. While it is perhaps just a matter of stating the obvious, to our knowledge no mention of the inefficiency of this construction is made in the computability theory literature. However, once we have noted this inefficiency, it is natural to ask whether a more efficient construction is possible.

In the main result of this paper, we show that in general, an efficient construction is not possible, by demonstrating a B-continuous functional, which is not $\lambda f \lambda x.(B(f,x))^n$-sequential for any $n \geq 0$. Before proving this result, however, we show that in the case where the continuous functional in question has a modulus functional B which is itself C-sequential, a construction which is efficient, relative to B *and* C, is possible. This implies that, for a certain natural class of moduli, continuity and sequentiality do coincide.

4. An efficient simulation

The result presented in this section is a generalization of "Blum's trick", ([2], [6], [21]), which relates certificate size and decision tree complexity for Boolean functions, to the case of the type-two functionals considered here. A similar proof is given in [8], but for a special case which is described in detail below.

We begin with the following simple fact about certificates. The following simple fact about certificates will prove essential in obtaining decision tree algorithms for functions with bounded certificates:

Lemma 4.1 *Suppose that c_y is a y-certificate for F at x and c_z is a z-certificate for F at x and $y \neq z$. Then there is some $x \in \mathrm{dom}(c_y) \cap \mathrm{dom}(c_z)$ so that $c_y(x) \neq c_z(x)$.*

PROOF: If this were not the case, consider any $h : \mathbb{N} \to \mathbb{N}$ so that $c_y \cup c_z \subseteq h$. We have $F(h,x) = y$ and $F(h,x) = z$, a contradiction. ■

Theorem 4.1 *Suppose that F is B-continuous, and B is C-sequential. Then F is $\lambda f \lambda x.(C(f,x) + [B(f,x)]^2)$-sequential.*

PROOF (sketch): We must show that there is a function g with the following properties.

1. For every f and x, $g_x|f = F(f,x)$.

2. For every f and x, the dialogue u between g_x and f has length bounded by $C(f,x) + [B(f,x)]^2$.

We begin by using Blum's trick to construct functions $g_{x,s}$ with the following properties:

1. For every f and x, if there is a y-certificate for F, x with size bounded by s, then $g_{x,s}|f = y$.

2. For every f and x, the dialogue u between $g_{x,s}$ and f has length bounded by s^2.

Once we are able to construct $g_{x,s}$, we can then construct g as follows: g, when given inputs f and x, first computes $B(f,x)$ via a dialogue of length $C(f,x)$. This is possible since B is C-sequential. Having done so, it then proceeds as $g_{x,B(f,x)}$. To construct $g_{x,s}$ we define for some $t \leq s+1$, a sequence $g^0, g^1, \dots, g^t$, of functions where g_0 is the everywhere undefined function and $g^t = g_{x,s}$. By u^i, $0 \leq i \leq s$, we will denote the initial segment of the dialogue between $g_{x,s}$ and f which is determined by g^i, and let $f^i \subseteq f$ be the portion of f determined by g^i. Fix an enumeration of all certificates for F, x of size at most s, such that all z certificates appear before any $(z+1)$-certificates. We start by assuming that $t = s+1$.

Suppose that g_{k-1} has been defined, where $1 \leq k \leq t$. Let c be the next certificate which is consistent with f^{k-1}. Suppose that c is a y-certificate. g_k will be an extension of g_{k-1} that is, for all proper initial segments u' of u^{k-1}, $g_k(u') = g_{k-1}(u')$. There are now three possible cases for the definition of g_k.

If there is a y-certificate $c' \subseteq f^{k-1}$, or if all the remaining certificates consistent with f^{k-1} are y-certificates, then $g_k(u^{k-1}) = 2y+1$, and t is set to k (i.e., the construction terminates at this point.)

If neither of the preceding cases hold, g_k is the extension of g_{k-1} determined by c. Suppose that $\mathrm{dom}(c) = \{x_1, \dots, x_r\}$, $r \leq s$. Then $g_k(u^{k-1}) = 2x_1$ and for $1 \leq i < r$,

$$g_k(u^{k-1} {}^\frown \langle f(x_1), \dots, f(x_i)\rangle) = 2x_{i+1},$$

where $^\frown$ denotes concatenation.

It remains to show that the construction always terminates with a g_t which produces an answer. Suppose that $F(f,x) = y$, and that $c^* \subseteq f$ is a y-certificate for F at x of size at most s. Suppose that we have not terminated with a g_t producing an answer before defining g_{s+1}. Let c^i, $1 \leq i \leq s$ be the certificate chosen at stage i of the construction. By the construction must be the case that c^s is a z-certificate for some $z \leq y$ (since the construction would not allow us to get past c^*.) Suppose first that $z < y$. Then by lemma 4.1, there must be an $x^i \in \mathrm{dom}(c^i)$ such that $c^i(x^i) \neq c^*(x^i)$. Moreover, by the construction each x^i is unique. Since $|c^*| \leq s$, we must have that $|\mathrm{dom}(c^*)| \leq s$ and so $c^* \subseteq f^s$. But then by the construction g_{s+1} *will* produce an answer. Now suppose that $z = y$. A similar argument shows that all the remaining certificates consistent with f^s must be y-certificates, so that the construction again will give a g_{s+1} which produces and answer.

Finally, we can see that at each stage of the construction the length of the dialogue is extended by at most s (since each certificate has size bounded by s.) So the overall length of the dialogue between f and $g_{x,s}$ is bounded by s^2. ■

In [8], Kapron presents a class of functionals which are *feasibly continuous*. Such functionals have certificates whose size is *feasible*, or *polynomial*, in terms of the size of the inputs f and x. Of course, it is not entirely clear what it means to be polynomial in the size of a *function* f. One approach is to work backwards from the situation presented by type-one functions: a type-one function f has *polynomial growth rate* if for some polynomial p, $|f(x)| \leq p(|x|)$ for all x. Thus we propose that to be polynomial in f and x means to be bounded, for all f and x, by $|F(f,x)|$ where F is a functional with polynomial growth rate. Now it remains to characterize the functionals with polynomial growth rate. We begin with Mehlhorn's [15] class of type-two, *polynomial time* functionals, which we will denote BFF (see [3] for an explanation of this terminology.) The following presentation of BFF is based on that of Townsend [22].

Definition 4.1 A rank (k,l) functional F is obtained by

- *Composition* from $H, G_1, \dots, G_l$ if

$$F(\vec{f},\vec{x}) = H(\vec{f}, G_1(\vec{f},\vec{x}), \dots, G_l(\vec{f},\vec{x}), \vec{x})$$

- *Expansion* from G if

$$F(\vec{f},\vec{g},\vec{x},\vec{y}) = G(\vec{f},\vec{x})$$

- *Limited recursion on notation* from G, H_0, H_1 and K if

$$\begin{aligned} F(\vec{f},\vec{x},0) &= G(\vec{f},\vec{x}) \\ F(\vec{f},\vec{x},2y) &= H_0(\vec{f},\vec{x},y,F(\vec{f},\vec{x},y)), \quad y>0 \\ F(\vec{f},\vec{x},2y+1) &= H_1(\vec{f},\vec{x},y,F(\vec{f},\vec{x},y)) \\ |F(\vec{f},\vec{x},y)| &\leq |K(\vec{f},\vec{x},y)| \end{aligned}$$

Let $\{F_1, \dots, F_k\}$ be a set of functionals. The class $\mathrm{BFF}(\{F_1, \dots, F_k\})$ of functionals *basic feasible in* $\{F_1, \dots, F_k\}$ is the smallest class containing all poly-time (type-1) functions, $F_1, \dots, F_k$, and the *application functional* Ap define by $\mathrm{Ap}(f,x) = f(x)$, which is closed under composition, expansion and limited recursion on notation. The class BFF of *basic feasible functionals* is just $\mathrm{BFF}(\emptyset)$.

Note: The system of [22] was formulated for computation over strings. We have made the straightforward modification to handle natural numbers. Also, we include all type-1 poly-time functions as initial functions. This simplifies the closure schemes required.

Certainly, if all functionals in BFF are polynomial time computable, then they all have polynomial growth rate. However, BFF adds an extra restriction, namely computability, which is not essential. This restriction may be relaxed by allowing computation relative to a (type-one) oracle with polynomial growth rate.. Thus we define the class $\mathbf{Poly}_1$ of second-order polynomial bounds to be all functionals of the form $\lambda f \lambda x.|F(f,x)|$, where F is a functional for which there exists a (type-one) function g with polynomial growth rate and a functional $G \in$ BFF such that $F(f,x) = G(g,f,x)$ for all f and x.

Note that every functional in $\mathbf{Poly}_1$ is $\mathbf{Poly}_1$-sequential, so we can apply Theorem 4.1 to obtain:

Corollary 4.1.1 **[8]** *Every* $\mathbf{Poly}_1$*-continuous functional is* $\mathbf{Poly}_1$*-sequential.*

5. A lower bound

The $\mathbf{Poly}_1$-continuous functionals have the property that they depend on only a *feasible* (polynomially bounded) amount of information about their function inputs (relative to some understanding of what it means to be polynomially bounded in a *function*.) It is possible to give a more direct definition of continuous functionals which depend on a polynomially bounded amount of information about their function inputs using *second-order polynomials*, and the notion of a *size* of a function.

Second-order polynomials were first introduced in [9], where they were used to give an alternate characterization of BFF.

Definition 5.1 A *second-order polynomial* over function variables $f_1, f_2, \ldots, f_k$ and number variables $x_1, x_2, \ldots, x_l$ is one of the following:

1. A number constant $n \in \mathbb{N}$
2. A number variable x_i
3. $\mathbf{p} + \mathbf{q}$, where $\mathbf{p}$ and $\mathbf{q}$ are second-order polynomials
4. $\mathbf{p} \cdot \mathbf{q}$, where $\mathbf{p}$ and $\mathbf{q}$ are second-order polynomials
5. $f_i(\mathbf{p})$, where $\mathbf{p}$ is a second-order polynomial.

If $\mathbf{p}$ is a second-order polynomial, $f_1, \ldots, f_k : \mathbb{N} \to \mathbb{N}$ and $x_1, \ldots, x_l \in \mathbb{N}$, then $\mathbf{p}(f_1, \ldots, f_k, x_1, \ldots, x_l)$ denotes a value in $\mathbb{N}$, defined in the obvious way.

Defining the size of a function f is somewhat trickier. The basic idea is that we want a measure of the worst (largest) output that could be produced by f for inputs of a given size (see [9] for a discussion of this.)

Definition 5.2 For any function $f : \mathbb{N} \to \mathbb{N}$, the *size* of f, denoted $|f|$, is the function $\lambda n. \max_{|y| \leq n} |f(y)|$.

The following result from [9] shows the importance of second-order polynomials in the characterization of type-two polynomial time computability.

Theorem 5.1 **[9]** *A functional is in* BFF *iff it is computed by an oracle Turing machine* M *such that for all inputs* f *and* x*,* M*'s running time is bounded by a second-order polynomial in* $|f|$ *and* $|x|$.

Let $\mathbf{Poly}_2$ be the set of all functionals of the form $\lambda f \lambda x.\mathbf{p}(|f|, |x|)$, where $\mathbf{p}$ is a second-order polynomial.

It is not hard to see that there is no second-order polynomial $\mathbf{p}$ such that the functional $\lambda f \lambda x.|f|(|x|)$ is $\mathbf{p}$-continuous. If it were, there must be an ordinary polynomial p such that for any 0-1 valued f, the value of $|f|(|x|)$ is determined by a certificate of size $p(|x|)$. Choose an x_0 such that $2^{|x_0|} > p(|x_0|)$. Let $f_0 = \lambda x.0$. Since $|f_0|(|x_0|) = 0$, there must be a 0-certificate $c_0 \subseteq f_0$ for $\lambda f \lambda x.|f|(|x|)$ at x_0, such that $|c_0| \leq p(|x_0|)$. Now let g be any function such that $c_0 \subseteq g$ but $g(y) > 0$ for some y such that $|y| \leq |x_0|$. This is possible by the choice of x_0. But now $|g|(|x_0|) > |f_0|(|x_0|)$, contradicting the fact that c_0 is a certificate.

So we cannot apply Theorem 4.1 to obtain a result analogous to Corollary 4.1.1 for the $\mathbf{Poly}_2$-continuous functionals. In fact, there is a $\mathbf{Poly}_2$-continuous functional which is not $\mathbf{Poly}_2$-sequential. The remainder of this section is devoted to demonstrating the existence of such a functional.

The functional F_R defined below is closely related to a Boolean function defined in [7], which is concerned with the power of Boolean decision trees for computing search functions.

The definition of F_R depends on a version of Ramsey's theorem (see [5]). F_R is defined in terms a a certain partial colouring function defined on complete graphs (denoted K_x):

Definition 5.3 For any function f and number $x > 0$, define the partial colouring function $\chi_{f,x}$ on K_x as follows: suppose that the edges of K_x are numbered $1, 2, \ldots, \binom{x}{2}$. Then for $1 \leq i \leq \binom{x}{2}$

$$\chi_{f,x}(i) = \begin{cases} 0 & \text{if } f(i) < 2^x - 1 \text{ and even} \\ 1 & \text{if } f(i) < 2^x - 1 \text{ and odd} \end{cases}$$

Note that if $f(i) \geq 2^x$, then $\chi_{f,x}(i)$ is undefined. Given a $\chi_{f,x}$, we say that it has a *monochromatic* K_k if there is a set of k nodes such that $\chi_{f,x}$ colours the complete subgraph on these nodes all 0 or all 1.

The functional F_R (for "Ramsey functional") is defined as follows: $F_R(f,0) = 0$, and for $x > 0$,

$$F_R(f,x) = \begin{cases} 0 & \text{if } \chi_{f,x} \text{ has a monochromatic} \\ & K_k \text{ with } k = \frac{1}{2}\log x; \\ 1 & \text{if } \chi_{f,x} \text{ has no monochromatic} \\ & K_k \text{ with } k = \frac{1}{2}\log x. \end{cases}$$

Theorem 5.2 *F_R is $\mathbf{Poly}_2$-continuous.*

PROOF: We will show that F_R is $\lambda f \lambda x.|x|^2(|f|(2|x|)^2 + |x|^2)$-continuous. First, suppose that $F_R(f,x) = 0$. Then there is a certificate, consisting of all the edges in the monochromatic subgraph. Such a certificate will have size bounded by $|x|^4$, since there will be $\binom{x}{2} \leq |x|^2$ values in the domain of the certificate (one for each edge of the monochromatic subgraph), and each value in the domain of the certificate is no greater than $\binom{x}{2}$. Now suppose that $F_R(f,x) = 1$. We now appeal to a result of Ramsey theory (see [5]) which states that any (total) 2-colouring of K_x has a monochromatic K_k where $k = \frac{1}{2}\log x$ nodes. Hence, if $F_R(f,x) = 1$. it follows that $\chi_{f,x}$ is undefined for some i, $1 \leq i \leq \binom{x}{2}$. Then is must be the case that $f(i) \geq 2^x - 1$, and so $|f(i)|^2 \geq x^2 \geq \binom{x}{2}$ and $|i| \leq 2|x|$. Thus there is a certificate, consisting of all the edges of K_x, of size $|x|^2|f|(2|x|)^2$. ■

Before proving that F_R is not $\mathbf{Poly}_2$-sequential, we need a simple result regarding the behaviour of sequential functionals.

Proposition 5.1 *Suppose $F = F_g$ is a sequential functional, f and f' are functions and $x \in \mathbb{N}$. Let $Q = \mathrm{Queries}(g,f,x)$ and $Q' = \mathrm{Queries}(g,f',x)$. If $f_Q = f'_{Q'}$, then g's dialogue with f is identical to its dialogue with f'.*

Theorem 5.3 *F_R is not $\mathbf{Poly}_2$-sequential.*

PROOF: The proof is by contradiction. Suppose that $F = F_g$, and that there is a second-order polynomial $\mathbf{p}$ such that for every f and x, the dialogue between g_x and f has length bounded by $\mathbf{p}(|f|,|x|)$.

First note that there must be an ordinary polynomial p such that for any 0-1 valued f, $\mathbf{p}(|f|,|x|) \leq p(|x|)$. Next, by a lower bound on Ramsey numbers we have that for sufficiently large x, there is a 2-colouring χ_0 of the complete graph on $x^{0.25}$ nodes with no monochromatic K_k where $k = \frac{1}{2}\log x$ (see [5], Theorem 4.2.1.).

Choose an x_0 for which such a χ_0 exists, and for which $x_0^{0.25} > p(|x_0|)$. For any 0-1 valued input function f, $|\mathrm{Queries}(g,f,x_0)| \leq p(x_0)$ (*).

Consider any f_0 for which $f_0(z) \in \{0,1\}$ for all $z \in \mathrm{Queries}(g,f_0,x_0)$. By Proposition 5.1 and the definition of χ_{f_0,x_0} it must be the case that for any such f_0, $F(f_0,x_0) = g_{x_0}|f_0 = 0$. But by Proposition 5.1 and (*), it is also the case that for any such f_0, g_{x_0} can query f_0 at no more than $p(|x_0|)$ different inputs, where $p(x_0) < x_0^{0.25} < \binom{x_0^{0.25}}{2}$. Combining this fact with the existence of the 2-colouring χ_0, we can construct such a f_0 for which $F_g(f_0,x_0) = 1$. Namely, f_0 will correspond to a colouring which uses χ_0 to colour the edges revealed to g_{x_0}, but leaves the colour of some remaining edge undefined. This gives us the required contradiction. ■

Note: This result could in fact be strengthened to show that any sequential algorithm for computing $F_R(f,x)$ must query f at exponentially many inputs. A proof will be given in the full paper.

6. Related results

All the results of this paper pertain to full collections of functionals at type level two. We have not addressed the issue of how effective versions of these collections are related. This question has been considered by Royer [19]. He defines an effective version of the $\mathbf{Poly}_1$-continuous functionals, using the notion of a *poly-time associate*. The effective version of the $\mathbf{Poly}_1$-sequential functionals is just BFF itself. Royer shows that if the effective $\mathbf{Poly}_1$-continuous functionals coincide with BFF, then there is a polynomial time algorithm for factoring integers.

7. Conclusions and future work

The relationship between continuity and sequentiality has long been an issue in the theory of computability for higher-type functions, and recent work has shed more light on the issue, in particular for total functionals. The work presented here formalizes the idea that, even in cases where classes of functionals coincide in terms of expressive power they may differ when considerations about efficiency are taken into account. We have shown that techniques from ordinary complexity theory may be applied fruitfully in this area.

This paper presents a starting-point for a number of possible threads of investigation. At type level two, a fuller investigation of the efficiency of various models of computation are possible. The relationship between bounded continuity, and computability with bounded nondeterminism should be more clearly delineated, perhaps in terms of a version of PCF++ with a bounded $\exists$ operator. Some of these issues will be dealt with in the full version of this paper. At higher types, we can investigate the recent results by Berger, Plotkin and Normann relating effective continuity and PCF definability with respect to efficiency. In another direction, it should be possible to develop some sort of complexity theory for computation over the sequential functionals of van Oosten and Longley. In short, recent developments in the theory of higher-type computability raise quite a number of interesting problems in the theory of higher-type complexity.

8. Acknowledgements

We would like to thank Martin Hoffman, Russell Impagliazzo, Hanno Nickau, and Jim Royer for helpful discussions and suggestions.

References

[1] U. berger. Total sets and objects in domain theory. *Annals of Pure and Applied Logic*, (60):91–117, 1993.

[2] M. Blum and R. Imagliazzo. Generic oracles and oracle classes. In *Proc. 28th Ann. IEEE Symposium on Foundations of Computer Science*, pages 118–126, 1987.

[3] S. Cook and B. Kapron. Characterizations of the basic feasible functionals of finite type. In S. Buss and P. Scott, editors, *Feasible Mathematics*, pages 71–93. Birkhauser, Boston, MA, 1990.

[4] R. Gandy and M. Hyland. Computable and recursively countable functions of higher type. In R. Gandy and M. Hyland, editors, *Logic Colloquium '76*, pages 407–438. North-Holland, Amsterdam, 1977.

[5] R. Graham, B. Rothschild, and J. Spencer. *Ramsey Theory*. John Wiley and Sons, Inc., New York, 2nd edition, 1990.

[6] J. Hartmanis and L. Hemachandra. Robust machines accept easy sets. *Theoretical Computer Science*, (74):217–225, 1990.

[7] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proc. 3rd Ann. IEEE Structure in Complexity Theory*, pages 29–38, 1988.

[8] B. Kapron. Feasibly continuous type-two functionals. *computational complexity*, (8):188–201, 1999.

[9] B. Kapron and S. Cook. A new characterization of type-2 feasibility. *SIAM J. Comput.*, (25):117–132, 1996.

[10] S. Kleene. Countable functionals. In A. Heyting, editor, *Constructivity in Mathematics*, pages 81–100. North-Holland, Amsterdam, 1959.

[11] S. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. Amer. Math. Soc.*, (91):1–52, 1959.

[12] S. Kleene. Recursive functionals and quantifiers of finite types II. *Trans. Amer. Math. Soc.*, (108):106–142, 1963.

[13] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland, Amsterdam, 1959.

[14] J. Longley. The sequentially realizable functionals. Technical Report ECS-LFCS-98-402, University of Edinburgh LFCS, December 1998.

[15] K. Mehlhorn. Polynomial and abstract subrecursive classes. *J. Comput. System Sci.*, (12):147–178, 1976.

[16] D. Normann. *Recursion on the Countable Functionals*, volume 811 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1980.

[17] D. Normann. Computability over the partial continuous functionals. Pure Mathematics Preprint Series 1998 2, Department of Mathematics, Univesity of Oslo, 1998. Revised version, October 1998.

[18] G. Plotkin. Full abstraction, totality and PCF. *Math. Struct. Comp. Sci.*, (9):1–20, 1999.

[19] J. Royer. On continuous type-2 feasible functionals. In preparation.

[20] W. Tait. Continuity properties of partial recursive functionals of finite type.

[21] G. Tardos. Query complexity, or why is it difficult to separate $NP^a \cap co\text{-}NP^a$ from P^a by a random oracle? *Combinatorica*, (9):385–392, 1989.

[22] M. Townsend. Complexity for type-2 relations. *Notre Dame J. Formal Logic*, (31):241–262, 1990.

[23] J. van Oosten. A combinatory algebra for sequential functionals of finite type. In S. Cooper and J. Truss, editors, *Models and Computability: Invited Papers from Logic Colloquium '97*. Cambridge University Press, Cambridge, 1999.