10

The importance of cardinality, separability, and compactness in computer science with an example from numerical signal analysis

Klaus E. Grue

Abstract

This chapter gives an example where topological guidance has been essential in developing a numerical algorithm for solving a problem from signal analysis. The chapter explains the importance of cardinality, separability, and compactness in numerical analysis, and provides examples of spaces that can be made separable or compact by a non-standard choice of topology. Furthermore, the chapter suggests a definition of 'approximate computability' and analyses some immediate consequences of the definition.

10.1 Introduction

Computers can perform accurate computations on some sets like the set of integers, and approximate computations on other sets like the set of real numbers. There are also sets on which computers can perform neither accurate nor approximate computations. One observation to be stated is that computers can only perform approximate computations over separable spaces (separability is necessary but not sufficient). In particular, computers can only perform approximate computations over spaces of cardinality at most $2^{\aleph_0}$.

In the example to be given, the problem is to find a bounded real function with certain properties. As the set of bounded real functions has cardinality exceeding $2^{\aleph_0}$, the problem is reformulated to take place in L_∞ which has cardinality $2^{\aleph_0}$.

We equip L_∞ with the so-called weak*-topology to obtain separability. The unit sphere (w.r.t. $\|\bullet\|_{1_\infty}$) in L_∞ is compact (w.r.t. the weak*-topology), and this result is used as follows:

1. to prove that solutions to the problem exist;
2. to prove that the sequence of approximations generated by the numerical algorithm has a subsequence that converges towards a solution;
3. to prove a property about the solutions.

From the point of view of numerical analysis, it may be somewhat surprising that it is possible to compute with bounded functions (or, rather, L_∞-functions) without imposing any further restrictions on these functions. Applications of computation on L_∞ are numerous. As one example, a grey tone screen image can be represented as an L_∞-function which gives the intensity of each point as an L_∞-function of time and two spatial coordinates. Colour images can be represented as three L_∞-functions. By representing computer images this way it becomes possible to write programs that produce images independently of screen resolution and refresh rate.

The concept of 'computability' is important in computer science. The chapter ends by giving some suggestions of what 'approximate computability' could mean and states the observation (well known from intuitionistic logic) that merely continuous functions can be 'approximately computable'.

The example is interdisciplinary in nature, but the results needed from each discipline are not particularly deep.

10.2 An optimization problem

The problem which was actually solved by topological guidance is described in (Grue 1985). However, this chapter considers a slightly different problem which is more interesting from the point of view of topology and exhibits less technicalities.

The problem to be considered is the following 'optimization problem'. Let $a_1, \ldots, a_n \in \mathbb{R}$ (the set of real numbers). Let $h_0, \ldots, h_n \in L_1$, that is, let $h_0, \ldots, h_n : \mathbb{R} \rightarrow \mathbb{R}$ be real functions such that

$$\|h_i\|_1 = \int_{-\infty}^{+\infty} |h_i(t)|\, dt < +\infty \quad i \in \{0, \ldots, n\}$$

(All integrals are Lebesgue integrals.) Find a function $g : \mathbb{R} \rightarrow \mathbb{R}$ such that

$$\forall t \in \mathbb{R} : |g(t)| \leqslant 1$$

and

$$\langle g, h_i \rangle = \int_{-\infty}^{+\infty} g(t)\, h_i(t)\, dt \leqslant a_i \quad i \in \{1, \ldots, n\}$$

$$\langle g, h_0 \rangle = \int_{-\infty}^{+\infty} g(t)\, h_0(t)\, dt \quad \text{is maximal}$$

(Also prove the existence of g.) The importance of this problem follows from Golomb and Weinberger (1959), except that Golomb and Weinberger deal mainly with finite energy signals (L_2) rather than bounded signals. Furthermore, an application of the solution to the problem is stated in the Appendix.

The importance of separability

Let $\mathbb{Z}$ denote the set of integers, and let $D = \{m2^n \mid m, n \in \mathbb{Z}\}$ be the set of finite binary expansions. The set D is countable, which means that elements of D can be represented using finitely many bits in a computer. Different elements of D may require different numbers of bits, and elements of D may require arbitrarily many bits, but each element of D merely requires finitely many bits. The set D is a dense subset of $\mathbb{R}$, which means that any element of $\mathbb{R}$ can be approximated arbitrarily well by elements of D. D is suited to numerical computations in $\mathbb{R}$ exactly because D is a countable dense subset of $\mathbb{R}$. A topological space is 'separable' if it has a countable dense subset. Hence, a necessary requirement to make approximate computations in a topological space is that the space is separable.

As any separable space has cardinality at most $2^{\aleph_0}$, another necessary requirement is that the space has cardinality at most $2^{\aleph_0}$.

Satisfaction of the cardinality condition

The optimization problem mentioned above aims at finding a bounded function g. However, the set of bounded functions has cardinality exceeding $2^{\aleph_0}$, and so we formulate the problem to require $g \in L_\infty, \|g\|_{1_\infty} \leqslant 1$ instead of $g : \mathbb{R} \to \mathbb{R}, \forall t \in \mathbb{R} : |g(t)| \leqslant 1$. Assuming $g \in L_\infty$ means the following.

1. We identify functions that are equal almost everywhere, that is, we identify functions g_1 and g_2 for which $\int_{-\infty}^{+\infty} |g_1(t) - g_2(t)|\, dt = 0$.
2. We assume that $\int_0^x g(t)\, dt$ is defined and finite for all $x \in \mathbb{R}$.

Both these restrictions are inessential to the optimization problem. The two restrictions together reduce the space to be considered to cardinality $2^{\aleph_0}$.

Satisfaction of the separability condition

The norm $\|\bullet\|_{1_\infty}$ induces a topology on L_∞ which, unfortunately, is not separable. Fortunately, however, the so-called weak*-topology on L_∞ makes L_∞ separable. The weak*-topology is uniquely determined by the following property.

Let $g, g_1, g_2, \ldots \in L_\infty$. We have $g_i \to g$ for $i \to +\infty$ w.r.t. the weak*-topology iff $\langle g_i, h\rangle \to \langle g, h\rangle$ for all $h \in L_1$.

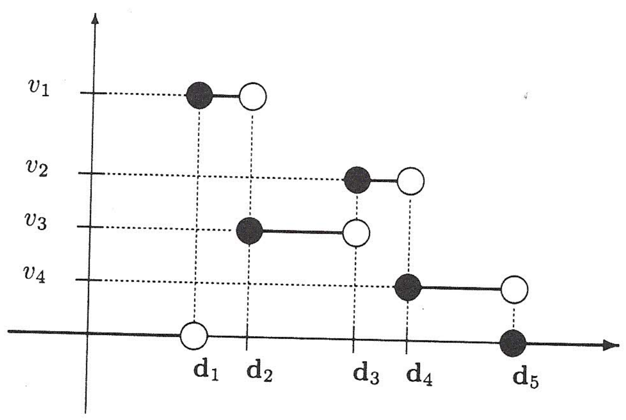

Fig. 10.1 An example step function

One may think of this concept of convergence as an adaptation of pointwise convergence to L_∞.

The choice of the weak*-topology has another benefit. Alaoglu's theorem (Rudin 1973) states that the unit sphere

$$B = \{g \in L_\infty \mid \|g\|_{1_\infty} \leqslant 1\}$$

is compact w.r.t. the weak*-topology, that is, any sequence $g_1, g_2, \ldots \in B$ has a subsequence which converges w.r.t. the weak*-topology (and the limit of any such sequence again belongs to B).

A countable dense subset of L_∞

Let $n \geqslant 1$ and let $v = (v_1, \ldots, v_{n-1}) \in D^{n-1}$. Let $d = \{d_1, \ldots, d_n\} \subseteq D$ be a set of n distinct elements of D (the set of finite binary expansions), and let $(\mathbf{d}_1, \ldots, \mathbf{d}_n)$ be the d_is sorted in ascending order. We define the step function $S_{v,d}(t)$ as

$$S_{v,d}(t) = \begin{cases} 0 & \text{for } t \in (-\infty, \mathbf{d}_1) \\ v_i & \text{for } t \in [\mathbf{d}_i, \mathbf{d}_{i+1}), i \in \{1, \ldots, n-1\} \\ 0 & \text{for } t \in [\mathfrak{a}_n, +\infty) \end{cases}$$

In particular, for $n = 1$, we define $S_{v,d}(t) = 0$ for $t \in (-\infty, \infty)$. Figure 10.1 shows $S_{(v_1,\ldots,v_4),\{d_1,\ldots,d_5\}}$ for some $v_1, \ldots, v_4$ and $d_1, \ldots, d_5$.

If $d \subseteq D$ has n elements, then we define

$$V_d = \{S_{v,d} \mid v \in D^{n-1}\}$$

The set V_d is an $(n-1)$-dimensional vector space. Further define V as the union of the V_ds:

$$V = \bigcup\{V_d \mid d \subseteq D, d \text{ finite }\}$$

The set V is a countable dense subset of L_∞ (with respect to the weak*-topology). Hence, V is a candidate for performing approximate computations with elements of L_∞. Any countable superset of V may also serve as a candidate, as any superset of V is also dense in L_∞.

A numerical solution to the optimization problem

Let $d_1, d_2, \ldots$ be an enumeration of D, that is, let $d_1, d_2, \ldots$ be any sequence without repetitions such that $\{d_1, d_2, \ldots\} = D$. Define $e(i) = \{d_1, \ldots, d_i\}$. We can easily verify that

$$V = \bigcup\{V_{e(i)} \mid i \in \mathbb{N}\}$$
$$V_{e(1)} \subseteq V_{e(2)} \subseteq \cdots$$

We may use this observation to outline a numerical algorithm for solving the optimization problem. The algorithm is not known to be present in the literature. Let

$$W = \{g \in L_\infty \mid \|g\|_{1\infty} \leqslant 1 \wedge \langle g, h_1\rangle \leqslant a_1 \wedge \cdots \wedge \langle g, h_n\rangle \leqslant a_n\}$$

We may state the optimization problem as follows: find $g \in W$ such that $\langle g, h_0\rangle$ is maximal (and prove the existence of g).

If $W = \emptyset$ then, obviously, the problem has no solutions, and so we rule out this case. In the problem considered in the Appendix, we have $a_i > 0$, $i \in \{1, \ldots, n\}$, and so the zero function is an element of W in that case.

Further, to rule out pathological cases, we assume that W is the closure of $W \cap V$, that is, that any element of W can be approximated arbitrarily well by elements of V. This will normally be easy to verify for practical applications.

As an example of a pathological case, let $n = 2$ and let h_1 be the constant function $h_1(t) = 1$. Let a_1 be any real number which has no finite binary expansion (for example, π) and let $h_2 = -h_1$ and $a_2 = -a_1$. We have $W = \{g \in L_\infty \mid \|g\|_{1\infty} \leqslant 1 \wedge \langle g, h_1\rangle = a_1\}$ so that $W \neq \emptyset$ and $W \cap V = \emptyset$.

From the assumptions we deduce that $W \cap V \neq \emptyset$. Hence, for any sufficiently large $k \in \mathbb{N}$, we have $W \cap V_{e(k)} \neq \emptyset$. Now, for each $i \geqslant k$ we may solve the following problem: find $\mathbf{g}_i \in W \cap V_{e(i)}$ such that $\langle \mathbf{g}_i, h_0\rangle$ is maximal. As we shall see later, this may be done using the simplex method (Rockafellar 1970). According to Alaoglu's theorem, as $\|\mathbf{g}_i\|_{1\infty} \leqslant 1$

for $i \geqslant k$, the series $\mathbf{g}_k, \mathbf{g}_{k+1}, \ldots$ has a convergent subseries (w.r.t. the weak*-topology), and the limit g for such a series obviously maximizes $\langle g, h_0 \rangle$ for $g \in W$.

Hence, we have a method for finding a series $\mathbf{g}_k, \mathbf{g}_{k+1}, \ldots$ which can be thinned into a convergent subseries. The process of actual thinning is irrelevant for the problem in the Appendix, where it is the value of $\langle g, h_0 \rangle$ which is needed. The series $\langle \mathbf{g}_k, h_0 \rangle, \langle \mathbf{g}_{k+1}, h_0 \rangle, \ldots$ of real numbers converges monotonically towards $\langle g, h_0 \rangle$.

As the sequence $d_1, d_2, \ldots$ may be any enumeration of D, a clever implementation of the above method may determine this sequence 'on the fly' such that d_i is placed where g is expected to be most dynamic.

The simplex method

The set $V_{e(i)}$ consists of all functions $S_{v,e(i)}$ where $v = (v_1, \ldots, v_{i-1}) \in D^{i-1}$. To find $\mathbf{g}_i \in W \cap V_{e(i)}$ such that $\langle \mathbf{g}_i, h_0 \rangle$ is maximal we need to solve the following problem: find $v = (v_1, \ldots, v_{i-1}) \in D^{n-1}$ such that

$$\begin{array}{rcll} -1 \leqslant & v_j & \leqslant 1 & j \in \{1, \ldots, i-1\} \\ v_1 c_{1k} + \cdots + v_{i-1} c_{(i-1)k} & & \leqslant a_j & k \in \{1, \ldots, n\} \\ v_1 c_{10} + \cdots + v_{i-1} c_{(i-1)0} & & & \text{is maximal} \end{array}$$

where

$$c_{ij} = \int_{\mathbf{d}_i}^{\mathbf{d}_{i+1}} h_j(t)\, dt$$

Hence we have $2i-2+n$ linear inequalities and one linear form to optimize, which is exactly what the simplex method can solve.

The above problem may have several solutions. Among these solutions one can find solutions for which at least $i-1$ of the inequalities become equalities (Rockafellar 1970). Hence, one can find solutions for which $|v_j| = 1$ for at least $i-1-n$ different j, and $|v_j| \neq 1$ for at most n different j.

A property of the solutions

We have now outlined a numerical algorithm which finds a sequence $\mathbf{g}_k, \mathbf{g}_{k+1}, \ldots$ where each $\mathbf{g}_i$ is found using the simplex method. The sequence is known to have a convergent subsequence, and the limit of any such sequence is a solution to the optimization problem. Each function $\mathbf{g}_i$ is a step function $S_{v,d}$ where $v = (v_1, \ldots, v_{i-1})$.

If we choose each $\mathbf{g}_i$ such that $|v_j| \neq 1$ holds for at most n values, then any limit g of any convergent subsequence must satisfy $|g(t)| = 1$ for almost all t. To see this, proceed as follows: for any $h \in L_1$ we have

$$\int_{-\infty}^{+\infty} (1 - |g_i(t)|)\, h(t)\, dt \;\rightarrow\; 0 \quad i \rightarrow +\infty$$

and hence

$$\int_{-\infty}^{+\infty} (1 - |g(t)|)\, h(t)\, dt \;=\; 0$$

As this holds for all $h \in L_1$, $1 - |g(t)| = 0$ for almost all t.

Hence, among the solutions to the optimization problem there are functions that bounce back and forth between -1 and 1. This indicates that it is reasonable to approximate solutions to the optimization problem by step functions. This is an important result seen from the point of view of numerical analysis. For other problems it might be more reasonable to approximate by piecewise linear continuous functions, splines, or other countable dense subsets of L_∞. The choice of a countable dense subset of L_∞ affects the pace of convergence and thereby the efficiency of the algorithm.

10.3 Further work

An interesting issue which remains to be studied is the notion of 'approximate computability'. It is clear that for example, addition of real numbers can be approximated arbitrarily well by computers. It is also possible to verify that the Fourier transform $F : L_2 \rightarrow L_2$ can be approximated arbitrarily well if, for example, we choose V defined earlier as a countable dense subset of L_2.

The signum function $f : \mathbb{R} \rightarrow \mathbb{R}$ defined by $f(x) = -1$ for $x < 0$, $f(0) = 0$, and $f(x) = 1$ for $x > 0$ is not approximately computable, for if $y = 0$, then regardless of the accuracy with which the computer knows y, the computer cannot decide whether $f(y) = -1$, $f(y) = 0$, or $f(y) = 1$. (This is a standard example from intuitionistic logic (Heyting 1966).) However, the function $g : \mathbb{R} \setminus \{0\} \rightarrow \mathbb{R}$ defined by $g(x) = -1$ for $x < 0$ and $g(x) = 1$ for $x > 0$ is computable. As we can see, a necessary condition for a function g to be approximately computable is that g is continuous.

Just as with the concept of 'computability', it is not obvious what 'approximate computability' should mean. For the concept of 'computability', a number of suggestions have been made by Markov, Turing, Herbrand-Gödel, and others, and all these suggestions have been proved to be equivalent (Mendelson 1979). It seems reasonable to follow the same approach for 'approximate computability', that is, to state definitions of the concept, to study consequences of the definitions, and to compare various definitions to see if they are equivalent.

One definition of 'approximate computability' could proceed as follows. We first choose a domain S for performing computations. The choice of domain is somewhat arbitrary. The choice and its impacts are discussed later.

Choice of domain

Let $(S, \leqslant)$ be the 'cpo' (complete partial order) given by the domain equation (Schmidt 1986, Scott 1982):

$$S = (\{nil\} + S \times S)_\perp$$

Further, let S_f be the finite elements of S, that is, let S_f be the least set such that $nil \in S_f$, $\perp \in S_f$, and $\forall x, y \in S_f : (x, y) \in S_f$. Let S_m be the set of maximal elements of S, that is, let $S_m = \{x \in S \mid \forall y \in S : x \not< y\}$. If $x, y \in S$ and $x \leqslant y$, then we say that x 'approximates' y.

One property of S is: for each maximal element y there is a chain $x_1 \leqslant x_2 \leqslant \cdots$ of finite elements such that y is the only element of S for which $x_i \leqslant y$, $i \in \{1, 2, \ldots\}$. In other words, maximal elements may be approximated arbitrarily well by finite elements.

The cpo S is interesting from a computer science point of view because computers can, to some extent, compute with elements of S_f. The qualification 'to some extent' covers the fact that computers cannot do just anything with elements of S_f. In particular, a computer cannot do anything reasonable with the bottom element $\perp$ of the cpo, because $\perp$ represents 'total absence of information'.

Representation of topological spaces

Let T be a topological space, let $S' \subseteq S_m$, and let t' be a surjective (or 'onto' or 'epimorphic') function of type $t' : S' \rightarrow T$. For each $x' \in S'$ we say that x' is a 'representation' of $t'(x') \in T$.

For all $x \in S$ define $t(x) = \{t'(x') \mid x' \in S' \wedge x \leqslant x'\}$. We have that t is a function of type $t : S \rightarrow \mathbb{P}(T)$ where $\mathbb{P}(T)$ denotes the powerset of T. For each $x \in S$ we say that x is an 'approximate representation' of each $y \in T$ for which $y \in t(x)$. Hence $x \in S$ is an approximate representation of any element of $t(x)$.

We say that $t : S \rightarrow \mathbb{P}(T)$ is a 'representation' function for T if it can be formed as above and the following holds: for any $y \in T$ and for any open neighbourhood Y of y, there is an $x \in S_f$ such that $y \in t(x) \subseteq Y$. In other words, t is a representation function if it is possible to approximate any element of T arbitrarily well by finite elements of S.

$$\begin{tikzcd}
S \arrow[r, "g"] \arrow[d, "t"'] & S \arrow[d, "u"] \\
\mathbb{P}(T) \arrow[r, "f \text{ applied to set of values}"'] & \mathbb{P}(U)
\end{tikzcd}$$

Fig. 10.2 Representing the function f by g

We say that a topological space T is 'representable' if there exists a representation function for T. As a direct consequence, any representable T is separable. To see this, choose (by the axiom of choice) an element $e_x \in t(x)$ for each $x \in S_f$, and form $E = \{e_x \mid x \in S_f\}$. E is a dense subset of T and, as S_f is countable, E is also at most countable.

It is trivial, but somewhat lengthy, to prove that the following topological spaces are representable: the real numbers with the usual topology; the integers with the point topology; any L_p space, $p \in [1, +\infty)$, with the $\|\bullet\|_{1_p}$-norm topology; the space L_∞ with the weak*-topology. Further, if A and B are representable, then the cartesian product $A \times B$ and the direct sum $A \oplus B$ are representable.

Representation of functions between spaces

Now let T and U be representable topological spaces and let t and u be representation functions for T and U respectively (see Figure 10.2). Let f and g be functions of type $f : T \rightarrow U$ and $g : S \rightarrow S$ respectively. We say that f and g are 'compatible' (w.r.t. t and u) if $f(t(x)) \subseteq u(g(x))$ for all $x \in S$ where $f(t(x))$ is shorthand for $\{f(y) \mid y \in t(x)\}$. Hence, if f and g are compatible, and if $x \in S$ represents $y \in T$, then $g(x)$ represents $f(y)$. Further, we say that g 'represents' f if f and g are compatible and the following holds: for all $y \in T$ and all neighbourhoods Z of $z = f(y) \in U$, there exists a neighbourhood Y of y such that

$$\forall x \in S : (t(x) \subseteq Y \Rightarrow u(g(x)) \subseteq Z)$$

In other words, we can approximate $f(y)$ arbitrarily well by $g(x)$ if only x approximates y sufficiently well.

We say that $f : T \rightarrow U$ is 'approximately computable' (w.r.t. t and u) if there exists a computable $g : S \rightarrow S$ which represents f. Hence, approximate computability is defined in terms of usual computability. Any of the equivalent definitions of usual computability will do. As an example we may state that the lazy Lisp (Friedman and Wise 1976, Henderson and Morris 1976, Henderson 1980) functions of type $S \to S$ are exactly the computable functions.

As a direct consequence, any approximately computable function is continuous. To see this, consider any element $y \in T$ and any neighbourhoods Z of $z = f(y) \in U$. There exists a neighbourhood Y of y such that $\forall x \in S : (t(x) \subseteq Y \Rightarrow u(g(x)) \subseteq Z)$. Combined with $f(t(x)) \subseteq u(g(x))$ this gives $\forall x \in S : (t(x) \subseteq Y \Rightarrow f(t(x)) \subseteq Z)$. For each $x' \in T$ there exists an $x \in S$ such that $t(x) = \{x'\}$. Hence $\forall x' \in T : (x' \in Y \Rightarrow f(x') \in Z)$. The continuity of f follows directly.

It is trivial but somewhat lengthy to prove that the following functions are approximately computable (w.r.t. suitable representation functions): addition, subtraction, and multiplication of real numbers and integers; division by non-zero real numbers and integers; addition of elements of L_p, $p \in [1, +\infty]$; multiplication of elements of L_p, $p \in [1, +\infty]$ by real numbers and integers; Fourier transformation of elements of L_2. As examples, addition of real numbers is an approximately computable function of type $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$, and division is an approximately computable function of type $\mathbb{R} \times (\mathbb{R} \setminus \{0\}) \to \mathbb{R}$.

Further, the following is trivial to prove: if $f_1 : T_1 \to T_2$ and $f_2 : T_2 \to T_3$ are approximately computable (w.r.t. suitable representation functions), then $f_2 \circ f_1$ is also approximately computable. The identity function on any representable topological space is approximately computable. If $f_1 : T \to T_1$ and $f_2 : T \to T_2$ are computable, then $f : T \to T_1 \times T_2$ given by $f(x) = (f_1(x), f_2(x))$ is computable. Further, the projections $f_1 : T \to T_1$ and $f_2 : T \to T_2$ of any approximately computable $f : T \to T_1 \times T_2$ are approximately computable. Similar results hold for direct sums.

Negative results

If T ($\neq \emptyset$) and U are topological spaces and U is of cardinality $2^{\aleph_0}$, then there exist constant functions $k : T \to U$ that are not computable, for there are $2^{\aleph_0}$ constant functions but merely $\aleph_0$ computable functions. Likewise, if U is a representable topological space of cardinality $2^{\aleph_0}$ with a representation function u, then there exists $x \in U$ such that $\{x\} = u(y)$ holds for no computable constant $y \in S$. Hence, representability of U does not ensure computability of each of its elements.

The domain S itself is not representable. An explanation of this phenomenon is that S contains elements that are only 'partially defined' in computer science terms. From a mathematical point of view, the set S is well defined and each element of S is a distinct object. From a computer science view, however, non-maximal elements are elements that are only partially defined. In particular, the bottom element $\perp$ is completely undefined, and any process that does anything with $\perp$ except passing it around, is doomed to loop indefinitely (as a consequence of Turing's halting problem (Mendelson 1979)). Hence, from a computer science point of view, $\perp$ is not a distinct object, for any attempt to distinguish it from any other object by means of a computer makes the computer loop indefinitely. Intuitively, the definition of 'representation function' rules out spaces with partial objects. The definition of representability has been chosen to fit the normal mathematical view that topological spaces consist of well-defined distinct objects.

Consequences of the choice of domain

The choice of the domain S above was somewhat arbitrary. Many other choices will lead to the same class of 'approximately computable functions'. The choice of one particular domain is useful when proving the approximate computability of compositions of approximately computable functions. The domain S above is the simplest that makes trivial the proof of the representability of cartesian products and direct sums.

Some relations to the literature and representation of real numbers

It is interesting to compare the above framework with the programs in (Boehm *et al.* 1986). Boehm *et al.* consider several possible representations of real numbers and provide computer programs for addition, subtraction, multiplication, and division with arbitrary precision for these representations. For simplicity, we merely consider a simplified version of one of the representations.

Expressed in the framework above, the representation of elements of $[-1,1]$ in (Boehm *et al.* 1986) is as follows. Consider the cpo $\underline{S}$ given by

$$\underline{S} = (\{-1,0,1\} \times \underline{S})_\perp$$

The maximal elements of $\underline{S}$ are infinite sequences $(a_1, a_2, \ldots)$ where $a_i \in \{-1,0,1\}$ for $i \in \{1,2,\ldots\}$. Now, for each maximal element $x = (a_1, a_2, \ldots)$ of $\underline{S}$ define

$$t'(x) = \sum_{i=1}^{+\infty} a_i 2^{-i}$$

Then define $t : \underline{S} \to \mathbb{P}([-1,1])$ by $t(x) = \{t'(x') \mid x' \in \underline{S}_m \wedge x \leqslant x'\}$, and let $\phi : S \to \underline{S}$ be any computable surjective function that maps maximal elements to maximal elements. We have that $t \circ \phi$ is a representation function for the topological space $[-1, 1]$ (with the usual topology). (Boehm *et al.* (1986) also state representations of all of $\mathbb{R}$.)

As explained in (Boehm *et al.* 1986), usual binary expansions, that is, elements of $\underline{S}' = (\{0, 1\} \times \underline{S}')_\perp$ cannot be used to represent elements of $[0, 1]$. One problem with $\underline{S}'$ is that the addition $1/3 + 1/6 = 1/2$ requires infinite carry look ahead and hence cannot be performed in finite time. If $\underline{S}'$ is substituted for $\underline{S}$ in the definition of t above, then $t \circ \phi$ does not become a representation function. Furthermore, as explained in (Boehm *et al.* 1986), the use of $\underline{S}'' = (\{-2, -1, 0, 1, 2\} \times \underline{S}'')_\perp$ instead of $\underline{S}$ increases the efficiency of algorithms by reducing the need for carry look ahead.

For further inspiration, Clenshaw and Olver (1980, 1986) provide algorithms for computing certain real functions with arbitrary precision, and Collatz (1966) is an exponent for using functional analysis in numerical analysis.

10.4 Conclusion

- Any set of cardinality greater than $2^{\aleph_0}$ is out of reach of computers.
- Approximate computations are only possible in separable spaces.
- In order to keep the cardinality at $2^{\aleph_0}$ when working with functions $f : \mathbb{R}^n \to \mathbb{R}^m$, identify functions that are equal almost everywhere, and require f to satisfy that $\int_0^{x_1} \cdots \int_0^{x_n} f(t_1, \ldots, t_n)\, dt_1 \cdots dt_n$ is defined and finite for all real $x_1, \ldots, x_n$.
- In order to obtain separability when working in L_∞, use the weak*-topology.
- Use Alaoglu's theorem, that is, make use of the fact that any closed subset of L_∞ which is bounded w.r.t. $\|\bullet\|_{1\infty}$ is compact w.r.t. the weak*-topology.
- In numerical analysis, do not hesitate to represent real functions as data structures rather than as computer programs.

In addition to the above conclusions, the chapter has suggested a definition of 'approximate computability' and investigated some immediate consequences.

Acknowledgements

My thanks are due to Nils Andersen, DIKU, for comments on the manuscript, and to Professor Dr. Dieter Spreen and others at the conference for useful comments and corrections.

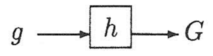

Fig. 10.3 The filter h

Appendix An example from numerical signal analysis

In signal analysis, a 'filter' is a physical device which takes a signal as input and delivers a signal as output. Such filters may be electrical filters that input and output electrical signals or they may be other kinds of filters such as microphones that input a sound signal and output an electrical signal.

The 'linear filters' are of particular interest because of their well understood properties. A linear filter is characterized by an 'impulse response' H, and its output G depends on the input g as follows:

$$G(t) = (g * H)(t) = \int_{-\infty}^{+\infty} g(\tau)H(t-\tau)\,d\tau$$

We define the 'inverse impulse response' h by $h(\tau) = H(-\tau)$. Hence

$$G(t) = (g * H)(t) = \int_{-\infty}^{+\infty} g(\tau)h(\tau - t)d\tau$$

In particular we have $G(0) = \langle g, h\rangle$.

Graphically, we represent the filter as in Figure 10.3. In signal analysis it is customary to assume $g \in L_2$ even though $g \in L_\infty$ is more reasonable for most physical signals. The assumption $g \in L_2$ leads to simpler mathematics. In this Appendix, however, we assume $g \in L_\infty$. Correspondingly, we assume $h \in L_1$.

Now assume that we want to build a filter with inverse impulse response $h \in L_1$ which is going to filter signals $g \in L_\infty$ for which $\|g\|_{1_\infty} \leqslant 1$. However, for economic reasons, we have to use a number of cheap filters with impulse responses $h_1, \dots, h_n \in L_1$ and to combine these into one filter with approximately the impulse response h as in Figure 10.4. In Figure 10.4, F represents a function of type $F : \mathbb{R}^n \to \mathbb{R}$. We have

$$\begin{aligned}\underline{G}(t) &= F(G_1(t), \dots, G_n(t)) \\ G_i(t) &= \int_{-\infty}^{+\infty} g(\tau)h_i(\tau - t)\,d\tau + e_i(t)\end{aligned}$$

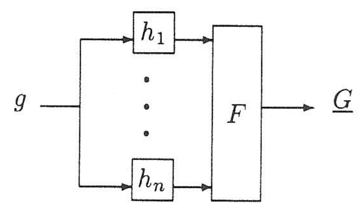

Fig. 10.4 Combined filters $h_1 \ldots h_n$

The functions e_i are unknown error functions which describe the imperfection of the filters $h_1, \ldots, h_n$. We assume $\|e_i\|_{1\infty} \leqslant \epsilon_i, i \in \{1, \ldots, n\}$, where $\epsilon_1, \ldots, \epsilon_n \in \mathbb{R}_+$ are known constants.

For any choice of F we define the 'error bound' E_F as the maximal difference between the G of Figure 10.3 and the $\underline{G}$ of Figure 10.4:

$$\begin{aligned} E_F &= \sup\{\|G - \underline{G}\|_{1\infty} \\ &\qquad \mid g \in L_\infty \wedge \|g\|_{1\infty} \leqslant 1 \wedge \|e_i\|_{1\infty} \leqslant \epsilon_i, i \in \{1, \ldots, n\}\} \end{aligned}$$

Hence

$$\begin{aligned} E_F &= \sup\{|\langle g, h\rangle - F(\langle g, h_1\rangle + d_1, \ldots, \langle g, h_n\rangle + d_n)| \\ &\qquad \mid g \in L_\infty \wedge \|g\|_{1\infty} \leqslant 1 \wedge |d_i| \leqslant \epsilon_i\} \end{aligned}$$

One somewhat surprising result (Golomb and Weinberger 1959) is that among the functions $F : \mathbb{R}^n \to \mathbb{R}$ which minimize E_F, there is at least one which is linear in its arguments. To see this, proceed as follows: define

$$\begin{aligned} C &= \{(\langle g, h\rangle, \langle g, h_1\rangle + d_1, \ldots, \langle g, h_n\rangle + d_n) \\ &\qquad \mid g \in L_\infty \wedge \|g\|_{1\infty} \leqslant 1 \wedge |d_i| \leqslant \epsilon_i\} \end{aligned}$$

Obviously, C is a closed bounded convex subset of $\mathbb{R}^{n+1}$, and C is symmetric around $(0, \ldots, 0)$. Define $A = \sup\{x_0 \in \mathbb{R} \mid (x_0, 0, \ldots, 0) \in C\}$. We have that $(A, 0, \ldots, 0)$ is on the boundary of C. From (Rockafellar 1970) we have that there is at least one tangent to C through $(A, 0, \ldots, 0)$ which is not parallel to $(1, 0, \ldots, 0)$. Let $x_0 = a_1 x_1 + \cdots + a_n x_n + A$ be such a tangent, and let $\underline{F}(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n$. We have

$$\forall (x_0, \ldots, x_n) \in C : |x_0 - \underline{F}(x_1, \ldots, x_n)| \leqslant A$$

and equality is obtained for $(A, 0, \ldots, 0)$ and $(-A, 0, \ldots, 0)$. Hence, $E_{\underline{F}} = A$ for this particular $\underline{F}$. However, as $(A, 0, \ldots, 0)$ and $(-A, 0, \ldots, 0)$ both belong to C, $E_F \geqslant A$ for any F, and so $\underline{F}$ minimizes E_F.

To find A we have to find $\sup\{x_0 \in \mathbb{R} \mid (x_0, 0, \ldots, 0) \in C\}$, that is, to find the maximal value of

$$x_0 = \langle g, h \rangle$$

where

$$\begin{array}{cc} g \in L_\infty & \|g\|_{1_\infty} \leqslant 1 \\ \langle g, h_1 \rangle \leqslant \epsilon_1 & \langle g, -h_1 \rangle \leqslant \epsilon_1 \\ \vdots & \vdots \\ \langle g, h_n \rangle \leqslant \epsilon_n & \langle g, -h_n \rangle \leqslant \epsilon_n \end{array}$$

This is exactly the optimization problem stated in the main text. To find $\underline{F}$, that is, to find $a_1, \ldots, a_n$, one has to perform numerical differentiation of the surface of C at the point $(A, 0, \ldots, 0)$ and hope that C is differentiable. The numerical differentiation requires the optimization problem to be solved n times with small perturbations of $e_1, \ldots, e_n$, and $2n$ times if differentiability has to be verified. We omit the details.

We have now outlined one optimization problem in signal analysis which can be solved by solving the optimization problem in the main text. Many similar optimization problems in signal analysis can be solved the same way.

References

Boehm, H. J., Cartwright, R., Riggle, M., and O'Donell, M. J. (1986) Exact Real Arithmetic: A Case Study in Higher Order Programming. *1986 ACM Symposium on Lisp and Functional Programming.*

Clenshaw, C. W., and Olver, F. W. J. (1980) An unrestricted algorithm for the exponential function. *SIAM Journal on Numerical Analysis.* Vol. 17, pp. 310–331.

Clenshaw, C. W., and Olver, F. W. J. (1986) Unrestricted algorithms for reciprocals and square roots. BIT, Vol. 26, pp. 476–492.

Collatz, L. (1966) *Functional Analysis and Numerical Mathematics.* Academic Press.

Friedman, D. P., and Wise, D. S. (1976) CONS Should Not Evaluate Its Arguments. In: S. Michaelson and R. Milner (Ed.), *Automata, Languages and Programming.* Edinburgh University Press, pp. 257–284.

Golomb, M., and Weinberger, H. F. (1959) Optimal Approximation and Error Bounds. In: R. Langer (Ed.), *On Numerical Approximation*, pp. 117–190. The University of Wisconsin Press.

Grue, K. E. (1985) Optimal Reconstruction of Bandlimited Bounded Signals. *IEEE Transactions on Information Theory*, Vol. IT-31, No. 5, pp. 594-601.

Henderson, P., and Morris, J. H. Jr. (1976) A Lazy Evaluator. *Conference Record of the Third ACM Symposium on Principles of Programming Languages.*

Henderson, P. (1980) *Functional Programming, Application and Implementation.* Prentice-Hall.

Heyting, A. (1966) *Intuitionism, An Introduction.* North-Holland Publishing Company.

Mendelson, E. (1979) *Introduction to Mathematical Logic.* D.Van Nostrand Company.

Rockafellar, R. T. (1970) *Convex Analysis.* Princeton University Press.

Rudin, W. (1973) *Functional Analysis.* McGraw-Hill Book Company.

Schmidt, D. A. (1986) *Denotational Semantics.* Allyn and Bacon Inc.

Scott, D. S. (1982) Domains for denotational semantics, in LNCS 140: *Proc. 9th ICALP*, pp. 577–613, Springer.