Math. Struct. in Comp. Science (2012), *vol.* 22, *pp.* 729–751. © Cambridge University Press 2012
doi:10.1017/S0960129511000508

Renormalisation and computation II: time cut-off and the Halting Problem

YURI I. MANIN

Max Planck Institut für Mathematik, Bonn, Germany
and
Northwestern University, Evanston, U.S.A.

Received 12 May 2010; revised 21 July 2010

This is the second instalment in the project initiated in Manin (2012). In the first Part, we argued that both the philosophy and technique of perturbative renormalisation in quantum field theory could be meaningfully transplanted to the theory of computation, and sketched several contexts supporting this view.

In this second part, we address some of the issues raised in Manin (2012) and develop them further in three contexts: a categorification of the algorithmic computations; time cut-off and anytime algorithms; and, finally, a Hopf algebra renormalisation of the Halting Problem.

1. Introduction

1.1. *Regularisation and anytime algorithms*

It is well known that the classical theory of computability includes *non*-computability phenomena as an integral part. In particular, an attempt to compute the value of a partially recursive function at a point where it is not defined might stall the computation forever, but we will never know whether this is so, or we have just not waited long enough ('the Halting Problem is undecidable').

The applied theory of computation deals with algorithms processing a finite amount of data into finite outputs. Nevertheless, even in such theoretically safe situations, time and/or memory requirements may make the implementation of a sound algorithm unfeasible.

The celebrated theory of polynomial time computations and the discovery of the P/NP problem served as a neutral zone meeting point between theoretical possibility and practical feasibility, and revealed beautiful new mathematical structures.

However, applied computer scientists consider other possible ways of turning unfeasible computations into feasible ones, known as 'anytime algorithms'. Basically, an 'anytime algorithm' allows the computation to stop in a feasible time, and it then supplies the result of this mutilated procedure with a measure of its quality: see Grass and Zilberstein (1995) and, for a nice short introduction, Grass (1996).

In Section 3 of the current paper, which is the second instalment in the project initiated in Manin (2012), we suggest that 'anytime algorithms' can be treated theoretically as a version of one of the regularisation schemes used in Quantum Field Theory, namely, *time cut-off* (for a more detailed description of the whole project, see the Introduction in Manin (2012)). More precisely, we analyse from this viewpoint the results of the stimulating paper Calude and Stay (2008).

One of the themes of the current paper, which the analogies with renormalisation and experience with anytime algorithms bring to the foreground in computation theory, is the stress on the structure of programs determined by the operation of the *'composition of programs'* and by *explicit parallelism*, which played a key role in our treatment of perturbative renormalisation as a model for regularising computations in Manin (2012).

Notice that many standard descriptions of programming methods are not stable with respect to composition, and have no natural means for expressing parallelism.

For example, composition $T_2 \circ T_1$ of two Turing machines, informally defined as a computation in which the output of T_1 becomes the oracular input ('program') for T_2, is not directly described as a new Turing machine T_3.

Language-like constructions like the lambda-calculus are inherently linear/sequential, and are thus not well suited for expressing options of parallelism.

The 'Flowcharts' imagery we propagandised for in Manin (2012) serves these goals much better. In Section 2 of the current paper, we will show that the same ideas admit a succinct categorical expression, and suggest that the flowchart constructions of Manin (2012) can be interpreted as a constructive existence theorem, to produce what we call 'an enriched programming method with unrestricted parallelism' (*cf.* Definition 2.5). This seems to be very much in the spirit of Baez and Stay (2010).

Finally, in Section 4 we use some ideas from quantum computation to give a Hopf renormalisation scheme for the Halting Problem.

1.2. *Computability as a mathematical structure and its interaction with other mathematical structures*

Most of the constructions considered in this paper refer to (un)feasible algorithms with infinite domains/ranges. When devising natural quantitative characteristics and regularisation schemes for them, we should bear in mind that they can be roughly subdivided into two large blocks.

Block A. This block consists of the inherent problems we enconter when referring to an infinite constructive world X, which depend *only on the class of 'admissible' recursively equivalent numberings of* X, and which are the same for all infinite X. From the computational viewpoint, any such X can be identified with $\mathbf{N}$ (natural numbers) or $\mathbf{Z}^+$ (non-zero natural numbers).

A typical example of such an X is some set of finite Bourbaki structures, such as words in a finite alphabet, or finite groups, or graphs, or their descriptions. In this context, the Bourbaki description is primarily used to define the class of admissible bijections (numberings) $\mathbf{Z}^+ \to X$ in question: they must be informally computable together with their inversions. One aspect of Church's thesis is the statement that recursive functions will provide an adequate notion of the algorithmic processing of elements of X whenever we can imagine an informal algorithm producing the numbering.

Once it is decided that the role of the respective Bourbaki structure is finished as soon as the class of numberings is determined, one can make explicit various secondary structures on X that can be defined exclusively in terms of admissible numberings.

One such structure is the algebra of enumerable subsets of X: the definition domains $D(f)$ of partial recursive functions. This family is stable with respect to finite intersections and enumerable unions. If we consider these sets modulo finite ones, we can prove interesting results about simple sets, maximal sets, and so on. For example, maximal $D(f)$ display a striking similarity to the holomorphy domains V in the theory of complex analytic functions of $\geqslant 2$ variables: in both cases, there are functions defined on $D(f)$ or V, respectively, that cannot be extended to a larger domain.

The proviso 'modulo finite subsets' can be formalised very naturally by changing the Constructive Universe $\mathscr{C}$ described in Section 2: simply consider the largest quotient of $\mathscr{C}$ that makes invertible those morphisms (computable maps) $X \to Y$ that become computably invertible after the restriction to some subsets of X and Y with finite complements.

Another such structure is the class of Kolmogorov orderings: total orders on X defined by increasing the Kolmogorov complexity with respect to various optimal enumerations.

Such orderings are not computable, but with respect to them, all recursive functions, including admissible numberings, become functions of linearly bounded growth. We will discuss this feature from the renormalisation viewpoint in Section 4.

Block B. This group consists of problems related to the interaction of computability with other Bourbaki sructures on X. An elementary example is the embedding $X = \mathbf{Q} \subset \mathbf{R}$ used in the theory of computable rational approximations to real numbers.

In this block, the greatest discovery was the Diophantine representability of enumerable subsets of $\mathbf{N}$ due to Davis, Robinson, Putnam and Matiyasevich.

A very interesting and unexpected example of such an interaction was elaborated in the work of Nabutovsky and Weinberger (*cf.* Nabutovsky and Weinberger (2003)), who showed that the computational complexity can be used to display a highly irregular landscape of minima of natural differential/geometric functionals on the space of Riemannian metrics modulo diffeomorphisms.

Since path integration over such a space is one of the key tools of quantum gravity, this can become an important next meeting space between renormalisation and computation.

2. Enriched programming methods with typing and parallelism: a categorical approach

2.1. *Preliminary remarks*

We use $\mathbf{N}$ to denote the set of natural numbers and $\mathbf{Z}^{+}$ to denote the positive natural numbers. $\mathbf{N}$ is usually taken as the basic set on which recursive functions are defined, though we used $\mathbf{Z}^+$ in Manin (2010), having found it more convenient in a Diophantine context.

As a Bourbaki structure, both $\mathbf{Z}^+$ and $\mathbf{N}$ are here (isomorphic) totally ordered sets, with minimal elements 1 and 0, respectively, and successor function suc(x): 'the smallest y such that $y > x$', or $suc(x) = x + 1$ in the standard notation. In a sense, this is the minimal structure needed to define the set of partial recursive functions that are partial maps $\mathbf{Z}^+ \to \mathbf{Z}^+$ or, more generally, $(\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$. The remaining components of the definition (see, for example, Manin (2010, V.2.1–2.4)) are just the standard category-theoretic constructions in a fixed monoidal category of sets $(ParSets, \times)$ with partial maps as morphisms and cartesian product: *cf.* Manin (2012, Section 3.7).

However, this total order structure *is not* invariant with respect to the structure we will define below in the sense that it is not preserved under the automorphisms of this structure. For this reason, we will not use one of the standard categorifications of computation theory in which $\mathbf{N}$ is replaced by a 'natural numbers object' $\mathcal{N}$ (of an abstract category), endowed with a morphism $suc_{\mathcal{N}} : \mathcal{N} \to \mathcal{N}$: this categorification puts undue stress on the role of this total order and the related iteration $suc_{\mathcal{N}} \circ \cdots \circ suc_{\mathcal{N}}$.

Instead, we adopt the version advocated in Manin (1999) of a subcategory $\mathcal{C}$ of $ParSets$ called the *Constructive Universe*. The relevant formal definitions are collected together in Sections 2.2–2.8, with some informal comments left until Section 2.10.

2.2. *Objects*

Objects of $\mathcal{C}$ will be called constructive worlds.

Definition 2.1. A constructive world X is either a finite set, or an infinite set endowed with a non-empty set $Num(X)$ of bijections $\nu : \mathbf{Z}^+ \to X$, called admissible numberings, satisfying the following conditions:

(i) If $\nu_1, \nu_2 \in Num(X)$, then $\nu_2^{-1} \circ \nu_1$ is a total recursive bijection.
(ii) If $\nu \in Num(X)$ and $f : \mathbf{Z}^+ \to \mathbf{Z}^+$ is a total recursive bijection, then $\nu \circ f \in Num(X)$.

Elements of the constructive world X are called constructive objects of the type X.

2.3. *Morphisms*

Let X, Y be two constructive worlds. Morphisms $X \to Y$ are induced by partial recursive maps on their structure numberings. More precisely, we have the following definition.

Definition 2.2. A morphism $X \to Y$ is a partial map $f : D(f) \to Y$, where $D(f) \subset X$ is a subset (possibly empty) satisfying the following conditions:

(i) If X is infinite and Y is finite, then for one (equivalently, any) admissible numbering $\nu : \mathbf{Z}^+ \to X$ and any $y \in Y$, the set $\nu^{-1}(f^{-1}(y))$ is recursive enumerable.
(ii) If X and Y are both infinite, then for one pair (equivalently, any pair) of admissible numberings $\nu_X : \mathbf{Z}^+ \to X$, $\nu_Y : \mathbf{Z}^+ \to Y$, the partial map $\nu_Y^{-1} \circ f \circ \nu_X : \mathbf{Z}^+ \to \mathbf{Z}^+$ is a partial recursive function.
(iii) If X is finite and Y is infinite, any partial map is a morphism.

With the standard composition of partial maps, constructive worlds form a category $\mathscr{C}$, which we will call the *Constructive Universe*. The set of morphisms $X \to Y$ will be denoted $\mathscr{C}(X, Y)$. Its subcategory consisting of infinite constructive worlds is equivalent to a very simple category consisting of one object, say $\mathbf{Z}^+$, and partial recursive maps as morphisms. Such categories are called *isotypical* ones in Heller (1990).

However, it is important to consider $\mathscr{C}$ as a (bi)monoidal category, with two symmetric monoidal structures $\times$ (direct product) and $\coprod$ (coproduct, or disjoint union (Heller 1990)), connected by the standard coherence diagrams.

These monoidal structures are induced by those in a small category of (unstructured) sets in which our constructive worlds lie, so it suffices to specify some privileged numberings of disjoint sums and direct products. Moreover, it suffices to consider numberings that are bijective maps $\mathbf{Z}^+ \to \coprod_{i=1}^{m} \mathbf{Z}^+$ and $\mathbf{Z}^+ \to (\mathbf{Z}^+)^m$.

For $\coprod_{i=1}^{m} \mathbf{Z}^+$, we simply assign to $m(k-1)+i$ the number k in the ith summand.

The cartesian product is more interesting because there are several numberings that become privileged in the context of Kolmogorov complexity.

We will generally assume that $\mathscr{C}$ is closed with respect to the monoidal structures $\times$ and $\coprod$.

2.4. *Constructive descriptions of morphisms*

We fix two constructive worlds X, Y. Since the set of recursive maps $\mathscr{C}(X, Y)$ is not a constructive world, we may try to replace it by descriptions.

Definition 2.3. A constructive world of descriptions is a pair $(P(X, Y), F)$, where $P(X, Y)$ is an object of $\mathscr{C}$, and $F : P(X, Y) \times X \to Y$ is a morphism in $\mathscr{C}$, satisfying the following condition:

— Letting $p \in P(X, Y)$, we use f_p to denote the partial map $x \mapsto F(p, x) \in Y, x \in X$. Then each f_p is a morphism $X \to Y$ in $\mathscr{C}$.

In other words, descriptions produce a set theoretic map $P(X, Y) \to \mathscr{C}(X, Y)$ constructively depending on (p, x).

2.4.1. *Translations.* Let $(P(X, Y), F)$ and $(Q(X, Y), G)$ be two constructive worlds of descriptions. A *translation* (or *compilation*) *method*

$$trans_{P,Q} : P(X, Y) \to Q(X, Y)$$

is an everywhere defined morphism in $\mathscr{C}$ such that for all $p \in P(X, Y)$, we have $trans(p) \in Q(X, Y)$ defines the same morphism $f_p : X \to Y$. In other words,

$$G \circ (trans_{P,Q} \times id_X) = F.$$

2.4.2. *Universal descriptions.* The world $(U(X, Y), W)$ is the world of *universal descriptions,* if for any other world of descriptions $(P(X, Y), F)$, there exists a translation morphism

$$trans_{P,U} : P(X, Y) \to U(X, Y).$$

In particular, it can compute any (semi)computable function, in the sense that the family of maps $f_u : X \to Y$, $u \in U(X, Y)$ contains all morphisms in $\mathscr{C}$.

2.4.3. *Complements.* Amongst the various constructive worlds of descriptions $P(X, Y)$ there exist some with better properties than the general definition would suggest. The terminology for them is rather unstable. We will sometimes say (see Section 3.5) that $P(X, Y)$ is *the base of a family* $f_p : X \to Y$, $p \in P(X, Y)$ of partial functions.

We will review below some relevant definitions and existence theorems. We consider four cases separately:

(i) If X and Y are infinite, then we may assume without loss of generality that $X = Y = \mathbf{Z}^+$.

Rogers (1958) calls such a world of descriptions U *semi-effective* if it computes all morphisms (partial recursive functions), and he says U is *fully effective* if it is universal in the sense of Section 2.4.2.

An easy construction in Rogers (1958) (see the example following Definition 3) shows that there are semi-effective descriptions that are not fully effective. A universal description world $U(X, Y) = \mathbf{Z}^+$ (or rather $\mathbf{N}$) is also called a *Gödel numbering* in Rogers (1958).

The main Theorem of Rogers (1958) implies in our language that *for any two universal description worlds (for infinite X, Y), there exist two mutually inverse translation isomorphisms between them: total recursive bijections that are compatible with functions that these descriptions compute.*

Schnorr (1974) considerably strengthened this result by saying that a universal description world $(U = \mathbf{Z}^+, W)$ is *an optimal Gödel numbering* if for any other world of descriptions $(P = \mathbf{Z}^+, F)$, there exists a translation morphism $t : P \to U$ that is a linearly bounded function $\mathbf{Z}^+ \to \mathbf{Z}^+$. We will simply say that such a description world is *optimal.*

Schnorr then proved that *optimal descriptions exist, and for any two optimal description worlds (with infinite X, Y), there exist two mutually inverse linearly bounded translation isomorphisms between them.*

Similar results hold in the case when only one of the worlds X, Y is infinite. The general situation can be reduced to the case when the relevant finite world is a one-element set – for simplicity, we will only consider this case.

(ii) The case $X = \{*\}$ is truly exceptional in the sense that $\mathscr{C}(\{*\}, Y)$, that is, 'Y-valued recursive functions of zero variables', can be canonically identified with the set Y, and *it is* thus a constructive world. Nevertheless, the ideas and main results of Rogers and Schnorr are also applicable in this case, and lead to a strengthening of the notion of a Kolmogorov optimal enumeration of constructive objects of a given type.

(iii) In the case $Y = \{*\}$, we can naturally identify $\mathscr{C}(X, \{*\})$ with the set of all enumerable subsets of X, domains of partial recursive functions with one value. Gödel and optimal numberings of enumerable subsets can also be easily defined, and again the Rogers and Schnorr theorems are valid for them.

(iv) Finally, when both X and Y are finite, the useful structurings of descriptions are those of Boolean polynomials, circuits and so on, and many complexity problems are centered around polynomial time computations – see Manin (1999) for an introduction that is close in style to the current paper.

2.5. *Enrichments of $\mathscr{C}$ over itself and programming methods*

There is a well-known general notion of a category C enriched over a monoidal category $(M, \otimes, I)$, where I is an identity object.

We will consider enrichments of $\mathscr{C}$ over $(\mathscr{C}, \times, I)$ where I is a fixed one-element constructive world. Products of empty families, such as $\mathbf{N}^a$ for $a = 0$, are interpreted as I.

According to the general pattern, such an enrichment must consist of the following data:

(a) For each pair of constructive worlds X, Y, an 'object of morphisms' $P(X, Y) \in \mathrm{Ob}\mathscr{C}$.
(b) For each triple of constructive worlds X, Y, Z, a 'composition morphism' in $\mathscr{C}$:

$$\circ : P(Y, Z) \times P(X, Y) \to P(X, Z) \tag{1}$$

(c) For each object X of $\mathscr{C}$, an identity morphism

$$\mathrm{id}_X : I \to P(X, X). \tag{2}$$

The standard axioms for morphisms in a category translate into commutativity requirements for three classes of diagrams in $\mathscr{C}$ expressing properties of associativity of enriched composition, and left and right identities.

Definition 2.4. An enrichment of $\mathscr{C}$ over $(\mathscr{C}, \times, I)$ as above is said to be an enriched programming method, denoted by $\mathscr{C}_P$, if:

(A) For each pair of constructive worlds X, Y, we are given a morphism

$$F_{X,Y} : P(X, Y) \times X \to Y$$

in $\mathscr{C}$ such that $P(X, Y)$ becomes a constructive world of descriptions in the sense of Definition 2.3. Thus, we have a family of set-theoretic maps

$$\Phi_{X,Y} : P(X, Y) \to \mathscr{C}(X, Y) : \quad p \mapsto f_p. \tag{3}$$

We will say that p is a description, or a program, computing f_p.

(B) The following axioms are satisfied:

(i) Morphisms (1) and (2) must be everywhere defined (total recursive) maps. (For (2), this simply means that they are non-empty maps.)

(ii) Compositions (1) must be compatible with the compositions of morphisms in $\mathscr{C}$:

$$f_{p \circ q} = f_p \circ f_q.$$

(iii) Element $\mathrm{id}_X(I) \in P(X, X)$ must be a description of the 'copying' program mapping x to x.
(Note that this should not be confused with the 'cloning' program computing the diagonal map $X \to X \times X, x \mapsto (x, x)$.)

Informally speaking, we have a functor

$$\Phi_P : \mathscr{C}_P \to \mathscr{C} \tag{4}$$

that is identical on objects and mapping a program to the function that this program computes.

The most important feature of this formalism is its explicit and systematic inclusion of the composition of programs into our formalism: this is a key requirement for all Hopf algebra renormalisation schemes.

Finally, we should mention that we use the word *program* as a synonym for *'description of a method to compute a given function'*, where the input to such a program is a specific value of the argument of this function, and the output is the value of the function. This praxis should not be confused with the one used in the theory of Turing machines, where programs are often understood as our inputs: the initial binary string on the tape.

2.6. *A (uni)versal enrichment*

An enrichment $\mathscr{C}_U$ as above is said to be *(uni)versal* if programs from U compute *all* partial recursive maps, and, moreover, if for each $\mathscr{C}_P$, there is a functor between enriched categories

$$\Psi : \mathscr{C}_P \to \mathscr{C}_U \tag{5}$$

that is identical on objects, with total recursive maps

$$\Psi_{X,Y} : P(X,Y) \to U(X,Y) \tag{6}$$

such that $\Psi_{X,Y}(p)$ for each p computes the same function as p. In other words, we have

$$\Phi_U \circ \Psi = \Phi_P.$$

Intuitively, we want the following properties of U as a programming method, which are somewhat stronger than those in Section 2.4.

(a) It can compute any (semi)computable function.

(b) For each other programming method P, there must exist a computable (on the world of P-programs) *translation* of P-programs into U-programs that compute the same functions.

(c) The translation must be compatible with the composition of programs and copying/identity programs (functorality of Ψ).

2.7. *Coproducts and typing*

Coproducts (disjoint sums) admit the most straightforward interpretation in the contexts where computer scientists speak about *typing*. In the simplest situation, a program $p \in P(X \coprod Y, Z)$ accepts inputs of either type X or Y, and produces outputs of type Z.

As far as I can judge, iterated application of similar interpretations, can be used in all contexts where the notion of typing is essential.

2.8. *Products and parallelism*

Let $X_i, Y_i, i = 1, \ldots, n$ be constructive worlds. Then the map

$$\pi : \mathscr{C}(X_1, Y_1) \times \mathscr{C}(X_2, Y_2) \times \cdots \times \mathscr{C}(X_n, Y_n) \to \mathscr{C}(X_1 \times X_2 \times \cdots \times X_n, Y_1 \times Y_2 \times \cdots \times Y_n),$$

$$\pi(f_1, \ldots, f_n)(x_1, \ldots, x_n) := (f_1(x_1), \ldots, f_n(x_n)) \tag{7}$$

defines families of computations with independent inputs/outputs that can be implemented in parallel. This can be generalised to programming methods as follows.

Definition 2.5. Let P be an enriched programming method. We will say that P admits unrestricted parallelism if the following additional structure is given.

Let $\{X_i\}$, $\{Y_i\}$, $i = 1, \ldots, n$, be any finite family σ of constructive worlds. We must be given maps

$$\pi_\sigma : \prod_{i=1}^{n} P(X_i, Y_i) \to P\left(\prod_{i=1}^{n} X_i, \prod_{i=1}^{n} Y_i\right), \tag{8}$$

that are lifts of (7). These maps must be equivariant with respect to the natural action of the symmetric group S_n permuting subscripts on both sides.

2.9. *Basic example: flowcharts*

The flowcharts defined in Manin (2012) form a convenient context for constructing enriched progamming methods with unrestricted parallelism. One example of such a method is the world P, which computes primitive recursive functions, and is described in Manin (2012, Definition 2.11). This definition uses the ideas of Yanofsky (2006).

Additional work remains to be done to produce a manageable construction of an universal enrichment with unrestricted parallelism.

2.10. *Comments: constructive worlds and admissible numberings*

Technically speaking, conditions (i) and (ii) of Definition 2.1 together mean that $Num(X)$ forms a principal homogeneous space over the group of total recursive permutations of $\mathbf{Z}^+$, which we may denote by $S_{\mathbf{Z}^+,rec}$, as an infinite analogue of þnite symmetric groups. Hence, the whole $Num(X)$ can be reconstructed from any single numbering in this set. There are usually some 'simplest', or 'privileged', numberings with which one mostly works, such as the numbering of binary words $w \in \{0,1\}^{\mathbf{N}}$ used in Calude and Stay (2008): $bin^{-1} : w \mapsto \overline{1w}$, where the line over a binary word means that it should be treated as the natural number given by its binary digits.

The idea of privileged numberings is essential, especially when we are dealing with polynomial time or, more generally, 'feasible' computations. In order to accommodate this idea, we can strengthen Definition 2.1 as follows. Consider the smaller 'symmetric group' $S_{\mathbf{Z}^+,pol}$ of total recursive permutations that are polynomial time computable together with their inverses, and define the structure of the *polynomial time constructive world* X by a set $Numpol(X)$ forming a principal homogeneous space over the group $S_{\mathbf{Z}^+,pol}$.

An (often implicit) part of the contemporary philosophy around Turing's Thesis consists of postulating that whenever we can informally speak about algorithms and (semi-) computable maps between two constructive worlds, we can always produce in the context of this discussion admissible numberings that are informally algorithmic and transform informal (semi)computable maps into (partial) recursive functions.

In any case, starting with such a class of numberings of X, we should stress that we study notions that are either invariant, or behave in a controlled way, under the action of the group $S_{X,rec}$.

A few further remarks are in order here.

Sometimes, a constructive world X itself is an unstructured set, in the sense that the only relevant structure on it is given by its set of admissible numberings. The typical example is a world A that in further constructions may serve as an alphabet. In this case, a numbering defining the whole $Num(A)$ is usually introduced *ad hoc*. But in most applications, X itself consists of certain sets (often finite and/or considered only up to isomorphism, or even organised into a category) endowed with a certain fixed Bourbaki structure, such as:

(a) finite words in an alphabet A;
(b) finite graphs up to an isomorphism;
(c) finite groups.

In such cases, the privileged numberings ('encodings') generating the whole $Num(X)$ are supposed to interact with this structure in such a way that the number of a constructive object can be 'algorithmically calculated' when we know this object as an instance of this structure, and, conversely, this instance must be algorithmically reconstructible from the number. It is only very rarely that such encodings can give good translations of the basic relations, composition laws and so on, involved in the definition of the structure. This is one reason to formulate models of computation directly in terms of this structure: *cf.* Church's lambda-calculus, Kolmogorov–Uspenski's graphs and Gács–Levin causal nets (see Gács and Levin (1982)).

On the other hand, such a simple task as choosing a privileged numbering of $\mathbf{Z}^+ \times \mathbf{Z}^+$ (upon which one of the monoidal structures on $\mathcal{C}$ is based) can lead to quite interesting constructions when this choice is related, for example, to complexity estimates – see the discussion in Sections 3.6–3.8 using L. Levin's norms for the definitions of such numberings.

Notice that the instances of constructive worlds of causal nets studied in Gács and Levin (1982) are themselves *categories*, and the study of the interaction of computability with symmetries of the respective constructive objects reveals interesting new phenomena.

We want to argue that our categorical framework suggests other possibilities for avoiding too close attention to the elementary steps of computation. In particular, the 'categorical Church Thesis' allows the following wonderfully succinct expression:

— *The category $\mathcal{C}$ is defined uniquely up to equivalence.*

An important complication and variation of the theme of admissible numberings arises when a structure S that we want to treat 'constructively' is then imposed on eventually inþnite sets. In such cases the relevant constructive objects are often not the structures themselves, but their finite *descriptions*: a group might be given by generators and relations, an affine scheme over $\mathbf{Z}$ by its equations, and so on.

The usual complication with descriptions is that many descriptions can produce one and the same (or canonically isomorphic) Bourbaki structure, and the relevant equivalence relation on the set of descriptions can be *undecidable,* or even *not recursively enumerable*. This is precisely the case of the structure constituted by recursive functions themselves, which is our main motivation for introducing enrichments as in Definitions 2.4 and 2.5.

3. Cut-off regularisation and anytime algorithms

3.1. *Cut-off regularisation*

In Quantum Field Theory, cut-off regularisation schemes have the following typical structure. The relevant Feynman integrals, say, in momentum space, may diverge when momentum becomes large or small. In this case, the formal integral I in question is replaced by the finite integral I_P taken over momenta $p \leqslant P := p_{cutoff}$ or $p \geqslant P := p_{cutoff}$, respectively. The behaviour of I_P as $P \to \infty$ or $P \to 0$, respectively, is then studied, and physical information is extracted from the behaviour of the polar part, or regular part, of I_P.

In computer science based on Turing machines and/or recursive functions, the natural 'divergence' occurs in space–time: a computation uses discrete memory (space) and runtime. A typical example of such a divergence is the infinite runtime of a Turing machine computing a partial recursive function f at an input (program) x that is outside the definition domain of f.

Application-oriented computer scientists, of course, recognise the practical necessity of time cut-offs, accompanied by sober estimates of the quality of the outputs. Systematic work on this problem resulted in the notion of 'Anytime Algorithms', *cf.* Grass and Zilberstein (1995). The usefulness of composition and the use of parallelism was stressed in Russell and Zilberstein (1991).

The stimulating paper Calude and Stay (2008) addressed the problem of cut-off of runtime theoretically, and gave meaningful quantitative characteristisations of such a cut-off. More precisely, let f be a partial recursive function ('a morphism of constructive worlds $X \to Y$') as above, and F be its description – a program calculating it. The *computation time (or runtime)* is another partial recursive function, which has the same domain $D(t(F)) = D(f) \subset X$ and target $\mathbf{Z}^+$, and whose precise definition depends on the details of the implied choice of our programming method.

For a Turing machine F, the number of steps required to halt and print $f(x)$ on the tape, where $x \in D(f)$, is $t(F)(x)$. One can similarly define another partial recursive function, 'memory volume' $m(F) : X \to \mathbf{Z}^+$ such that for $x \in D(f)$, the minimal length of tape required to compute $f(x)$ is $m(F)(x)$. Here W is the constructive world of binary words $\{0,1\}^{\mathbf{Z}^+}$. Yet another partial recursive function, $s(F)$, is the sum total of lengths of filled parts of the tape over all steps of computation. Notice that Soare's *settling function* (Soare 2004, Definition 8.2), which is essentially $max\,\{t(F)(y) \,|\, y < x, y \in D(f)\}$, is generally not partial recursive, but some of the inequalities stated below, such as (11), are valid for it too.

One can define natural analogues of functions $t(F)$, $m(F)$ and $s(F)$ for rather general conventional programming methods F, as discussed in Manin (1999) and Chapter IX of the new edition of Manin (2010).

Returning to Calude and Stay (2008), we will first of all show that some of the basic results of that paper related to cut-offs admit a straightforward reformulation in such a way that they become applicable and true for *any* partial recursive function, including, of course, $t(F)$, $m(F)$ and $s(F)$.

This naturally raises the question of what is so specific about $t(F)$, $m(F)$ and $s(F)$. We will treat this question in Section 3.5 in the context of the categorification developed in Section 2, and will show that this provides some meaningful insights into these measures for computation processes.

3.2. *Complexity*

We will first recall the definition and properties of the Kolmogorov ('exponential' or 'program') complexity $C_u : \mathbf{Z}^+ \to \mathbf{Z}^+$: *cf.* Manin (2010, VI.9). Calude and Stay (2008) called it the *the natural complexity* and denoted it by ∇_U or simply ∇.

This complexity measure is defined with respect to a partial recursive function $u : \mathbf{Z}^+ \to \mathbf{Z}^+$, which is surjective:

$$C_u(x) := \min \{y \mid u(y) = x\}.$$

This function u is an arbitrary element of the set of *Kolmogorov, or Gödel, optimal* functions, representatives of which can be effectively constructed: *cf.* Rogers (1958), Schnorr (1974) and Manin (2010). Optimality implies that for any other partial recursive $v : \mathbf{Z}^+ \to \mathbf{Z}^+$, there exists a constant $c_{u,v} > 0$ such that for all x, $C_u(x) \leqslant c_{u,v} C_v(x)$. (The right-hand side is interpreted as ∞ if x is not in the range of v).

It follows that another choice of optimal function replaces C_u by a function $C_{u'} = 2^{O(1)} C_u$. We will say that two such functions belong to the same *bounded equivalence* class.

Moreover, as we discussed earlier, the definition of the complexity of integers can be extended to a definition of the complexity of partial recursive functions of any fixed number of variables m, as in Manin (2010, VI.9.1). This requires a choice of Kolmogorov optimal recursive function of $m+1$ variables. We then have the following simple result (where we omit the subscripts at C specifying the choices of optimal families, and use c with subscripts to denote various constants, which also depend on these choices).

Proposition 3.1. For any partial recursive function $f : \mathbf{Z}^+ \to \mathbf{Z}^+$ and $x \in D(f)$, we have

$$C(f(x)) \leqslant c_f C(x) \leqslant c'_f x. \tag{9}$$

If f and $x \in D(f)$ are allowed to vary, we have

$$C(f(x)) \leqslant c\, C(f) C(x) \log\left(C(f) C(x)\right). \tag{10}$$

In particular, if f is a total recursive permutation, then complexities of x and $f(x)$ are bounded equivalent. It follows that we can define the complexity function, up to bounded equivalence, $C : X \to \mathbf{Z}^+$ for any infinite constructive world X by choosing an admissible numbering $v : \mathbf{Z}^+ \to X$ and putting $C(x) := C_u(v^{-1}(x))$ for some optimal u.

3.3. *Runtimes according to Calude and Stay (2008)*

Proposition 3.1 is a special case of Manin (2010, VI.9, Proposition 9.6). In turn, it implies as special cases inequality (2) and Theorem 4 of Calude and Stay (2008).

It is easy to see this by simply comparing the terminology and notation.

Calude and Stay (2008) deals with the complexity C (their $\nabla = \nabla_U$) of binary words, which reduces to the complexity of integers via the admissible numbering denoted by *bin* in Calude and Stay (2008). This is defined using a 'universal Turing machine' U, which in our language is a programming method computing one of the Kolmogorov optimal functions u. Consider the partial recursive function $x \mapsto t(U)(x)$, which is the runtime of U at the argument $x \in \mathbf{Z}^+$. Inequality (2) of Calude and Stay (2008) can be rewritten in our notation as

$$C(t(U)(x)) \leqslant cx, \tag{11}$$

which is our (9) for $f = t(U)$. The same inequality is valid for $m(U)$ and $s(U)$, but also for $t(F)$, $m(F)$ and $s(F)$ for any F, and even for Soare's settling functions: see Section 3.1.

3.3.1. *Growth of recursive functions and algorithmic randomness.*

The central argument of Calude and Stay (2008) is based on two statements:

(a) The runtime of the Kolmogorov optimal program at a point x of its definition domain is either $\leqslant cx^2$, or is not 'algorithmically random' (Calude and Stay 2008, Theorem 5).

(b) 'Algorithmically random' integers have density zero for a class of computable probability distributions.

The last statement justifies the time cut-off prescription, which is the main result of Calude and Stay (2008):

— *if the computation on the input x did not halt after cx^2 Turing steps, stop it, decide that the function is not determined at x, and proceed to $x + 1$.*

Proposition 3.2 will generalise the statement (a) somewhat.

3.4. *Randomness and growth*

Consider a pair of functions $\varphi, \psi : \mathbf{R}_{>0} \to \mathbf{R}_{>0}$ satisfying the following conditions:

(a) $\varphi(x)$ and $\frac{x}{\varphi(x)}$ are strictly increasing, starting with a certain x_0, and tend to infinity as $x \to \infty$.

(b) $\psi(x)$ and $\frac{\psi(x)}{x\varphi(\psi(x))}$ are increasing and tend to infinity as $x \to \infty$.

The simplest examples are $\varphi(x) = \log(x + 2)$ and $\psi(x) = (x + 1)^{1+\varepsilon}$, with $\varepsilon > 0$.

In our context, φ will play the role of a 'randomness scale'. We will say $x \in \mathbf{Z}^+$ is *algorithmically φ-random* if $C(x) > x/\varphi(x)$. The second function ψ will then play the role of the associated growth scale.

Proposition 3.2. Let f be a partial recursive function. Then for all sufficiently large x exactly one of the following alternatives holds:

(i) $x \in D(f)$ and $f(x) \leqslant \psi(x)$.
(ii) $x \notin D(f)$.
(iii) $x \in D(f)$, and $f(x)$ is not algorithmically φ-random.

Proof. We only need to check that if $x \in D(f)$ and $f(x) > \psi(x)$, then $f(x)$ is not algorithmically φ-random, that is

$$C(f(x)) \leqslant \frac{f(x)}{\varphi(f(x))}. \tag{12}$$

In fact, in view of (9),

$$C(f(x)) \leqslant cx \tag{13}$$

for some constant c (depending on u and f). Furthermore, for sufficiently large x, in view of (b) at the beginning of this section, we have

$$cx \leqslant \frac{\psi(x)}{\varphi(\psi(x))} \leqslant \frac{f(x)}{\varphi(f(x))}. \tag{14}$$

Clearly, (13) and (14) imply (12). □

3.5. *Cost estimate functions*

Since, as we have argued, the randomness/growth alternative holds for arbitrary recursive functions, and not only for runtimes and the like, we will briefly discuss specific properties of runtimes, considered from the perspective of categorification, explained in Section 2.

Let P be an enriched programming method, as in Definition 2.4. We will say that a partial function $\delta : P(X,Y) \times X \to \mathbf{Z}^+$ is *a cost estimate function* if the following conditions are satisfied:

(i) *δ is partial recursive (morphism in $\mathscr{C}$) and* $D(\delta) = \{(p,x) \mid x \in D(f_p)\}$.
(ii) $\delta(p \circ q, x) = \delta(q,x) + \delta(p, q(x))$ *whenever both sides are defined.*

Requirement (i) is natural because the 'run-cost' of a computation (time, maximum storage size) must be computable in terms of cost increments required at each step. Requirement (ii) then expresses the additivity of such increments. We may, or may not, ascribe a non-zero cost to the program calculating identical function ('data transfer').

Requirement (i) complemented by the requirement of the *decidability of the graph of* δ constitute two axioms that are due to M. Blum. The latter property also has a clear intuitive meaning.

Finally, if our cost estimate function refers to time only, and we allow unrestricted parallelism, the following property is natural. Using the notation of (8), we must have:

(iii) $\delta(\pi_\sigma(p_1,\ldots,p_n),(x_1,\ldots,x_n)) = \max(\delta(p_1,x_1),\ldots,\delta(p_n,x_n))$.

3.6. *Constants related to Kolmogorov complexity estimates*

Since inequalities (9) and (10), and their extensions are often useful, we will say a few words about their computability.

As in Manin (2012, VI.9), we call partial maps $(\mathbf{Z}^+)^m \to \mathbf{Z}^+$, $m \geqslant 0$, *m-functions.* A Kolmogorov optimal family of m-functions $u(x_1, \dots, x_m; k)$, $k \in \mathbf{Z}^+$, is produced from two inputs:

(a) A fully effective (in the sense of Rogers, *cf.* Section 2.4.3) family of $(m+1)$-functions U.

(b) A recursive embedding $\theta : \mathbf{Z}^+ \times \mathbf{Z}^+ \to \mathbf{Z}^+$ with decidable image satisfying a linear growth condition

$$\theta(k, j) \leqslant k \cdot \varphi(j) \tag{15}$$

where $\varphi : \mathbf{Z}^+ \to \mathbf{Z}^+$ an appropriate function.

Having made these choices, we put

$$u(x_1, \dots, x_m; k) := U(x_1, \dots, x_m; \theta^{-1}(k)). \tag{16}$$

Then for any other family v of m-functions v with base $\mathbf{Z}^+$ and each m-function f, we have the inequality

$$C_u(f) \leqslant c_{u,v} C_v(f) \tag{17}$$

with

$$c_{u,v} := \varphi(C_U(v)). \tag{18}$$

(*cf.* Manin (2010, VI.9.4)).

Clearly, (9) is a special case of (17). An effective estimate of (18) will be assured, if φ is computable and increasing, and if, knowing a P-description of v, we can find some member of the family U coinciding with v. The latter, in turn, is automatic if P is supplied with a translation morphism $trans_{P,U}$.

We will now discuss the numberings θ.

3.7. *Slowly growing numberings*

Let $R = (R_k \,|\, k \in \mathbf{Z}^+)$ be a sequence of positive numbers tending to infinity with k. For $M \in \mathbf{Z}^+$, we put

$$V_R(M) := \{(k, l) \in (\mathbf{Z}^+)^2 \,|\, kR_l \leqslant M\}. \tag{19}$$

Clearly,

$$card\, V_R(M) \leqslant \sum_{l=1}^{\infty} \left[\frac{M}{R_l}\right] < \infty, \tag{20}$$

where $[a]$ denotes the integral part of a.

We have

$$V_R(M) \subset V_R(M+1),\ (\mathbf{Z}^+)^2 = \cup_{M=1}^{\infty} V_R(M).$$

Therefore, we can define a bijection $N_R : \mathbf{Z}^+ \to (\mathbf{Z}^+)^2$ by saying $N_R(k,l)$ is the number of (k,l) in the total ordering $<_R$ of $(\mathbf{Z}^+)^2$ determined inductively by the rule that $(i,j) <_R (k,l)$ if and only if one of the following alternatives holds:

(a) $iR_j < kR_l$.
(b) $iR_j = kR_l$ and $j < l$.

Proposition 3.3. The numbering N_R is well defined and has the property that all elements of $V_R(M+1) \setminus V_R(M)$ have strictly larger numbers than those of $V_R(M)$. Moreover:

(i) If each R_l is rational, or computable from above, then N_R is computable (total recursive).

(ii) If the series $\sum_{l=1}^{\infty} R_l^{-1}$ converges and its sum is bounded by a constant c, then

$$N_R(k,l) \leqslant c(kR_l + 1). \tag{21}$$

(iii) If the series $\sum_{l=1}^{\infty} R_l^{-1}$ diverges, and

$$\sum_{l=1}^{M} R_l^{-1} \leqslant F(M) \tag{22}$$

for a certain increasing function $F = F_R$, then

$$N_R(k,l) \leqslant (kR_l + 1)F(kR_l + 1). \tag{23}$$

Proof. The first statements are an easy exercise. For (21) and (22), notice that if M is the minimal value for which $(k,l) \in V_R(M)$, we have $M - 1 < kR_l \leqslant M$ and

$$N_R(k,l) \leqslant card\, V_R(M).$$

Then for part (ii), we have from (20) that

$$card\, V_R(M) \leqslant \sum_{m=1}^{\infty} MR_m^{-1} \leqslant c(kR_l + 1).$$

Similarly, for part (iii), we have

$$card\, V_R(M) \leqslant M \sum_{m=1}^{M} R_m^{-1} \leqslant (kR_l + 1)F(kR_l + 1).$$

□

3.8. *Levin's norms*

From (21), any sequence $\{R_l\}$ with converging $\sum_l R_l^{-1}$ can be used in order to construct the bijection $\mathbf{Z}^+ \times \mathbf{Z}^+ \to \mathbf{Z}^+$, $(k,l) \mapsto N_R(k,l)$ growing linearly with respect to k. Assume that it is computable and therefore can play the role of θ in condition (b) in Section 3.6. In this case, for any integer M the set $V_R(M)$ must be decidable. It follows that for any l, the set of rational numbers $k/M \leqslant r_l := R_l^{-1}$ is decidable.

Even if we weaken the last condition, requiring only recursivity of the set $k/M \leqslant r_l$ (that is, asking each r_l to be computable from below), the convergence of $\sum_l r_l$ implies that there is a universal upper bound (up to a constant) for such r_l. Namely, let CP be the *prefix Kolmogorov complexity* on $\mathbf{Z}^+$ defined with the help of a certain optimal prefix enumeration.

Proposition 3.4 (Levin 1976). For any sequence of computable from below numbers r_l with convergent $\sum_l r_l$, there exists a constant c such that for all l, we have $r_l \leqslant c \cdot CP(l)^{-1}$.

More generally, Levin constructs in this way a hierarchy of complexity measures associated with a class of abstract norms, which are functionals on sequences computable from below.

4. Regularisation and renormalisation of the Halting Problem

4.1. *Introduction*

In this section, we devise some simple regularisation/renormalisation schemes tailored to the Halting Problem. The general structure of such a scheme is sketched in Manin (2012, Section 0.2), and involves the following components.

(a) *Deforming the Halting Problem.*

In this step, we transform the problem of recognising whether a number $k \in \mathbf{Z}^+$ belongs to the definition domain $D(f)$ of a partial recursive function f to the problem of deciding whether an analytic function $\Phi(k, f; z)$ of a complex parameter z has a singularity (in our case, a pole) at $z = 1$.

In fact, using an idea from quantum computing, we may reduce the case of arbitrary f to the case of a partial recursive permutation $\sigma = \sigma_f : D(\sigma) \to D(\sigma)$ of its definition domain, and construct $\Phi(k, \sigma; z)$ for such permutations. This reduction is described in Manin (2012, Sections 3.6–3.8).

(b) *Choosing a minimal subtraction algebra.*

Our choice of an appropriate minimal subtraction algebra (see the definition in Manin (2012, Section 4.2)) is based on the established properties of the functions $\Phi(k, \sigma; z)$ – *cf.* Proposition 4.4.

Specifically, let $\mathscr{A}_+$ be the algebra of analytic functions in $|z| < 1$, continuous at $|z| = 1$. This is a unital algebra, and we endow it with the augmentation $\varepsilon_{\mathscr{A}} : \Phi(z) \mapsto \Phi(1)$, and put

$$\mathscr{A}_- := (1 - z)^{-1}\mathbf{C}[(1 - z)^{-1}].$$

Finally, we let

$$\mathscr{A} := \mathscr{A}_+ \oplus \mathscr{A}_-.$$

We can now use Manin (2012, Theorem 4.4.1) in renormalisation schemes involving a connected filtered Hopf algebra $\mathscr{H}$ (*cf.* Ebrahimi-Fard and Manchon (2007, Section 2.5, Theorem 1) and Manin (2012, Section 4.1)). However, we still need to indicate which Hopf algebras and which of their $\mathscr{A}$-characters will be involved in this game.

(c) *Hopf algebra of an enriched programming method.*

A class of such algebras is described in Manin (2012, Sections 3.3–3.4). This construction explicitly refers to flowcharts, but can be easily modified and generalised so that they can be applied to enriched programming methods P in the sense of Definition 2.4.

Basically, $\mathscr{H} = \mathscr{H}_P$ is the symmetric algebra spanned by isomorphism classes $[p]$ of certain descriptions belonging to, say, $P(\mathbf{Z}^+, \mathbf{Z}^+)$. Comultiplication in $\mathscr{H}_P$ is dual to the composition of descriptions:

$$\Delta([p]) := \sum_{q,r|q\circ r=p} [r] \otimes [q]. \tag{24}$$

(Recall that the composition of descriptions is associative).

In order to ensure the finiteness of the right-hand side of (24) and to produce a Hopf filtration, we must also postulate the existence of a 'size function' on descriptions. The simplest properties of such a function $p \mapsto |p|$ that will serve our goal are finiteness of the set of descriptions of bounded size and additivity

$$|q \circ r| = |q| + |r|.$$

For a concrete example, see Manin (2012, Section 3.4).

(d) *Characters, corresponding to the Halting Problem.*
Finally, we assume that we have constructed $\mathscr{H}_P$ and $\Phi(k, f; z)$ as above. Then the character $\varphi_k : \mathscr{H}_P \to \mathscr{A}$ (*cf.* Manin (2012, Section 4.4)) corresponding to the Halting Problem at a point $k \in \mathbf{Z}^+$ for the partial recursive function computable with the help of a description $p \in P(\mathbf{Z}^+, \mathbf{Z}^+)$ is defined by

$$\varphi_k([p]) := \Phi(k, f; z) \in \mathscr{A}. \tag{25}$$

As soon as this definition is adopted, the machinery and philosophy of Hopf renormalisation and Birkhoff decomposition (Manin 2012, Theorem 4.4.1) becomes applicable to the classical Halting Problem.

It may be even more relevant for quantum computation schemes based on infinite-dimensional Hilbert spaces.

4.2. *The simplest construction*

Let $f : \mathbf{Z}^+ \to \mathbf{Z}^+$ be a partial recursive function. Consider its extension $\bar{f} : \mathbf{N} \to \mathbf{N}$ defined by $\bar{f}(x) = f(x)$ if $x \in D(f)$ and $f(x) = 0$ otherwise.

Put

$$\Psi(k, f; z) := \sum_{n=0}^{\infty} \frac{z^n}{(1 + n\bar{f}(k))^2}. \tag{26}$$

Proposition 4.1.

(i) If $k \notin D(f)$, then

$$\Psi(k, f; z) = \frac{1}{1 - z}. \tag{27}$$

(ii) If $k \in D(f)$, then $\Psi(x, \sigma; z)$ is the Taylor series of a function analytic at $|z| < 1$ and continuous at the boundary $|z| = 1$. The value $\bar{f}(k) = f(k)$ can be uniquely reconstructed from Ψ: for example,

$$f(k) = \sqrt{\left.\frac{dz}{d\Psi}\right|_{z=0}} - 1. \tag{28}$$

Proof. The proof is obvious. □

In fact, formula (26) can be seen in its natural context if we borrow the general prescription of reducing any function to a permutation from the theory of quantum computation.

We will now briefly recall this prescription, following Manin (2012, Sections 3.6–3.8).

4.3. *Reduction of the general Halting Problem to the recognition of fixed points of permutations*

We begin with a partial recursive function $f : X \to X$ where X is an infinite constructive world and extend X by one point, that is, form $X \coprod \{*_X\}$. We then choose a total recursive structure of an additive group without torsion on $X \coprod \{*_X\}$ with zero $*_X$. We now extend f to the everywhere defined (but generally uncomputable) function $g : X \coprod \{*_X\} \to X \coprod \{*_X\}$ by

$$g(y) := *_X \text{ if } y \notin D(f).$$

We then define the map

$$\tau_f : \left(X \coprod \{*_X\}\right)^2 \to \left(X \coprod \{*_X\}\right)^2$$

by

$$\tau_f(x, y) := (x + g(y), y). \tag{29}$$

This is clearly a permutation, and since $(X \coprod \{*_X\}, +)$ has no torsion, the only finite orbits of $\tau_f^{\mathbf{Z}}$ are fixed points.

Moreover, the restriction of τ_f to the recursive enumerable subset

$$D(\sigma_f) := \left(X \coprod \{*_X\}\right) \times D(f)$$

of the constructive world $Y := (X \coprod \{*_X\})^2$ induces a partial recursive permutation σ_f of this subset.

Since $g(y)$ never takes the zero value $*_X$ on $y \in D(f)$, but is always zero outside it, the complement of $D(\sigma_f)$ in Y consists entirely of fixed points of τ_f.

Thus, the Halting Problem for f reduces to the fixed point recognition for τ_f.

4.4. *Permutations with bounded shift*

The formula (26) can be generalised as follows.

Definition 4.2. Let σ be a permutation of $\mathbf{Z}^+$, $k \in \mathbf{Z}^+$. We say that σ has a bounded shift at k if there exist constants a, b, c (depending on σ and k) such that for all $n \in \mathbf{Z}$,

$$c \cdot |n + a| \leqslant \sigma^n(k) \leqslant c \cdot |n + b|. \tag{30}$$

Lemma 4.3. If σ has bounded shift at k, then the $\sigma^{\mathbf{Z}}$-orbit of k is infinite, and for any $m \neq 0$ and any point of this orbit l, we have σ^m has bounded shift at l.

Proof. Let $l = \sigma^d(k)$. From (26) we get

$$c \cdot |mn + d + a| \leqslant \sigma^{mn}(l) = \sigma^{mn+d}(k) \leqslant c \cdot |mn + d + b|,$$

that is

$$c|m| \cdot \left| n + \frac{d+a}{m} \right| \leqslant (\sigma^m)^n(l) \leqslant c|m| \cdot \left| n + \frac{d+b}{m} \right|. \tag{31}$$

This inequality has the same form as (30), but with different constants. □

Proposition 4.4. Let σ be a permutation of $\mathbf{Z}^+$, $k \in \mathbf{Z}^+$. Put

$$\Psi(k, \sigma; z) := \sum_{n=1}^{\infty} \frac{z^n}{\left(\sigma^n(k)\right)^2}. \tag{32}$$

Then we have:

(i) If $\sigma^{\mathbf{Z}}$-orbit of k is finite, then $\Phi(\sigma, x; z)$ is a rational function in z for which all poles are first order and lie at roots of unity.

(ii) If this orbit is infinite, and σ has bounded shift at (any point of) this orbit, then $\Phi(\sigma, k; z)$ is the Taylor series of a function analytic at $|z| < 1$ and continuous at the boundary $|z| = 1$.

Proof. If $\sigma^{\mathbf{Z}}$-orbit of k is finite, then (28) is a finite sum of several geometric progressions each of which sums to a rational function of the type

$$const \cdot \frac{z^k}{1 - z^l}.$$

Otherwise, because of (30), we get a series that is absolutely convergent for $|z| \leqslant 1$. □

4.5. *The Kolmogorov order*

There are many interesting σ, such as total recursive permutations, that are not permutations of bounded shift. To cope with this situation, we will (uncomputably) reorder $\mathbf{Z}^+$, and show that after this reordering, *all* partial recursive functions and permutations corresponding to them will satisfy a version of the bounded shift property, which will allow us to construct a modification of $\Psi(z)$.

Slightly more generally, let X be an infinite constructive world, and consider an optimal enumeration $u : \mathbf{Z}^+ \to X$ in the sense of Kolmogorov or Schnorr (see Section 2.4.3). This means that u is total recursive and surjective, and the function $C_u : X \to \mathbf{Z}^+$,

$$C_u(x) := \min\{k \,|\, u(k) = x\}$$

is (a representative of) the Kolmogorov complexity of constructive objects of type X.

We now define the Kolmogorov total order on X associated with u by

$$x < y \;\Leftrightarrow\; C_u(x) < C_u(y),$$

and write $\mathbf{K} = \mathbf{K}_u : X \to \mathbf{Z}^+$ to denote the function

$$\mathbf{K}(x) := 1 + card\,\{y \,|\, C_u(y) < C_u(x)\}.$$

$\mathbf{K}$ is clearly a bijection. If we arrange X in the order of growing Kolmogorov complexity, $\mathbf{K}(x)$ is precisely the number of x in this order.

It is also convenient to introduce a Kolmogorov order on $\mathbf{Z}^+$. We will denote the numbering for this order by the same letter $\mathbf{K}$ since this should not lead to any confusion.

It is straightforward to check that for some constant $c_0 > 0$ and all $x \in X$, we have

$$c_0\, C_u(x) \leqslant \mathbf{K}(x) \leqslant C_u(x). \tag{33}$$

We now let $\sigma : X \to X$ be a partial recursive map, such that σ maps $D(\sigma)$ to $D(\sigma)$ and induces a permutation of this set. We put

$$\sigma_{\mathbf{K}} := \mathbf{K} \circ \sigma \circ \mathbf{K}^{-1}$$

and consider it as a permutation of the subset

$$D(\sigma_{\mathbf{K}}) := \mathbf{K}(D(\sigma)) \subset \mathbf{Z}^+$$

consisting of numbers of elements of $D(\sigma)$ in the Kolmogorov order. We then have the following modified version of (30):

Proposition 4.5. Let $x \in D(\sigma)$. If the orbit $\sigma^{\mathbf{Z}}(x)$ is infinite, then there exist constants $c_1, c_2 > 0$ such that for $k := \mathbf{K}(x)$ and all $n \in \mathbf{Z}$ we have

$$c_1 \cdot \mathbf{K}(n) \leqslant \sigma^n_{\mathbf{K}}(k) \leqslant c_2 \cdot \mathbf{K}(n). \tag{34}$$

Proof. Let $k = \mathbf{K}(x)$, $x \in X$. We have for $n > 0$ that

$$\sigma^n_{\mathbf{K}}(k) = \mathbf{K}(\sigma^n(x)) \leqslant c \cdot \mathbf{K}(n) \tag{35}$$

for any fixed Kolmogorov complexity order on $\mathbf{Z}^+$ (which we will denote by the same letter $\mathbf{K}$ to simplify the notation). In fact, if we replace $\mathbf{K}$ in (35) by the appropriate complexity C, this will follow from (9), since $n \mapsto \sigma^n(x)$ is an everywhere defined morphism $\mathbf{Z}^+ \to X$ in $\mathscr{C}$. It remains to invoke (33).

Furthermore, let $Y := \{\sigma^n(x) \,|\, n \in \mathbf{Z}^+\}$. This is a recursively enumerable subset of X, and the partial function $\lambda : X \to \mathbf{Z}^+$ with definition domain Y,

$$\lambda(y) = n \;\text{ if }\; y = \sigma^n(x),$$

is partial recursive. Hence, again in view of (29) and (9),

$$\mathbf{K}(n) = \mathbf{K}(\lambda(y)) \leqslant c' \cdot \mathbf{K}(y) = c' \cdot \mathbf{K}(\sigma^n(x)). \tag{36}$$

Combining (35) and (36), we get (34) for $n \geqslant 0$. Applying the same reasoning to σ^{-1} in place of σ, we obtain (34) for negative n. □

Proposition 4.6. With the same notation as in Proposition 4.5, we put

$$\Phi(k,\sigma;z) := \frac{1}{k^2} + \sum_{n=1}^{\infty} \frac{z^{\mathbf{K}(n)}}{\left(\sigma_{\mathbf{K}}^n(k)\right)^2}. \tag{37}$$

Then we have:

(i) If $\sigma^{\mathbf{Z}}$-orbit of x is finite, then $\Phi(x,\sigma;z)$ is a rational function in z for which all the poles are first order and lie at roots of unity.

(ii) If this orbit is infinite, then $\Phi(x,\sigma;z)$ is the Taylor series of a function analytic at $|z| < 1$ and continuous at the boundary $|z| = 1$.

4.6. *Remarks*

(a) In the proofs of Propositions 4.5 and 4.4, we actually only used the fact that σ restricted to the particular orbit $Y := \{\sigma^n(x) \,|\, n \in \mathbf{Z}^+\}$ is recursive, which justifies our choice of $\mathscr{A}_-$ in component (b) of Section 4.1.

(b) Although Kolmogorov's order is as uncomputable as Kolmogorov's complexity, there are serious arguments for studying constructions explicitly involving it, such as our renormalisation characters.

One can argue that all cognitive activity of our civilisation that is based on symbolic (in particular, mathematical) representations of reality actually deals with the *initial Kolmogorov segments* of potentially infinite linguistic constructions, thereby *always* replacing vast volumes of data by their compressed descriptions. This is especially apparent in the outputs of the modern genome projects.

In this sense, such linguistic cognitive activity can be metaphorically compared to a gigantic precomputation process, shellsorting infinite worlds of expressions in their Kolmogorov order.

Acknowledgements

I am very grateful to Cristian Calude, Leonid Levin, Mike Stay and Noson Yanofsky, who sent me remarks and suggestions that have been incorporated in the paper.

References

Baez, J. and Stay, M. (2010) Physics, topology, logic and computation: a Rosetta stone. In: Coecke, B. (ed.) New Structures for Physics. *Springer-Verlag Lecture Notes in Physics* **813** 95–172. (Available at arXiv:0903.0340.)

Calude, C. and Stay, M. (2008) Most programs stop quickly or never halt. *Advances in Applied Mathematics* **40** 295–308.

Calude, C. and Stay, M. (2006) Natural halting probabilities, partial randomness, and zeta functions. *Information and Computation* **204** 1718–1739.

Ebrahimi-Fard, K. and Manchon, D. (2007) The combinatorics of Bogolyubov's recursion in renormalization. (Available at arXiv0710.3675 [math-ph].)

Gács, P. and Levin, A. (1982) Causal nets or what is a deterministic computation? *International Journal of Theoretical Physics* **21** (12) 961–971.

Grass, J. (1996) Reasoning about Computational Resource Allocation. An introduction to anytime algorithms. Posted on the XRDS (Crossroads) website.

Grass, J. and Zilberstein, S. (1995) Programming with anytime algorithms. In: *Proceedings of the IJCAI-95 Workshop on Anytime Algorithms and Deliberation Scheduling*.

Heller, A. (1990) An existence theorem for recursive categories. *Journal of Symbolic Logic* **55** (3) 1252–1268.

Levin, L. (1976) Various measures of complexity for finite objects (axiomatic description). *Soviet Math. Dokl.* **17** (2) 522 –526.

Li, M. and Vitányi, P. (1993) *An Introduction to Kolmogorov Complexity and its Applications*, Springer.

Manin, Y. (2010) *A Course in Mathematical Logic* (the second, expanded Edition), Springer-Verlag.

Manin, Y. (1999) Classical computing, quantum computing, and Shor's factoring algorithm. *Séminaire Bourbaki*, Exposée 862 (June 1999), Astérisque **266** 375–404. (Available at arXiv:quant-ph/9903008).

Manin, Y. (2012) Renormalization and computation I. Motivation and background. In: Loday, J-L. and Vallette, B. (eds.) Proceedings OPERADS 2009. *Séminaires et Congrès* **26**, Société Mathématique de France 181–223. (Preprint available at arXiv:0904.492.)

Nabutovsky, A. and Weinberger, S. (2003) The fractal nature of Riem/Diff I. *Geometriae Dedicata* **101** 145–250.

Rogers, H. (1958) Gödel numberings of partial recursive functions. *Journal of Symbolic Logic* **23** 331–341.

Russell, S. J. and Zilberstein, S. (1991) Composing real–time systems. In: Mylopoulos, J. and Reiter, R. (eds.) *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia*, Morgan Kaufmann 212–217.

Schnorr, C. P. (1974) Optimal enumerations and optimal Gödel numberings. *Mathematical Systems Theory* **8** (2) 182–191.

Soare, R. I. (2004) Computability theory and differential geometry. *Bulletin of Symbolic Logic* **10** (4) 457–486.

Yanofsky, N. S. (2006) Towards a definition of an algorithm. (Available at arXiv:math/0602053v3 [math.LO].)