RENORMALIZATION AND COMPUTATION I: MOTIVATION AND BACKGROUND

Yuri I. Manin

Max–Planck–Institut für Mathematik, Bonn, Germany,
and Northwestern University, Evanston, USA

To Grigori Mints for his 70th anniversary

Abstract. The main observable quantities in Quantum Field Theory, *correlation functions*, are expressed by the celebrated Feynman path integrals which are not well defined mathematical objects.

Perturbation formalism interprets such an integral as a formal series of finite–dimensional but *divergent integrals,* indexed by Feynman graphs, the list of which is determined by the Lagrangian of the theory.

Renormalization is a prescription that allows one to systematically "subtract infinities" from these divergent terms producing an asymptotic series for quantum correlation functions.

On the other hand, graphs treated as *"flowcharts"*, also form a combinatorial skeleton of the abstract computation theory and various operadic formalisms in abstract algebra.

In this role of *descriptions* of various (classes of) computable functions, such as recursive functions, functions computable by a Turing machine with oracles etc., graphs can be used to replace standard formalisms having linguistic flavor, such as Church's λ–calculus and various programming languages.

The functions in question are generally not everywhere defined due to potentially infinite loops and/or necessity to search in an infinite haystack for a needle which is not there.

In this paper I argue that such infinities in *classical computation theory* can be addressed in the same way as Feynman divergences, and that meaningful versions of renormalization in this context can be devised. Connections with quantum computation are also touched upon.

Contents

0. Introduction

0.1. Feynman integrals. The main observable quantities in Quantum Field Theory, *partition and correlation functions*, are expressed by the celebrated Feynman path integrals which are not well defined mathematical objects.

Perturbation formalism in Quantum Field Theory (QFT) interprets a Feynman path integral as a formal series $\sum_\tau I_\tau$ of multidimensional integrals, which are generally divergent. Formal expressions for these divergent terms are labeled by (decorated) Feynman graphs τ (whose exact structure and totality depend on the Lagrangian of the Quantum Field Theory in question): see [Po], [Fra], [Cos], and sec. 2 below based upon [Ma3], Ch. IV.3.

0.2. Renormalization. *A renormalization scheme* is a prescription (depending on QFT) that allows one to systematically "subtract infinities" from these divergent terms and produce an asymptotic series for quantum correlation functions. After the initial breakthroughs made by physicists in the 40s– 70s, several mathematical versions of renormalization gradually crystallized.

We will use the following version which consists of two parts that have essentially different natures.

Part 1: A regularization scheme. The divergent integrals corresponding to all individual Feynman graphs are first "deformed", by systematical introduction of a parameter, say z, such that for (sufficiently small) $z \neq 0$ the integrals converge, whereas for $z = 0$ we get an initial divergent expression. As the simplest example, imagine that an individual deformed Feynman integral $I_\tau(z)$ defines a germ of a meromorphic function of z with the only singularity at zero. Then one can define *the regularized value of* I_τ as the difference

$$I_{\tau,reg} := (I_\tau(z) - \ \textit{the polar part of}\ I_\tau(z))\,|_{z=0} \tag{0.1}$$

("minimal subtraction" of counterterms).

The choice of a specific z and the respective deformation is an indispensable input of physics at this stage. It has not been formalized in a single concise mathematical scheme: several known regularizations display remarkable mathematical and physical variety.

Moreover, even the prescriptions to read off the physically meaningful numbers from a chosen regularization scheme are wonderfully ambiguous.

A number of the standard choices is based on the idea of *cut–off*: e. g. one introduces a finite scale of integration where the infinite one led to divergences (such as ultraviolet ones), and then calculates, say, an "effective action" at this scale.

In this case the infinity of the polar part is not subtracted but rather "kept at a safe distance" and declared to be the bearer of the physical meaning.

A totally non–obvious choice is done in the remarkable schemes of *dimensional regularization*: the dimension of space–time, 4, is deformed into a "complex dimension" $4+z$, by appropriate formal changes in integrands. There is a series of other very interesting examples.

Part 2: Renormalization as simultaneous regularization. Feyman integrals corresponding to different Feynman graphs τ are mutually interrelated in a way reflecting the combinatorics of graphs: typically, a divergent integral contributes to the formal integrand of a larger integral, that remains divergent even after the (partial) regularization of the integrand.

This means that the regularization schemes chosen for individual integrals must be united in a coherent whole.

After the seminal papers [Kr1], [ConKr], this can be done in the following way, in two steps.

First, the linear span of isomorphism classes of Feynman graphs τ is given the structure of a Hopf algebra H. Multiplication in this algebra is induced by the disjoint union of graphs, whereas comultiplication is more sophisticated and roughly speaking, reflects how bigger graphs are constructed from smaller ones.

Second, the totality of deformed integrals $I_\tau(z)$ induces a character (linear functional) of the Hopf algebra H, $\varphi : H \to A$. In the simplest case of the regulariztion scheme explained above, this character takes values in the algebra $A = A_+ \oplus A_-$ of germs of analytic functions meromorphic near $z = 0$. Here A_+ is the unital subalgebra of holomorphic germs, and A_- is the subalgebra of polar parts.

The appropriate characters of H form a group (defined with the help of the comultiplication of H and the multiplication of A). In this group, there is a version

of Birkhoff decomposition: in particular, φ above can be presented the product of its regular part φ_+ taking values in A_+ and a polar part φ_- in $1+A_-$. The regular part furnishes all regularized Feynman integrals simultaneously:

$$I_{\tau,reg} := \varphi_+(\tau, z)\,|_{z=0} \tag{0.2}$$

A concise and well–rounded exposition of the relevant mathematics can be found in [E-FMan].

A warning. The terminology adopted in the sketch above is not universally accepted one. The distinction we make between regularization and renormalization in physics is generally blurred. In mathematics, the word regularization is used in different senses in a variety of contexts having only a vague common intuitive kernel.

Therefore, I summarize once again our current usage.

Regularization schemes in general deal with the problem of "extracting finite information" from potentially infinite expressions: summing divergent sums, evaluating divergent integrals, or, as we try to do, taming more general computation processes. The central, but by no means all–inclusive, intuitive image is that of *"subtracting the divergent part"*. The first important examples and insights go back at least to Euler.

Renormalizaton studies ways of doing it *compatibly with composition* of potentially divergent expressions.

Finally, *the Hopf algebra renormalization* adds to this picture the idea that all relevant compositions can be heaped into one big Hopf algebra, or group. This allows one to replace the term–by–term subtraction of divergences with an overall *division* by the divergent part in the respective group (Birkhoff decomposition).

0.3. Renormalization and computation. In this paper I argue that certain computational problems could benefit from a systematic development of regularization/renormalization technique. Moreover, some basic ingredients of Feynman's renormalization can be very naturally transposed into computational contexts.

Here I will start with a brief discussion of such problems, related Hopf algebras, and regularization schemes.

0.4. Computation and infinity. Adopting the classical Church thesis, I will identify the universe of classical computation theory with that of *partial recursive functions*. Classically, such a function is a partial map $f : \mathbf{Z}^+ \to \mathbf{Z}^+$ (or, more

generally, $f : (\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$). Values $f(x)$ can be computed in finite running time, if x belongs to the definition domain $D(f)$ of f. However, outside of $D(f)$ the algorithm computing f (Turing machine, or any other programming method) will generally work indefinitely long without producing a definite answer. This is called the undecidability of the Halting Problem.

In order to avoid this kind of "computing block", it is often useful to recede and replace the initial problem by that of computation of the graph $\Gamma_f \subset \mathbf{Z}^+ \times \mathbf{Z}^+$. The point is that Γ_f is the total image of a *general, or total, recursive function*, say $F : \mathbf{Z}^+ \to \mathbf{Z}^+ \times \mathbf{Z}^+$, which is everywhere defined. Then all points $(x, f(x)), x \in D(f)$, will be printed out in some order.

Both the domain and the target of a partial recursive map f need not be $\mathbf{Z}^+$: any enumerable set X having a natural ("computable") numbering will do. It is convenient to consider such sets ("constructive worlds") as objects of a category ("constructive universe"), and partial recursive maps between them as morphisms.

For example, computability studies in classical analysis usually start with a working definition of a computable real number. One can declare, say, that such a number α is represented by a computable Cauchy sequence computably approximating it, that is, by a total recursive function $\mathbf{Z}^+ \to \mathbf{Q}$, $n \mapsto r_n$ such that $|\alpha - r_n| < 2^{-n}$. A recent paper by M. Yoshinaga [Yo] shows that all *periods* in the sense of [KoZa] are computable (see also [Fri] for background). In fact, (certain) periods directly appear as values of Feynman integrals: cf. [BlEsKr], [Bl], [AlMar1–3] and references therein.

More generally, challenging problems arise when the target domain is (a constructive approximation to) a world of geometric images, say, subsets of $\mathbf{R}$ or $\mathbf{R}^d$. The case $d = 2$ can be motivated by imagining computer screen pictures with arbitrary high resolution.

0.5. Example: computability of Julia sets. In [BrYa], the authors define the constructive world B^d consisting of finite unions of closed balls with rational centers and rational radii in $\mathbf{R}^d$ (they in fact restrict themselves by dyadic rational numbers of the form $p/2^n$.) Such sets are dense in the Hausdorff metric on the set of compact subsets.

A compact set $K \subset \mathbf{R}^d$ is called computable, if there exists a computable sequence in B^d computably approximating K wrt the Hausdorff metric. Braverman and Yampolsky then proceed to show, that for certain *computable numbers* c, the Julia sets $J(z^2 + c)$ are *not computable.* (Julia sets belong to the class of most popular fractal pictures).

0.6. Example: Code domains and isolated codes. Another class of two–dimensional pictures, *code domains,* naturally arises in the theory of error–correcting codes. The following results were proved by the author in 1981; for detailed exposition, see cf. [VlMa], Ch. 1, and [TsVlN], sec. 1.3.

Let F be a finite set, *alphabet*, of cardinality q. Consider the constructive world of *codes*: pairs (n, C) where $n \in \mathbf{Z}^+$ and $C \subset F^n$. We write also $n(C)$ in place of n. The Hamming distance $d(a, b)$ between two words $a = (a_i)$, $b = (b_i)$ in F^n is defined as the number of i with $a_i \neq b_i$. Define the following computable functions of codes with integer (resp. rational) values:

$$d(C) = \min \{d(a, b) \mid (a, b) \in C \times C, a \neq b\},\ k(C) := [\log_q \operatorname{card} C]$$

$$R(C) := \frac{k(C)}{n(C)}, \quad \delta(C) = \frac{d(C)}{n(C)}.$$

Denote by V_q the set of all points $(\delta(C), R(C))$ (q being fixed, n variable). Let $\overline{V}_q$ be the closure of V_q in the square $[0, 1]^2$ of the (δ, R)–plane.

It is proved in [VlMa] that $\overline{V}_q$ consists of all points, lying below or on some continuous curve $R = \alpha_q(\delta)$, plus a subset of isolated points lying above this curve.

Similar results were proved for subclasses of codes: *linear codes*, which are linear subspaces of $\mathbf{F}_q^n$, and polynomial time computable linear codes.

It is not known whether $\alpha_q(\delta)$ is a differentiable function. Arguably, a high resolution picture could help make an educated guess. It is not known whether the set of isolated codes or its complement in V_q are computable.

0.7. Kolmogorov complexity as ultimate "computational infinity". I consider both problems discussed in 0.5 and 0.6 as classes of computations on which the problem of regularization might be tested. As Braverman and Yampolski ask in [BrYa], "What would a computer really draw, when $J(z^2 + c)$ is uncomputable?" Regularization is expected to be a (theoretical) modification of computation that produces better pictures at least in most cases.

I expect that mathematical content of regularization will be a reduction of other "computationally challenged" problems to the universal one: computation of Kolmogorov complexity.

The simplest model is this (cf. [Ma2], sec. 5). Denote by $\mathbf{K} : \mathbf{Z}^+ \to \mathbf{Z}^+$ a (non–computable) permutation rearranging positive integers in the order of increasing Kolmogorov complexity (with respect to some fixed optimal numeration). Then

any recursive function f becomes bounded by a linear function in the following sense. There exists a constant $c = c_f$ such that

$$\mathbf{K} \circ f \circ \mathbf{K}^{-1}(n) \leq cn \quad \text{for all } n \in \mathbf{K}(D(f)).$$

Recall that $\mathbf{K}$ is bounded by a linear function, but $\mathbf{K}^{-1}$ is not bounded by any recursive function.

Now, imagining the conjugation $\mathbf{K} \circ f \circ \mathbf{K}^{-1}$ as a regularized version of f, we see that it become "computable" in a Pickwickean sense: regularized values of f, regularized running time and memory of an algorithm, computing φ, all become functions of no more than linear growth.

The following verbal reformulation might serve as a justification of this scheme. $\mathbf{K}(x)$ is *a short(est) description* of the number x, and $\mathbf{K}(f(*))$ is a short(est) description of the number $f(*)$. Thus, $\mathbf{K} \circ f \circ \mathbf{K}^{-1}$ is an avatar of f which replaces operations on numbers by operations on their short descriptions. An oracle computing $\mathbf{K}$ and $\mathbf{K}^{-1}$ would be helpful.

However, regularization in computation remains a challenging problem, and it is not satisfactorily solved in this paper. (Cf. also remarks on tropical geometry in the subsection 4.6).

Much better is the situation with the Hopf algebra analogs.

0.8. Graphs as programming methods. Analogs of Feynman graphs appear as "flowcharts" used for visualization of information flows in various processes of computation: cf. for example [Sc], [Zo], [Mar]. As such, they often serve only illustrative and mnemonic purposes, comparable with the role of pictures in schoolbook versions of Euclid's geometry.

One notable exception is [Sc] where a class of decorated graphs is structured to form a Boolean category, and infinite graphs are invoked to encode loops.

Graphs as a category of structured sets in Bourbaki's sense arise very naturally in the theory of operads of various kinds. In [BoMa], we developed a formalism allowing one to consider (decorated) graphs as objects of a monoidal category and to interpret uniformly many versions of algebraic operads, PROPs etc. as functors on various categories of graphs.

As a necessary step, I show that the combinatorics involved in the definition of the Hopf comultiplication for Feynman diagrams can be transported to flowcharts and in fact to more general categories of *programming methods.*

0.9. Is there an analog of "action" in the theory of computation? Although the combinatorial skeleton of perturbative series naturally emerges in theoretical computing, the latter notoriously lacks an analog of the basic physical quantity, which is called *action.*

Recall that describing (an isolated piece of) physical reality, in the classical or quantum mode, we must start with the following two steps:

a) Define a (mathematical image of) the space of virtual paths/histories $\mathcal{P}$.

b) Introduce a (real–valued) functional $S : \mathcal{P} \to \mathbf{R}$ on this space, satisfying additivity properties wrt space–time domains ("conservation laws", "locality").

Then in the classical mode of description, physical histories correspond to the stationary points of S ("principle of the least action").

In the quantum mode, calculation of correlators and transition amplitudes requires understanding of Feynman integrals $\int_{\mathcal{P}} e^{iS(\varphi)} D\varphi$.

The theory of computation badly needs a quantity that would play the role of S. (Half a century ago, clearly seeing this need, I. M. Gelfand coined an expression "principle of the least interaction" in the theory of finite automata).

A version of Kolmogorov complexity again seems to me a promising candidate.

Acknowledgements. During the preparation of this paper, I profited from correspondence with Kevin Costello, Leonid A. Levin, Matilde Marcolli, Gregori Mints, Alexander Shen, Noson Yanofsky. I am very grateful to them all.

1. Feynman graphs and perturbation series: a toy model

This section is a very brief and elementary introduction to the perturbative formalism for Feynman path integrals. On a toy finite–dimensional model we demonstrate, first, how a series over Feynman diagrams arises, and second, how the structure of its terms depends on the action functional.

1.1. Notation. *Feynman path integral* is an heuristic expression of the form

$$\frac{\int_{\mathcal{P}} e^{S(\varphi)} D(\varphi)}{\int_{\mathcal{P}} e^{S_0(\varphi)} D(\varphi)} \tag{1.1}$$

or, more generally, a similar heuristic expression for *correlation functions.*

In the expression (1.1), $\mathcal{P}$ is imagined as a functional space of *classical fields* φ on a *space–time manifold* M. Space–time may be endowed with Minkovski or

Euclidean metric, but not necessarily: in models of quantum gravity metric is one of the fields. Fields in general may include scalar functions, tensors of various ranks, sections of vector bundles, connections etc.

$S : \mathcal{P} \to \mathbf{C}$ is a functional of *classical action*: usually $S(\varphi)$ is expressed as an integral over M of a local density on M which is called *Lagrangian.* In our notation (1.1) (differing by the sign from the standard one), $S(\varphi) = -\int_M L(\varphi(x))dx$. Lagrangian density may depend on derivatives, include distributions etc.

Usually $S(\varphi)$ is represented as the sum of *a quadratic part* $S_0(\varphi)$ (Lagrangian of free fields) and remaining terms which are interpreted as interaction and treated perturbatively.

Finally, the integration measure $D(\varphi)$ and the integral itself $\int_{\mathcal{P}}$ should be considered as simply a part of the total expression (1.1) expressing the idea of "summing over trajectories".

In our toy model, we will replace $\mathcal{P}$ by a finite–dimensional real space. For the time being, we endow it with a basis indexed by a finite set of "colors" A, and an Euclidean metric g encoded by the symmetric tensor (g^{ab}), $a, b \in A$. We put $(g^{ab}) = (g_{ab})^{-1}$.

The action functional $S(\varphi)$ will be a formal series in linear coordinates on $\mathcal{P}$, (φ^a), of the form

$$S(\varphi) = S_0(\varphi) + S_1(\varphi), \quad S_0(\varphi) := -\frac{1}{2}\sum_{a,b} g_{ab}\varphi^a\varphi^b,$$

$$S_1(\varphi) := \sum_{k=1}^{\infty} \frac{1}{k!} \sum_{a_1,\dots,a_k \in A} C_{a_1,\dots,a_k}\varphi^{a_1}\dots\varphi^{a_k} \tag{1.2}$$

where $(C_{a_1,\dots,a_n})$ are certain symmetric tensors. If these tensors vanish for all sufficiently large ranks n, $S(\varphi)$ becomes a polynomial and can be considered as a genuine function on $\mathcal{P}$.

On the other hand, below we will mostly consider (g_{ab}) and $(C_{a_1,\dots,a_n})$ as independent formal variables as well, "formal coordinates on the space of theories".

We can now state our first theorem, expressing the toy version of (1.1) as a series over (isomorphism classes of) graphs.

For the time being, a graph τ for us consists of two finite sets, edges E_τ and vertices V_τ, and the incidence map sending E_τ to the set of unordered pairs of

vertices. Each vertex is supposed to be incident to at least one edge. There is one *empty graph.*

1.2. Theorem. *We have, for a formal parameter* λ

$$\frac{\int_{\mathcal{P}} e^{\lambda^{-1}S(\varphi)}D(\varphi)}{\int_{\mathcal{P}} e^{\lambda^{-1}S_0(\varphi)}D(\varphi)} = \sum_{\tau\in\Gamma} \frac{\lambda^{-\chi(\tau)}}{|\mathrm{Aut}\,\tau|}\, w(\tau) \tag{1.3}$$

In the r.h.s. of (1.3) the summation is taken over (representatives of) all isomorphism classes of all finite graphs τ*. The weight* $w(\tau)$ *of such a graph is determined by the action functional (1.2) as follows:*

$$w(\tau) := \sum_{u\colon F_\tau\to A} \prod_{e\in E_\tau} g^{u(\partial e)} \prod_{v\in V_\tau} C_{u(F_\tau(v))}\,. \tag{1.4}$$

Here F_τ *is the set of flags, or "half–edges" of* τ*. Each edge* e *consists of a pair of flags denoted* ∂e*, and each vertex* v *determines the set of flags incident to it denoted* $F_\tau(v)$*. Finally,* $\chi(\tau)$ *is the Euler characteristic of* τ*.*

1.3. Comments. Even in this toy version, the meaning of the "perturbation series expansion" (1.3) requires some explanations.

First of all, the numerator of the left hand side of (1.3) is *by definition* the result of *term–wise integration* of the formal series which can be obtained from the Taylor series of the exponent in the integrand. Concretely

$$\int_{\mathcal{P}} e^{\lambda^{-1}S(\varphi)}D(\varphi) = \int_{\mathcal{P}} e^{\lambda^{-1}S_0(\varphi)} \left(1+\sum_{N=1}^{\infty}\frac{\lambda^{-N}S_1(\varphi)^N}{N!}\right)\prod_a d\varphi^a \;:=$$

$$\int_{\mathcal{P}} e^{\lambda^{-1}S_0(\varphi)}\prod_a d\varphi^a \;+$$

$$\sum_{N=1}^{\infty}\frac{\lambda^{-N}}{N!}\sum_{k_1,\dots,k_N=1}^{\infty}\frac{1}{k_1!\dots k_N!}\sum_{a_j^{(i)}\in A,1\le j\le k_i}\prod_{i=1}^{N} C_{a_1^{(i)},\dots,a_{k_i}^{(i)}}\int_{\mathcal{P}} e^{\lambda^{-1}S_0(\varphi)}\prod_{i,j}^{N}\varphi^{a_j^{(i)}}\prod_a d\varphi^a\,. \tag{1.5}$$

This definition makes sense if the right hand side of (1.5) is understood as a formal series of infinitely many independent weighted variables $C_{a_1,\dots,a_k}$, weight

of $C_{a_1,\dots,a_k}$ being k. In fact, the Gaussian integrals in the coefficients uniformly converge.

More precisely, putting $d := \dim \mathcal{P} = \operatorname{card} A$ and $D := \det(g_{ab})$, we have

$$\int_{\mathcal{P}} e^{\lambda^{-1} S_0(\varphi)} \prod_a d\varphi^a \ = \frac{(2\pi\lambda)^{d/2}}{D^{1/2}}.$$

Furthermore, put for a polynomial $F(\varphi) \in \mathbf{C}[\varphi]$

$$\langle F(\varphi) \rangle := \frac{\int_{\mathcal{P}} e^{\lambda^{-1} S_0(\varphi)} F(\varphi) \prod_a d\varphi^a}{\int_{\mathcal{P}} e^{\lambda^{-1} S_0(\varphi)} \prod_a d\varphi^a} \ .$$

Then we have the following

1.3.1. Wick's Lemma. *In the notations above, we have*

a) $\langle \varphi^{a_1} \dots \varphi^{a_n} \rangle = 0$ *for* $n \equiv 1 \bmod 2$.

b) $\langle \varphi^a \varphi^b \rangle = \lambda g^{ab}$.

c) $\langle \varphi^{a_1} \dots \varphi^{a_{2m}} \rangle = \lambda^m \sum g^{a_{i_1} a_{j_1}} \dots g^{a_{i_m} a_{j_m}}$ *where the summation is taken over all unordered partitions of* $\{1, \dots, 2m\}$ *into* m *unordered pairs* $\{i_1, j_1\}, \dots, \{i_m, j_m\}$ *(pairings.)*

1.4. Proof of Theorem 1.2. Let us now calculate the l.h.s. of (1.3). From the definition and (1.5) we get

$$\sum_{N=1}^{\infty} \frac{\lambda^{-N}}{N!} \sum_{k_1,\dots,k_N=1}^{\infty} \frac{1}{k_1! \dots k_N!} \sum_{a_j^{(i)} \in A, 1 \le j \le k_i} \prod_{i=1}^{N} C_{a_1^{(i)},\dots,a_{k_i}^{(i)}} \langle \prod_{i,j}^{N} \varphi^{a_j^{(i)}} \rangle \,. \tag{1.6}$$

Choose some $(N; k_1, \dots, k_N)$. A typical monomial of degree N in $C_\bullet$ in the decomposition of (1.6) will be

$$\lambda^{-N} \frac{1}{N!} \prod_{i=1}^{N} \frac{1}{k_i!} C_{a_1^{(i)},\dots,a_{k_i}^{(i)}} \langle \prod_{i=1}^{N} \varphi^{a_1^{(i)}} \dots \varphi^{a_{k_i}^{(i)}} \rangle \,. \tag{1.7}$$

It vanishes if $k_1 + \dots + k_N$ is odd. Otherwise, in view of Wick's Lemma (1.7) can be rewritten as

$$\lambda^{-N + \frac{1}{2}\sum k_i} \frac{1}{N!} \prod_{i=1}^{N} \frac{1}{k_i!} C_{a_1^{(i)},\dots,a_{k_i}^{(i)}} \left(\sum g^{a_{l_1}^{(i_1)} a_{m_1}^{(j_1)}} \dots g^{a_{l_r}^{(i_r)} a_{m_r}^{(j_r)}} \right) \tag{1.8}$$

where $r = \frac{1}{2}\sum k_i$ and the inner sum is taken over all pairings of the set of ordered pairs $F = F(N;\, k_1, \dots, k_N) = \cup_{i=1}^{N}\{(i,1), \dots, (i,k_i)\}$.

Construct now a family of graphs τ, corresponding to the monomials in g^{**} appearing in (1.8). They will have a common set of flags $F_\tau := F$, and a common set of vertices $V_\tau = \{1, \dots, N\}$, $\partial_\tau(i,l) = i$. Declare that two flags constitute halves of an edge, if these flags are paired as in the respective monomial in (1.8). If we color the flags of one such graph by the map $F_\tau \to A : (i,l) \mapsto a_l^{(i)}$, then we will get a monomial in $(C_\bullet, g^{**})$ appearing in the weight function (1.4).

It remains to perform some bookkeeping in order to identify the coefficients at this monomials appearing respectively in (1.6) and the r.h.s. of (1.3). Here is a sketch.

The graphs constructed above represent all isomorphism classes of graphs in our sense. In fact, a choice of $(N;\, k_1, \dots, k_N)$ determines the number of vertices of any valence, and the choice of a pairing determines which pairs of flags become edges ($N = 0$ produces the empty graph.) Moreover, a non–empty graph comes thus equipped with a total ordering of its vertices and all sets of flags belonging to one vertex. The sum over graphs does not take care of these orderings. The group $\operatorname{Aut}\tau$ effectively acts on the whole set of them consisting of $N!\,\prod_{i=1}^{N} k_i!$ elements. Summing over isomorphism classes, we may replace the numerical coefficient in (1.8) by $|\operatorname{Aut}\tau|^{-1}$.

Finally,

$$-N + \frac{1}{2}\sum_{i=1}^{N} k_i = -|V_\tau| + |E_\tau| = \chi(\tau).$$

This completes the proof.

There are several more useful identities in the framework of our toy model. We mention two of them; proofs can be found in [Ma3], IV.3.

1.5.1. Theorem. *We have*

$$\log \sum_{\tau\in\Gamma} \frac{\lambda^{-\chi(\tau)}}{|\operatorname{Aut}\tau|}\, w(\tau) = \sum_{\tau\in\Gamma_0} \frac{\lambda^{-\chi(\tau)}}{|\operatorname{Aut}\tau|}\, w(\tau) \tag{1.9}$$

where Γ_0 is a set of representatives of isomorphism classes of connected graphs.

1.6. Summation over trees and the stationary value of the classical action. It is well known that quantum path integral is expected to be dominated

by small fluctuations around stationary points of the classical action functional $S(\varphi)$. In our formal context, one can define a natural stationary point φ_0, and it turns out that $S(\varphi_0)$ is again a sum over graphs with non–empty set of edges, this time simply connected and connected that is, trees T.

We will again treat $C_{a_1,\dots,a_k}$ as independent graded formal variables over a ring containing g_{ab}, g^{ab} and $\mathbf{Q}$. Then all our sums make sense as formal series.

Put $C^a = \sum_{b\in A} g^{ab}C_b$ and denote by $N \subset R$ the ideal generated by $C_{a_1,\dots,a_k}$ for all $k \geq 2$.

1.7. Theorem. *a) The equations*

$$\forall a \in A,\ \frac{\partial S(\varphi)}{\partial \varphi^a} = 0 \tag{1.10}$$

admit the unique solution $\varphi_0 = \{\varphi_0^a\}$, $a \in A$, satisfying the condition

$$\varphi_0^a \equiv C^a \bmod N\,. \tag{1.11}$$

b) The series over trees

$$Z := \sum_{\tau\in T} \frac{\lambda^{-\chi(\tau)}}{|\mathrm{Aut}\,\tau|}\, w(\tau) \tag{1.12}$$

satisfies the differential equations

$$\frac{\partial Z}{\partial C_a} = \varphi_0^a,\ a \in A, \tag{1.13}$$

and is the respective critical value of $S(\varphi)$:

$$Z = S(\varphi_0)\,. \tag{1.14}$$

1.8. Quantum fields: general indications. Using the toy model as an inspiration for developing the respective perturbative formalism for more realistic quantum field theories, one meets many new problems. The simplest of them can be illustrated already in the case of scalar field theories.

Basically, in this case our combinatorial finite set of indices A is replaced by the set of points of space–time M, and a vector $\varphi = \{\varphi^a\}$ by a function $\varphi = \varphi(x)$,

$x \in M$. The right hand side of (1.3), suitably interpreted, becomes *a definition* of the path integral. When one devises this interpretation, summations over A are replaced by integrations over M. In particular, weights of Feyman's graphs (1.4) turn into multidimensional integrals, and these integrals generally turn out to be *divergent.*

The simplest divergences occur already in the interpretation of (the products of) the terms g^{ab}. When $S_0(\varphi)$ is given by the Lagrangian density of the free field on $\mathbf{R}^D$

$$L(\varphi) := \frac{1}{2}(\sum_{k=1}^{d} |\partial_k \varphi(x)|^2 + m^2 \varphi(x)^2),$$

the classical "stationary phase" equation is Klein–Gordon $(-\Delta + m^2)\varphi(x) = 0$, and passing to the Fourier transform in the momentum space, we easily get an integral representation for the relevant continuous analog of (g^{ab}), "free propagator":

$$G_0(x-y) = \int_{\mathbf{R}^D} \frac{e^{-ip\cdot(x-y)}}{p^2+m^2} \frac{d^D p}{(2\pi)^D}$$

For $D = 1$, it is a continuous function, but for $D > 1$, it is a distribution singular on the diagonal $x = y$. Trying to make sense of the product of terms g^{ab} corresponding, for example, to the graph with two vertices, connected by two edges, and supplied with one aditional flag at each vertex, we get a formal expression

$$\int d^D u \, d^D v \, G_0(x-y) \, G_0(u-v)^2 \, G_0(v-z)$$

which after Fourier transform in q, with fixed p, becomes

$$\int \frac{1}{q^2+m^2} \frac{1}{(p-q)^2+m^2} \frac{d^D q}{(2\pi)^D}. \tag{1.15}$$

This integral diverges for $D \geq 4$.

Therefore, such integrals must be suitably regularized.

The choice of a regularization method for individual integrals (weights $w(\tau)$), such as introducing an auxiliary parameter and then defining a minimal subtraction of divergent terms, must take into account the basic fact that the perturbation series contains *all graphs*, and if one say, subtracts the divergence of weight of some subgraph, this changes the weight of the total graph and its initial divergence.

Beautiful inductive combinatorial prescriptions for doing subtractions coherently were devised by physicists in the early years of path integration. Nowadays they are organized into a well–structured algebraic theory involving Hopf algebras.

Basically, comultiplication in this Hopf algebra reflects the ways a graph can be composed from its subgraphs, and weight function is translated into a character of this Hopf algebra. Subtraction of divergences is replaced by dividing out the polar part in the group of characters. The following sections of the article are dedicated to various implementations of this general scheme.

In conclusion, we remark that integrals of the type (1.15), when they absolutely converge, as well as the regularized values of more general Feynman integrals, are *periods* in the sense of Kontsevich–Zagier ([KoZa]), and therefore are closely related to (mixed) motives. Study of various classes of such integrals from this perspective is very active: see, in particular, [AlMar1–3], [Bl], [BlEsKr], and references therein.

2. From Feynman graphs to flowcharts and programming methods

In this section, we review several contexts in which graphs and their weights such as (1.4) appear without reference to path integrals. We stress their interpretation as flowcharts processing input data into output data, and briefly discuss two generalizations: *programming methods* and (generalized) *operads.* This requires introduction of more general notion of decorated graphs. We start with some preparatory remarks.

2.1. Weights of connected graphs and tensor networks. The expression (1.4) for a toy Feynman weight

$$w(\tau) := \sum_{u\colon F_\tau \to A} \prod_{e\in E_\tau} g^{u(\partial e)} \prod_{v\in V_\tau} C_{u(F_\tau(v))}\,. \tag{2.1}$$

can be slightly generalized and reinterpreted as a polylinear map

$$\otimes_{v\in V_\tau} S^{|F_\tau(v)|}(\mathcal{P}) \to k. \tag{2.2}$$

Namely, let the former finite–dimensional "path" vector space $\mathcal{P}$ be now defined over a field k and endowed with symmetric non–degenerate metric $g : S^2(\mathcal{P}) \to k$. Let $C_v \in S^{|F_\tau(v)|}(\mathcal{P})$ be a family of symmetric tensors indexed by vertices of τ.

Such a family can be interpreted as a *tensor network*: cf. [Zo]. The weight (2.1) is then the result of contraction of all these tensors along all pairs of indices

corresponding to (pairs of flags forming) edges of τ. More precisely, factors $g^{u(\partial e)}$ raise subscripts, and summation over u comes from writing tensors in a basis.

2.2. Variants and generalizations. (i) First, expressions (2.1) and (2.2) suggest to extend the notion of graph by allowing "freely hanging" flags (they are variously called leaves, tails, or legs). Then the basic constituents of any graph are "corollas".

By definition, a corolla is an one–vertex graph with several flags attached to it at one end. Supplying an additional information specifying which flags are paired to form halves of an edge, we can construct any graph from the corollas of all its vertices.

(ii) Second, graphs should be considered as objects of a category. We have already used a notion of isomorphism/automorphism of graphs which clearly involved only the combinatorial part of the structure.

Developing this idea further, we can introduce other classes of morphisms. One may, for example, interpret as the morphism a list of pairs of tails of a given graph (source) that must be glued together in order to produce edges out of a new graph (target). Another class of morphisms consists of contractions of a subset of edges.

Moreover, one is naturally led to the consideration of *decorated graphs*, for example, oriented ones. This is essential when graphs are interpreted as flowcharts. In fact, in (2.2) we could bypass orientation only because the metric made the space $\mathcal{P}$ self–dual.

In more general models, the set of flags is subdivided into two subsets: inputs (oriented towards the vertex) and outputs (oriented outwards). If an input f is decorated by a linear space, the output f' forming the other half of the edge must be decorated by the dual space, so that "contraction of indices" will still be possible.

Then morphisms must be compatible with decorations. The compatibility conditions are usually motivated by envisioned applications.

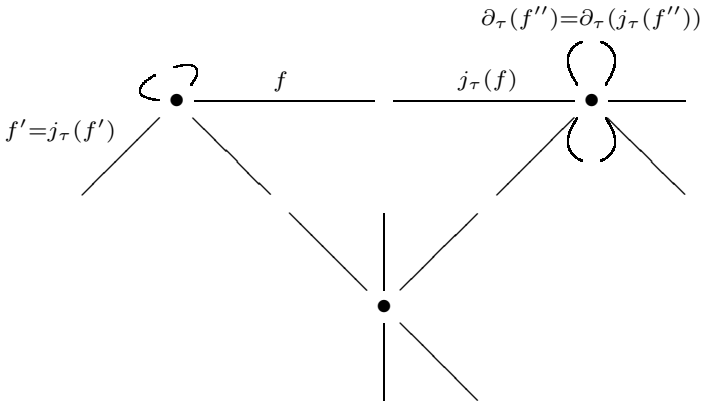
(iii) Third, having set upon a category of decorated graphs, one naturally wonders what are interesting functors defined on graphs. It turns out that all known types of *operads* can be treated as such functors, and operadic algebras become natural transformations of functors. In particular, graphs of the type that appeared in the toy model produce *cyclic operads*, and tensors $\{g^{ab}, C_{a_1,\dots,a_n}\}$ can be treated as structure constants of respective algebras.

A systematic development of this formalism is given in [BoMa]. See also [vdL], [Va1–2], [I1–2], [Bl] for various other versions. Pseudo–tensor categories of [BeDr] can be introduced in a similar way.

Below, I will sketch only those basic definitions that are relevant in the computation/renormalization context.

2.3. Combinatorial graphs and geometric graphs. We make a distinction between *combinatorial graphs* which are certain Bourbaki structures defined via discrete (in our case even finite) sets, and their *geometric realizations*, which are topological spaces. Combinatorial graphs form a category Gr. A general description of morphisms of Gr is given in [BoMa]. Some subclasses of morphisms will be described where they are needed.

A combinatorial graph τ consists of two sets F_τ (flags), V_τ (vertices) and two incidence relations. Each flag $f \in F_\tau$ is incident to exactly one vertex $v \in V_\tau$, its *boundary* which is $v = \partial_\tau(f)$, and the map $\partial_\tau : F_\tau \to V_\tau$ is a part of the data. Finally, some pairs of flags form "halves of an edge": this incidence relation is represented by an involution $j_\tau : F_\tau \to F_\tau$, $j_\tau^2 = \mathrm{id}$.



A geometric graph, geometric realization of τ, is the topological space $|\tau|$ which is obtained from the family of segments $\{[0,1/2]_f\}$ indexed by $f \in F_\tau$. We have to identify some groups of end–points. Namely, we glue together 0's of all segments, whose respective flags have one and the same boundary vertex v. Finally, we glue the end–points $1/2$ of each pair of segments indexed by two flags $f \neq f'$ such that $j_\tau(f) = f'$.

Pairs of flags $f \neq f'$ with $j_\tau(f) = f'$ are elements of the set of *edges* E_τ. Fixed points of j_τ are called *tails*, their set is denoted T_τ.

A graph τ is called *connected* (resp. *simply connected*, etc) iff its geometric realization is such. A connected and simply connected graph is a *tree*, a disjoint union of trees is a *forest.* A tree with one vertex is a *corolla.* Each vertex $v \in V_\tau$

defines its corolla $F_\tau(v)$ consisting of flags incident to v, v itself, induced ∂_τ and trivial j_τ.

We will not consider isolated vertices, so that our ∂_τ will be a surjection. However, we do allow *empty graphs.*

Finally, a few words about *morphisms.* Any morphism of combinatorial graphs $h : \tau \to \sigma$ that we will consider, will be *uniquely defined by two maps*:

$$h_V : V_\tau \to V_\sigma, \quad h^F : F_\sigma \to F_\tau$$

However, conditions, restricting allowed maps, will depend on the class of morphisms, and eventually on the decorations (cf. below). Composition is composition of maps.

In particular, h is *an isomorphism*, iff h_V, h^F are *bijections, identifying the incidence maps.*

Notice one peculiarity: h_V is covariant, whereas h^F is *contravariant.* This choice can be explained using the flowcharts intuition discussed below: a change of arguments produces the lift of functions *in reverse direction.*

2.4. Decorated graphs. Let $L = (L_F, L_V)$ be two sets: *labels of flags and vertices*, respectfully.

An L–decoration of the combinatorial graph τ consists of two maps $F_\tau \to L_F$, $V_\tau \to L_V$. Usually these maps are restricted by certain *compatibility with incidence relations* conditions that we will not try to axiomatize, and only illustrate on several basic examples.

L–decorated graphs for various specific sets of labels will also form a category. Any morphism h will be, as above, determined by h_V, h^F, but this time these maps will be restricted by certain *compatibility with decorations.*

An isomorphism of decorated graphs is simply an isomophism of underlying combinatorial graphs, preserving the decorations.

2.5. Orientation and flowcharts. Consider the set of labels $L_F := \{in, out\}$. A decoration $F_\tau \to \{in, out\}$ such that halves of any edge are oriented by different labels, is called *an orientation of* τ. On the geometric realization, a flag marked by *in* (resp. *out*) is oriented towards (resp. outwards) its vertex.

Tails of τ oriented *in* (resp. *out*) are called *(global) inputs* T^{in}_τ (resp. *(global) outputs* T^{out}_τ) of τ. Similarly, $F_\tau(v)$ is partitioned into inputs and outputs of the vertex v.

Consider an orientation of τ. Its edge is called *an oriented loop*, if both its halves belong to the same vertex. Otherwise an oriented edge starts at a source vertex and ends at a different target vertex.

More generally, a sequence of pairwise distinct edges $e_1, \dots, e_n$, is called *a simple path* of length n, if e_i and e_{i+1} have a common vertex, and the $n-1$ vertices obtained in this way are pairwise distinct. If moreover e_1 and e_n also have a common vertex distinct from the mentioned ones, this path is *a wheel* of length n. A loop is a wheel of length one.

A wheel in an oriented graph is called *oriented wheel*, if of each two flags of a wheel, sharing a common vertex, one is oriented *in*, and and another *out*.

Imagine now that we have a set of operations Op that can be performed on certain objects from a set Ob and produce other objects of this kind. Take (names of) this set of operations as the set of labels of vertices L_V. Then an oriented graph τ with vertices decorated by L_V can sometimes be considered as a flowchart describing a set of computations.

Here is the simplest case. Let τ be an oriented corolla, with vertex labeled by the name of an operation f which describes a map $Ob^a \to Ob^c$, where $a, c \in \mathbf{N}$. Call a (resp. c) the *arity* (resp. *co–arity*) of f. Require the following compatibility:

$$\operatorname{card} T_\tau^{in} = a, \quad \operatorname{card} T_\tau^{out} = c.$$

Enrich the decoration of inputs by a bijection $T_\tau^{in} \to \{1, \dots, a\}$ and that of outputs by a bijection $T_\tau^{in} \to \{1, \dots, c\}$. Then such a corolla can be considered as a replacement of the expression $f(x_1, \dots, x_a) = (y_1, \dots, y_c)$ with variable $x_i, y_j \in Ob$.

More general flowcharts τ are obtained, if we allow outputs of a set of operations to be taken as inputs of another set of operations. At the end we will get global outputs.

Notice that if we choose $Ob \times \{in, out\}$ as the label set for decorating flags L_F, then certain decorated graphs can be interpreted as concrete *instances* of a computation.

Three remarks are in order here.

(i) An oriented corolla with *empty* set of inputs, but non–empty set of outputs, can be used to formalize the notion of *oracle* in our context. In theoretical computer science, one is allowed to imagine an oracle providing one with a piece of information which did not come from any sensible computation.

(ii) One can also imagine a corolla with empty set of global outputs. But a more intuitive formalization of such a "device" is a corolla with one output which always

produces a specific "empty" object. Its vertex may also be decorated by a name of "empty" operation.

(iii) If our purported flowchart has oriented loops or wheels, we may be in trouble, because intuitively they require explicit inclusion of "time" in our calculation: the vertex of a loop must feed its output into its input – how many times? When this circling of information will stop?

Without adopting specific prescriptions, we better avoid oriented wheels in our flowcharts by using only directed graphs (but cf. the definitions in [Sc]).

An oriented graph τ is called *directed* if it satisfies the following condition:

($\bullet$) *On each connected component of the geometric realization $|\tau|$, one can define a continuous real valued function ("height") in such a way that moving in the direction of orientation along each flag decreases the value of this function.*

In particular, oriented trees and forests are always directed.

Of course, a directed graph admits infinitely many compatible height functions, but only a partial order induced by such a function on vertices, will be used below. However, if difference of heights models time of computation, then the function itself becomes an essential element of the structure.

2.6. Example: Dana Scott's flow diagrams. In [Sc], a class of (eventually infinite) decorated graphs is introduced called *flow diagrams.* It was explicitly designed to model computations with loops, possibly infinite ones, and it forms a special kind of category which Dana Scott treats as a lattice.

Here we give only basic definitions.

2.6.1. Labels. The set L_V (labels of vertices) has the following structure:

$$L_V = \mathcal{F} \coprod \mathcal{S} \coprod \{\bot, \top\}.$$

Here $\mathcal{F}$ are functional labels: names of *operators* transforming its input to its output. This set includes the name I of identical operator. Furthermore, $\mathcal{S}$ are names of *switches*: a switch tests its input and redirects it to *one of its outputs,* depending on the results. Finally, symbols $\bot$ and $\top$ are used to generate some "improper" diagrams. In particular, a vertex labeled by $\bot$ describes "the smallest" flow diagram, whereas $\top$ corresponds to the "largest" one.

2.6.2. Graphs. The first subclass of decorated graphs qualifying as Scott's flow diagrams are *oriented trees.* The orientation (describing the direction of the information flow) goes from one tail (input) to many tails (outputs). Each vertex

has one input and either one output, or two outputs. If it has one output, it must be labeled by a symbol from $\mathcal{F}$ or else $\top, \bot$. If it has two outputs, it must be labeled by a switch from $\mathcal{S}$.

Clearly, when such a finite tree processes a concrete input, the output will show *only at one of the output tails*, because of the semantics of switching. Hence we might imagine a special vertex accepting many inputs and producing the one which is "not silent". A price for it will be that our graphs will not be trees anymore. They will still remain *directed graphs.*

We now pass to another example.

2.7. Example: Yanofsky's algorithms. Recursive functions are certain (partial) maps $f : (\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$. The number of arguments a is called *arity* of f, c its *coarity*. The definition domain of f is denoted $D(f)$.

Basic recursive functions are: successor $x \mapsto x + 1$, projections, and constant functions. The world of recursive functions is the smallest set of functions containing all basic functions and closed with respect to four *elementary operations*: composition (or substitution), bracketing, recursion and the μ–operator. The first three, applied to everywhere defined recursive functions, produce an everywhere defined function as well. The μ–operator generally creates only a partial function. For more details, see e. g. [Ma1], V.2, and below.

Recursive functions can be introduced by their *descriptions* in a formal or programming language: essentially, a sequence of functions whose first term is (the name of) a basic function, and the name of a new function is supplied by the name and arguments of elementary operation, applied to some formerly constructed functions. If the operator μ is not used in a description, the resulting function is called *primitive recursive.*

In [Ya], N. Yanofsky suggested using graphs (modulo appropriate equivalence relation) as a replacement of the descriptions above, at least for primitive recursive functions. I will present some of his constructions below to illustrate our general approach.

2.8. Decorated graphs $Prim$**.** Elements of $Prim$ are disjoint unions of trees τ, in which each vertex is the boundary of at least two flags. Moreover, τ must be endowed with an *admissible decoration.* The latter consists of the following data. They can be chosen independently on each connected component so that in the following discussion we speak about trees, if we did not explicitly mentioned the general case.

(a) *A marked tail*, which is called *root*, or *the (global) output* of τ. Its vertex is called the *root vertex*. The remaining tails are called *(global) inputs* of τ. Global inputs form a set $F_\tau^{in} \subset F_\tau$, and we consider the global output as an one–element subset $F_\tau^{out} \subset F_\tau$.

A choice of root determines (and is equivalent to) the choice of a specific *orientation*: a map $F_\tau \to \{in, out\}$. Namely, in each shortest path (sequence of flags) from a global input to the root, assign *out* to the flag that leaves its vertex, and *in* to the flag that enters it. This defines the partition of all flags into two subsets: (local) inputs and outputs.

We will say that τ with such a decoration is *an oriented tree.* We repeat that by definition, each oriented tree must have exactly one global output, and at least one global input.

(b) All corollas of an oriented tree are also oriented trees. The next part of decoration is a choice of *total order on the set of inputs of each corolla of* τ*,* and, if τ is not connected, a choice of total order on the set of its connected component.

(c) A map *arity/coarity*: $F_\tau \to \mathbf{N} : f \mapsto (a(f), c(f))$. If two flags are halves of an edge, they must be assigned the same arity/coarity.

(d) A map $op : V_\tau \to \{\mathbf{c}, \mathbf{b}, \mathbf{r}\}$. The value $op\,(v)$ assigned to a vertex is called the respective *operator*: $\mathbf{c}, \mathbf{b}, \mathbf{r}$ stand respectively for *composition, bracketing, recursion.*

(e) A map $in : F_\tau^{in} \to \{basic\ recursive\ functions\}$ such that for each $i \in F_\tau^{in}$, $in(i)$ is a basic function of arity $a(i)$ and coarity $c(i)$.

All these data must be *compatible*. A part of compatibility conditions was already included in the description. We will now formally introduce the remaining set, and simultaneously explain an interpretation of graphs in $Prim$ (without decoration 2.8 (e)) as operations acting on families of input functions.

2.9. Objects of $Prim$ as flowcharts. Given an oriented tree τ with a decoration as above, we interpret the whole τ as a symbol of an *operation* $Op(\tau)$ that can be performed over families of functions, indexed by global inputs of τ.

More precisely, let $f = \{f_i \,|\, i \in F_\tau^{in}\}$ be a family of functions (or even partial functions) such that $f_i : (\mathbf{Z}^+)^{a(i)} \to (\mathbf{Z}^+)^{c(i)}$. Then

$$Op\,(\tau)(f) = g : \ (\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$$

where (a, c) is the arity/coarity of the root.

The prescription for getting g, given f, runs as follows.

One vertex case. Let τ be a corolla whose vertex is decorated by $\mathbf{c}, \mathbf{b}$, or $\mathbf{r}$. Then g is obtained by applying to the family $\{f_i\}$, $i \in F_\tau^{in}$, the respective elementary operation: composition, bracketing or recursion. This requires the following compatibilities which vary depending on the label of the vertex.

(a) *Composition.* Let $(a_1, c_1), \ldots, (a_r, c_r)$ be the family of arities/coarities of inputs ordered as the respective flags. They must then be constrained by the condition $c_1 = a_2, \ldots, c_{r-1} = a_r$, and the arity/coarity of the output must be (a_1, c_r).

For a general τ, these compatibility conditions must be satisfied for all corollas τ_v of all vertices decorated by $\mathbf{c}$.

In the flowchart interpretation, such a corolla transforms an input family $(f_1, \ldots, f_r)$, $f_i : (\mathbf{Z}^+)^{a_i} \to (\mathbf{Z}^+)^{c_i}$, into the composition $f_r \circ f_{r-1} \circ \ldots f_1$.

(b) *Bracket.* With the same notation as in (a), the compatibility condition reads $a_\bullet := a_1 = \cdots = a_r$, and the arity/coarity of the output must be $(a_\bullet, c_1 + \ldots c_r)$.

For a general τ, these compatibility conditions must be satisfied for all corollas τ_v of all vertices decorated by $\mathbf{b}$ and respective orderings.

In the flowchart interpretation, such a corolla transforms an input family $(f_1, \ldots, f_r)$, $f_i : (\mathbf{Z}^+)^{a_\bullet} \to (\mathbf{Z}^+)^{c_i}$, into the map

$$\langle f_1, \ldots, f_r \rangle : (\mathbf{Z}^+)^{a_\bullet} \to (\mathbf{Z}^+)^{c_1 + \cdots + c_r}.$$

It was called juxtaposition in [Ma1], V.2.3 (b).

(c) *Recursion.* If a vertex is decorated by $\mathbf{r}$, it must have exactly *two local inputs.* If the arity/coarity of the first one (in their structure order) is (a, c), for the second one it must be $(a + c, c)$, and for the local output it must be $(a + 1, c)$: this is our compatibility condition.

In the flowchart interpretation, such a vertex takes as input two arbitrary maps $f_1 : (\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$, $f_2 : (\mathbf{Z}^+)^{a+c} \to (\mathbf{Z}^+)^c$ and produces the output

$$g : (\mathbf{Z}^+)^{a+1} \to (\mathbf{Z}^+)^c$$

defined recursively as

$$g(x, 1) := f_1(x),$$

$$g(x, k + 1) := f_2(x, g(x, k))$$

for each $x \in (\mathbf{Z}^+)^a$, $k \in \mathbf{Z}^+$.

This form of recursion is more restrictive than the one which is often used: it does not allow f_2 to depend explicitly on the recursion parameter k. However, R. M. Robinson has proved in [Ro] that it suffices to use it in order to get all primitive recursive functions, if an extension of the list of basic functions is allowed. Afterwards, M. D. Gladstone in [Gl] has shown that such an extension is unnecessary.

General case. At first, consider a connected graph τ. Assume that it has ≥ 2 vertices. We define the operation $Op(\tau)$ by induction on the number of vertices.

Namely, for a vertex v which is the boundary of a global input, consider the subfamily $f_v := \{f_i \,|\, \partial_\tau(i) = v\}$. Denoting by τ_v the corolla of v (an *in–corolla*), calculate $g_v := Op(\tau_v)(f_v)$ as specified above.

One can check that this prescription produces the result independent on arbitrary choices.

Now, consider the maximal decorated subgraph τ^0 of τ, whose flags and vertices do not belong to this *in*–corolla. Its global inputs consist of all global inputs of τ not adjacent to v, and $j_\tau(r)$, where r is the root of our corolla. Decoration of τ^0 is the restriction of that of τ; global inputs of τ retain also their input functions f_i. Decorate the input $j_\tau(r)$ by g_v and put

$$Op(\tau)(\{f_i\}) := Op(\tau^0)(\{f_i, g_v \,|\, \partial(i) \neq v\}).$$

The right hand side is defined due to the inductive assumption.

Finally, if τ is the disjoint union of connected components $\coprod_{a\in A} \tau_a$, we put

$$Op(\coprod_{a\in A} \tau_a) := \times_{a\in A} Op(\tau_a)$$

in the sense that $Op(\tau)$ acts on the family, naturally indexed by A, of (families of) global inputs of connected components, and produces the family of outputs, as well indexed naturally by A.

As we implied in the previous discussion, we can apply $Op(\tau)$ to families, consisting non–necessarily of basic, or even recursive, functions.

But if we want to define programming methods based upon $Prim$, then we must decorate global inputs by some basic functions, and interpret the resulting decorated tree as as a program producing one concrete recursive function.

Here the choice becomes ambiguous: we may change the list of basic functions, and we may allow the application of $\mathbf{c}, \mathbf{b}, \mathbf{r}$ to some restricted class of subfamilies, getting the more general cases from trees larger than corollas.

For $\mathbf{c}$ and $\mathbf{b}$, we allowed arbitrary natural families, implicitly using associativity of intended interpretations. Yanofsky allows only two inputs. For $\mathbf{r}$, we essentially adhered to the choice made by Yanofsky.

2.10. *Prim* **as a world of programming methods.** We may consider $(\mathbf{Z}^+)^a$, $a \geq 0$, as objects of a category $\mathcal{C}$ whose morphisms are recursive functions. There are many other enumerable sets S endowed with a computable bijection (enumeration) $\mathbf{Z}^+ \to S$. For example, consider some finite Bourbaki structure, such as a finite group, or a finite graph. One can easily construct a set of representatives of isomorphism classes of such a structure, one from each class, together with its enumeration.

Given such a set $\mathcal{S}$, we can add it to $\mathcal{C}$ as a new object, with evident morphisms. Such objects may be called (infinite) *constructive worlds.* For a more detailed discussion, see [Ma2] and the 2nd Edition of [Ma1]. The new category will be equivalent to $\mathcal{C}$. We may always assume it to be monoidal with respect to the direct product $\times$.

Now, the morphism sets $\mathcal{C}((\mathbf{Z}^+)^a, (\mathbf{Z}^+)^c)$ that is, recursive functions of a fixed arity and coarity, *do not form a constructive world*: only their descriptions do. Whenever a constructive world of descriptions is chosen, its definition must be completed by a family of *evaluation maps* and *composition maps* which must be computable, that is, morphisms in the (extended) category $\mathcal{C}$. When this is done, we will elevate the constructive world of descriptions to that of *programming methods.*

In order to illustrate these general considerations, define $P(a, c)$ as the subset of $Prim$, consisting of (isomorphism classes of) graphs whose outputs (roots of connected components) have the total arity/coarity (a, c).

The evaluation morphism in $\mathcal{C}$

$$\mathrm{ev}_{P(a,c)} : P(a, c) \times (\mathbf{Z}^+)^a \to (\mathbf{Z}^+)^c$$

we have already essentially described. Namely,

$$\mathrm{ev}_{P(m,n)}(\tau, (x_1, \ldots, x_m)) := f_\tau(x_1, \ldots, x_m)$$

where f_τ is the total output of the flowchart τ which we formerly denoted $Op(\tau)$, applied to the input decorations of τ.

A computable multiple composition morphism

$$\mathrm{comp}: \ P(m_{r-1}, m_r) \times \cdots \times P(m_2, m_3) \times P(m_1, m_2) \to P(m_1, m_r)$$

can be constructed as follows. For simplicity, we will only describe the composite $\mathrm{comp}\,(\tau_r, \tau_{r-1}, \ldots, \tau_1)$ for an r–tuple of decorated trees $\tau_1, \tau_2, \ldots, \tau_r$.

Consider a corolla with vertex decorated by $\mathbf{c}$, r inputs decorated by the arities $(m_1, m_2), \ldots, (m_{r-1}, m_r)$, and an output decorated by (m_1, m_r). Graft inputs of ths corolla to the roots of $\tau_1, \ldots, \tau_r$ respectively. The resulting tree represents the composition.

Of course, on the combinatorial level, we will have to make a stupid choice of some "concrete" vertex and flags of this corolla, but the result will be unique up to unique isomorphism identical on the component trees τ_i.

However, if we iterate partial compositions which on the level of maps correspond, say, to $h \circ g \circ f$, $(h \circ g) \circ f$ and $h \circ (g \circ f)$ respectively, we will get three different decorated trees, say $\sigma_{123}, \sigma_{12,3}, \sigma_{1,23}$.

On the combinatorial/geometric level these trees are interconnected by two contraction morphisms: $\sigma_{12,3} \to \sigma_{123}$ and $\sigma_{1,23} \to \sigma_{123}$ which contract the edges entering to the root vertices, whose ends are marked by $\mathbf{c}$. One can simply declare that such contractions generate an equivalence relation on the elements of $Prim$, and that algorithms encoded by $Prim$ are actually such (or even bigger) equivalence classes rather than isomorphism classes of the decorated trees.

However, since we work in a categorical context, a better way to proceed is to organize $Prim$ into a constructive category, and then to *localize* it with respect to those morphisms $\tau \to \sigma$ that produce a natural identification $Op\,(\tau)$ and $Op\,(\sigma)$.

Recall that the *localization of a category $\mathcal{B}$ with respect to a set of its morphisms S* is a functor $L: \mathcal{B} \to \mathcal{B}[S^{-1}]$ which makes all morphisms in S invertible and which is the initial object among all functors with this property.

Here is a simple version of this construction.

2.11. Definition – Claim. *Consider the category Pr whose set of objects is the set $Prim$, and morphisms are compositions of the following maps of decorated graphs:*

(i) Isomorphisms.

(ii) Contractions of subtrees of the following type: all vertices of such a subtree are decorated by $\mathbf{c}$. After the contraction, the resulting vertex must be marked by $\mathbf{c}$. The remaining decorations do not change.

(iii) Contractions of subtrees, whose all vertices are decorated by **b**. *After the contraction, the resulting vertex must be marked by* **b**. *The remaining decorations do not change.*

Denote by P the localization of Pr with respect to all morphisms. It has the natural structure of a category of programming methods for which composition and bracket operations become associative.

One can similarly accommodate more sophisticated equivalence relations between decorated trees, studied by Yanofsky.

To this end one can extend the category Pr by some extra morphisms, and then localize with respect to them as well.

3. Bialgebras and Hopf algebras related to computation

Graphs which in the Feynman formalism enumerate terms of a perturbation series, for the purposes of renormalization are organized into another structure: they serve as formal generators of an algebra endowed with a diagonal map, that is, a bialgebra. The nature of this diagonal map Δ becomes even more transparent when graphs are treated as flowcharts: Δ simply sums the different ways of decomposing a flowchart into a "sub–" and a "quotient–" component. Thus, a general context for introducing the relevant bialgebras is that of morphisms in categories, enriched categories, and programming methods.

In this section, we give a sample of such constructions. The most basic and elementary example is that of finite categories.

3.1. Proposition. *Let C be a finite category. Let k be a base field (or unital commutative ring). Denote by $B := B_C$ the symmetric algebra of the linear space freely generated by all morphisms in C. Define the diagonal morphism of algebras $\Delta : B \to B \otimes B$ on generators:*

$$\Delta(f) := \sum_{g,h \,|\, gh=f} h \otimes g \tag{3.1}$$

Then Δ is coassociative so that B becomes a bialgebra with unit and counit.

Proof. We have

$$(\Delta \otimes \mathrm{id}) \circ \Delta(f) = \sum_{k,l \,|\, kl=h} \left(\sum_{g,h \,|\, gh=f} l \otimes k \otimes g \right), \tag{3.2}$$

$$(\mathrm{id}\otimes\Delta)\circ\Delta(f)=\sum_{d,e\,|de=g}\left(\sum_{g,h\,|gh=f}h\otimes e\otimes d\right). \tag{3.3}$$

It remains to check that each term at the right hand side of (3.2) appears exactly once in the rhs of (3.3), and vice versa. In fact, both sets are in a natural bijection with triple decompostions $f=f_1f_2f_3$. This ends the proof.

To put it simply, coassociativity of Δ is a formal consequence of associativity of composition of morphisms.

Composition of programming methods often is not strictly associative. Moreover, relevant categories are not always finite. Even Boolean circuits computing maps between finite sets of a restricted cardinality do not form a finite set.

The latter problem can be alleviated by an additional structure that is always relevant for programming methods: some natural $\mathbf{Z}^+$–measure of *program size* and/or *computation time*, which is more or less additive with respect to decompositions of programming methods (as opposed to the decomposition of the respective functions) $F=GH$. Such an additivity ensures that the right hand side of formulas similar to (3.1) consists of a finite number of terms. Associativity/coassociativity then is achieved by appropriate modifications of constructive objects associated with programming methods.

The following example, based on graphs, displays some relevant constructions. It is a version of Kreimer–Connes Hopf algebra in quantum field theory which can be re–interpreted as a composition bialgebra (3.1) of programming methods as soon as an appropriate category of decorated graphs is chosen.

3.2. Definition. *Let τ be an oriented graph. Call a proper cut C of τ any partition of V_τ into a disjoint union of two non–empty subsets V_τ^C (upper vertices) and $V_{\tau,C}$ (lower vertices) satisfying the following conditions:*

(i) For each oriented wheel in τ, all its vertices belong either to V_τ^C, or to $V_{\tau,C}$.

(ii) If an edge e connects a vertex $v_1\in V_\tau^C$ to $v_2\in V_{\tau,C}$, then it is oriented from v_1 to v_2 ("information flows down").

(iii) There are also two improper cuts: the upper improper cut is the partition $V_\tau^C=\emptyset$, $V_{\tau,C}=V_\tau$, whereas the lower one is the partition $V_\tau^C=V_\tau$, $V_{\tau,C}=\emptyset$.

Having chosen a cut C, we may define two graphs: τ^C (upper part of τ wrt C) and τ_C (lower part wrt C) by the following conditions.

Vertices of τ^C (resp. τ_C) are V_τ^C (resp. $V_{\tau,C}$). Flags of τ^C (resp. τ_C) are all flags of τ incident a vertex of τ^C (resp. τ_C). Edges of τ^C (resp. τ_C) are all edges of τ whose both boundaries belong to τ^C (resp. τ_C).

Finally, all orientations remain the same as they were on τ, and if τ was not only oriented by additionally decorated, or labels remain the same.

The term "cut" is motivated by the following simple observation. Given (τ, C), we may define one more graph σ, with $(F_\sigma, V_\sigma, \partial_\sigma)$= $(F_\tau, V_\tau, \partial_\tau)$. Furthermore, all edges remain the same except for those that lead from a vertex in τ^C to a vertex in τ_C: their halves become some global outputs (resp. inputs) of τ^C (resp. τ_C).

Thus, we have

$$\sigma = \tau^C \coprod \tau_C$$

where $\coprod$ is the disjoint union.

In other words, we get $|\sigma|$ by cutting all edges leading from $|\tau^C|$ to $|\tau_C|$ at their midpoints. This implies that C can be also identified with the respective subset of edges in $|\tau|$: this is the definition of a cut often accepted in physics literature.

3.3. Bialgebras of decorated graphs. Fix a set of labels $L = (L_F, L_V)$. Assume that $L_F = L_F^0 \times \{in, out\}$, When we speak about orientation of an L–decorated graph, we always mean the one that is obtained by forgetting labels from L_F^0. The isomorphism class of a decorated graph τ is denoted $[\tau]$.

3.3.1. Definition. *A set Fl ("flowcharts") of L–decorated graphs is called admissible, if the following conditions are satisfied:*

(i) Each connected component of a graph in Fl belongs to Fl. Each disjoint union of a family of graphs from Fl belongs to Fl. Empty graph $\emptyset$ is in Fl.

(ii) For each $\tau \in Fl$ and each cut C of τ, τ^C and τ_C belong to Fl.

Let now Fl be an admissible set of graphs, and k a commutative ring. Denote by $H = H_{Fl}$ the k–linear span of isomorphism classes of graphs in Fl: the k–module of formal finite linear combinations $\{ \sum_{\tau \in Fl} a_{[\tau]}[\tau] \}$.

Define two linear maps

$$m : H \otimes H \to H, \quad \Delta : H \to H \otimes H$$

by the following formulas extended by k–linearity:

$$m([\sigma] \otimes [\tau]) := [\sigma \coprod \tau], \tag{3.4}$$

$$\Delta([\tau]) := \sum_C [\tau^C] \otimes [\tau_C], \tag{3.5}$$

where the sum is taken over all cuts of τ.

3.3.2. Proposition. *(i) m defines on H the structure of a commutative k–algebra with unit $[\emptyset]$. Denote the respective ring homomorphism*

$$\eta : k \to H,\ 1_k \mapsto [\emptyset]\,.$$

(ii) Δ is a coassociative comultiplication on H, with counit

$$\varepsilon : H \to k,\ \sum_{\tau \in Fl} a_{[\tau]}[\tau] \mapsto a_{[\emptyset]} \tag{3.6}$$

(iii) $(H, m, \Delta, \varepsilon, \eta)$ is a commutative bialgebra with unit and counit.

Proof. (i) The first statement is straightforward. It is worth mentioning that (H, m, η) is in fact the symmetric algebra freely generated by the set of isomorphism classes Fl_{con} of connected non–empty graphs in Fl: $[\coprod_i \tau_i]$ corresponds to $\prod_i [\tau_i]$.

(ii) The least obvious in this statement is the coassociativity of Δ. Omitting for brevity square brackets indicating isomorphism classes at the rhs, we can write:

$$(\Delta \otimes id) \circ \Delta([\tau]) = \sum_C \sum_{C'} (\tau^C)^{C'} \otimes (\tau^C)_{C'} \otimes \tau_C, \tag{3.7}$$

$$(id \otimes \Delta) \circ \Delta([\tau]) = \sum_C \sum_{C''} \tau^C \otimes (\tau_C)^{C''} \otimes (\tau_C)_{C''}, \tag{3.8}$$

where C runs over cuts of τ, C' runs over cuts of τ^C, and C'' runs over cuts of τ_C.

We want to establish a bijection between the sets of tensor monomials in the rhs of both formulas.

To this end, consider triple partitions of V_τ,

$$V_\tau = V_1 \coprod V_2 \coprod V_3$$

satisfying the conditions similar to those in Definition 3.2:

(a) For each oriented wheel in τ, all its vertices belong to one of the sets V_i.

(b) If an edge e connects a vertex $v_1 \in V_i$ to $v_2 \in V_j$, $i < j$, then it is oriented from v_1 to v_2.

From such a triple partition, we can produce two double partitions: $(V_1 \coprod V_2) \coprod V_3$ and $V_1 \coprod (V_2 \coprod V_3)$. Both of them satisfy conditions of Definition 3.2. Hence they define two cuts of τ, say C_{12} and C_{23}.

Moreover, $V_1 \coprod V_2$ defines a cut of $\tau^{C_{12}}$, say C'_{12}, and hence a term in the rhs of (3.7). Similarly, $V_2 \coprod V_3$ defines a cut of $\tau_{C_{23}}$, say C''_{23}, and hence a term in the rhs of (3.8).

We claim that this construction establishes a bijection between the respective terms. The reasoning is somewhat cumbersome, but straightforward, and we leave it to the reader.

The coidentity axiom reads

$$(\varepsilon \otimes id) \circ \Delta([\tau]) = (id \otimes \varepsilon) \circ \Delta([\tau]) = [\tau]. \tag{3.9}$$

To check it, we refer to (3.5) and (3.6): only two terms in (3.5), corresponding to improper cuts, can contribute to (3.9). One gives the identity for the left counit, another for the right one.

(iii) It remains to check that Δ and ε are algebra homomorphisms. For ε, this follows from (3.4) and (3.5). For Δ, this follows from the fact that the a cut C of $\sigma \coprod \tau$ is the same as a pair of cuts (C_σ, C_τ) of σ, τ respectively, so that $(\sigma \coprod \tau)^C = \sigma^{C_\sigma} \coprod \tau^{C_\tau}$ and similarly for lower parts.

3.4. Hopf algebra of decorated graphs. In order to construct an antipode on the bialgebra $H = H_{Fl}$ we will show that one can introduce on H_{Fl} a grading by $\mathbf{N}$ turning it into a *connected graded bialgebra* in the sense of [E–FMan], 2.1. Then the existence of an antipode (and an explicit construction of it) is provided by the Corollary 1 in 2.3 of [E–FMan].

There are two kinds of natural gradings. One can simply define

$$H_n := \textit{ the } k-\textit{submodule of } H \textit{ spanned by } [\tau] \textit{ in } Fl \textit{ with } |F_\tau| = n.$$

One can also introduce a weight function on the set of labels in 2.4: $|\cdot| : L \to \mathbf{N}$ and put

$$H_n := \textit{ the } k-\textit{submodule of } H$$
$$\textit{spanned by } [\tau] \textit{ in } Fl \textit{ with } n = \sum_{f \in F_\tau} (|l(f)| + 1) + \sum_{v \in V_\tau} |l(v)|\,,$$

where $l : V_\tau \coprod F_\tau \to L$ is the structure decoration of τ.

From the definitions, it follows that for either choice we have

$$m(H_p \otimes H_q) \subset H_{p+q}, \quad \Delta(H_n) \subset \oplus_{p+q=n} H_p \otimes H_q,$$

and moreover, $H_0 = k[\emptyset]$ is one–dimensional, so that H is connected.

3.5. Hopf algebras from quantum computation: a review. The last subsections of this part are dedicated to one more class of constructions that are directly related to some ideas in the quantum computation project.

One standard model of quantum computation starts with a classical Boolean circuit B which computes a map $f : X \to X$, where X is a finite set of Boolean words, say, of length n. After replacing bits with qubits, and X with the 2^n–dimensional Hilbert space H spanned by the ortho–basis of binary words, we have to calculate the linear operator $U_f : H \to H$, linearization of f. Only unitary operators U_f can be physically implemented. Clearly, U_f are unitary only for bijective f; moreover, they must be calculated by "reversible" Boolean circuits.

On the other hand, interesting f are only rarely permutations. For example, in search problems f is the characteristic function of a subset $X_0 \subset X$ ("a needle in a haystack").

There is a well–known trick, allowing one to transform any Boolean circuit B_f calculating f into another Boolean circuit B_F of comparable length consisting only of reversible gates and calculating a bijection F of another finite set, such that information about f is easily read off from the corresponding information about F (for more details, cf. [Ma2], 3.2).

If we now focus on permutations of X, there naturally arise two Hopf algebras related to them: *group algebra of permutations* and a dual Hopf algebra. For infinite X, there are several versions of Hopf algebras associated to the group of unitary operators $H_X \to H_X$.

Below we reproduce the combinatorial part of these constructions, having in mind applications to quantum computations of *recursive* functions f, say, $\mathbf{Z}^+ \to \mathbf{Z}^+$. An additional complication arises here: f can be only partial, hence before processing it into a permutation, we must make it everywhere defined.

Before proceeding to details, notice a change of perspective. Working with $Prim$ as flowcharts in 2.9, and by extension in 3.3, we considered *programming methods*, whose inputs and outputs were functions. Here we work with what earlier was

called *instances of computation:* inputs and outputs are now arguments/values of a function to be computed.

3.6. Reduction of total maps to bijections. Consider a set X and a class $\mathcal{F}$ of everywhere defined maps $f : X \to X$.

In order to reduce $\mathcal{F}$ to permutations, introduce on X a structure of, say, abelian group with composition law denoted $+$. Produce from f the map

$$\tilde{f} : X^2 \to X^2,\ \tilde{f}(x, y) := (x + f(y), y). \tag{3.10}$$

This is a bijection: $\tilde{f}^{-1}$ maps (x', y') to $(x' - f(y'), y')$. Knowing $\tilde{f}$, we can compute f: take the first coordinate of $\tilde{f}(0, y)$.

If X is endowed with a natural enumeration, one should choose an (easily) computable group law $+ : X^2 \to X$, and thus reduce the computation of f to that of $\tilde{f}$ using a (hopefully manageable) additional amount of memory and time. In turn, if f is computable, $\tilde{f}$ will be as well. Identical permutations and composition of two computable permutations are computable.

3.7. Reduction of partial maps to total maps. First recall how one composes partial maps.

Formally, a partial map from a set X to a set Y is a pair $(\varphi, D(\varphi))$ where $D(\varphi)$ is a subset of X (possibly empty), and $\varphi : D(\varphi) \to Y$ is an actual map. We put $\operatorname{Im}\varphi := \varphi(D(\varphi))$. Denote $Par\,(X, Y)$ the set of partial maps. The composition $Par\,(Y, Z) \times Par\,(X, Y) \to Par\,(X, Z)$ is defined as

$$(\chi, D(\chi)) \circ (\varphi, D(\varphi)) := (\chi \circ \varphi, \varphi^{-1}(D(\chi) \cap \operatorname{Im}\varphi)\,).$$

One easily sees that in this way we get a category, say $ParSets$.

Notice that each set of morphisms $Par\,(X, Y)$ is *pointed*, in the sense that it has a canonical element, "empty map", say, $\emptyset_{X,Y}$. Its composition with any other morphism is again the respective empty map.

This last remark motivates the consideration of another category: that of *pointed sets* $PSets$. An object of $PSets$ is a pair $(X, *_X)$ where $*_X \in X$ (so that X cannot be empty). A morphism $(X, *_X) \to (Y, *_Y)$ is an everywhere defined map $f : X \to Y$ such that $f(*_X) = *_Y$. The composition is evident.

Deleting marked points, we get a functor $PSets \to ParSets$:

$$X \mapsto X^\circ := X \setminus \{*_X\}, \quad f \mapsto f^\circ := (\varphi, D(\varphi)), \tag{3.11}$$

where, for $f : X \to Y$, $D(\varphi)$ is defined as $f^{-1}(Y^\circ)$ and φ as the restriction of f to $D(\varphi)$.

This construction is obviously invertible in the sense that there exists a quasi–inverse functor $ParSets \to PSets$. It can be constructed by formally adding an extra marked point $*_X$ to each object X in $ParSets$, and extending each partial map $(\varphi, D(\varphi))$ from X to Y by sending $X \setminus D(\varphi)$ to $*_Y$.

Let us return now to the situation where X, Y are endowed with computable numberings, and restrict ourselves to semi–computable partial functions φ. Then the passage from φ to an everywhere defined (total) function $f : X \cup \{*_X\} \to Y \cup \{*_Y\}$ involves a simple extension of φ: we put

$$f(x) = \varphi(x) \text{ for all } x \in D(\varphi),\ f(x) = *_Y \text{ otherwise.} \tag{3.12}$$

Of course, $Y \cup \{*_Y\}$ is endowed with obvious computable numberings compatible with that of Y, say, one can simply augment by 1 the initial numbering of Y and put $*_Y$ at the first place. But from the viewpoint of computability $*_Y$ looks rather as "infinite" , or "transfinite" element: if $D(\varphi)$ is only enumerable but not decidable, a Turing machine trying to calculate $f(x)$ for $x \in D(\varphi)$ might never stop. For the same reason, f as a total function might become uncomputable.

Nevertheless, we can apply to f the trick (3.10) and get a permutation $\tilde{f}$ of $(X \cup \{*_X\})^2$ which will be uncomputable outside $(X \cup \{*_X\}) \times (D(\varphi) \cup \{*_X\})$. Choosing a computable structure of abelian group on $X \cup \{*_X\}$ we will have to treat $*_X$ as an ordinary element. Choosing it to be zero, we get the following nice statement.

3.8. Proposition. *Let $\varphi : X \to X$ be a partial recursive function. Construct its extension as above $f : X \cup \{*_X\} \to X \cup \{*_X\}$. Choose a computable (general recursive) structure of additive group on $X \cup \{*_X\}$ with zero $*_X$.*

Denote by $\tilde{f}$ the permutation of $Z := (X \cup \{*_X\})^2$ produced from f by the analog of formula (3.10) in this context. Then $\tilde{f}$ is a permutation with the following properties:

(i) $\tilde{f}$ is an extension of the partial recursive function $g : Z \to Z$ with the definition domain

$$D(g) := (X \cup \{*_X\}) \times (D(\varphi) \cup \{*_X\}). \tag{3.13}$$

(ii) $\tilde{f}$ induces a permutation of $D(g)$ with unique fixed point $(*_X, *_X)$. The complement of $D(g)$ consists of all remaining fixed points of $\tilde{f}$.

Proof. By definition, we have

$$x \in D(\varphi) \implies f(x) = \varphi(x),$$

$$x \in X \cup \{*_X\} \setminus D(\varphi) \implies f(x) = *_X.$$

Therefore, denoting by x, y variable elements of $X \cup \{*_X\}$, we have:

$$\tilde{f}(x, y) = (x + \varphi(y), y), \text{ if } y \in D(\varphi),$$

$$\tilde{f}(x, y) = (x, y), \text{ if } y \notin D(\varphi),$$

Therefore, $\tilde{f}$ is computable on (3.13), and has there a unique fixed point $(*_X, *_Y)$. Moreover, since in the case $y \in D(\varphi)$, $\varphi(y)$ is never zero, we easily obtain (ii).

4. Regularization and renormalization

The subsections 4.1–4.5 are dedicated to a review of the relevant parts of the renormalization formalism in Quantum Field Theory, following [E–FMan]. The reader should keep in mind that this scheme does not cover all versions, used by physicists: see e.g. [Cos] for a detailed treatment of the so called Wilsonian renormalization, and [Po] for a brief introduction to the peculiarities related to the quantization of gauge fields.

In the subsections 4.6–4.9 I review some regularization schemes that might be relevant in theoretical computation.

4.1. Connected filtered Hopf algebras. Let $\mathcal{H}$ be a unital associative and counital coassociative bialgebra over a field K of characteristic zero. The relevant structure maps are denoted

$$m : \mathcal{H} \otimes \mathcal{H} \to \mathcal{H}, \ \Delta : \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}, \ u : K \to \mathcal{H}, \ \varepsilon : K \to \mathcal{H}.$$

In our main applications $\mathcal{H}$ will be a bialgebra of programming methods, such as flowcharts (see 3.3). It will satisfy two main assumptions of [E–FMan]: it will be filtered and connected. This means that we are given a filtration $\mathcal{H} = \cup_{n=0}^{\infty} \mathcal{H}^n$, compatible with m and Δ in the standard sense:

$$m(\mathcal{H}^p \otimes \mathcal{H}^q) \subset \mathcal{H}^{p+q}, \quad \Delta(\mathcal{H}^n) \subset \sum_{p+q=n} \mathcal{H}^p \otimes \mathcal{H}^q,$$

and moreover, $\mathcal{H}^0$ is identified with K by means of u and ε.

In this case, $\mathcal{H}$ automatically has an antipode S with $S(\mathcal{H}^n) \subset \mathcal{H}^n$, and hence is a Hopf algebra. The antipode can be given explicitly, by induction on the filtration degree. Namely, we have $S(1) = 1$, and for any $x \in \mathcal{H}^n$, $n \geq 1$, in a version of Sweedler's notation

$$\widetilde{\Delta}x := \Delta x - (x \otimes 1 + 1 \otimes x) = \sum_{(x)} x' \otimes x'' \in \bigoplus_{\substack{p,q \geq 1 \\ p+q=n}} \mathcal{H}^p \otimes \mathcal{H}^q \tag{4.1}$$

and

$$S(x) = -x - \sum_{(x)} S(x')x'' = -x - \sum_{(x)} x'S(x'') \tag{4.2}$$

The antipode is a crucial ingredient in the renormalization formulas (4.6) and (4.7) below.

Since its existence is guaranteed by (4.1), (4.2), in applications to programming methods, when the relevant bialgebra is constructed, we must define *a compatible filtration.* Often it comes from a bialgebra *grading,* whose intuitive meaning is quite transparent: it is a measure of complexity/volume of the relevant programming method which is additive with respect to the composition of programs.

For example, the total number of flags is additive with respect to cuts (cf. Definition 3.2), so it can be used to define the grading of the respective bialgebra. When graphs, such as flowcharts, are decorated by labels from a countable set L , one can choose a "weight" numbering of L and define the grading degree of a decorated graph as *the sum of all labels of its flags and vertices.*

4.2. "Minimal subtraction" algebras. A "minimal subtraction" scheme, that merges well with Hopf algebra renormalization techniques (cf. [E-FMan]), formally is based upon a commutative associative algebra $\mathcal{A}$ over a field K, together with two subalgebras $\mathcal{A}_-$ ("polar part") and $\mathcal{A}_+$ (regular part), such that $\mathcal{A} = \mathcal{A}_- \oplus \mathcal{A}_+$ as a linear space. One usually assumes $\mathcal{A}$ unital, and $1_A \in \mathcal{A}_+$. Moreover, an augmentation homomorphism $\varepsilon_{\mathcal{A}} : \mathcal{A}_+ \to K$ must be given.

Then any element $a \in \mathcal{A}$ is the sum of its polar part a_- and regular part a_+. The "regularized value" of a is $\varepsilon_{\mathcal{A}}(a_+)$.

4.3. Example: germs of meromorphic functions. Here $K = \mathbf{C}$, $\mathcal{A} :=$ the ring of germs of meromorphic functions of z, say, at $z = 0$; $\mathcal{A}_- := z^{-1}\mathbf{C}[z^{-1}]$, $\mathcal{A}_+$ consists of germs of regular functions at $z = 0$, $\varepsilon_{\mathcal{A}}(f) := f(0)$.

Notice that for the same algebra, a complementary choice could have been made: one could put $\mathcal{A}'_- :=$ germs regular and vanishing at $z = 0$, and $\mathcal{A}'_+ := \mathbf{C}[z^{-1}]$, with $\varepsilon'_A(f) = f(z_0)$ for a constant z_0.

This is a toy model of situations arising in cut–off regularization schemes: z_0 is an input from physics, the scale of a parameter z_0 (such as energy) to which our observed quantities refer. As a physics justification of the choice of such a scale one might postulate the belief that beyond this scale " a new physics" starts.

4.4. Hopf renormalization scheme. We summarize its algebraic version here, restating Theorem 1 from sec. 2.5, [E–FMan].

Let $\mathcal{H}$ be a Hopf algebra as above, $\mathcal{A}_+, \mathcal{A}_- \subset \mathcal{A}$ a minimal subtraction unital algebra. Consider the set $G(\mathcal{A})$ of K–linear maps $\varphi : \mathcal{H} \to \mathcal{A}$ such that $\varphi(1_{\mathcal{H}}) = 1_{\mathcal{A}}$.

Then $G(\mathcal{A})$ with the convolution product

$$\varphi * \psi(x) := m_{\mathcal{A}}(\varphi \otimes \psi)\Delta(x) = \varphi(x) + \psi(x) + \sum_{(x)} \varphi(x')\psi(x'') \tag{4.3}$$

is a group, with identity $e(x) := u_{\mathcal{A}} \circ \varepsilon(x)$ and inversion

$$\varphi^{*-1}(x) = e(x) + \sum_{m=1}^{\infty} (e - \varphi)^{*m}(x) \tag{4.4}$$

where for any $x \in \ker \varepsilon$ the latter sum contains only finitely many non–zero summands.

An important subset of $G(\mathcal{A})$ consists of *characters:* those linear maps $\mathcal{H} \to \mathcal{A}$ that are homomorphisms of algebras.

4.4.1. Theorem on the Birkhoff decomposition. *If $\mathcal{A}$ is a minimal subtraction algebra, each $\varphi \in G(\mathcal{A})$ admits a unique decomposition of the form*

$$\varphi = \varphi_-^{*-1} * \varphi_+; \quad \varphi_-(1) = 1_{\mathcal{A}},\ \varphi_-(\ker \varepsilon) \subset \mathcal{A}_-,\ \varphi_+(\mathcal{H}) \subset \mathcal{A}_+. \tag{4.5}$$

Values of renormalized polar (resp. regular) parts φ_- (resp. φ_+) on $\ker \varepsilon$ are given by the inductive formulas

$$\varphi_-(x) = -\pi \left(\varphi(x) + \sum_{(x)} \varphi_-(x')\varphi(x'') \right), \tag{4.6}$$

$$\varphi_+(x) = (\mathrm{id} - \pi)\left(\varphi(x) + \sum_{(x)} \varphi_-(x')\varphi(x'')\right). \tag{4.7}$$

Here $\pi : \mathcal{A} \to \mathcal{A}_-$ is the polar part projection in the algebra $\mathcal{A}$.

If φ is a character, φ_+ and φ_- are characters as well.

4.5. Rota–Baxter operators as generalized polar parts. The general definition of a Rota–Baxter (RB) operator of weight θ on an associative (not necessarily unital or commutative) algebra $\mathcal{A}$ is this: it is a linear operator $R : \mathcal{A} \to \mathcal{A}$, satisfying the identity

$$R(f)R(g) = R(R(f)g + fR(g) + \theta fg) \tag{4.8}$$

If $\mathcal{A}$ is a minimal subtraction algebra, the polar part projection $\pi : \mathcal{A} \to \mathcal{A}_-$ is an RB operator of weight -1; moreover, $\pi^2 = \pi$.

Dropping the restrictions $\theta = -1$ and $R^2 = R$, but still imagining R as a generalized "polar part" operator, one gets more freedom in using the recursive renormalization schemes (4.6), (4.7): see [E–FMan] and below.

We now pass to the discussion of this scheme in possible applications to programming methods. We have already explained that to construct the relevant Hopf algebra $\mathcal{H}$ we need a composition bialgebra as in section 3, completed by a filtration coming, say, from an additive complexity function on programming methods.

4.6. Target algebras and tropical geometry. We now turn to possible interpretations of target algebras $\mathcal{A}$ and linear functionals/characters $\varphi : \mathcal{H} \to \mathcal{A}$ in the theory of computation.

Roughly speaking, there are at least two classes of meaningful pairs $(\mathcal{A}, \varphi)$:

(A) $\varphi(x)$ can be a quantitative characteristic (measure) *of a program* x (e. g. running time and/or memory needed to complete the computation as a function on the size of input);

(B) $\varphi(x)$ is quantitative characteristics *of the output* produced by a variable program x (on a particular input, or on the set of all inputs).

Some natural measures take their values not in commutative rings, but in *commutative semirings*, say, of the type *Max–Plus*: cf. [Cas1], [Cas2] for a recent review and references. Such a structure is similar to a commutative ring, but its additive group axioms are weakened to those of additive monoid. The name *Max–Plus* is

explained by a typical example of such a semiring: $\mathbf{R}_{\geq 0}$ with "addition" $\max(x, y)$ and "multiplication" $x + y$.

Such measures do not fit directly in the framework of Hopf renormalization theory as it was formulated, but I want to stress their importance here, by providing two examples. I plan to return to Hopf semiring renormalization in a sequel to this article.

4.6.1. Example: parallel computation. Let a world W of histories of computations be represented by decorated flowcharts as in 3.3. Then a reasonable idealization of *running time* might be a function $T : W \to \mathbf{R}_{\geq 0}$ with the following property: for any flowchart τ and its cut C,

$$T(\tau) = T(\tau^C) + T^C(\tau_C) \tag{4.9}$$

where the superscript in T^C is supposed to remind that input of τ_C is the output of τ^C. On the other hand, the idea of *parallel computation* is reflected in a formal requirement: running time of a disjoint sum of flowcharts is

$$T(\tau_1 \coprod \tau_2) = \max(T(\tau_1), T(\tau_2)). \tag{4.10}$$

Such a function can be thought of as a semiring–valued "quasi–character" sending the composition of programs to the semiring product, and disjoint sum of programs to the semiring sum.

For more details, see sec. 2 of [Ma4].

4.6.2. Example: Bayesian networks. A Bayesian network (see e. g. [PaSt1–2]) is a directed graph whose vertices are (decorated by) two groups of variables: observable random variables $Y_1, \dots, Y_n$ and hidden random variables $X_1, \dots, X_m$, whereas edges are decorated by (matrices of) transition probabilities. A Bayesian network can be considered as a programming method for computing certain characteristics of hidden random variables, such as so called *marginal probabilities*, and *maximal a posteriori* log *probabilities (MAP)* (cf. [PaSt1]). Although Bayesian networks are not, strictly speaking, flowcharts in the sense of our sec. 3, it is not difficult to recast them in a similar form. Then the outputs such as marginal and MAP probabilities become semiring–valued characters.

4.7. Algebras of sequences. We start in a formal setting. For a field K, consider the space $\mathcal{S} = \mathcal{S}_K$ of infinite sequences $f = (f_1, f_2, \dots)$, $f_i \in K$. There

are three different multiplications that furnish relevant structures of commutative algebra on this space, $\bullet$, $\times$ and $*$:

$$(f \bullet g)_n := f_n g_n,\ 1_\bullet = (1, 1, 1, \dots), \tag{4.11}$$

$$(f * g)_n := \sum_{\max(p,q)=n} f_p f_q,\ 1_* = (1, 0, 0, \dots), \tag{4.12}$$

$$(f \times g)_n := \sum_{p+q=n} f_p f_q, \quad (\text{non} - \text{unital}). \tag{4.13}$$

Clearly, $(\mathcal{S}_K, \times)$ is simply the algebra of formal series $\sum_{n=1}^{\infty} f_n z^n$. We could formally adjoin an identity to it, that is, allow sequences starting with $n = 0$. Multiplications $\bullet$ and $*$ extend to this case without problems. However, we will have to avoid non–vanishing f_0 in other contexts, cf. below.

We can similarly interpret $(\mathcal{S}_K, *)$ as the algebra of formal "tropical" series $\sum_{n=1}^{\infty} f_n z^n$, with multiplication $z^p * z^q := z^{\max(p,q)}$. Notice that z is the identity in this algebra. However, if we extend it by allowing $f_0 \neq 0$, the role of identity will pass to the constant formal series 1.

The map "partial summation":

$$S : (\mathcal{S}_K, *) \to (\mathcal{S}_K, \bullet), \quad S(f)_N := \sum_{n=1}^{N} f_n \tag{4.14}$$

is an isomorphism of unital algebras:

$$S(f * g) = S(f) \bullet S(g).$$

Considered as a map of the algebra $(\mathcal{S}_K, *)$ *into itself*, S is *a Rota–Baxter (RB) operator of weight 1* in the sense of [E–FMan], Sec. 3: we have

$$S(f) * S(g) = S(S(f) * g + f * S(g) + f * g). \tag{4.15}$$

The role of this remark in the context of renormalization is explained by the fact that in the simplest "minimal subtraction" scheme as in the subsection 1 above, the projection to the "polar part" $\pi : \mathcal{A} \to \mathcal{A}_-$ is an idempotent Rota–Baxter operator of weight -1.

Notice in conclusion that the slightly modified summation operator S',

$$S'(f)_N := \sum_{n=1}^{N+1} f_n, \tag{4.16}$$

is again Yang–Baxter, of the same weight -1 as a polar projection π is.

4.7. Boutet de Monvel's regularization. Let now $K = \mathbf{C}$. Consider the subspace $\mathcal{A} = \mathcal{A}_{\mathbf{C}}$ in $\mathcal{S}_{\mathbf{C}}$ consisting of such sequences $f = (f_n)$ for which there exists a polynomial $P = P_f$ and an integer $A = A_f$ with the property

$$S(f)_N = P_f(\log N) + O((\log N)^A/N) \tag{4.17}$$

4.7.1. Theorem. *(i) $S(\mathcal{A})$ is a subalgebra in $(\mathcal{S}_{\mathbf{C}}, \bullet)$, so that $\mathcal{A}$ is a subalgebra of tropic power series $(\mathcal{S}_{\mathbf{C}}, *)$.*

(ii) Considered as a subset of $(\mathcal{S}_{\mathbf{C}}, \times) = z\mathbf{C}[[z]]$, $\mathcal{A}$ can be described as a subspace of the algebra $\mathcal{B}$ of power series $f(z)$ convergent in $|z| < 1$ and vanishing at 0, such that for a polynomial $Q = Q_f$ and an integer $B = B_f$ we have as $z \to 1 - 0$ along the real axis

$$f(z) = Q_f(-\log(1-z)) + O((1-z)\cdot \log^B(1-z)). \tag{4.18}$$

(iii) The "singular part" Q_f is uniquely defined by P_f. It can be derived from P_f by applying the formal differential operator of infinite order obtained from the Taylor series of the gamma–function:

$$Q_f(t) = \Gamma(1+\partial_t)\, P_f(t), \quad \Gamma(1+x) = 1 - \gamma x + \dots. \tag{4.19}$$

The first part is straightforward. The second and third ones constitute a theorem by Boutet de Monvel: see [BdeM1] for a more precise statement, and [De], [Ra] for some proofs.

NB The embedding $\mathcal{A} \subset \mathcal{B}$ is strict.

4.7.2. Comments. Consider the subalgebra $\mathcal{B}_- \subset \mathcal{B}$ consisting of all functions $Q(-\log(1-z))$, $Q \in t\,\mathbf{C}[t]$. In fact, we have $\mathcal{B}_- \subset \mathcal{A}$, so we may alternatively call it $\mathcal{A}_-$.

The map $\pi : f \mapsto Q_f$ is a surjective algebra homomorphism $\mathcal{B} \to \mathcal{B}_- \subset \mathcal{B}$. It is a natural singular part of f at $z = 1$ replacing the polar part of a meromorphic function. Its kernel $\mathcal{B}_+$ (resp $\mathcal{A}_+$) consists of elements of $\mathcal{B}$ (resp. $\mathcal{A}$) vanishing at $z = 1$. Hence we get a minimal subtraction algebra $\mathbf{C} \oplus \mathcal{B}$ that can serve as an input to a natural regularization scheme. In fact, Boutet de Monvel, Racinet and Deligne used it to regularize the integral expressions for multiple zeta values which are closely related to special Feynman integrals.

Restricted to $\mathcal{A}$, π is a linear surjective map $\mathcal{A} \to \mathcal{A}_-$.

As at the end of 4.3, we can also interchange the nominations of "polar" and "regular" parts.

Finally, notice that $-\log(1-z)$ corresponds to the sequence

$$\mathbf{l} := (1, 1/2, 1/3, \dots). \tag{4.20}$$

This sequence satisfies (4.17) with $P_f(t) = t + \gamma$. Hence it is natural to declare that the subalgebra of polar parts in $(\mathcal{A}, *)$ (i. e. in $\mathcal{A}$ interpreted as tropical series) is the algebra of $*$–polynomials in $\mathbf{l}$. The formula (4.19) says that the change of multiplication from $*$ to $\times$ does not change the space of polar parts.

Results of A. Levin ([Le]) suggest that in (one of the) computation contexts one can meaningfully replace (4.20) by a sequence related to exponential Kolmogorov complexity, in the spirit of the remark made in 0.6 above that Kolmogorov complexity is the ultimate computational infinity. I will finish this paper by briefly explaining Levin's theorem.

4.8. One–sided enumerability. The standard definition of a computable real number x involves a recursive sequence of rational approximations (r_n) to it, together with a recursive sequence of bounds for the error $|x - r_n| \le b_n$, $b_n \to 0$.

We will consider here, following [Le], *one–sided versions* of computability that are related to various versions of Kolmogorov complexity.

4.8.1. Definition. *A real number $x \in \mathbf{R}$ is called enumerable from below iff the following equivalent conditions (i), (ii) are satisfied:*

(i) there exists a general recursive function $\varphi : \mathbf{Z}^+ \to \mathbf{Q}$ such that

$$\varphi(1) \le \varphi(2) \le \varphi(3) \le \dots, \quad \lim \varphi(n) = r. \tag{4.21}$$

(ii) The set $\{r \in \mathbf{Q} \,|\, r < x\}$ is recursively enumerable.

For obvious reasons, we consider the symbol $+\infty$ as computable from below as well (this symbol is useful in tropical contexts): it is approximated by $\varphi(n) = n$.

In [BrYa] such numbers are called left computable (or rather, $-r$ are called right computable).

4.8.2. Definition. *A sequence of real numbers x_n, $n = 1, 2, \dots$ is called enumerable from below iff the following equivalent conditions (i), (ii) are satisfied:*

(i) there exists a general recursive function $\varphi : \mathbf{Z}^+ \to \mathbf{Z}^+ \times \mathbf{Q}$, $\varphi(n) = (m_n, r_n)$ such that the map $\mathbf{Z}^+ \to \mathbf{Z}^+ : n \mapsto m_n$ is surjective; if $m_a = m_b$, $a < b$, then $r_a \leq r_b$; and finally, the limit of the sequence $r_a^{(n)}$ corresponding to one and the same first coordinate n, is x_n.

(ii) The set $\{(n, r_n) \in \mathbf{Z}^+ \times \mathbf{Q} \,|\, r_n < x_n\}$ is recursively enumerable.

Again, we may include the symbol $+\infty$ as a possible value of x_n.

We will use results of [Le], where such sequences of non–negative real numbers were considered.

Remarks. (a) Enumerable from below reals form an additive subsemigroup of $\mathbf{R}$. The check is straightforward. The same is true for reals enumerable from above (one can iclude the symbol $-\infty$ in place for $+\infty$).

(b) There exist reals enumerable from below but not from above, and vice versa. In fact, if a real x is enumerable both from below and above, than it is computable: the differences between the upper and the lower n–th approximation form a recursive sequence of bounds for each of the approximations.

It is crucial that there exist one–sided enumerable numbers which are not computable: see e. g. Proposition 2.2 in [BrYa].

(c) For this reason, enumerable from below reals *do not form a subring of* $\mathbf{R}$: multiplication, say, by -1 reverses enumerability from below to enumerability from above.

Similarly, *inversion* $x \mapsto x^{-1}$ interchanges enumerability from below and enumerability from above.

However, *non–negative reals* enumerable from below (resp. from above) form a semiring with respect to the usual addition and multiplication.

They also form a tropical semiring with respect to the operations $\oplus := \max, \otimes := \cdot$ (resp. $\oplus := \max, \otimes := \cdot$).

4.9. Kolmogorov complexity reappears. The simplest construction due to L. Levin starts with a particular "norm functional" defined on the set of enumerable from below sequences $f : \mathbf{Z}^+ \to \mathbf{R}_{\geq 0}$ (our former x_n is now $f(n)$):

$$N(f) := \sup \{r \cdot \operatorname{card} \{x \mid f(x) \geq r\}\}$$

4.9.1. Proposition. *There exists an enumerable from below sequence of finite norm F such that any enumerable from below sequence f is majorized by cF for an appropriate constant c.*

The sequence $(-\log F(n))$ coincides with the sequence of values of (logarithmic) Kolmogorov complexity, up to an $O(1)$ function.

l intend to study in the sequel of this paper a version of regularization where (4.20) is replaced by $(F(n))$.

Appendix: Renormalization at large

Warning: the gentle reader is kindly invited to skip the following musings.

A.1. Is physical infinity real? The basic intuitive picture behind mathematical formalism of renormalization seems to be an image of "finite" *observable* reality as *a difference (or quotient)* of two unobservable and infinite *physical* realities. The same intuition in theoretical physics produces such expressions as "vacuum energy is infinite": the observable finite lumps of energy/matter are interpreted as finite differences between the the two excitation levels of vacuum, both of which are infinite.

This intuition is supported by technological achievements: the energy of a nuclear explosion is freed, when the two infinities, tightly balanced at the nuclear scale, are made unbalanced by subtly controlled technological processes – their difference then destroys everything around them.

Finally, the question in the title of this subsection should not be confused with a totally different question: *"Is physical reality infinite?"* (Wittgenstein's angry laugh thunders from the Great Beyond ...)

A2. Epistemology of mathematics and infinite. A possible parallel in the platonic world of ideas can be traced in the on–going epistemological shift related to the foundations of mathematics: *discrete and finite* nowadays often comes from looking at (homotopy types of) *continuous and infinite.*

After Cantor, Dedekind, Hausdorff, Bourbaki and up to the last decades we have been always moving in reverse direction. Not anymore.

To help an uninitiated reader to see what is going on, here are simple examples: imagine that "two" counts not fingers or stones, but orientations of an Euclidean space, and **Z** counts homotopy classes of closed oriented loops around zero in a plane.

Notice that in the Set Theory, where we, after Cantor, Frege and Russell, interpret integers as cardinalities of finite discrete sets, only natural numbers **N** appear directly, whereas negative numbers require a psychologically difficult and historically late leap of imagination.

A.3. Renormalization of financial markets? Finally, it might be amusing to think about the current financial crisis in similar terms.

This is not a simple fancy: recall that one of the early psychologically acceptable interpretations of negative numbers was formulated in terms of *debt.*

Now, to put it crudely, consider the sum total of values of contracts defining what all players at the global financial market *owe* to their counterparts ("minus infinity") at a given time, and the respective sum total of values of contracts defining what all players at the financial market *are owed* ("plus infinity"). In the world of material reality where one can lend only what one owns and get back only what one has lent, these two "infinities for dummies" would exactly cancel.

Money adds to this material world an ideal dimension of *credit*, and (in good times) a creative force. The difference between credits and debits pays for a private house and public education. In a healthy economy, this difference however must be reasonably stable and remain on a considerably lesser scale than the two infinities.

Technically, debts and credits do not cancel anymore for many reasons. Banks are required to hold monetary reserves which constitute a small percentage of their deposits; the rest can be invested, loaned etc. Debts must be paid at different times in future, at various rates of interest. This line of thought is iterated, which leads to the creation of contracts buying and selling risks, debts, etc. Such contracts are *derivatives*, financial instruments whose value is derived from the value of other financial instruments. The two infinities, and their difference, start fluctuating and eventually lose their contacts with reality.[1]

In a remarkable agreement with our metaphor, one of the great players and keen observers of financial markets, Warren Buffett, once called derivatives "financial weapons of mass destruction".

[1] Cf. [Mas] for a related quantitative discussion.

Renormalization of finances probably needs intellects of Richard Feynman's scale.

A.4. Computational viewpoint on human civilization. From the computational viewpoint, human civilization is a supplier of software, hardware, and oracles producing programs and inputs. It is also a consumer of outputs.

Scientific laws such as Newton's law of gravity, are short (Kolmogorov simple) oracular prescriptions for writing software that will be calculating (predicting), say, visible movements of planets.

Scientific observations are systematic methods of obtaining oracular prescriptions for inputs into resulting programs.

Gods write equations of motion, devils choose initial and boundary conditions, experimenters store them in databases.

References

[AlMar1] P. Aluffi, M. Marcolli. *Feyman motives of banana graphs.* hep–th/0807.1690

[AlMar2] P. Aluffi, M. Marcolli. *Algebro–geometric Feynman rules.* hep–th/0811.2514

[AlMar3] P. Aluffi, M. Marcolli. *Parametric Feynman integrals and determinant hypersurfaces.* math.AG/0901.2107

[BeDr] A. Beilinson, V. Drinfeld. *Chiral Algebras.* AMS Colloquium Publications, vol. 51, AMS, Providence, Rhode Island, 2004.

[Bl] S. Bloch. *Motives associated to graphs.* Japan J. Math., vol. 2 (2007), 165–196.

[BlEsKr] S. Bloch, E. Esnault, D. Kreimer. *On motives associated to graph polynomials.* Comm. Math. Phys., vol. 267 (2006), 181–225.

[BoMa] D. Borisov, Yu. Manin. *Generalized operads and their inner cohomomorhisms.* In: Geometry and Dynamics of Groups and spaces (In memory of Aleksader Reznikov). Ed. by M. Kapranov et al. Progress in Math., vol. 265. Birkhäuser, Boston, pp. 247–308. math.CT/0609748

[BdeM1] L. Boutet de Monvel. *Remark on divergent multizeta series.* Posted at http://people.math.jussieu.fr/∼boutet, 2004.

[BdeM2] L. Boutet de Monvel. *Algèbre de Hopf des diagrammes de Feynman, renormalisation et factorisation de Wiener–Hopf (d'après A. Connes et D. Kreimer).* Séminaire Bourbaki, no. 900, 2002.

[BrYa] M. Braverman, M. Yampolsky. *Computability of Julia sets.* Moscow Math. Journ., 8:2 (2008), 185–231.

[Cas1] D. Castella. *L'Algèbre tropicale comme algèbre de la caracteristique 1: Algèbre linéaire sur les semi–corps idempotents.* math.AC/0807.3088

[Cas2] D. Castella. *L'Algèbre tropicale comme algèbre de la caracteristique 1: Polynômes rationnels et fonctions polynomiales.* math.RA/0809.023

[ConKr1] A. Connes, D. Kreimer. *Hopf algebras, renormalization and noncommutative geometry.* hep-th/9808042

[ConKr1] A. Connes, D. Kreimer. *Renormalization in quantum field theory and the Riemann–Hilbert problem. I. The Hopf algebra structure of graphs and the main theorem.* Comm. Math. Phys. 210, no. 1 (2000), 249–273.

[Cos] K. Costello. *Renormalization of quantum dield theories.* Available at author's home page.

[De] P. Deligne. *Multizéta values.* Lecture notes, IAS Princeton, 2001.

[E-FMan] K. Ebrahimi–Fard and D. Manchon. *The combinatoris of Bogolyubov's recursion in renormalization.* math-ph/0710.3675

[Fra] A. Frabetti. *Renormalization Hopf algebras and combinatorial groups.* arXiv:0805.4385

[Fri] B. Friedrich. *Periods and algebraic de Rham cohomology.* math.AG/0506113

[GaLe] P. Gács, A. Levin. *Causal Nets or What Is a Deterministic Computation?* Int. Journ. Theor. Phys., vol. 21, No. 12 (1982), 961–971.

[GeMa] S. Gelfand, Yu. Manin. *Methods of homological algebra.* Second edition, Springer, 2003.

[Gl] M. D. Gladstone. *A reduction of the recursion scheme.* Journ. Symb. Logic, 32:4 (1967), 505–508.

[I1] L. Ionescu. *The Feynman legacy.* math.QA/0701069

[I2] L. Ionescu. *From operads and PROPs to Feynman processes.* math.QA/0701299

[Ka] M. Kapranov. *Noncommutative geometry and path integrals.* arXiv:math/0612411

[KoZa] M. Kontsevich, D. Zagier. *Periods.* In: Mathematics unlimited —2001 and beyond, Springer, Berlin 2001, 771–808.

[Kr1] D. Kreimer. *On the Hopf algebra structure of perturbative quantum field theory.* Adv. Theor. Math. Phys., 2 (1998), 303–334.

[Kr2] D. Kreimer. *Structures in Feynman graphs – Hopf algebras and symmetries.* hep-th/020211

[Kr3] D. Kreimer. *Dyson–Schwinger equations: from Hopf algebras to number theory.* hep–th/0609004

[KremSz] K. Kremnizer, M. Szczesny. *Feynman graphs, rooted trees, and Ringel–Hall algebras.* math.QA/0806.1179

[Le] L. Levin. *Various measures of complexity for finite objects (axiomatic description).* Soviet Math. Dokl., vol 17, No. 2 (1976), 522 –526.

[Manch] D. Manchon. *Hopf algebras, from basics to application to renormalization.* math.QA/0408405

[Ma1] Yu. Manin. *A Course in Mathematical Logic.* Springer Verlag, 1977. XIII+286 pp. (The second, expanded Edition in preparation).

[Ma2] Yu. Manin. *Classical computing, quantum computing, and Shor's factoring algorithm.* Séminaire Bourbaki, no. 862 (June 1999), Astérisque, vol 266, 2000, 375–404. quant-ph/9903008.

[Ma3] Yu. Manin. *Frobenius manifolds, quantum cohomology, and moduli spaces.* AMS Colloquium Publications, vol. 47, Providence, RI, 1999, xiii+303 pp.

[Ma4] Yu. Manin. *Renormalization and computation II: Time cut–off and the Halting Problem. math.QA/0908.34.30*

[Mar] M. Markl. *Operads and PROPs.* math.AT/0601129

[MarkShi] I. L. Markov, Y. Shi. *Simulating quantum computation by contracting tensor networks.* quant–ph/0511069

[Mas] V. Maslov. *Economic law of increase of Kolmogorov complexity. Transition from financial crisis 2008 to the zero–order phase transition (social explosion).* q-fin.GN/0812.4737

[Mi] R. Miller. *Computable fields and Galois theory.* Notices of the AMS, vol. 55, no. 7 (2008), 798–807.

[Mo] Y. N. Moschovakis. *What is an Algorithm.* Available at author's home page.

[Po] M Polyak. *Feynman diagrams for pedestrians and mathematicians.* math.GT/0406251

[PaSt1] L. Pachter, B. Sturmfels. *Tropical geometry of statistical models.* Proc. Nat. Ac. Sci. USA, vol. 101, no. 46 (2004), 16132–16137.

[PaSt2] L. Pachter, B. Sturmfels. *Parametric inference for biological sequence analysis.* q-bio.GN/0401033

[Ra] G. Racinet. *Doubles mélanges des polylogarithmes multiples aux racines del'unité.* Publ. Math. Inst. Hautes Études Sci., No. 95 (2002), 185–231.

[Ro] R. M. Robinson. *Primitive recursive functions.* Bull. AMS, 53 (1947), 925–942.

[Sc] D. Scott. *The lattice of flow diagrams.* In: Symposium on Semantics of Algorithmic Languages, Springer LN of Mathematics, 188 (1971), 311–372.

[TsVlN] M. Tsfasman, S. Vlăduţ, D. Nogin. *Algebraic geometric codes: basic notions.* Mathematical Surveys and Monographs, 139. American Mathematical Society, Providence, RI, 2007. xx+338 pp.

[Va1] B. Vallette. *A Koszul duality for PROPs.* math.AT/0411542

[Va2] B. Vallette. *Manin products, Koszul duality, Loday algebras and Deligne conjecture.* math.QA/0609002

[vdL] P. van der Laan. *Operads and the Hopf algebras of renormalization.* math-ph/0311013

[VlMa] S. Vlăduţ, Yu. Manin. *Linear codes and modular curves. (Russian).* In: Current problems in mathematics, Vol. 25, 209–257, Itogi Nauki i Tekhniki, Akad. Nauk SSSR, VINITI, Moscow, 1984.

[Ya] N. S. Yanofsky. *Towards a definition of an algorithm.* math.LO/0602053

[Yo] M. Yoshinaga. *Periods and elementary real numbers.* math.AG/0805.0349

[Zo] P. Zograf. *Tensor networks and the enumeration of regular subgraphs.* math.CO/0605256