An answer to the Gamma question

Benoit Monin
Associate Professor
LACL
Université Paris-Est Créteil
Créteil, France
benoit.monin@u-pec.fr

Abstract

We answer in this paper an open question (known as the "Gamma question"), related to the recent notion of coarse computability, which stems from complexity theory. The question was formulated by Andrews, Cai, Diamondstone, Jockusch and Lempp in "Asymptotic density, computable traceability and 1-randomness" [1]. The Gamma value of an oracle set measures to what extent each set computable with the oracle is approximable in the sense of density by a computable set. The closer to 1 this value is, the closer the oracle is to being computable. The Gamma question asks whether this value can be strictly in between 0 and $1/2$.

In this paper, we pursue some work initiated by Monin and Nies in "A unifying approach to the Gamma question" [19]. Using notions from computability theory, developed by Monin and Nies, together with some basic techniques from the field of error-correcting codes, we are able to give a negative answer to this question.

The proof we give also provides an answer to a related question, asked by Denis Hirschfeldt in the expository paper "Some questions in computable mathematics" [12]. We also solve the Gamma problem for bases other than 2, answering another question of Monin and Nies.

CCS Concepts • **Theory of computation → Computability**;

Keywords Computability, Coarse computability, Turing degrees

ACM Reference Format:
Benoit Monin. 2018. An answer to the Gamma question. In *LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3209108.3209117

1 Introduction

1.1 A brief history of the Γ question

Generic-case complexity is a subfield of computational complexity. It started with the observation that some problems that are difficult to solve in full are easy to solve on "most inputs", namely on a set of inputs of density 1. This notion was introduced by Kapovich, Myasnikov, Schupp and Shpilrain [17]. They showed among other things that for a large class of finitely generated groups, the generic case complexity of the word problem is linear.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
LICS '18, July 9–12, 2018, Oxford, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209117

This notion has recently been extended to computability by Jockusch and Schupp [15]. The authors identified two notions that can be proved to be incomparable. The first is generic computability, where one must always give the right answer, without having to provide an answer for a small set of inputs. The second is coarse computability, for which one always has to provide an answer, with the right to be wrong on a small set of inputs. In both cases, a set of inputs is considered small if it is of density 0; this will be made precise in Section 2.

Hirschfeldt, Jockusch, McNicholl and Schupp [13] introduced and studied a value $\gamma(A)$ to an arbitrary subset A of natural numbers which measures how closely A is from being coarsely computable, that is, how closely A can be approximated by computable sets. Namely, $\gamma(A)$ is the least upper bound of lower asymptotic densities of all sets $\{n \in \mathbb{N} : A(n) = C(n)\}$ where C is computable (see Section 2.2 for a more formal definition). Then Andrews, Cai, Diamondstone, Jockusch and Lempp [1] assigned a value Γ to each Turing degree. For a Turing degree d, they defined $\Gamma(d)$ to be the greatest lower bound of $\gamma(A)$ where A is a set of degree d (see Section 2.2 for a more formal definition). Hirschfeldt et al [13] had shown that for any degree d, if $\Gamma(d) > 1/2$, then d must be the computable degree, implying that $\Gamma(d) = 1$. This implied no real $1/2 < r < 1$ can be realized by the Γ value of a degree. They also showed the existence of degrees d such that $\Gamma(d) = 0$. Later Andrews et al [1] showed the existence of degrees d such that $\Gamma(d) = 1/2$, and they asked whether a Turing degree could have a Γ value strictly between 0 and $1/2$.

In this paper, we answer this question by showing that if a Turing degree has a Γ value strictly smaller than $1/2$, then its Γ value must be 0. This implies, together with the results of [1] and [13] mentioned in the previous paragraph, that $0, 1/2$ and 1 are the only reals that can be realized by the Γ value of a degree, giving a natural trichotomy of the Turing degrees.

1.2 On the relevance of the Γ question

From the second half of the 20th century until today, a considerable amount of work has been conducted in order to understand the realm of non-computable objects - most of the time elements of $2^{\mathbb{N}}$. Such sequences, as non-computable, will always remain somehow *inaccessible*. Nonetheless it is still possible to study some of their properties, in particular properties relating to their computational power. Doing so, it becomes apparent that it is often not the properties of the object itself, that we are interested in, but the properties of its equivalence class under the relation "X is computable using Y as an oracle and Y is computable using X as an oracle" : The Turing degrees.

Many such properties were introduced and studied, leading to more and more detailed characterizations of the Turing degrees.

Some of them became very popular, due to their numerous non-trivially equivalent characterizations, as well as their use in many situations. We give here two examples of such properties, that also have interactions with the Γ question.

For the first example, we say that a degree d is PA, if d can compute a complete and consistent extension of Peano Arithmetic (by the well known Gödel's first incompleteness theorem, no such computable extension exists). The PA degrees have been widely used and studied in computability theory. We provide here a non-trivially equivalent (and rather intriguing) characterization, in order to illustrate a typical theorem about characterizations of Turing degrees. One direction is due to Dana Scott [20] and the other to Solovay (unpublished) : "A degree d is PA iff for any computable tree $T \subseteq 2^{<\mathbb{N}}$ (a set of strings closed under prefixes), such that T has at least one infinite path (a set of compatible elements of T), d computes an infinite path of T.

For the second example, we say that a degree d is computably dominated if every function that d computes, is dominated by a computable function. The computable degree is of course computably dominated, but it is not the only one. There are for instance computably dominated PA degrees. A non-trivially equivalent (and still rather intriguing) characterization of the non-computably dominated degrees is as follow [19]: A degree d is non-computably dominated, iff d can compute a function from $\mathbb{N}$ to $\mathbb{N}$ which equals infinitely often every computable function from $\mathbb{N}$ to $\mathbb{N}$.

Let us now come back to the Γ question. Here again, we question a property related to the computational power a set A might have : "being able to compute sets which are hard to approximate by computable sets". Before we continue, let us mention how the Γ question relates with the two notions of computability introduced above : Andrews et al. [1] showed that if A is of PA or non-computably dominated degree, then we must have $\Gamma(A) = 0$. Monin and Nies [19] showed later that the converse does not hold. In this paper, we show that the only possible Γ values for a degree are 0, $1/2$ and 1. In order to help the reader understand what is surprising in this trichotomy, we provide here a simplified version of this paper's main achievement.

Given any $A \in 2^{\mathbb{N}}$, we are interested in all the sequences of binary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = 2^n$, that are computable from A. We are then interested in algorithms P (not using oracles) taking n in parameter, and trying then to approximate each string σ_n by a string τ_n of the same length. We will show that for any A, exactly one of the following is true:

1. There exists an A-computable sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = 2^n$, such that for any algorithm P, there are infinitely many n such that P is wrong on every bit of σ_n.
2. Both (2a) and (2b):

(2a) For any A-computable sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = 2^n$, for any $\varepsilon > 0$, there is an algorithm which (asymptotically) correctly guesses a fraction of $1/2 - \varepsilon$ bits of every σ_n.

(2b) There is also an A-computable sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = 2^n$, such that for any $\varepsilon > 0$, no algorithm (asymptotically) correctly guesses a fraction of $1/2 + \varepsilon$ bits of every σ_n.

3. The set A is computable.

The first non-trivial part of this trichotomy is "Suppose (2b) is not true. Then why must A be computable?". This had already been answered by Hirschfeldt, Jockusch, McNicholl and Schupp [13]. The second non-trivial part of this trichotomy is "Suppose (2a) is not true. Then why must we have (1) ?". A slightly more complicated version of this question is answered with Theorem 3.11, the main result of this paper.

1.3 The content of this paper

In order to solve the Γ question, we use the key notion of bounded infinite often equality, that was introduced by Monin and Nies [19]. Informally, given two computable functions f, H from $\mathbb{N}$ to $\mathbb{N}$, we say that f is infinitely often equal with bound H, if it equals infinitely often every computable function bounded by H. Monin and Nies [19] showed that if a Turing degree contains a function which is infinitely often equal with bound $2^{(2^n)}$, then its Γ value must be 0. We show here that the following are equivalent for a Turing degree d:

1. The Γ value of d is strictly smaller than $1/2$.
2. d computes a function which is infinitely often equal with bound $2^{(2^n)}$.
3. The Γ value of d is equal to 0.

We then deal with related questions that naturally arise from the Γ question and have been asked previously. In particular Monin and Nies assigned [19] for any integer $q > 2$ a value Γ in base q, to any Turing degree. This new notion is naturally derived from the original one, by considering infinite q-ary sequences instead of infinite binary sequences. Monin and Nies showed that if the Γ value in base q of a Turing degree is strictly bigger than $1/q$, the degree is in fact the computable degree and its Γ value in base q must be one. It is worth mentioning that the proof is not a straightforward modification of the one from Hirschfeldt et al. in base 2. Monin and Nies also argued that every known example of Turing degree with a Γ value of $1/2$ has a Γ value in base q of $1/q$. They asked whether this is always the case. We answer the question here by showing that the following are equivalent for a Turing degree d:

1. The Γ value in base q of d is strictly smaller than $1/q$.
2. d contains a function which is infinitely-often-equal with bound $2^{(2^n)}$.
3. The Γ value (in any base) of d is equal to 0.

This will also imply that the only possible Γ value in base q of a Turing degree are $0, 1/q$ and 1. Again, the proof will not be a straightforward modification of the proof of the equivalent statement for base 2.

We will finally argue that our proofs also give the same answer to a version of the Γ question in the tt-degrees, asked in [11]. We will briefly defend the relevance of this question and explain why our proofs automatically settles the Γ question in the tt-degrees.

An analogue of the Γ question was also studied for the many-one degrees, and in this case, Matthew Harrison-Trainor [10] was able to provide the opposite answer, in extreme contrast to the situation for Turing and truth-table degrees : For any $r \in [0, 1/2]$, there exists an m-degree d with $\Gamma_m(d) = r$, where Γ_m is the version of Γ for many-one degrees.

1.4 On the relation between Γ and the error-correcting codes

For one key step of the proof that $\Gamma(d) < 1/2$ implies $\Gamma(d) = 0$, we will need to borrow some basic techniques from the field of error-correcting codes, which copes with the problem of reliable transmission of information on a noisy channel. If one wants to transmit, say, a binary message of length m, the idea is to encode this message in a codeword of length $n > m$, and to transmit this codeword instead. The codeword must then be robust to the noise, say random bit flip, that may occur during the transmission: If the number of bit flips is not too large, the receiver must be able to retrieve the initial message. The research in this area focuses on improving robustness to various type of noise, as well as providing efficient encoding and decoding algorithms.

Error-correcting codes are not only used in many data communication protocols. They also have been an important tool in theoretical computer sciences. They play a central role in different areas such as cryptography (e.g. [21] and [4]), probabilistic proofs (e.g. [3] [22] and [2]), pseudorandomness theory (e.g. [6]), and many more. To our knowledge, it is one of the first time they are used in computability theory. A previous example occurs in [5].

2 Preliminaries and notations

In the following, by $\mathbb{N}^\star$, we mean the set of all strictly positive integers (and by $\mathbb{N}$ the set of integers which are positive or null). We work in the space of infinite sequences over a finite alphabet $\{0,\dots,q-1\}$ for an integer $q > 1$, denoted by $q^{\mathbb{N}}$. We call q-ary sequences, or sequences elements of $q^{\mathbb{N}}$ and q-ary strings, or strings, the finite sequences over the alphabet $\{0,\dots,q-1\}$. The set of q-ary strings is denoted by $q^{<\mathbb{N}}$. We sometimes also use the word sequence to denote sequences of various objects (typically integers), and when we do so we will always specify it to avoid any ambiguity.

For $q > 1$ and for a string $\sigma \in q^{<\mathbb{N}}$, we denote the set of elements of $q^{\mathbb{N}}$ extending σ by $[\sigma]$ and we call those sets *cylinders*. We denote by λ the unique probability measure on $q^{\mathbb{N}}$ such that $\lambda([\sigma]) = q^{-|\sigma|}$ for any string σ, where $|\sigma|$ denotes the length of σ. For a sequence $X \in q^{\mathbb{N}}$ and a finite interval $I \subset \mathbb{N}$, we denote by $X\restriction_I$ the string $X(I(0))^\frown \dots {}^\frown X(I(m-1))$, where m is the length of I. The notation $X\restriction_n$ for $n \in \mathbb{N}$ means $X\restriction_{[0,\dots,n-1]}$.

In this paper we will be interested in having a canonical coding between sequences and functions $f : \mathbb{N} \to \mathbb{N}$ which are strictly bounded by some $H : \mathbb{N} \to \mathbb{N}$. Such a function H will generally be an *order function*, that is, a computable function H such that $H(n) \leq H(n+1)$ and $\lim_n H(n) = +\infty$. In the context of q-ary sequences and strings, to make the coding work nicely we will consider that the bound H is always of the form $q^{\widetilde{H}(n)}$. Given a q-ary sequence X and such a bound $H(n) = q^{\widetilde{H}(n)}$, we denote by f_X the function bounded by H, whose values are the integers encoded by successive chunks of bits of X, of length $\widetilde{H}(n)$. Formally we define $H'(n) = \sum_{m<n} \widetilde{H}(m)$ (with $H'(0) = 0$), and $f_X(n)$ to be the integer smaller than $q^{\widetilde{H}(n)}$ which is encoded by the string $X\restriction_{[H'(n),H'(n+1))}$. Conversely, given f with $f(n) < H(n) = q^{\widetilde{H}(n)}$, we write X_f to denotes the sequence X such that $f_X = f$.

Finally, for a q-ary sequence X and a p-ary sequence Y, we write $Y \leq_T X$ to mean that Y is Turing computable from X. We write $Y \equiv_T X$ if $Y \leq_T X$ and $X \leq_T Y$. We write $Y <_T X$ if $Y \leq_T X$ but not $X \leq_T Y$. The equivalence classes for the relation $\equiv_T$ are the *Turing degrees*. Turing degrees are usually defined only over binary sequences. Here the use of q-ary sequences for $q > 2$ will have its importance. We thus consider that Turing degrees are defined over the union of the q-ary sequences for all integers $q > 1$.

2.1 Concentration inequalities

We briefly recall here the few concentration inequalities that will be used in this paper.

Definition 2.1. For $q > 1$, for a given $n \in \mathbb{N}$ and two strings $\sigma_1, \sigma_2 \in q^n$, the Hamming distance between σ_1 and σ_2 is the number of positions where σ_1 and σ_2 differ. We denote by $\delta(\sigma_1,\sigma_2)$ the normalized hamming distance:

$$\delta(\sigma_1,\sigma_2) = \frac{\#\{i < n : \sigma_1(i) \neq \sigma_2(i)\}}{n}$$

For $q > 1$, $n > 0$ and σ a q-ary string of length n, Hoeffding's inequality bounds the measure of the set of q-ary strings of the same length, whose Hamming distance with σ is smaller than $1 - 1/q - \varepsilon$ or larger than $1 - 1/q + \varepsilon$ (with $\varepsilon > 0$ such that both values are greater than 0 and smaller than 1):

$$\lambda(\{\tau \mid \delta(\sigma,\tau) < 1 - 1/q - \varepsilon\}) \leq e^{-2\varepsilon^2 n} \tag{1}$$

$$\lambda(\{\tau \mid \delta(\sigma,\tau) > 1 - 1/q + \varepsilon\}) \leq e^{-2\varepsilon^2 n} \tag{2}$$

In the first case, the set $\{\tau \mid \delta(\sigma,\tau) < 1-1/q-\varepsilon\}$ is the *Hamming ball* of radius $1-1/q-\varepsilon$ and centered in σ, that we will denote with $B(\sigma, 1-1/q-\varepsilon)$. There are known slightly sharper bounds in this case:

$$\lambda(B(\sigma, 1-1/q-\varepsilon)) \leq q^{-n(1-H_q(1-1/q-\varepsilon))} \tag{3}$$

where $H_q(\alpha)$ is the q-ary entropy function defined by:

$$H_q(\alpha) = \alpha \log_q(q-1) - \alpha \log_q(\alpha) - (1-\alpha)\log_q(1-\alpha)$$

Note that $H_q(0) = 0$, that $H_q(1-1/q) = 1$ and that H_q is strictly increasing on $[0, 1-1/q]$.

Equations (1), (2), (3) will be referred to in this paper as the *concentration inequalities*. The reader can see [14] for (1) and (2), and chapter 1 of [23] for (3).

2.2 Preliminaries on coarse computability

The notion of coarse computability received quite a lot of recent attention by various authors (see for example [1] and [13]).

Definition 2.2. A sequence X is *coarsely computable* if there is a computable sequence A such that the lim inf of the frequency of positions n on which $X(n) = A(n)$, equals 1. More formally, let us introduce the function:

$$\underline{\rho}(X,A) = \liminf_n \; 1 - \delta(X\restriction_n, A\restriction_n)$$

The sequence X is coarsely computable if for some computable sequence A we have $\underline{\rho}(X,A) = 1$.

A real number can naturally be assigned to non-coarsely computable objects. This number can be seen as an indication of how far the object is from being coarsely computable.

$$\gamma(X) = \sup_{A \text{ computable}} \underline{\rho}(X,A)$$

We will refer to this as the γ value of X. Note that for any $q > 1$ and $X \in q^{\mathbb{N}}$, the supremum in the above definition is reached equivalently by considering only computable q-ary sequences. Andrews, Cai, Diamondstone, Jockusch and Lempp [1] had the interesting idea to define a similar value for Turing degrees, which indicates how far a degree is from being coarsely computable. Here on the contrary, the definition depends on the base. We first present the definition only for binary sequences, as it was defined first. We will discuss later the case of q-ary sequences for $q > 2$.

$$\Gamma(d) = \inf\{\gamma(X) \colon X \in 2^{\mathbb{N}} \text{ is in the Turing degree } d\}$$

This will be referred to in this paper as the Γ value of d. In practice we will often write $\Gamma(X)$ for a set $X \in 2^{\mathbb{N}}$ to mean $\Gamma(d)$ where d is the Turing degree of X. It is easy to see that one can equivalently consider $\Gamma(X)$ to be the infimum over the values $\gamma(Y)$ for every $Y \leq_T X$, rather than just every $Y \equiv_T X$. The reason is that given any $Y <_T X$, we can add to the sequence Y all the information about X at some very sparse computable set of positions, giving a new set Turing equivalent to X, with the same γ value as Y's.

The Γ question is: which real numbers can be realized by the Γ value of a degree? Hirschfeldt, Jockusch, McNicholl and Schupp [13] showed that every real number r can be realized by the γ value of a binary sequence. Their results also show that $\Gamma(X) > 1/2$ if and only if X is computable and thus $\Gamma(X) = 1$ and gave examples of sequences X with $\Gamma(X) = 0$. Then Andrews et al. [1] gave examples of sequences X with $\Gamma(X) = 1/2$. Monin and Nies [19] then provided new examples of sequences X with $\Gamma(X) = 0$, and new examples of sequences X with $\Gamma(X) = 1/2$. The work of all these authors had left so far open the existence of sequences X with $0 < \Gamma(X) < 1/2$. This paper achieve to fully answer the Γ question, by showing that if $\Gamma(X) < 1/2$ then we must have $\Gamma(X) = 0$. Thus only the reals 0, $1/2$ and 1 can be realized by the Γ value of a degree.

Monin and Nies also defined Γ values for bases other than 2. For an integer $q \geq 2$ and $X \in q^{\mathbb{N}}$ we define the value $\Gamma_q(X)$ as before except we now consider an infimum over elements of $q^{\mathbb{N}}$ which are Turing equivalent to X. Finally for a (non rational) real $r \in \mathbb{R}$ we define $\Gamma_q(r)$ to be $\Gamma_q(X)$ for $X \in q^{\mathbb{N}}$ the canonical representation of r in base q.

Monin and Nies showed that for any $q \geq 2$ and any $X \in q^{\mathbb{N}}$, we have $\Gamma_q(X) > 1/q$ iff $\Gamma_q(X) = 1$ iff X is computable. They showed that for any known example of real r such that $\Gamma_2(r) = 1/2$, we also have $\Gamma_q(r) = 1/q$ for any $q \geq 2$. They asked whether this is always the case. We will also answer here this question in the affirmative.

2.3 Preliminaries on error-correcting codes

An *encoding function* with parameters k, n is a function $E : q^k \to q^n$ that maps a q-ary string m of length k into a longer, redundant q-ary string $E(m)$ of length n. An input m of the function is referred to as the *message*, whereas its output $E(m)$ is referred to as a *codeword*. The *error-correcting code* itself, or simply the *code*, is defined to be the image of the encoding function. In other words, it is the set of all codewords which are used to encode the various messages.

In practice, we often work directly with subsets $C \subseteq q^n$ of size q^k. The encoding function is then induced by taking any enumeration of C, and such a set is said to be a $(n, k)_q$-code.

A key parameter of a $(n, k)_q$-code C is its *distance*, which refers to the smallest Hamming distance between two codewords. We write $d(C)$ for the distance of C and $\delta(C)$ for its normalized distance, $d(C)/n$. If d is the distance of C, the receiver of a codeword would be able to correct up to $d/2$ errors (assuming he knows the encoding function): If the number of errors is smaller than $d/2$, the originally transmitted string is simply given by the codeword that is the closest to the received message.

Another important parameter of a code is its *rate*: For the sake of efficiency, one would like the length n of a codeword not to be much longer than the length k of messages we have to transmit. The rate is the quantity k/n, and measures, in some sense, the amount of redundancy added by the encoding.

The following proposition is almost trivial and says that it is possible to build a code with good rate (independent from the messages' length) and with relative distance as close to $1 - 1/q$ as we want (in the following $\lfloor r \rfloor$ for a real r is the largest integer smaller than or equal to r).

Proposition 2.3 (Asymptotic Gilbert bound [23])**.** *Let $q > 1$. Let ε with $0 < \varepsilon < 1 - 1/q$, let $\alpha = 1 - H_q(1 - 1/q - \varepsilon)$ and $n \in \mathbb{N}$. There exists a $(n, \lfloor \alpha n \rfloor)_q$-code C of normalized distance $\delta \geq 1 - 1/q - \varepsilon$.*

Proof. Suppose that we have a set $C \subseteq q^n$ such that for every string $\tau \in q^n$, there exists $\sigma \in C$ with $\delta(\tau, \sigma) < 1 - 1/q - \varepsilon$. Thus $\bigcup_{\sigma \in C} B(\sigma, 1 - 1/q - \varepsilon) = q^{\mathbb{N}}$ and in particular $\sum_{\sigma \in C} \lambda(B(\sigma, 1 - 1/q - \varepsilon)) \geq 1$. By the concentration inequality (3) above, $\lambda(B(\sigma, 1 - 1/q - \varepsilon)) \leq q^{-n(1-H_q(1-1/q-\varepsilon))}$ which implies that

$$|C| q^{-n(1-H_q(1-1/q-\varepsilon))} \geq 1$$

Thus we must have at least $q^{\lfloor n(1-H_q(1-1/q-\varepsilon)) \rfloor}$ elements in C.

It follows that the code satisfying the proposition can be built in a greedy manner: at any step, put in the code any string such that its relative distance to every previously picked string is large enough, until this is not possible. □

From the previous proposition, for every $\varepsilon > 0$ as small as we want, there exists $\alpha > 0$ such that we can build a $(n, \lfloor \alpha n \rfloor)_2$-code for any $n \in \mathbb{N}$ with normalized distance of at least $1/2 - \varepsilon$. Thus we can correct as close as we want to $1/4$ of errors. It is not possible with binary codes to correct more errors, and in general, with q-ary codes, to correct more than $(1 - 1/q)/2$ errors: By the well known Plotkin bound, the maximal number of strings of length n with pairwise normalized distance larger than $1 - 1/q + \varepsilon$ is a constant independent of n. This will not be good enough for us, and to solve the Γ question, we will need to be able to correct up to a ratio of $1/2$ errors for binary codes.

We will then use an alternate notion of decoding called *list decoding*, proposed independently by Elias [8] and Wozencraft [24] in the late 50s. List decoding allows the decoder to output a list of possible codewords rather than a unique one. Even when constrained to output a relatively small number of answers, list decoding permits recovery from errors well beyond the $d/2$ barrier that comes with unique decoding.

This is formally done in the following theorem, known as the *list decoding capacity theorem*. It says that for any ε, it is possible to perform list decoding with lists of constant size and successfully correct up to a ratio of $1 - 1/q - \varepsilon$ errors. Moreover, the rate of the code can be made as close as we want to $1 - H_q(1 - 1/q - \varepsilon)$ by taking larger and larger lists.

The word "capacity" in the theorem's name refers to the quantity $1 - H_q(1 - 1/q - \varepsilon)$ and goes back to the study of noisy channels. The

reader can see for instance chapter 8 of [7] for more information about the capacity of noisy channels.

Theorem 2.4 (The list decoding capacity theorem [9]). *Let $q > 1$. Let ε with $0 < \varepsilon < 1 - 1/q$ and $n \in \mathbb{N}$. For any $L \in \mathbb{N}$ and $0 < \alpha < 1 - H_q(1 - 1/q - \varepsilon) - 1/L$, there exists a set C of $q^{\lfloor \alpha n \rfloor}$ many q-ary strings of length n such that for any q-ary string σ of length n, there are at most L elements τ of C with $\delta(\sigma, \tau) \leq 1 - 1/q - \varepsilon$.*

Note that for $0 < \varepsilon < 1-1/q$, we always have $0 < H_q(1-1/q-\varepsilon) < 1$ and then one can always find suitable α and L.

The version of this theorem with $q = 2$ will suffice to solve the Γ question in base 2. To solve the Γ question in bases larger than 2, we will need a variation of the list decoding theorem. This variation will be introduced later, in Section 4.1. The reader who is interested in the proof of the list decoding theorem can also refer to the proof we will give of its variation, which is very similar.

3 The possible Γ values

3.1 Previous work

Denis Hirschfeldt, Carl Jockusch, Timothy McNicholl, and Paul Schupp showed the following theorem:

Theorem 3.1 ([13]). *Let $X \in 2^{\mathbb{N}}$. If $\Gamma(X) > 1/2$ then X is computable and $\Gamma(X) = 1$.*

Informally, given a binary sequence X, the idea is to compute from it another sequence Y where each bit of X is repeated in Y a number of times much larger than the sum of the number of times each previous bit is repeated. As some computable sequence R must almost always agree with Y more than half of the time, it is possible to use a so called "majority vote" technique: With at most finitely many exceptions, the bit of X is the one that occurs the most in R when looking at the positions where this bit is repeated in Y. Andrews, Cai, Diamondstone, Jockush and Lempp showed the first part of the following theorem (the second part being from Hirschfeldt et al.):

Theorem 3.2 ([1, 13]). *There exists $X \in 2^{\mathbb{N}}$ such that $\Gamma(X) = 1/2$. There exists $X \in 2^{\mathbb{N}}$ such that $\Gamma(X) = 0$.*

Andrews et al. [1] gave two distinct examples of sequences with a Γ value of $1/2$, and two distinct examples of sequences with a Γ value of 0 (as mentioned in the introduction, the sequences of non-computably dominated or PA degree). Later Monin and Nies identified a unique notion covering the possible known examples of sequences having a Γ value of $1/2$, and they identified a unique notion covering possible known examples of sequences having a Γ value of 0: being respectively non-weakly Schnorr engulfing and infinitely often equal with bound $2^{(2^n)}$.

We won't give here more details about the property of being weakly Schnorr engulfing and having a Γ value of $1/2$. The reader can refer to [19] for more details on the subject. The notion of being infinitely often equal with bound h, introduced in [19], was new and appears to be a key step in the resolution of the Γ question.

3.2 Being infinitely often equal

Definition 3.3. Given a bound $H : \mathbb{N} \mapsto \mathbb{N}$ we say that $f : \mathbb{N} \mapsto \mathbb{N}$ is *H-infinitely often equal* (or H-i.o.e.) if f equals infinitely often every computable function strictly bounded by H. A sequence A is of H-i.o.e. degree if A Turing computes an H-i.o.e. function.

Note that as long as $H \geq 2$, we obtain the same notion by replacing "infinitely often" by "at least once" in the definition: If f equals only finitely often to some computable function bounded by $H \geq 2$, then there also exists a computable function bounded by H which never equals f. Therefore if f equals every computable function bounded by H at least once, it must equals every computable function bounded by H infinitely often.

If A is of $2^{(2^n)}$-i.o.e. degree, then $\Gamma(A) = 0$. This uses a very simple but key idea, that will also be reused to solve the Γ question. In the following $f(2n)$ and $H(2n)$ are short notations to denote respectively the functions $n \mapsto f(2n)$ and $n \mapsto H(2n)$.

Proposition 3.4 ([19]). *Let $H : \mathbb{N} \to \mathbb{N}$. Let $f : \mathbb{N} \to \mathbb{N}$ be bounded by H. If f is H-i.o.e., then $f(2n)$ must be $H(2n)$-i.o.e. or $f(2n+1)$ must be $H(2n+1)$-i.o.e.*

Proof. By contrapositive, suppose there exist computable functions $g_1 \leq H(2n)$ and $g_2 \leq H(2n+1)$ such that $g_1(n) \neq f(2n)$ for all but finitely many n and such that $g_2(n) \neq f(2n+1)$ for all but finitely many n. Then the computable function g such that $g(2n) = g_1(n)$ and $g(2n+1) = g_2(n)$ equals to f only finitely often, and f is then not H-i.o.e. □

Corollary 3.5 ([19]). *If a sequence A is of $2^{(2^n)}$-i.o.e. degree, then for any $a \in \mathbb{N}^*$, the sequence A is of $2^{(a^n)}$-i.o.e. degree.*

Proof. Suppose that A is of $2^{(2^n)}$-i.o.e. degree. Then from Proposition 3.4, A is of $2^{(2^{2n})}$-i.o.e. degree or A is of $2^{(2^{2n+1})}$-i.o.e. degree. Also it should be clear that if a function is H-i.o.e., then it is also H'-i.o.e. for any $H' \leq H$. In any case we have that A is of $2^{(2^{2n})}$-i.o.e. degree.

By iterating the same argument, we have that A is of $2^{(2^{kn})}$-i.o.e. degree for any $k \in \mathbb{N}^*$, and therefore that A is of $2^{(a^n)}$-i.o.e. degree for any $a \in \mathbb{N}^*$. □

Theorem 3.6 ([19]). *If a sequence A is of $2^{(2^n)}$-i.o.e. degree, then $\Gamma(A) = 0$.*

Proof. By Corollary 3.5 it is enough to show that if A is of $2^{(a^n)}$-i.o.e. degree for a natural number $a > 1$, then $\Gamma(A) \leq 1/a$. Let $a > 1$ be an integer and suppose that A computes a $2^{(a^n)}$-i.o.e. function f, that we can bound without loss of generality by $2^{(a^n)}$. Let J_n be the integer interval $[a^{n-1}, a^n)$ for $n > 0$, and let $J_0 = \{0\}$. Note that the length of J_n is at most of a^n. We define the f-computable sequence B such that $B \upharpoonright_{J_n}$ is equal to the string corresponding to the n-th value of f. Consider now any computable sequence X and its bitwise complement $\overline{X}$, together with the function g which to n associates the integer corresponding to the string $\overline{X} \upharpoonright_{J_n}$. As f equals g infinitely often, then for infintely many n we have that $\overline{X}$ and B agree on every point in J_n, and thus that X and B disagree on every point in J_n. For any such n, the density below a^n of the set of points where X and B agree is at most a^{n-1}/a^n. As this is true for every computable X we have $\gamma(B) \leq 1/a$ and hence $\Gamma(A) \leq 1/a$. □

We now introduce a concept related to infinite often equality, which directly comes from inspecting aspects of the Γ question. This notion is also what will connect the Γ question to the field of error-correcting codes:

Definition 3.7. Let $H : \mathbb{N} \to \mathbb{N}$ be a computable function and $\alpha \in [0,1]$. A sequence of binary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ where $|\sigma_n| = H(n)$ is $2^{H(n)}$-infinitely often α-equal (or i.o.α-e.) if for every computable sequence of binary strings $\{\tau_n\}_{n\in\mathbb{N}}$, with $|\tau_n| = |\sigma_n|$, we have:

$$\limsup_n 1 - \delta(\sigma_n, \tau_n) \geq \alpha$$

Informally, we want $\{\sigma_n\}_{n\in\mathbb{N}}$ to be equal infinitely often, on a fraction of at least α bits, to every computable sequence $\{\tau_n\}_{n\in\mathbb{N}}$ where $|\tau_n| = H(n)$.

3.3 $\Gamma(X) < 1/2$ implies $\Gamma(X) = 0$

The γ value of a sequence is defined by considering longer and longer initial segments. In order to solve the question, we would like to consider successive chunks of bits instead of initial segments. This is done in the following theorem, which will also be used to solve the Γ question for bases other than 2.

Theorem 3.8. *Let $\varepsilon > 0$. Let $q \geq 2$ and $X \in q^{\mathbb{N}}$. We have (2) implies (1):*

1. $\Gamma_q(X) \geq 1/q - \varepsilon$.
2. *For every $k \in \mathbb{N}^*$, for any X-computable sequence of q-ary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ where $|\sigma_n| = \lfloor 2^{n/k} \rfloor$, there is a computable sequence of q-ary strings $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$ such that for every n, we have $1 - \delta(\sigma_n, \tau_n) \geq 1/q - \varepsilon$.*

Proof. Suppose that (2) is true. Consider any sequence $Y \in q^{\mathbb{N}}$ computed by X and fix $c \in \mathbb{N}^*$. Pick k such that for n large enough $\lfloor 2^{(n+1)/k} \rfloor$ is smaller than $1/c$ times the sum of $\lfloor 2^{i/k} \rfloor$ for $i \leq n$. Such a thing is always possible: By the sum of the geometric series we have that $2^{(n+1)/k} - 1$ is equal to $(2^{1/k} - 1)\sum_{i=0}^{n} 2^{i/k}$. The larger k is, the closer to 0 the quantity $(2^{1/k} - 1)$ is. To have $2^{(n+1)/k} \leq 1/c \sum_{i=0}^{n} \lfloor 2^{i/k} \rfloor$ we need to compensate the imprecisions due to the additional -1 constant and due to the use of the floor function. This is easily done for instance with k such that $(2^{1/k} - 1) < 1/(2c)$. Split then Y into chunks of bits $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = \lfloor 2^{n/k} \rfloor$.

Thus the length of σ_{n+1} is smaller than $1/c$ times the sum of the length of $|\sigma_i|$ for $i \leq n$. As $\{\sigma_n\}_{n\in\mathbb{N}}$ is an X-computable sequence satisfying (2), there must exist a computable sequence $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$ such that for every n, we have $1 - \delta(\sigma_n, \tau_n) \geq 1/q - \varepsilon$. Let R be the concatenation of the τ_n. Let $H(n) = \sum_{i=0}^{n} |\sigma_i|$. For every n we have $1 - \delta(X\upharpoonright_{H(n)}, R\upharpoonright_{H(n)}) \geq 1/q - \varepsilon$.

We shall now show that the above quantity does not drop too much for prefixes of length $H(n) + i$ for $i < H(n+1) - H(n)$. By hypothesis, among the $H(n) + i$ first bits (for n large enough), there are at least $H(n)(1/q - \varepsilon)$ bits which are guessed correctly by R. Also the number of total bits is at most $H(n+1)$, which is at most $H(n) + H(n)/c$. Thus for each $i < H(n+1) - H(n)$ the fraction of bits which are guessed correctly before $H(n) + i$ is at least:

$$\frac{H(n)(1/q - \varepsilon)}{H(n) + H(n)/c}$$

Which equals:

$$\frac{1/q - \varepsilon}{1 + 1/c}$$

Thus $\gamma(Y) \geq \frac{1/q-\varepsilon}{1+1/c}$. We can repeat this operation for c larger and larger, making lower bounds on the γ values of Y closer and closer to $1/q - \varepsilon$. If this happens for every $Y \in q^{\mathbb{N}}$ computable by X, we then have $\Gamma_q(X) \geq 1/q - \varepsilon$. □

Note that the part (1) implies (2) in the previous theorem is also true, and will follow from the proof of Theorem 4.3, which answers the Γ question in base q: In the proof of Theorem 4.3, we deduce from $\neg(2)$ that X must have a Γ_q value of 0, then making $\neg(1)$ true.

The following corollary relies on the fact that for binary sequences, if we know one value a bit does not take, we actually know the value of the bit. We will need to elaborate a bit more for q-ary sequences where $q > 2$.

Corollary 3.9. *Let $\varepsilon > 0$. Let $X \in 2^{\mathbb{N}}$. We have (1) implies (2):*

1. $\Gamma(X) < 1/2 - \varepsilon$.
2. *There exists $k \in \mathbb{N}^*$, and an X-computable sequence of strings $\{\sigma_n\}_{n\in\mathbb{N}}$, which is $2^{\lfloor 2^{n/k} \rfloor}$-infinitely often $(1/2 + \varepsilon)$-equal.*

Proof. Suppose (1). By Theorem 3.8. There exists $k \in \mathbb{N}^*$, and an X-computable sequence of binary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ where $|\sigma_n| = \lfloor 2^{n/k} \rfloor$, such that for any computable sequence of binary strings $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$, we have $1 - \delta(\sigma_n, \tau_n) < 1/2 - \varepsilon$ for infinitely many n.

Such a sequence must also be $2^{\lfloor 2^{n/k} \rfloor}$-infinitely often $(1/2 + \varepsilon)$-equal: For any computable sequence of strings $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$, we can consider the computable sequence of strings $\{\tau'_n\}_{n\in\mathbb{N}}$ where each τ'_n is the complement bitwise of τ_n. There must be infinitely many n such that $1 - \delta(\sigma_n, \tau'_n) < 1/2 - \varepsilon$ and thus infinitely many n such that $1 - \delta(\sigma_n, \tau_n) > 1/2 + \varepsilon$. □

Again, in the previous corollary, the direction (2) implies (1) is also true and is proved similarly, but using the direction (1) implies (2) of Theorem 3.8. Using this last corollary, we can now prove with the help of the list decoding theorem, that if the Γ value of a sequence is strictly smaller than $1/2$, then it must actually be 0. Before we prove Theorem 3.11, we need to introduce a simple technical tool:

Definition 3.10. Let $\{T_n\}_{n\in\omega}$ be a sequence of finite sets of integers, that is, for each n we have $T_n \subseteq \mathbb{N}$ and $|T_n| < \infty$. Such a sequence $\{T_n\}_{n\in\omega}$ is called a *trace*. We say that a trace *captures infinitely often* a function $g : \mathbb{N} \to \mathbb{N}$, if there are infinitely many n such that $g(n) \in T_n$.

Finally we say that a trace $\{T_n\}_{n\in\omega}$ is computable (resp. A-computable) if each T_n is computable (resp. A-computable) uniformly in n, in a strong sense : An algorithm should tell us at once all the integers which belong to T_n. Formally, there must exist a computable (resp. A-computable) function $f : \mathbb{N} \to \mathbb{N}$ such that for every n, $f(n) = 2^{x_0} + \cdots + 2^{x_r}$ and $T_n = \{x_0, \ldots, x_r\}$.

Theorem 3.11. *Let $X \in 2^{\mathbb{N}}$. The following are equivalent:*

1. $\Gamma(X) < 1/2 - \varepsilon$ *for some* $\varepsilon > 0$.
2. *X is of $2^{(2^n)}$-i.o.e. degree*
3. $\Gamma(X) = 0$.

Proof. (2) → (3) is given by Theorem 3.6. (3) → (1) is trivial. We prove here (1) → (2). Suppose $\Gamma(X) < 1/2 - \varepsilon$ for some $\varepsilon > 0$. In particular from Corollary 3.9, X computes a sequence of binary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = \lfloor 2^{n/k} \rfloor$ and which is $2^{\lfloor 2^{n/k} \rfloor}$-i.o.$(1/2 + \varepsilon)$-e. for some $k \in \mathbb{N}^*$. Let us pick ε' with $0 < \varepsilon' < \varepsilon$.

Using the list decoding theorem (Theorem 2.4 with the special case $q = 2$), we pick $L \in \mathbb{N}$ and $0 < \alpha < 1$ such that for any n, there exists a collection C_n of $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$ strings of length $\lfloor 2^{n/k} \rfloor$, such that no string σ of length $\lfloor 2^{n/k} \rfloor$ has a relative Hamming distance smaller than or equal to $1/2 - \varepsilon'$ with more than L strings of C_n. Note that such a collection of strings C_n is computable uniformly in n. Fix an enumeration $\tau_0^n, \tau_1^n, \dots$ of the elements of each C_n.

We define the following X-computable trace $\{T_n\}_{n\in\mathbb{N}}$: For any n, T_n is the collection of integers i such that the $\delta(\sigma_n, \tau_i^n) \leq 1/2 - \varepsilon'$. Note that each T_n is X-computable uniformly in n and that $|T_n| \leq L$ (possibly T_n is also empty). Note also that the values of T_n are bounded by $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$.

We claim that every computable function $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \leq 2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$, is captured infinitely often by $\{T_n\}_{n\in\mathbb{N}}$. Indeed, given such a computable function g bounded by $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$, let us consider the computable sequence of strings $\{\rho_n\}_{n\in\mathbb{N}}$ with $|\rho_n| = \lfloor 2^{n/k} \rfloor$ defined by $\rho_n = \tau_i^n$ if $g(n) = i$. As $\{\sigma_n\}_{n\in\mathbb{N}}$ is $2^{\lfloor 2^{n/k} \rfloor}$-i.o.$(1/2+\varepsilon)$-e. and as $\varepsilon' < \varepsilon$, there must exist infinitely many values n such that $\delta(\sigma_n, \rho_n) \leq 1 - (1/2 + \varepsilon')$. But then by definition of $\{T_n\}_{n\in\mathbb{N}}$, for each of these n, we must have $g(n) \in T_n$.

Thus for every computable function g which is bounded by $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$, there exist infinitely many n such that $g(n) \in T_n$. Now, by the same argument than the one of Proposition 3.4, either $\{T_{2n}\}_{n\in\mathbb{N}}$ must capture infinitely often every computable function bounded by $2^{\lfloor \alpha \lfloor 2^{2n/k} \rfloor \rfloor}$, or $\{T_{2n+1}\}_{n\in\mathbb{N}}$ must capture infinitely often every computable function bounded by $2^{\lfloor \alpha \lfloor 2^{(2n+1)/k} \rfloor \rfloor}$. In either case, $\{T_n\}_{n\in\mathbb{N}}$ can compute a trace that captures infinitely often every computable function bounded by $2^{\lfloor \alpha \lfloor 2^{2n/k} \rfloor \rfloor}$.

By iterating the same idea, X can compute a trace $\{T'_n\}_{n\in\mathbb{N}}$ with $|T'_n| \leq L$, which captures infinitely often every computable function bounded by 2^{L2^n}. We can also without loss of generality assume that each element of each T'_n is bounded by 2^{L2^n}, and thus assume that each element of each T_n is coded on exactly $L2^n$ bits.

We now use the fact that $|T'_n| \leq L$ for every n, to compute using T'_n a function $h \leq 2^{(2^n)}$ which equals infinitely often every computable function bounded by $2^{(2^n)}$. First for every n, we add if necessary some elements in T'_n such that $|T'_n| = L$. Then we view each element e_i of T'_n as an L-tuple $\langle e_i^1, \dots, e_i^L \rangle$. Formally e_i^j is the j-th chunk of 2^n consecutive bits coding e_i. Consider the L distinct X-computable functions $h_1, \dots, h_L$ given by $h_i(n) = e_i^i$ where $e_i = \langle e_i^1, \dots, e_i^i, \dots, e_i^L \rangle$ is the i-th element of T'_n. We claim that at least one h_i is $2^{(2^n)}$-i.o.e. Suppose otherwise, and consider the L computable functions $p_1, \dots, p_L$ witnessing that: for each $i \leq L$, p_i never equals h_i. Then the computable function $p(n) = \langle p_1(n), \dots, p_L(n) \rangle$ is never captured by $\{T'_n\}_{n\in\mathbb{N}}$, as for every n, the i-th component of $p(n)$ (seen as an L-tuple) is different from the i-th component of the i-th element of T'_n (seen as an L-tuple). This contradicts our hypothesis, and then at least one h_i must be $2^{(2^n)}$-i.o.e. Note that h_i is computable from X. This concludes the proof. □

4 Related questions

4.1 The Γ question for bases other than 2

In this section we show how to deal with bases other than 2, using variants of the list decoding theorem. We start by using list decoding to simplify the proof of the following result from Monin and Nies [19]: If $\Gamma_q(X) > 1/q + \varepsilon$, then X is computable. The case $q = 2$ was done by Hirschfeldt, Jockusch, McNicholl and Schupp [13] using a majority vote technique. Monin and Nies called on a quite unrelated and difficult theorem from Kummer [18] to generalize to bases other than 2. The use of list decoding gives a more direct and simple proof:

Theorem 4.1 ([19]). *Let $q > 2$. Let $\varepsilon > 0$. Let $X \in q^{\mathbb{N}}$. If $\Gamma_q(X) > 1/q + \varepsilon$, then X is computable.*

Proof. Suppose $\Gamma_q(X) > 1/q + \varepsilon$. Let $F(n) = (3/\varepsilon)^n$. By the sum of the geometric series, we have $F(n) > (2/\varepsilon) \sum_{i=0}^{i<n} F(i)$ for every n. We let $F'(n) = \sum_{i\leq n} F(n)$.

Using the list decoding theorem, we pick $L \in \mathbb{N}$ and $0 < \alpha < 1$ such that for any n, there exist a collection C_n of $q^{\lfloor \alpha F(n) \rfloor}$ strings of length $F(n)$, such that no string σ of length $F(n)$ has a relative Hamming distance smaller than $1 - 1/q - \varepsilon/2$ with more than L strings of C_n.

Let the X-computable sequence of strings $\{\sigma_n\}_{n\in\mathbb{N}}$ with $|\sigma_n| = F(n)$ be the encoding of $X \upharpoonright_{\lfloor \alpha F'(n) \rfloor}$ using the code C_n. Let Y be the concatenation of the strings $\{\sigma_n\}_{n\in\mathbb{N}}$. As $\gamma_q(Y) > 1/q + \varepsilon$, there must exist a computable sequence of strings $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$ and such that $\delta(\tau_n, \sigma_n) < 1 - 1/q - \varepsilon/2$ for every n. Suppose otherwise, then for every computable sequence $\{\tau_n\}_{n\in\mathbb{N}}$ with $|\tau_n| = |\sigma_n|$, there exists infinitely many n such that τ_n agrees on less than $1/q + \varepsilon/2$ bits with σ_n. As $|\tau_n|$ is larger than $2/\varepsilon$ times $\sum_{i=0}^{i<n} |\tau_i|$, a simple computation shows that also the concatenation of the strings τ_i for $i \leq n$ agrees with less than $1/q + \varepsilon$ bits with the concatenation of the strings σ_i for $i \leq n$. In particular it would contradict that $\gamma_q(Y) > 1/q + \varepsilon$.

Now to compute X we proceed as follow: For every n, there are at most L strings of C_n which agree with τ_n on more than $1/q + \varepsilon/2$ bits. Among them we know there must be the correct image of $X \upharpoonright_{\lfloor \alpha F'(n) \rfloor}$ by the code C_n. Using $\{\tau_n\}_{n\in\mathbb{N}}$, we can compute for every n the sets T_n consisting of all the preimages of the strings of C_n which agree with τ_n on more than $1/q + \varepsilon/2$ bits. We can then compute the tree $T \subseteq q^{<\mathbb{N}}$ such that $\sigma \in T$ iff $\exists n$ such that $\sigma \in T_n$ and such that $\forall m \leq n$ σ extends a string in T_m. For every n this tree contains at most L strings of length n. Also we must have $X \in [T]$. In particular, there must be a prefix σ of X such that X is the only infinite path of T extending σ. This makes X computable by a well known following argument : suppose we have computed τ with $\sigma \prec \tau \prec X$, to know if the next bit is 0 or 1, we look for the smallest n such that there are either no extensions of $\tau 0$ of length n in T, or no extensions of $\tau 1$ of length n in T. By Koenig's lemma, as X is the only extension of σ, the algorithm will find such an n and therefore will know the next bit of X. □

Monin and Nies [19] asked whether for any real r and any $q > 2$, we must have $\Gamma_q(r) = 1/q$ iff $\Gamma(r) = 1/2$. We now answer the question by showing that if $\Gamma_q(r) < 1/q$, then r Turing computes a 2^{2^n}-i.o.e. function which implies $\Gamma(r) = 0$. One can easily verify that $\Gamma_{b+1}(r) \leq \Gamma_b(r)$, which allows us to conclude that $\Gamma_q(r) = 0$.

In base 2 we use Corollary 3.9 which comes with the symmetry of $2^{\mathbb{N}}$: If σ is far from a string τ, then σ is close to $\bar{\tau}$, the complement bitwise of τ. Of course, no such thing is possible in base $q > 2$. But what is important is the use of unlikely events: in base 2 it is unlikely to be far from a given string τ, and symmetrically it is as unlikely to be close from $\bar{\tau}$. We actually do not need this symmetry. We can simply use the fact that it is unlikely in base q to be far from a string τ.

This is why we need a variation of the list decoding theorem, which does not correspond anymore to a concrete problem as it was the case with error-correcting codes : Instead of assuming that we never have too much error during our transition, we now assume that we always have a lot. Being wrong many times is as unlikely as being correct many times. Thus list decoding is also possible when the number of errors is always sufficiently big.

Theorem 4.2 (Variation on list decoding)**.** *Let $q > 2$. Let $\varepsilon > 0$ and $n \in \mathbb{N}$. For $L \in \mathbb{N}$ large enough and $\alpha \in \mathbb{R}^+$ small enough, there exists a set C of $q^{\lfloor \alpha n \rfloor}$ many q-ary strings of length n such that for any q-ary string σ of length n, there are at most L elements τ of C such that $\delta(\sigma, \tau) > 1 - 1/q + \varepsilon$.*

Proof. We prove that if we pick at random the strings in C, the theorem is true with positive probability. For a string $\sigma \in q^n$, let us define the somehow opposite of a Hamming ball : $D(\sigma, 1-1/q+\varepsilon) = \{\tau \in 2^n \mid \delta(\sigma, \tau) > 1-1/q+\varepsilon\}$. By the second concentration inequality, we have $\lambda(D(\sigma, 1-1/q+\varepsilon)) \leq q^{-n\beta}$ with $\beta = 2\varepsilon^2 \log_q(e)$ (Note that a proof of the list decoding theorem can be done like the present proof, but using the bound of the Hamming ball $B(\sigma, 1 - 1/q - \varepsilon)$) given by the third concentration inequality).

Consider L large enough and α small enough, such that $\alpha < \beta - 1/L$. Let $k = \lfloor \alpha n \rfloor$ and let C be a collection of q^k strings picked at random. For any subset of $L + 1$ of these strings, the probability that a given string σ has a relative Hamming distance larger than $1 - 1/q + \varepsilon$ with each of them is bounded by $q^{-\beta n(L+1)}$. Thus the probability that a given σ has a relative Hamming distance larger than $1 - 1/q + \varepsilon$ with all the strings in any possible subsets of size $L + 1$ of C is bounded by $\binom{q^k}{L+1} q^{-\beta n(L+1)}$. And the probability that this happens for any string σ is bounded by $q^n \binom{q^k}{L+1} q^{-\beta n(L+1)}$. The following computation shows that this quantity is smaller than 1:

$$\begin{aligned}
q^n \tbinom{q^k}{L+1} q^{-\beta n(L+1)} &\leq q^n q^{\alpha n(L+1)} q^{-\beta n(L+1)} \\
&\quad \text{(using } k \leq \alpha n\text{)} \\
&\leq q^{-n(L+1)(-\alpha - 1/(L+1) + \beta)} \\
&\leq q^{-n(L+1)(-\beta + 1/L - 1/(L+1) + \beta)} \\
&\quad \text{(using } \alpha < \beta - 1/L\text{)} \\
&\leq q^{-n(L+1)((L+1-L)/L(L+1))} \\
&\leq q^{-n/L}
\end{aligned}$$

It follows that that for any n, if C is a collection of $q^{\lfloor \alpha n \rfloor}$ strings of length n that we pick at random, the probability that no string σ of length n has a relative Hamming distance bigger than $1 - 1/q + \varepsilon$ with more than L strings of C is positive and goes to 1 as n goes to infinity. In particular, for any n, there exists such a collection of strings. □

It is now clear how to call on this new list decoding theorem, to solve the Γ question in any base:

Theorem 4.3. *Let $q > 2$. Let $\varepsilon > 0$. Let $X \in q^{\mathbb{N}}$. Suppose $\Gamma_q(X) < 1/q - \varepsilon$. Then X computes a $2^{(2^n)}$-i.o.e. function.*

Proof. From Theorem 3.8, there must exist $k \in \mathbb{N}^*$ and an X-computable sequence of q-ary strings $\{\sigma_n\}_{n\in\mathbb{N}}$ where $|\sigma_n| = \lfloor 2^{n/k} \rfloor$, such that for every computable sequence of q-ary strings $\{\tau_n\}_{n\in\mathbb{N}}$, we have $1 - \delta(\sigma_n, \tau_n) < 1/q - \varepsilon$ for infinitely many n.

Using the variation of the list decoding theorem (Theorem 4.2), we pick $L \in \mathbb{N}$ and $0 < \alpha < 1$ such that for any n, there exists a collection C_n of $q^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$ strings of length $\lfloor 2^{n/k} \rfloor$, such that no string σ of length $\lfloor 2^{n/k} \rfloor$ has a relative Hamming distance bigger than $1 - 1/q + \varepsilon$ with more than L strings of C_n. Fix an enumeration $\tau_0^n, \tau_1^n, \ldots$ of the elements of each C_n.

The proof continues similarly to the one in base 2, except that it is now impossible to be far from more than L strings of C_n instead of impossible to be close. We define similarly as in the proof of Theorem 3.11, the following X-computable trace $\{T_n\}_{n\in\mathbb{N}}$: For any n, T_n is the collection of integers i such that the $\delta(\sigma_n, \tau_i^n) > 1 - 1/q + \varepsilon$. As in Theorem 3.11, $\{T_n\}_{n\in\mathbb{N}}$ is X-computable, the values of T_n are bounded by $q^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$ and from Theorem 4.2 we know that $|T_n| \leq L$.

As in the proof of Theorem 3.11, we show that every computable function $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \leq q^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$, is *captured infinitely often* by $\{T_n\}_{n\in\mathbb{N}}$. Indeed, given such a computable function g bounded by $q^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$, consider the computable sequence of q-ary strings $\{\rho_n\}_{n\in\mathbb{N}}$ with $|\rho_n| = \lfloor 2^{n/k} \rfloor$ defined by $\rho_n = \tau_i^n$ if $g(n) = i$. By hypothesis on the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$, there must exist infinitely many values n such that $\delta(\sigma_n, \rho_n) > 1 - 1/q + \varepsilon$. But then by definition of $\{T_n\}_{n\in\mathbb{N}}$, for each of these n, we must have $g(n) \in T_n$.

Now, As in the proof of Theorem 3.11, one easily argue that X computes a trace $\{T'_n\}_{n\in\mathbb{N}}$ with $|T'_n| = L$ which captures infinitely often every computable function bounded by q^{L2^n}. We can also without loss of generality assume that each element of each T'_n is bounded by q^{L2^n}, and thus coded on exactly $L2^n$ bits. As in the proof of Theorem 3.11, one easily shows that $\{T'_n\}_{n\in\mathbb{N}}$ (and thus X) computes a function $h \leq q^{(2^n)}$ which is equal infinitely often to every computable function bounded by $q^{(2^n)}$. Therefore as every $q^{(2^n)}$-i.o.e. function is also a $2^{(2^n)}$-i.o.e. function, X computes a $2^{(2^n)}$-i.o.e. function. □

4.2 The Gamma question in the tt-degrees

A well understood and studied notion in the Turing degrees is the one of being computably dominated. We say that $X \in 2^{\mathbb{N}}$ is computably dominated if every function from $\mathbb{N}$ to $\mathbb{N}$ that X computes is bounded by a computable one. One of the first given examples of sequences with a Γ value of 0 are the non-computably dominated sequences. This class is quite large in several senses (it has measure 1 and it is co-meager). Also for a given sequence X, we have the following well known equivalence [16]:

1. X is computably dominated
2. the truth-table degrees containing X coincides with the Turing degrees containing X.

We say that $X \geq_{tt} f$ if there exists a functional which is total using any oracle, and which gives f when X is used as an oracle. So (2) above means that whenever X is used as an oracle to compute a function, then the functional using X can be made total on every oracle other than X. This property was used for instance in [1] to show that the computably dominated randoms have a Γ value of $1/2$.

As the Γ question was solved already on the Turing degree which are not computably dominated (their Γ value is 0), this gave rise to an analogue of the Γ question in the tt-degrees. The function $\Gamma_{tt}(X)$ is defined like the function $\Gamma(X)$, except the infimum is now taken over all the binary sequences in the tt degrees of X. Denis Hirschfeldt [11] asked about the possible Γ values for the tt-degrees. Our proofs actually also solves the Γ question in the tt-degrees:

The equivalent of Corollary 3.9 for the tt-degrees is the following: If $\Gamma_{tt}(X) < 1/2 - \varepsilon$, then there exists $k \in \mathbb{N}^*$, and a $2^{\lfloor 2^{n/k} \rfloor}$-infinitely often $(1/2 + \varepsilon)$-equal sequence of strings $\{\sigma_n\}_{n \in \mathbb{N}}$ which is truth-table computable from X. Also in the proof of Theorem 3.11, the computation of a trace of at most L values which traces infinitely often every function bounded by $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$ from a $2^{\lfloor 2^{n/k} \rfloor}$-infinitely often $(1/2 + \varepsilon)$-equal sequence of strings is clearly tt. Then the computation (the extraction at a set of computable positions) of such a trace with bound 2^{L2^n} instead of $2^{\lfloor \alpha \lfloor 2^{n/k} \rfloor \rfloor}$ is also clearly tt. Finally the extraction of a $2^{(2^n)}$-i.o.e. function from such a trace is also tt.

References

[1] Uri Andrews, Mingzhong Cai, David Diamondstone, Carl Jockusch, and Steffen Lempp. 2016. Asymptotic density, computable traceability, and 1-randomness. *Fundamenta Mathematicae* 234, 1 (2016), 41–53. http://eudml.org/doc/286624

[2] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)* 45, 3 (1998), 501–555.

[3] László Babai, Lance Fortnow, and Carsten Lund. 1991. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity* 1, 1 (1991), 3–40.

[4] Donald Beaver and Joan Feigenbaum. 1990. Hiding instances in multioracle queries. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 37–48.

[5] Laurent Bienvenu, Adam R Day, and Rupert Hölzl. 2009. From bi-immunity to absolute undecidability. *The Journal of Symbolic Logic* 78, 4 (2009), 1218–1228.

[6] Andrej Bogdanov and Emanuele Viola. 2010. Pseudorandom bits for polynomials. *SIAM J. Comput.* 39, 6 (2010), 2464–2486.

[7] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

[8] Peter Elias. 1957. List decoding for noisy channels. *Technical report, MIT* (1957).

[9] Peter Elias. 1991. Error-correcting codes for list decoding. *Information Theory, IEEE Transactions on* 37, 1 (1991), 5–12.

[10] Matthew Harrison-Trainor. 2017. The Gamma question for many-one degrees. *Annals of Pure and Applied Logic* (2017).

[11] Denis Hirschfeldt. 2016. Some Questions in Computable Mathematics. *To Appear* (2016).

[12] Denis R Hirschfeldt. 2017. Some questions in computable mathematics. In *Computability and Complexity*. Springer, 22–55.

[13] Denis R Hirschfeldt, Carl G Jockusch Jr, Timothy H McNicholl, and Paul E Schupp. 2016. Asymptotic density and the coarse computability bound. *Computability* 5, 1 (2016), 13–27.

[14] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[15] Carl Jockusch and Paul Schupp. 2012. Generic computability, Turing degrees, and asymptotic density. *Journal of the London Mathematical Society* 85, 2 (2012), 472–490.

[16] Carl G Jockusch. 1969. Relationships between reducibilities. *Trans. Amer. Math. Soc.* 142 (1969), 229–237.

[17] Ilya Kapovich, Alexei Myasnikov, Paul Schupp, and Vladimir Shpilrain. 2003. Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra* 264, 2 (2003), 665–694.

[18] Martin Kummer. 1992. A proof of Beigel's cardinality conjecture. *The Journal of Symbolic Logic* 57, 02 (1992), 677–681.

[19] Benoit Monin and André Nies. 2015. A unifying approach to the Gamma question. In *Logic in Computer Science (LICS), 2015 30th Annual ACM/IEEE Symposium on*. IEEE, 585–596.

[20] Dana Scott. 1962. Algebras of sets binumerable in complete extensions of arithmetic. In *Proceedings of the Symposium on Pure and Applied Mathematics*, Vol. 5. 117–121.

[21] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[22] Adi Shamir. 1992. Ip = pspace. *Journal of the ACM (JACM)* 39, 4 (1992), 869–877.

[23] Jacobus Hendricus Van Lint. 2012. *Introduction to coding theory*. Vol. 86. Springer Science & Business Media.

[24] John Wozencraft. 1958. List decoding. *Quarterly Progress Report* 48 (1958), 90–95.