Information and Computation 232 (2013) 1–9

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

A polynomial-time algorithm for computing absolutely normal numbers

Verónica Becher [a,b,*], Pablo Ariel Heiber [a], Theodore A. Slaman [c]

[a] Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
[b] CONICET, Argentina
[c] Department of Mathematics, University of California Berkeley, USA

ARTICLE INFO

Article history:
Received 8 March 2013
Received in revised form 6 July 2013
Available online 22 August 2013

ABSTRACT

We give an algorithm to compute an absolutely normal number so that the first n digits in its binary expansion are obtained in time polynomial in n; in fact, just above quadratic. The algorithm uses combinatorial tools to control divergence from normality. Speed of computation is achieved at the sacrifice of speed of convergence to normality.

1. Introduction

"Show me an absolutely normal number." Émile Borel posed this problem over one hundred years ago, but it still has no fully satisfactory solution. For example, it is unknown whether the numbers π or e or any irrational algebraic number is absolutely normal. Recall that a real number is absolutely normal if the digits in its infinite expansion in each base are distributed uniformly.

One solution to the problem would be to give an algorithm and actually compute each of the digits in the binary expansion of an absolutely normal number, one after the other. The first such algorithm was given by Alan Turing [11,3]. Unfortunately this algorithm is unfeasible because determining the first n digits would require time that is double exponential in n. Another algorithm is the computable reformulation [2] of Wacław Sierpiński's construction [10], which also requires double exponential time. Yet another construction of an absolutely normal number was given by Wolfgang Schmidt [9]. He remarks his number is "clearly defined" and, in fact, it is clearly computable, but its time complexity has not been analyzed.

Jack Lutz and Elvira Mayordomo were the first to announce[1] the existence of an absolutely normal number computable in polynomial time. Santiago Figueira and André Nies reported another proof. See [8] and [6]. Their arguments analyze polynomial-time martingales, a device from the theory of algorithmic randomness, and the means to diagonalize against them.

Here, we give an algorithm that computes an absolutely normal number in polynomial time, indeed just above quadratic. Our algorithm is an efficient variant of Turing's approach on absolutely normal numbers, and as such, uses combinatorial tools to control divergence from normality directly. In [1], we report on the implementation of the algorithm and analysis of its output.

Our algorithm achieves speed of computation at the cost of slowness of convergence to normality. There are known limits on the rate of convergence to normality and there are examples that are nearly optimal [5, Chapter 4]. We are left

* Corresponding author at: Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina.
E-mail addresses: vbecher@dc.uba.ar (V. Becher), pheiber@dc.uba.ar (P.A. Heiber), slaman@math.berkeley.edu (T.A. Slaman).
URLs: http://www.dc.uba.ar/people/profesores/becher/ (V. Becher), http://math.berkeley.edu/~slaman/ (T.A. Slaman).

[1] Seventh International Conference on Computability, Complexity and Randomness, Cambridge, Great Britain, July 2012.

0890-5401/$ – see front matter
http://dx.doi.org/10.1016/j.ic.2013.08.013

with the question of whether the trade-off between rate of computation and rate of convergence to normal is an inherent aspect of any computation of an absolutely normal number or an artifact of our construction.

Question. Is there an absolutely normal number computable in polynomial time having a nearly optimal rate of convergence to normality?

2. Preliminaries

A *base* is an integer b greater than or equal to 2, a *digit* in base b is an element in $\{0,\dots,b-1\}$, and a *block* in base b is a finite sequence of digits in base b. The length of a block x is $|x|$, $x\restriction\ell$ is the subblock of the first ℓ digits of x and $x[i]$ is the ith digit of x; the same notation applies when x is an infinite sequence of digits. A digit d *occurs* in a block x at position i if $x[i]=d$. The number of occurrences of the digit d in the block x is $\mathrm{occ}(x,d)=\#\{i\colon x[i]=d\}$. If x and u are blocks, xu is the concatenation of x and then u. If u_i, for $1\leqslant i\leqslant m$, are blocks, $\prod_{i=1}^{m}u_i$ is the concatenation of the u_i, in increasing order of i. We use $\mu()$ to denote Lebesgue measure, and log to denote logarithm in base 2.

2.1. Base-b representations and b-adic intervals

For each real number R in the unit interval we consider the unique expansion in base b of the form $R=\sum_{i=1}^{\infty}a_ib^{-i}$, where the integers $0\leqslant a_i<b$, and $a_i<b-1$ infinitely many times. This last condition over a_i ensures a unique representation of every rational number, and leads us to consider semi-open intervals $[p,q)$ in the real line. We write $(R)_b$ to denote the representation of a real R in base b. We use the phrase *b-adic interval* to refer to a semi-open interval I of the form $[a/b^m,(a+1)/b^m)$, for an integer a such that $0\leqslant a<b^m$. We move freely between b-adic intervals and base-b representations. If x is a base-b block and it is understood that we are working in base b, then we let $.x$ denote the rational number whose expansion in base b has exactly the digits appearing in x. Given the block x, the reals with base-b representations whose sequences of digits extend x are exactly those belonging to the b-adic interval $[.x,.x+b^{-|x|})$, written in base b. Conversely, every b-adic interval $[a/b^m,(a+1)/b^m)$ corresponds to a block x as above, where x is obtained by writing a in base b and then prepending a sufficient number of zeros to obtain a block of length m.

2.2. Absolute normality

Among the several equivalent definitions of absolute normality the following is the most convenient to present our algorithm.

Definition 2.1. A real number R is *simply normal* to base b if each digit d in base b has the same asymptotic frequency $1/b$ in the representation of R in base b:

$$\lim_{n\to\infty}\mathrm{occ}\big((R)_b\restriction n,d\big)/n=1/b.$$

R is *normal to base b* if it is simply normal to the bases b^i, for every $i\geqslant 1$. R is *absolutely normal* if it is normal to every integer base (equivalently, simply normal to every integer base).

Émile Borel not only isolated the notion of normal number[2] but also proved that almost every real number is absolutely normal.

2.3. Discrepancy

The simple discrepancy of an initial segment of the base b representation of a real number R indicates how much the digits in that initial segment vary from their expected average. Note that $D(u,b)$ is a number between 0 and $1-1/b$.

Definition 2.2. Let u be a finite block of digits in base b. The *simple discrepancy*, $D(u,b)$ of u in the base b is the maximum for $d\in\{0,\dots,b-1\}$ of $|\mathrm{occ}(u,d)/|u|-1/b|$.

Lemma 2.3. *A real number R is simply normal in base b if and only if*

$$\lim_{n\to\infty}D\big((R)_b\restriction n,b\big)=0.$$

[2] Borel's original definition, given in [4], says that a real number R is normal to base b if each of the numbers R, bR, $b^2R,\dots$ is simply normal to the bases b^i, for every integer $i\geqslant 1$. Although it seems more demanding, this last condition is equivalent to requiring that just R be simply normal to the bases b^i, for every integer $i\geqslant 1$. A proof can be read in [5].

Borel's theorem is underpinned by the fact that for any base almost every sufficiently long block has small discrepancy. We will need an explicit bound for the number of blocks of a given length having larger discrepancy than a given value. To make the paper self-contained we include the proof of the next lemma, which gives such a bound. We follow Hardy and Wright's classic text [7], but sharpen the value obtained there.

Let the number of blocks of length k in base b where a given digit occurs exactly i times be $p_b(k,i) = \binom{k}{i}(b-1)^{k-i}$.

Lemma 2.4. *(Adapted from [7, Theorem 148].) Fix a base b and a block length k. For every real ε such that $6/k \leqslant \varepsilon \leqslant 1/b$, $\sum_{0\leqslant i\leqslant k/b-\varepsilon k} p_b(k,i)$ and $\sum_{k/b+\varepsilon k\leqslant i\leqslant k} p_b(k,i)$ are at most $b^k e^{-b\varepsilon^2 k/6}$.*

Proof. Observe that for each i such that $i \leqslant k/b$, $p_b(k,i-1) < p_b(k,i)$ holds; and for each i such that $i > k/b$, $p_b(k,i) < p_b(k,i-1)$. The strategy to prove the wanted bounds is to "shift" the first sum to the right by $m = \lfloor \varepsilon k/2 \rfloor$ positions, and the second sum to the left by $m+1$ positions. We start with the first sum. Let $a = k/b - \varepsilon k$. For each i such that $0 \leqslant i \leqslant a$,

$$p_b(k,i) = \frac{p_b(k,i)}{p_b(k,i+1)} \cdot \frac{p_b(k,i+1)}{p_b(k,i+2)} \cdot \ldots \cdot \frac{p_b(k,i+m-1)}{p_b(k,i+m)} \cdot p_b(k,i+m).$$

The largest quotient in the expression above is $\frac{p_b(k,i+m-1)}{p_b(k,i+m)}$. Using the symbolic expression for $p_b(k,i)$, $\frac{p_b(k,i)}{p_b(k,i+1)} = \frac{(i+1)(b-1)}{k-i}$. Then,

$$\begin{aligned}
\frac{p_b(k,i)}{p_b(k,i+1)} &\leqslant \frac{p_b(k,\lfloor a\rfloor + m - 1)}{p_b(k,\lfloor a\rfloor + m)} = \frac{(\lfloor a\rfloor + m)(b-1)}{k - \lfloor a\rfloor - m + 1} \\
&< \frac{(k/b - \varepsilon k/2)(b-1)}{k - k/b + \varepsilon k/2} = 1 - \frac{\varepsilon b/2}{1 - 1/b + \varepsilon/2} \\
&< 1 - \varepsilon b/2 \quad (\text{using } \varepsilon \leqslant 1/b) \\
&< e^{-b\varepsilon/2}.
\end{aligned}$$

Since $m = \lfloor \varepsilon k/2 \rfloor$ and $\varepsilon k \geqslant 6$, $e^{-b\varepsilon m/2} \leqslant e^{-b\varepsilon(\varepsilon k/2-1)/2} = e^{-b\varepsilon^2 k/4 + b\varepsilon/2} \leqslant e^{-b\varepsilon^2 k/6}$. We obtain, $p_b(k,i) < e^{-b\varepsilon^2 k/6} p_b(k,i+m)$. Since $\sum_{0\leqslant i\leqslant k} p_b(k,i) = b^k$ we conclude

$$\sum_{0\leqslant i\leqslant a} p_b(k,i) < e^{-b\varepsilon^2 k/6} \sum_{0\leqslant i\leqslant a} p_b(k,i+m) \leqslant b^k e^{-b\varepsilon^2 k/6}.$$

To bound the second sum we shift the sum to the left by $m+1$ positions. Let $z = \lceil k/b + \varepsilon k \rceil$. For any integer i such that $z - m < i \leqslant k$,

$$p_b(k,i) = \frac{p_b(k,i)}{p_b(k,i-1)} \cdot \frac{p_b(k,i-1)}{p_b(k,i-2)} \cdot \ldots \cdot \frac{p_b(k,i-m)}{p_b(k,i-m-1)} \cdot p_b(k,i-m-1)$$

where these quotients increase as the indices decrease. So,

$$\begin{aligned}
\frac{p_b(k,i)}{p_b(k,i-1)} &\leqslant \frac{p_b(k,\lceil z\rceil - m)}{p_b(k,\lceil z\rceil - m - 1)} = \frac{k - \lceil z\rceil + m + 1}{(\lceil z\rceil - m)(b-1)} \\
&\leqslant \frac{k - k/b - \varepsilon k/2 + 1}{(k/b + \varepsilon k/2)(b-1)} \\
&< 1 - \varepsilon b/3.
\end{aligned}$$

To see the last inequality observe that it is equivalent to $\varepsilon b - 2/b - \varepsilon < 1 - \frac{6}{\varepsilon bk}$, which is implied by $1 - 2/b < 1 - \frac{6}{\varepsilon bk}$, because $\varepsilon b \leqslant 1$ and $\varepsilon > 3/k$. Therefore, $\frac{p_b(k,i)}{p_b(k,i-1)} < e^{-b\varepsilon/3}$. Then, using $m+1$,

$$p_b(k,i) < e^{-b\varepsilon(m+1)/3} p_b(k,i-m-1) \leqslant e^{-\frac{b\varepsilon^2 k}{6}} p_b(k,i-m-1).$$

From this last inequality and $\sum_{0\leqslant i\leqslant k} p_b(k,i) = b^k$ conclude $\sum_{z\leqslant i\leqslant k} p_b(k,i) < b^k e^{-b\varepsilon^2 k/6}$. □

Lemma 2.5. *Let $t \geqslant 2$ be an integer and let ε and δ be between 0 and 1, with $\varepsilon \leqslant 1/t$. Let k be the least integer greater than the maximum of $\lceil 6/\varepsilon \rceil$ and $-\ln(\delta/2t)6/\varepsilon^2$. Then, for all $b \leqslant t$ and all $k' \geqslant k$, the fraction of blocks x of length k' in base b for which $D(x,b) > \varepsilon$ is less than δ.*

Proof. Consider the case of a base b less than or equal to t and suppose that d is a digit in base b. By Lemma 2.4, the number of blocks x of length k such that $|\mathrm{occ}(x,d)/|x| - 1/b|$ is greater than ε is bounded by $2b^k e^{-b\varepsilon^2 k/6}$. Thus, the number

of blocks x of length k such that $D(x,b) > \varepsilon$ is bounded by $2b^{k+1}e^{-b\varepsilon^2 k/6}$. To have this constitute a fraction of no more than δ of all the b^k sequences, it is sufficient that $\delta > 2be^{-\varepsilon^2 k/6}$. This is implied by $k > -\ln(\delta/2b)6/\varepsilon^2$.

Since δ is less than or equal to 1, if $b \leqslant t$ is a base then $-\ln(\delta/2t) \geqslant -\ln(\delta/2b)$, and hence $k \geqslant -\ln(\delta/2b)6/\varepsilon^2$. But, then for any $k' > k$, the number of blocks x of length k' such that $D(x,b) > \varepsilon$ is a fraction of no more than δ of all the $b^{k'}$ sequences, as required. □

3. The algorithm

3.1. Simple normality in a single base

First, we discuss the ingredients for ensuring that a constructed real number X be simply normal to a single base b. We will employ these means later for several bases simultaneously.

We consider a sequence of b-adic intervals $(I_i)_{i\geqslant 1}$ by recursion on i such that for each i, the I_{i+1} is a subset of I_i, and such that $\lim_{i\to\infty}\mu(I_i) = 0$. The real number X determined by the algorithm will be the unique element of $\bigcap_{i\geqslant 1} I_i$, i.e. the limit of the left endpoints of these intervals. We let x_i be the block in base b such that I_i is the b-adic interval $[.x_i, .x_i + b^{-|x_i|})$. We let u_{i+1} be the block such that $x_i u_{i+1} = x_{i+1}$, i.e. x_{i+1} is the concatenation of x_i followed by u_{i+1}. Thus, for any $k \leqslant i$, $x_i = x_k \prod_{j=k+1}^{i} u_j$ and X is equal to $.x_k \prod_{j=k+1}^{\infty} u_j$.

We will be working toward ensuring that simple discrepancy decreases as we consider longer initial segments in the base-b expansion of X. We do so by choosing u_{i+1} so that $D(u_{i+1}, b)$ is smaller than a self-imposed threshold ε_{i+1}, where the function $i \mapsto \varepsilon_i$ is monotonically decreasing. Then, for any k and any $i+1$ sufficiently large relative to k, x_{i+1} is the concatenation of x_k with $\prod_{j=1}^{i+1} u_j$, a long string of discrepancy less than ε_j. It follows that $D(x_i, b)$ is not much larger than ε_j.

We need to determine the appropriate length of u_{i+1}. By allowing $|u_{i+1}|$ be sufficiently large, it is ensured that there is some block u_{i+1} such that $D(u_{i+1}, b) < \varepsilon_{i+1}$. By allowing $|u_{i+1}|$ be sufficiently small in comparison to $|x_i|$, it is ensured that for each ℓ less than or equal to $|u_{i+1}|$, $D(x_{i+1} \upharpoonright (|x_i| + \ell), b)$ is not much larger than $D(x_i, b)$, i.e. the variations of simple discrepancy within prefixes of u_{i+1} introduce only small variations of simple discrepancy within prefixes of x_{i+1}. Our task is to arrange for $\lim_{i\to\infty}\varepsilon_i = 0$ while maintaining the appropriate proportions in length between x_i and u_{i+1}.

Lemma 3.1. *Suppose that x and u are blocks in base b and $\varepsilon \in (0,1)$. If $D(x,b) < \varepsilon$ and $|u|/|x| < \varepsilon$, then for every ℓ less than or equal to $|u|$, $D((xu) \upharpoonright (|x| + \ell), b) < 2\varepsilon$.*

Proof. Let ℓ be fixed as above and let d be a digit in base b.

$$\begin{aligned}
\frac{\operatorname{occ}((xu) \upharpoonright (|x|+\ell), d)}{|x|+\ell} &\geqslant \frac{\operatorname{occ}(x,d)}{|x|+|u|} \\
&> \frac{(1/b-\varepsilon)|x|}{|x|+|u|} \quad \text{by assumption on } D(x,b) \\
&> 1/b - \varepsilon - (1/b)\frac{|u|}{|x|+|u|} \quad \text{by elementary means} \\
&> 1/b - 2\varepsilon \quad \text{since } \varepsilon > |u|/|x|.
\end{aligned}$$

Therefore, if $\varepsilon > |u|/|x|$, then for each $\ell \leqslant |u|$,

$$1/b - \frac{\operatorname{occ}((xu) \upharpoonright (|x|+\ell), d)}{|x|+\ell} < 2\varepsilon.$$

That $\frac{\operatorname{occ}((xu)\upharpoonright(|x|+\ell),d)}{|x|+\ell} - 1/b < 2\varepsilon$ can be verified similarly, which is sufficient to prove the lemma. □

Note that if u_1 and u_2 are blocks in base b such that $D(u_1,b) < \varepsilon$ and $D(u_2,b) < \varepsilon$, then $D(u_1u_2,b) < \varepsilon$. By applying this observation and Lemma 3.1, we obtain the following corollary by induction.

Corollary 3.2. *Take as given blocks x and u_i in base b, for $i \leqslant m$. Suppose ε satisfies the following conditions.*

1. *$D(x,b) < \varepsilon$.*
2. *For each $i \leqslant m$, $D(u_i,b) < \varepsilon$.*
3. *For each $i \leqslant m$, $|u_i|/|x\prod_{j=1}^{i-1} u_j| < \varepsilon$.*

Then for every ℓ less than or equal to $|\prod_{i=1}^{m} u_i|$, $D((x\prod_{i=1}^{m} u_i) \upharpoonright (|x|+\ell), b) < 2\varepsilon$.

The next lemma is essentially a special case of Lemma 3.1, with the roles of x and u reversed. We will apply it to analyze the effect of iteratively appending blocks of small discrepancy to an initial one.

Lemma 3.3. *Suppose that x and u are base b blocks. If ε is given so that $D(u,b) < \varepsilon$ and $|x|/|u| < \varepsilon$, then $D(xu,b) < 2\varepsilon$.*

3.2. Simple normality in multiple bases

We turn to working simultaneously with bases $b \in \{2,\dots,t\}$ in the context of stage i of a construction by recursion. Instead of one interval I_i, we will work with a nested sequence of intervals, $I_{i,2} \supset I_{i,3} \supset \cdots \supset I_{i,t}$, such that each $I_{i,b}$ is b-adic. Lemma 3.4 shows that the lengths of these intervals need not shrink too quickly.

Lemma 3.4. *For any interval I and any base b, there is a b-adic subinterval I_b such that $\mu(I_b) \geqslant \mu(I)/(2b)$.*

Proof. Let m be least such that $1/b^m$ is less than $\mu(I)$, i.e. $m = \lceil -\log_b(\mu(I))\rceil$. Note that $1/b^m$ is greater than or equal to $\mu(I)/b$, since $1/b^{m-1} \geqslant \mu(I)$. If there is a b-adic interval of length $1/b^m$ strictly contained in I, then let I_b be such an interval, and note that I_b has length greater than or equal to $\mu(I)/b$. Otherwise, there must be an a such that a/b^m is in I and neither $(a-1)/b^m$ nor $(a+1)/b^m$ belongs to I. Thus, $2/b^m$ is greater than $\mu(I)$. However, since $1/b^m < \mu(I)$ and b is greater than or equal to 2, $2/b^{m+1}$ is less than $\mu(I)$. So, at least one of the two intervals $[\frac{ba-1}{b^{m+1}}, \frac{ba}{b^{m+1}})$ or $[\frac{ba}{b^{m+1}}, \frac{ba+1}{b^{m+1}})$ must be contained in I. Let I_b be such. Then, the length of I_b is $\frac{1}{b^{m+1}} = \frac{1}{2b}\frac{2}{b^m} > \mu(I)/(2b)$. In either case, the length of I_b is greater than $\mu(I)/(2b)$. □

Definition 3.5. A *t-sequence* is a nested sequence of intervals, $\vec{I} = (I_2,\dots,I_t)$, such that I_2 is dyadic and for each base $b \geqslant 2$, I_{b+1} is a $(b+1)$-adic subinterval of I_b such that $\mu(I_{b+1}) \geqslant \mu(I_b)/2(b+1)$. We let $x_b(\vec{I})$ be the block in base b such that $.x_b(\vec{I})$ is the representation of the left endpoint of I_b in base b.

We can iteratively apply Lemma 3.4 for the following corollary.

Corollary 3.6. *For every dyadic interval I_2 and integer $t \geqslant 2$ there is a t-sequence $\vec{I}$ starting with I_2.*

In Corollary 3.6, we establish the existence of t-sequences. In Section 4, see especially Lemma 4.1, we analyze the number of operations needed to compute them.

If $\vec{I}$ is a t-sequence, then for any $b \leqslant t$ and any real $X \in I_t$, X has $x_b(\vec{I})$ as an initial segment of its representation in base b. If, further, $\vec{I}' = (I'_2,\dots,I'_{t'})$ is a t'-sequence with $t \leqslant t'$ such that $I'_2 \subset I_t$ and $X \in I'_{t'}$, then for each b less than or equal to t, $\vec{I}'$ specifies how to extend $x_b(\vec{I})$ to a longer initial segment $x_b(\vec{I}')$ of the base b representation of X. As opposed to arbitrary nested sequences, in t-sequences there is a function of t that gives a lower bound of the ratio between the measures of I_t and I_2. That is, $\mu(I_t)$ is at least $\mu(I_2)/(2^t t!)$.

3.3. Construction by recursion

Our construction of the real X is by recursion and written in terms of two given functions, $i \mapsto t_i$ and $i \mapsto \varepsilon_i$. The first determines the number of bases to be considered at stage i and the second determines a rational number upper bound on the allowed discrepancies of the blocks of new digits added to the representations of X in those bases. In stage $i+1$, we will have a t_i-sequence $\vec{I}_i = (I_{i,2}, I_{i,3},\dots,I_{i,t_i})$ given from the previous stage, with associated blocks $x_b(\vec{I}_i)$, for $b \leqslant t_i$.

Definition 3.7. Following Definition 3.5, for $b \leqslant t_i$, let $x_{i+1,b}$ be $x_b(\vec{I}_{i+1})$, the base b representation of the left endpoint of $I_{i+1,b}$, and let $u_{i+1,b}$ be $u_b(\vec{I}_{i+1})$, i.e. $x_{i+1,b} = x_{i,b}u_{i+1,b}$.

Algorithm 3.8. Assume given computable functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ such that t_i and $1/\varepsilon_i$ are non-decreasing in i and unbounded, with $\varepsilon_i \leqslant 1/t_i$. Let δ_{i+1} be the upper bound of the fraction of blocks in base b for $b \leqslant t_i$, of the length considered at stage $i+1$, that can be discarded,

$$\delta_{i+1} = \frac{1}{8t_i}\frac{1}{2^{t_i+t_{i+1}}t_i!t_{i+1}!}.$$

Let k_{i+1} be the length for the block in base t_i to be added at stage $i+1$,

$$k_{i+1} = \max\bigl(\lceil 6/\varepsilon_{i+1}\rceil, \bigl\lceil -\ln\bigl(\delta_{i+1}/(2t_i)\bigr)6/\varepsilon_{i+1}^2\bigr\rceil\bigr) + 1.$$

Initialization. Start with $\vec{I}_0 = ((I_{0,2}))$, with $I_{0,2} = [0,1)$.

Recursion step $i+1$. Determine the t_{i+1}-sequence $\vec{I}_{i+1}$ for stage $i+1$ as follows.

1. Let L be a dyadic subinterval of I_{i,t_i} such that $\mu(L) \geqslant \mu(I_{i,t_i})/4$.
2. For each dyadic subinterval J_2 of L of measure $2^{-\lceil \log t_i \rceil k_{i+1}} \mu(L)$, let $\vec{J} = (J_2, J_3, \ldots, J_{t_{i+1}})$ be a t_{i+1}-sequence for J_2.
3. Let $\vec{I}_{i+1}$ be the leftmost of the t_{i+1}-sequences $\vec{J}$ considered above such that for each $b \leqslant t_i$, $D(u_b(\vec{J}), b) \leqslant \varepsilon_{i+1}$.

We let X be the unique real in the intersection of the intervals in the sequences $\vec{I}_i$. Expressed in base b, $X = \lim_{i\to\infty} .x_{i,b}$. Expressed in terms of representations, $(X)_b = \prod_{i=1}^{\infty} u_{i,b}$.

To show that X is well-defined, we just need to verify that at each stage $i+1$ there is t_{i+1}-sequence $\vec{I}_{i+1}$. To prove that at each stage $i+1$ there is $\vec{I}_{i+1}$, we compare the measures of two sets. Let S be the union of the set of intervals $J_{t_{i+1}}$ over the $2^{\lceil \log t_i \rceil k_{i+1}}$-many t_{i+1}-sequences $\vec{J} = (J_2, \ldots, J_{t_{i+1}})$. By Lemma 3.4, $\mu(L) \geqslant \mu(I_{i,t_i})/4$, and for each $\vec{J}$, $\mu(J_{t_{i+1}}) \geqslant \frac{1}{2^{t_{i+1} t_{i+1}!}} \mu(J_2)$. Observe that the possibilities for J_2 form a partition of L. Hence, $\mu(S) \geqslant \frac{1}{2^{t_{i+1} t_{i+1}!}} \mu(L)$. Combining inequalities, $\mu(S) \geqslant \frac{1}{2^{t_i t_i!}} \frac{1}{4} \frac{1}{2^{t_{i+1} t_{i+1}!}} \mu(I_{i,2})$. Let N be the subset of S defined as the union of the set of intervals $J_{t_{i+1}}$ which occur in t_{i+1}-sequences which are *not suitable*. A t_{i+1}-sequence $\vec{J}$ is *not suitable* if for some $b \leqslant t_i$, $D(u_b(\vec{J}), b) > \varepsilon_{i+1}$. By construction, $u_2(\vec{J})$ has length $\lceil \log t_i \rceil k_{i+1}$ and for each $b \leqslant t_i$, $u_b(\vec{J})$ has length greater than or equal to k_{i+1}. Each $\vec{J}$ considered at stage $i+1$ is such that for every $b \leqslant t_i$ each interval J_b is a subinterval of $I_{i,b}$. According to Lemma 2.5 and by the choice of k_{i+1}, for each $b \leqslant t_i$, the subset of $I_{i,b}$ consisting of reals with base b representations $.x_{i,b}u_b(\vec{J})$ for which $D(u_b(\vec{J}), b) > \varepsilon_{i+1}$ has measure less than $\delta_{i+1}\mu(I_{i,b})$, and hence less than $\delta_{i+1}\mu(I_{i,2})$. Hence, $\mu(N) < t_i \delta_{i+1} \mu(I_{i,2})$. By the choice of δ_{i+1}, $\mu(N) < \frac{1}{42^{t_i} t_i! \, 2^{t_{i+1} t_{i+1}!}} \mu(I_{i,2})$. Then, $\mu(N) < \mu(S)$. Since S is a superset of N, this proves that at stage $i+1$ there is a suitable t_{i+1}-sequence $\vec{I}_{i+1}$.

3.4. Absolute normality

We give sufficient conditions on the functions $i \mapsto \varepsilon_i$ and $i \mapsto t_i$ to ensure absolute normality.

Theorem 3.9. *Suppose that the functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ are monotonic and such that $\lim_{i\to\infty} t_i = \infty$ and $\lim_{i\to\infty} \varepsilon_i = 0$. Further, suppose that for each i and for each $b \leqslant t_i$, $|u_{i+1,b}|/|x_{i,b}| < \varepsilon_{i+1}$. Then, the real X constructed in terms of these functions is absolutely normal.*

Proof. Let b be an integer greater than or equal to 2 and let $\varepsilon \in (0,1)$. Choose s so that b is less than t_s and $4\varepsilon_s$ is less than ε. During stages $i+1$ after s, we ensure of the constructed real X that the base b representation of X is obtained by appending blocks $u_{i+1,b}$ to $x_{i,b}$ for which $D(u_{i+1,b}, b) < \varepsilon_s$. Thus, for any n, $D(\prod_{i=s}^{n-1} u_{i+1,b}, b) < \varepsilon_s$. Fix s_1 so that $|x_{s,b}|/(s_1 - |x_{s,b}|) < \varepsilon_s$. By noting that we add at least one new digit in the base b representation of X during every stage after s and applying Lemma 3.4, we have that $D(x_{s,b} \prod_{i=s}^{s_1-1} u_{i+1,b}, b)$ is less than $2\varepsilon_s$. Then, Corollary 3.2 applies to conclude that for every ℓ,

$$D\left(X \upharpoonright \left|x_{s,b} \prod_{i=s}^{s_1-1} u_{i+1,b}\right| + \ell, b\right) < 2 \cdot 2\varepsilon_s < \varepsilon.$$

By Lemma 2.3, this is sufficient to prove the theorem. □

4. Implementation and time complexity

We consider the time complexity of the algorithm to be the number of elementary operations required to output the first i digits, where an elementary operation takes a fixed amount of time. We also count the number of mathematical operations performed by the algorithm, where mathematical operations include addition, subtraction, comparison, multiplication, division and logarithm. We use the big O notation standard in computer science, which illustrates the asymptotic behavior of a given function. A function $g(x)$ is $O(h(x))$ when there are constants x_0 and c such that for every $x \geqslant x_0$, $g(x) < c\,h(x)$.

Algorithm 3.8 depends on two given monotonic functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$. By controlling the rates at which t_i and ε_i approach their limits, we can control the number of operations required to run the construction. Thus, the count of the performed operations up to step i is given as a product of two factors, one that depends only on t_i and ε_i which can be made arbitrarily small, and the other that does not, which is the significant factor. We say that a number is *small* if it can be bounded by a function of t_i and ε_{i+1}. By the virtue of the algorithm all values are polynomial in the inverse of the measure of the smallest interval I being considered, so they can be represented by $O(-\log \mu(I))$ digits. *Expensive* mathematical operations are multiplications and divisions having both operands non-small. *Non-expensive* mathematical operations are operations having at least one small operand and also all additions, subtractions and comparisons. Expensive operations

require $O((-\log\mu(I))^2)$ elementary operations, whilst for the non-expensive $O(g(x)(-\log\mu(I)))$ elementary operations suffice, where g is some increasing function and x is small.

We represent b-adic intervals as tuples of four integers $\langle a, b, m, p\rangle$ such that the represented intervals are $[a/b^m, (a+1)/b^m)$ and $p = b^m$. The last terms p are kept just for efficiency of computation. For a b_1-adic interval $I_1 = \langle a_1, b_1, m_1, p_1\rangle$ and a b_2-adic interval $I_2 = \langle a_2, b_2, m_2, p_2\rangle$ we define $\mathrm{left}(I_1, I_2) = a_1 p_2$ and $\mathrm{right}(I_1, I_2) = a_2 p_1$.

The next lemma bounds the needed operations to find a b-adic subinterval of a given interval. It is intended that the given values be previously computed data; the proof revisits the existential result given in Lemma 3.4.

Lemma 4.1. *Suppose we are given two bases b_1 and b_2 and two b_1-adic intervals J_1 and I_1. We are also given a b_2-adic interval I_2 such that $J_1 \subseteq I_2 \subseteq I_1$, and the integers $\ell_I = \mathrm{left}(I_1, I_2)$ and $r_I = \mathrm{right}(I_1, I_2)$. Suppose we want to compute a b_2-adic subinterval J_2 of J_1 such that $\mu(J_2) \geqslant \mu(J_1)/(2b_2)$, and also compute the integers $\ell_J = \mathrm{left}(J_1, J_2)$ and $r_J = \mathrm{right}(J_1, J_2)$. The result can be obtained by two alternative computations, one takes $O((-\log\mu(J_1))^2)$ elementary operations; the other takes $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1)))(-\log\mu(J_1)))$ elementary operations, where g is some increasing function. In either case, $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1))))$ mathematical operations suffice.*

Proof. For $s = 1, 2$, let I_s be given by $\langle e_s, b_s, n_s, q_s\rangle$ and J_s be given by $\langle a_s, b_s, m_s, p_s\rangle$. Notice that $\mu(I_s) = 1/q_s = 1/b_s^{n_s}$ and $\mu(J_s) = 1/p_s = 1/b_s^{m_s}$. Within this proof, small values are those that can be bounded by the factor $g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1)))$. In particular, later in the proof it becomes clear that for each s, $m_s - n_s$ and $a_s - e_s$ are small.

First we give a computation that uses $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1))))$ non-expensive mathematical operations. We start calculating the small values $b_s^{m_s - n_s}$. Using iterated squaring it takes $O(\log(m_s - n_s))$ multiplications, requiring $O(\log b_s^{2(m_s - n_s)}) = O((-\log(\mu(J_s)/\mu(I_s)))^2)$ elementary operations, in total. Notice that $\mu(J_2)/\mu(I_2) > \mu(J_1)/(2b_2\mu(I_1))$ and so

$$O\big(-\log\big(\mu(J_2)/\mu(I_2)\big)\big) \subseteq O\big(-\log\big(\mu(J_1)/\mu(I_1)\big)\big).$$

We need to find a_2, m_2, p_2, ℓ_J and r_J, such that:

(1) $a_1/b_1^{m_1} \leqslant a_2/b_2^{m_2}$ and $(a_2+1)/b_2^{m_2} \leqslant (a_1+1)/b_1^{m_1}$,

(2) $1/b_1^{m_1} \leqslant 2b_2/b_2^{m_2}$,

(3) $p_2 = b_2^{m_2}$,

(4) $\ell_J = \mathrm{left}(J_1, J_2) = a_1 p_2$ and $r_J = \mathrm{right}(J_1, J_2) = a_2 p_1$.

Since $J_2 \subseteq I_2$, $n_2 \leqslant m_2$. From $\mu(J_2) \geqslant \mu(J_1)/(2b_2)$ and $\mu(I_2) \leqslant \mu(I_1)$ we can conclude that $\mu(J_2) \geqslant (\mu(I_2)/(2b_2))(\mu(J_1)/\mu(I_1))$. Application of $-\log_{b_2}$ to both sides yields $m_2 \leqslant n_2 + (m_1 - n_1)\log_{b_2} b_1 + 2$. So there are at most $(m_1 - n_1)\log_{b_2} b_1 + 2$ possible values for m_2, and we can iterate through each of them. From $J_2 \subseteq I_2$ we also infer that $e_2 b_2^{m_2 - n_2} \leqslant a_2 \leqslant (e_2+1)b_2^{m_2-n_2} - 1$, which means that there are $b_2^{m_2-n_2}$ possible values for a_2 and we can iterate through each of them. Since the number of iterations required to try the possibilities for both m_2 and a_2 are small numbers, they can be bounded by choosing g appropriately. To compute the starting and ending values in such iterations we only need a small number of non-expensive mathematical operations, and to change between consecutive values we need only addition. We then check for each pair if all requirements are met. Since Lemma 3.4 ensures that m_2 and a_2 exist, the described procedure will eventually find a suitable pair meeting the requirements.

For a given pair a_2 and m_2, we can compute p_2 by $p_2 = q_2 b_2^{m_2 - n_2}$ with a single non-expensive mathematical operation. To calculate r_J first notice that

$$\begin{aligned} r_J &= a_2 p_1 \\ &= a_2 b_1^{m_1} \\ &= (a_2 - e_2) q_1 b_1^{m_1 - n_1} + e_2 q_1 b_1^{m_1 - n_1} \\ &= q_1 (a_2 - e_2) b_1^{m_1 - n_1} + r_I b_1^{m_1 - n_1}. \end{aligned}$$

Since $a_s - e_s$ is small, r_J can be obtained from the last expression using only a constant number of non-expensive mathematical operations, because all factors are small except the first one of each term. The calculation of ℓ_J is similar. At this point, ℓ_J, r_J and p_2 meet the requirements by their construction. To check the requirements for a_2 and m_2, notice that requirement (1) is equivalent to $0 \leqslant \mathrm{right}(J_1, J_2) - \mathrm{left}(J_1, J_2) \leqslant p_2 - p_1$ and requirement (2) is equivalent to $p_2 \leqslant 2b_2 p_1$, and both can be checked with a constant number of non-expensive mathematical operations, given that we already calculated $\ell_J = \mathrm{left}(J_1, J_2)$ and $r_J = \mathrm{right}(J_1, J_2)$.

An alternative way of computing can be achieved by replacing the iteration through possible values of a_2 and m_2 by their direct computation using the given bounds and rounding. This entails a constant number of expensive mathematical operations. □

The next lemma counts the steps in one complete stage of our algorithm. As in the previous lemma, it is intended that the given values be previously computed data. We count all operations except the computation of t_{i+1}, ε_{i+1} and k_{i+1}, which is postponed until the subsequent theorem.

Lemma 4.2. *Assume we are given i, t_i, ε_{i+1} and t_{i+1}. Then, there is computable function $h(t,\varepsilon)$, increasing in t and $1/\varepsilon$, such that stage $i+1$ of Algorithm 3.8 can be completed in $O(h(t_{i+1},\varepsilon_{i+1}))$ mathematical operations. Let n be the minimum number of digits that are sufficient to represent each of the endpoints of the intervals of $\vec{I}_i$. In case $t_{i+1}=t_{i-1}$ stage $i+1$ requires $O(h(t_{i+1},\varepsilon_{i+1})n)$ elementary operations; otherwise it requires $O(h(t_{i+1},\varepsilon_{i+1})n^2)$ elementary operations.*

Proof. We count the operations needed to run all the steps of stage $i+1$. Assume first that $t_{i+1}=t_{i-1}$. Then, all bases considered in stage $i+1$ were also considered in stages i and $i-1$. Lemma 4.1 applies to count the operations needed to find subintervals. In each application of the lemma, the values of I_1, I_2, ℓ_I and r_I in the hypothesis are carried forward from the computation in the previous stage. Then, $-\log(\mu(J_1)/\mu(I_1))$ is bounded by $\lceil \log t_i\rceil k_{i+1}$ and hence is a small value. Since $O(-\log\mu(J_1))=O(n)$, finding a subinterval requires at most $O(g(t_{i+1},\varepsilon_{i+1}))$ mathematical operations or $O(g(t_{i+1},\varepsilon_i)n)$ elementary operations.

We write h with a subindex to indicate a function of t_{i+1} and ε_{i+1}. Let h_o be such that each non-expensive mathematical operation in this procedure uses at most $O(h_o)$ elementary operations and each expensive mathematical operation uses at most $O(h_o n)$. The computation can be organized in the following steps.

- Compute $\lceil\log(t_i)\rceil$, δ_{i+1} and k_{i+1}. This takes a constant number of non-expensive mathematical operations or $O(h_o)$ elementary operations.
- Compute a dyadic subinterval L of I_{i,t_i} such that $\mu(L)\geqslant\mu(I_{i,t_i})/4$. This takes $O(g(t_{i+1},\varepsilon_{i+1}))$ mathematical operations or $O(g(t_{i+1},\varepsilon_{i+1})n)$ elementary operations.
- In increasing order of left endpoint, consider the dyadic subintervals J_2 of L:
  1. For each possible J_2, we determine a t_{i+1}-sequence $\vec{J}$ starting with J_2. This takes $O(t_{i+1}\,g(t_i,\varepsilon_{i+1}))$ mathematical operations or $O(h_1\,g(t_i,\varepsilon_{i+1})n)$ elementary operations.
  2. For each $b\leqslant t_i$, compute the base-b representation of the left endpoint $u_b(\vec{J})$ of $\vec{J}_b$. This requires $O(|u_b(\vec{J})|)$ non-expensive mathematical operations, because each operation has at least one operand which is a base and hence depends only on t_i, and $|u_b(\vec{J})|\leqslant\lceil\log t_i\rceil k_{i+1}$ also depends only on t_i. Therefore, this takes $O(h_2)$ mathematical operations or $O(h_2 h_o)$ elementary operations.
  3. Calculate thresholds for the number of occurrences of each digit to check $D(u_b(\vec{J}),b)\leqslant\varepsilon_{i+1}$. Such thresholds are of the form $(1/b+\varepsilon_{i+1})|u_b(\vec{J})|$ and therefore can be calculated with a constant number of non-expensive mathematical operations for each base. Hence, $O(t_i)=O(h_3)$ non-expensive mathematical operations or $O(h_3 h_o)$ elementary operations in total.
  4. The counting of occurrences and the comparison against the threshold operate only on small values (bounded by $\max(t_i,|u_b(\vec{J})|)$). This step takes $O(|u_b(\vec{J})|\max(t_i,|u_b(\vec{J})|))$ non-expensive mathematical operations for each base. In total, this is

$$O\big(t_i\big|u_b(\vec{J})\big|\max\big(t_i,\big|u_b(\vec{J})\big|\big)\big)=O(h_4)$$

  mathematical operations or $O(h_4 h_o)$ elementary operations.

The search stops upon finding a suitable t_{i+1}-sequence, before exhausting the $2^{\lceil\log t_i\rceil k_{i+1}}$ many intervals J_2. This requires at most $h_*=2^{\lceil\log t_i\rceil k_{i+1}}$ iterations. We can complete the proof for the case $t_{i+1}=t_{i-1}$ by setting $h=h_*(h_1\,g+h_2+h_3+h_4)h_o$.

If $t_{i+1}>t_{i-1}$, then it is possible that for some uses of Lemma 4.1 we do not have any previously computed data. In this case we set the intervals in the hypothesis of the lemma as $I_1=I_2=[0,1)$ and $\ell_I=r_I=0$, making $-\log(\mu(J_s)/\mu(I_s))=-\log(\mu(J_s))=O(n)$ for $s=1,2$, and thus requiring $O(g\,n^2)$ elementary operations for each application of the alternative computation in Lemma 4.1. This requires $O(h_* h_1\,g\,n^2)$ elementary operations more than in the previous case. Using the same h as before, this case entail at most $O(h\,n^2)$ elementary operations. □

Theorem 4.3. *Suppose f is a computable non-decreasing unbounded function. Algorithm 3.8 computes an absolutely normal number X such that, for any base b, it outputs the first i digits in the base b representation of X after performing $O(f(i)i)$ mathematical operations or $O(f(i)i^2)$ elementary operations.*

Proof. We define functions $i\mapsto t_i$ and $i\mapsto\varepsilon_i$ simultaneously with running an implementation of Algorithm 3.8. Let $t_1=2$ and $\varepsilon_1=1/2$. Assume $k_1=1$ and $f(1)$ is known data, having a value greater than $h(2,1)$, for the h as in Lemma 4.2 For the recursion stage $i+1$, assume that $t_i=v$ and $\varepsilon_i=1/v$ are given, with $v\geqslant 2$, and that $\vec{I}_i$ is the result of the construction as determined by the first i many values of t and ε with associated blocks $x_{i,b}$. If the number of stage $i+1$ is a power of 2, we execute i many elementary operations in the computation of the initial values of f, obtaining the values of f on the numbers less than or equal to some integer m. Notice that $1\leqslant m\leqslant i$. Define δ by

$$\delta = \frac{1}{8t_i}\frac{1}{2^{t_i+\nu+1}\,t_i!(\nu+1)!},$$

which would be the value of δ_{i+1} if we were to define $t_{i+1} = \nu + 1$. Let $k(\varepsilon, \delta, t)$ be the function defined by the calculation of k as given in Lemma 2.5. Finally, we execute i many elementary operations in the computations of the functions $k(1/(\nu+1), \delta, \nu+1)$ and $h(\nu+1, 1/(\nu+1))$. If we obtain values for these functions within the allotted number of operations and they satisfy the inequalities $h(\nu+1, 1/(\nu+1)) < f(m)$ and, for each $b \leqslant t_i$,

$$\frac{\lceil \log(\nu+1)\rceil k(1/(\nu+1), \delta, \nu+1) + \lceil -\log(\delta)\rceil}{|x_{i,b}|} < \frac{1}{\nu+1},$$

then define $t_{i+1} = \nu + 1$ and $\varepsilon_{i+1} = 1/(\nu+1)$. Otherwise, let $t_{i+1} = t_i = \nu$ and $\varepsilon_{i+1} = \varepsilon_i = 1/\nu$. We then complete stage $i+1$ of the construction and thereby complete the recursion step in the definitions of the functions t and ε. Clearly, $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ are computable, $i \mapsto t_i$ is non-decreasing, and $i \mapsto \varepsilon_i$ is non-increasing. Applying the assumptions on f, $\lim_{i\to\infty} t_{i+1} = \infty$ and $\lim_{i\to\infty} \varepsilon_{i+1} = 0$. Further, in the construction determined by these functions, if during stage $j+1$ the value of ε_{j+1} is lowered from $1/(\nu-1)$ to $1/\nu$, then for each $b \leqslant t_j$,

$$\frac{\lceil \log(\nu+1)\rceil k(1/\nu, \delta_{j+1}, \nu+1) + \lceil -\log(\delta_{j+1})\rceil}{|x_{j,b}|} < \frac{1}{\nu}.$$

For every subsequent stage $i+1$ during which $\varepsilon_{i+1} = 1/\nu$ and for every $b \leqslant \nu+1$,

$$|u_{i+1,b}| \leqslant \lceil \log(\nu+1)\rceil k\big(1/(\nu+1), \delta_{j+1}, \nu+1\big) + \lceil -\log(\delta_{j+1})\rceil$$

and $|x_{i+1,b}| \geqslant |x_{j,b}|$, so $|u_{i+1,b}|/|x_{i,b}|$ is less than or equal to $1/(\nu+1)$. Thus, the construction satisfies the hypotheses of Theorem 3.9 and thereby produces an absolutely normal number.

All mentioned mathematical operations are non-expensive, because the only non-small operands in them are of the form $|x_{i,b}|$ and all those appear on independent calculations. The computations of the values of t and ε during stage $i+1$ add only $O(i)$ elementary operations to the construction itself. Since that calculation is only done when the stage number is a power of two, in total this adds $O(i)$ extra elementary operations. Since f is non-decreasing, for every i, if $t_{i+1} = \nu+1$ then $h(\nu+1, 1/(\nu+1)) < f(i+1)$. From the way t_i is defined, $t_{i+1} > t_{i-1}$ in at most $O(\log i)$ stages; therefore, using Lemma 4.2 the total number of required elementary operations is $O(i\,f(i+1)i + \sum_{j=1}^{\log i} f(2^j+1)2^{j^2}) = O(f(i+1)i^2)$. Since each stage produces at least one extra digit in every base considered, i stages are enough to produce the first i digits in any of those bases. □

Acknowledgments

Joos Heintz raised the problem of finding a polynomial time algorithm to produce absolutely normal numbers more than ten years ago, and since then he has consistently encouraged V. Becher to work on it.

Becher and Heiber are members of Laboratoire International Associé INFINIS, Université Paris Diderot–CNRS/Universidad de Buenos Aires–CONICET. Becher's research was supported by CONICET and Agencia Nacional de Promoción Científica y Tecnológica, Argentina. Slaman's research was partially supported by the National Science Foundation, USA, under Grant No. DMS-1001551 and by the Simons Foundation. This work was done while the authors participated in the Buenos Aires Semester in Computability, Complexity and Randomness, 2013.

References

[1] Verónica Becher, Martin Epszteyn, Pablo Ariel Heiber, Theodore A. Slaman, Efficiently computing an absolutely normal number, preprint, 2013.
[2] Verónica Becher, Santiago Figueira, An example of a computable absolutely normal number, Theor. Comput. Sci. 270 (2002) 947–958.
[3] Verónica Becher, Santiago Figueira, Rafael Picchi, Turing's unpublished algorithm for normal numbers, Theor. Comput. Sci. 377 (2007) 126–138.
[4] Émile Borel, Les probabilités d'enombrables et leurs applications arithmétiques, Suppl. Rend. Circ. Mat. Palermo 27 (1909) 247–271.
[5] Yann Bugeaud, Distribution Modulo One and Diophantine Approximation, Cambridge Tracts in Mathematics, vol. 193, Cambridge University Press, 2012.
[6] Santiago Figuiera, André Nies, Feasible analysis and randomness, preprint, 2013.
[7] G.H. Hardy, E.M. Wright, An Introduction to the Theory of Numbers, sixth ed., Oxford University Press, Oxford, 2008.
[8] Elvira Mayordomo, Construction of an absolutely normal real number in polynomial time, preprint, 2013.
[9] Wolfgang M. Schmidt, Über die Normalität von Zahlen zu verschiedenen Basen, Acta Arith. 7 (1961/1962) 299–309.
[10] Wacław Sierpiński, Démonstration élémentaire du théorème de M. Borel sur les nombres absolument normaux et détermination effective d'un tel nombre, Bull. Soc. Math. Fr. 45 (1917) 127–132.
[11] Alan Turing, A note on normal numbers, in: J.L. Britton (Ed.), Collected Works of A.M. Turing: Pure Mathematics, North-Holland, Amsterdam, 1992, pp. 117–119, with notes of the editor on pp. 263–265.