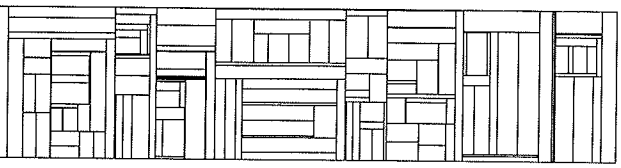
ISSN 0105-8517

The Complexity of Malign Ensembles

Peter Bro Miltersen

DAIMI PB – 335
September 1990

COMPUTER SCIENCE DEPARTMENT
AARHUS UNIVERSITY
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark



The complexity of malign ensembles*

Peter Bro Miltersen
Aarhus University, Computer Science Department
Ny Munkegade, DK 8000 Aarhus C.

September 20, 1990

Abstract

We analyze the concept of *malignness*, which is the property of probability ensembles of making the average case running time equal to the worst case running time for a class of algorithms. We derive lower and upper bounds on the complexity of malign ensembles, which are tight for exponential time algorithms, and which show that no polynomial time computable malign ensemble exists for the class of polynomial time algorithms. Furthermore, we show that for no class of superlinear algorithms a polynomial time computable malign ensemble exists, unless every language in P has an expected polynomial time constructor.

1 Introduction

The average case time complexity of specific algorithms has for a number of years been an active area of research, often showing significant improvement over the worst case complexity when specific distributions of the inputs were assumed. Recently, Li and Vitanyi [1] studied the Solomonoff-Levin measure $\mathbf{m}$ and found that when the inputs to any algorithm are distributed according to this measure, it holds that the algorithm's average case complexity is of the same order of magnitude as its worst case complexity. More precisely,

*Work partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

$$\exists c > 0 : \sum_{x \in \Sigma^n} \frac{\mathbf{m}(x)}{\sum_{y \in \Sigma^n} \mathbf{m}(y)} T_A(x) \geq c \max_{x \in \Sigma^n} T_A(x).$$

In this paper, we use the term *malign* for such a measure. This result seems interesting for at least two reasons.

- The Solomonoff-Levin measure assigns large amounts of probability to strings with lots of pattern, and small amounts to random strings. Therefore, the result seems to imply that worst case strings or strings close to worst case will be easily describable. In [1], the example of quicksort is used, where the worst case strings are the sorted or almost sorted ones. These have short descriptions, and hence high Solomonoff-Levin measure. That worst case strings in general are easily describable seems counterintuitive.
- Li and Vitanyi argue in [3] that the Solomonoff-Levin measure should be used as the a priori probability in Bayesian reasoning, because, in a certain sense, it lies close to any r.e. measure (it dominates them multiplicatively). Similarly, one could argue that if inputs are given from a natural source, the results imply that the average case time will be close to the worst case time, so that no improvement is possible.

However, since the Solomonoff-Levin measure is not even recursive, their result fits poorly with the traditional complexity theory view on average case complexity, founded by Levin in [6] and extended in [7] and [8]. In Levins approach to average case complexity, the distribution function of the input measure is required to be polynomial time computable. Therefore, unless we can derive some kind of time bounded version of Li and Vitanyi's result, the two subjects seem quite unrelated, although some of the consequences appear similar [1]. In this paper, we analyze from a complexity-theoretic perspective the property of malignness. For this, we restate Li and Vitanyi's result and give a simple proof. It seems that the counterintuitive property of malignness is dependent upon an exponential time pattern, which makes the above interpretations less obvious. We present a number of results which support our intuition. Our results pose a limit on the results achievable in the average case direction by the time bounded versions of the Solomonoff-Levin measure, which are also discussed in [1]. In general, they suggest that if inputs are given by any polynomial time adversary, phenomena like the above will not arise.

2 Notation

- We consider the fixed binary alphabet $\Sigma = \{0,1\}$. Σ^* is the set of binary strings with the usual ordering, first by length and then lexicographically, and Σ^n is the set of strings of length n. By $x-1$ we denote the string preceding x, and by $x+1$ the string following x. The empty string is denoted Λ. The length of the string x is denoted by $|x|$. $< \cdot, \cdot >$ denotes a polynomial time bijective pairing function $\Sigma^* \times \Sigma^* \to \Sigma^*$ with polynomial time projections.

- A *measure* μ on a finite or countable set M is a function from M to the real numbers, with $\mu(x) \geq 0$ for all x. Given a subset $A \subseteq M$, we define

$$\mu(A) = \sum_{x \in A} \mu(x).$$

  A *probability measure* is a measure with $\sum_{x \in M} \mu(x) = 1$. Given a measure μ on Σ^*, we denote by μ^* its distribution function $\mu^*(x) = \mu(\{y | y \leq x\})$. If $\mu(\Sigma^n) \neq 0$ for all n, we define the n'th derived measure of μ to be the probability measure on Σ^n defined by

$$\mu_n(x) = \frac{\mu(x)}{\mu(\Sigma^n)}.$$

- A function $f : \Sigma^* \to \mathbf{R}$ is called *polynomial time computable* if there exists a polynomial time Turing machine transducer which on input $< x, 1^i >$ produces the binary notation of an integer t with

$$|f(x) - t2^{-i}| \leq 2^{-i}.$$

  In general, if the machine runs in time at most $g(|x|, i)$, we say that f is computable in time $g(|x|, i)$.

- By *algorithms* we mean the algorithms in a fixed machine model, which takes an input x, and always halts. These can not be effectively enumerated, of course, but we will assume that all the machines, including those which do not halt on every input, are enumerated $A_1, A_2, \ldots$ such that simulation, including stepcounting etc., of n steps of A_i on input x can be performed in time $p(i, |x|, n)$ on a Turing machine, where p is a polynomial. We assume that algorithms can simulate Turing machines in real time.

- Given an algorithm A, we define $T_A(x)$, $x \in \Sigma^*$ to be its running time on the binary string x. Given a function $f : \boldsymbol{N} \rightarrow \boldsymbol{N}$, we define

$$ALG(f) = \{A | T_A(x) \leq f(|x|) \text{ for all } x, \text{ except a finite number}\}.$$

$$T_A^w(n) = \max_{|x|=n} T_A(x)$$

is the algorithm's worst case running time. We denote by $w_A(n)$ the lexicographically least string in Σ^n with $T(w_A(n)) = T_A^w(n)$.

$$T_A^a(\boldsymbol{\mu}, n) = \sum_{|x|=n} \mu_n(x) T_A(x)$$

is the algorithm's average case running time with respect to the measure $\boldsymbol{\mu}$.

3 Malign measures

In this section malignness is defined and Li and Vitanyi's result on the Solomonoff-Levin measure is presented. We give a direct proof and skip the conceptual developments of [1]. We consider the class of Turing machines, where each machine has three tapes,

- A binary input tape, infinite in one direction, with the restriction that the head can only move in this direction. Thus, the input tape is one-way, one-way infinite.
- A two-way, infinite, work tape.
- A one-way, one-way infinite binary output tape.

The input tape and the output tape are started with their heads on the first square, and a machine must be able to determine by itself when it has read its input. Now consider an effective enumeration $M_1, M_2, \ldots$ of the above class of Turing machines, and let U be a machine universal for the class, i.e. U will on input $1^i 0 t$, where t is an infinite tape, halt if and only if the machine M_i on input tape t halts, and U will in that case output whatever M_i outputs.

We next consider the input tape of U filled with the results of an infinite sequence of coin tosses. The Solomonoff-Levin measure of a string

$x \in \Sigma^n$, $\mathbf{m}(x)$, is then defined as the probability that U halts, outputting x. Since U of course has a non-zero probability of not halting, we have that

$$\sum_{x \in \Sigma^*} \mathbf{m}(x) < 1.$$

The Solomonoff-Levin measure was first defined rigorously in [5]. Intuitively, it gives a large amount of measure to strings with lots of pattern, since these have short programs which have a high probability of appearing. Actually, it is closely tied to self-delimiting Kolmogorov complexity, since $\mathbf{m}(x) = 2^{-\Theta(K(x))}$, where $K(x)$ is the self-delimiting Kolmogorov complexity of x, but we do not need this result here (see [1] for a proof, and [3] and [4] for general discussions of the properties of $\mathbf{m}$).

Definition 3.1 *A measure μ is* malign *for an algorithm A if and only if there exists a $c > 0$ such that for all sufficiently large n*

$$T_A^a(\mu, n) \geq cT_A^w(n).$$

It is malign for a class of algorithms $\mathcal{A}$ if it is malign for each $A \in \mathcal{A}$.

The following is the main result from [1] on average case complexity.

Theorem 3.1 (Li and Vitanyi) *The Solomonoff-Levin measure $\mathbf{m}$ is malign for all algorithms.*

Proof Consider the following Turing machine M, of the above kind.

```
Read the prefix 1^i0, i ≥ 0 from the tape.
Simulate U on the rest of the tape.
if U halts then
    n := |U's output|
    simulate A_i on all inputs of length n, finding the lexicographically
      least worst case output, w_{A_i}(n) (This may not halt).
    output w_{A_i}(n)
fi
```

Assume $M = M_k$ in the above enumeration, and assume that i is the index of an algorithm, i.e. that A_i halts on all inputs. If U is started with the tape 1^k01^i0t, where U, started on t, outputs a string of length n, U will output $w_{A_i}(n)$. The events of reading 1^k01^i0 and reading t are independent. But this means that for all n

$$\mathbf{m}(w_{A_i}(n)) \geq 2^{-k-i-2}\mathbf{m}(\Sigma^n).$$

But then

$$T^a_{A_i}(\mathbf{m}, n) \geq \frac{\mathbf{m}(w_{A_i}(n))}{\mathbf{m}(\Sigma^n)} T^w_{A_i}(n) \geq 2^{-k-i-2} T^w_{A_i}(n).$$

□

As mentioned in the introduction, theorem 3.1 seems interesting, because it suggests that generally inputs which require a lot of time contains lots of pattern - otherwise the average case complexity with respect to the Solomonoff-Levin measure could not be of the same order as the worst case complexity. In [1], Li and Vitanyi use quicksort as an example, where the worst case inputs are the ones already sorted - i.e. inputs with lots of pattern. Theorem 3.1 then suggests that this is a general phenomenon. By examining the proof we see that it indeed holds that the worst case input has a lot of pattern. The proof uses that the worst case input to A_i of length n can be described in the following way: "The worst case input to A_i of length n". Thus $w_{A_i}(n)$ has a short description, i.e. lots of pattern. Of course, this seems to make our interpretation of the theorem a bit less interesting: Worst case inputs have a pattern, but it is the very pattern of being a worst case input. Further reflection of the notion of pattern seems to be called for. The problem with the pattern "The worst case input to A_i of length n" is, of course, that it takes exponential time to get from the description to the result. Thus, the pattern is difficult (we should mention that it is implicitly stated in [1] that this is exactly the kind of pattern that makes the theorem work). The main object of this paper is to establish that this is the way it has to be - In general it does not hold that inputs which are slow to process have an easy pattern. To make this more precise, we want to answer questions of the following kind:

Suppose $\boldsymbol{\mu}$ is malign for a class of algorithms $\mathcal{A}$. What complexity does $\boldsymbol{\mu}$ have?

As in the theory of Average-NP [6][7][8], it seems natural to take the computational complexity of the distribution function $\boldsymbol{\mu}^*$ as the complexity of $\boldsymbol{\mu}$. However, this does not seem to be a good idea in this context without further restrictions on $\boldsymbol{\mu}$, as the following theorem shows.

Theorem 3.2 *For each general recursive function f there exists a measure μ which is malign for $ALG(f)$ and whose distribution function μ^* is polynomial time computable.*

Proof Define

$$T(x,i,t) = \min(T_{A_i}(x), t).$$

$$w(n,i,t) = \min\{y \in \Sigma^n | \forall x \in \Sigma^n : T(y,i,t) \geq T(x,i,t)\},$$

i.e. $w(n,i,t)$ is the lexicographically least worst case input of A_i of size n, when A_i is restricted to run for at most t time steps. Let g be a time constructible function with $g(n) \geq f(n)$ for all n. Define

$$v(n,i) = w(n,i,g(n)).$$

Observe that $v(n,i)$ can be computed in time $t(n,i) = q(2^n p(n,i,g(n)))$ where p and q are polynomials, by exhaustively simulating A_i on all inputs of length n. Put

$$u(n) = t(n,n).$$

By choosing p and q appropriately (sufficiently large), u can be made time constructible. We can without loss of generality assume that $u(n+1) \geq u(n) + n$. Now define

$$b(x,i) = \begin{cases} 0 & \text{if } i \leq u(|x|) \\ 0 & \text{if } u(|x|) < i \leq u(|x|) + |x| \text{ and } v(|x|, i - u(|x|)) \neq x \\ 1 & \text{if } u(|x|) < i \leq u(|x|) + |x| \text{ and } v(|x|, i - u(|x|)) = x \\ 0 & \text{if } u(|x|) + |x| < i \end{cases}$$

$$\mu(x) = \sum_{i=1}^{\infty} b(x,i) 2^{-i}.$$

Since $v(n,j) \in \Sigma^n$, we have

$$\mu(\Sigma^n) = \sum_{i=u(n)+1}^{u(n)+n} 2^{-i} < 2^{-u(n)}.$$

Fix an algorithm $A_j \in \mathcal{A}$. We have that $v(n,j) = w_{A_j}(n)$ for all n. But then

$$b(w_{A_j}(n), u(n) + j) = 1 \text{ for } n \geq j$$

and therefore

$$\boldsymbol{\mu}(w_{A_j}(n)) \geq 2^{-u(n)-j}$$

$$\mu_n(w_{A_j}(n)) = \frac{\boldsymbol{\mu}(w_{A_j}(n))}{\boldsymbol{\mu}(\Sigma^n)} > 2^{-j}$$

$$T^a_{A_j}(\boldsymbol{\mu}, n) \geq \mu_n(w_{A_j}(n)) T_{A_j}(w_{A_j}(n)) > 2^{-j} T^w_{A_j}(n).$$

Thus $\boldsymbol{\mu}$ is malign for $\mathcal{A}$. It remains to show that $\boldsymbol{\mu}^*$ is polynomial time computable.

$$\boldsymbol{\mu}^*(x) = \sum_{j<|x|} \boldsymbol{\mu}_b(\Sigma^j) + \boldsymbol{\mu}(\{y||y| = |x| \wedge y \leq x\}).$$

The i'th binary digit of $\sum_{j<|x|} \boldsymbol{\mu}(\Sigma^j)$ is 1 if and only if there exists an $m \in \{0, \ldots, |x|-1\}$ so that $u(m) < i \leq u(m) + m$. This can be decided in time polynomial in $|x|$ and i, since u is time constructible. For the second term, observe that the i'th binary digit of $\boldsymbol{\mu}(\{y||y| = |x| \wedge y \leq x\})$ is 1 if and only if $u(|x|) < i \leq u(|x|) + |x|$ and $v(|x|, i - u(|x|)) \leq x$. To decide if this is the case, we execute the following algorithm.

```
input x,1^i
decide if u(|x|) < i ≤ u(|x|) + |x|
if "yes" then
    calculate the exact value of u(|x|)
    calculate v(|x|, i − u(|x|))
fi
```

Except for the calculation of $v(|x|, i-u(|x|))$, it immediately follows from the time constructibility of u that the algorithm is polynomial time. But observe that this calculation can be done in time

$$t(|x|, i - u(|x|)) \leq t(|x|, |x|) = u(|x|) < i.$$

□

Of course, the theorem (and in particular its proof) suggests that the complexity of the measure $\boldsymbol{\mu}$ itself is not particularly relevant when we are interested in properties such as being malign for a class. This is no big surprise, since we only use the derived probability measures μ_n. If we examine the proof, we see that it is based on the trick of letting $\boldsymbol{\mu}(\Sigma^n)$ vanish very rapidly, so that the non-zero digits of $\boldsymbol{\mu}(x)$ appear so late

that we have a sufficient amount of time to compute them. The fact that μ must be normalized in the definition of malignness thus seems to be giving us an unreasonable advantage in the computation of μ^*. If we instead look at the time required to compute the normalized $\mu^*_{|x|}(x)$, no such trick will work. It is therefore still reasonable to conjecture that it requires time exponential in x to compute this number, since the computation of a digit seems to require running an algorithm on all inputs of size $|x|$. Consequently, from now on we require that the measures we consider are normalized.

4 Malign ensembles

Definition 4.1 *A* probability ensemble *(or merely ensemble) is a measure* $\mu : \Sigma^* \to [0,1]$, *with* $\mu(\Sigma^n) = 1$ *for all* n.

Thus, if μ is an ensemble, $\mu(x) = \mu_{|x|}(x)$ for all x.

Definition 4.2 *An ensemble* μ *is called* polynomial time computable *if and only if its family of distribution functions* $x \to \mu^*_{|x|}(x)$ *is polynomial time computable. We denote by PE the class of polynomial time computable ensembles. In general, we say that an ensemble is computable in time* $f(|x|,i)$ *if* $\mu^*_{|x|}$ *is computable in time* $f(|x|,i)$.

This definition relativizes in the obvious way.

We can, through the use of essentially the same technique as in our proofs of theorem 3.1 and theorem 3.2, construct malign ensembles and hence provide upper bounds for the time required to compute malign ensembles for certain classes of algorithms. The upper bounds reflect our intuition on how the proof of theorem 3.1 works.

Theorem 4.1 *There exists a polynomial* p *so that for any time constructible function* f *there exists an ensemble* μ, *computable in time* $p(2^{|x|}f(|x|),i)$ *so that* μ *is malign for* $ALG(f)$.

Proof Define

$$T(x,i) = \min(T_{A_i}(x), f(|x|)).$$

By our assumptions on the enumeration A_i, $T(x,i)$ can be computed in time $p_1(f(|x|),i)$, where p_1 is a polynomial, which, by the time constructibility of f, do not depend upon f. Define

$$w(n,i) = \min\{x \in \Sigma^n | \forall y \in \Sigma^n : T(y,i) \leq T(x,i)\}.$$

The function w can be computed in time $p_2(2^n, p_1(f(n),i))$ i.e. in time $p_3(2^n f(n), i)$.

$$b(x,i) = \begin{cases} 1 & \text{if } w(|x|,i) = x \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{|x|}(x) = \sum_{i=0}^{\infty} b(x,i)2^{-i}.$$

$\boldsymbol{\mu}$ is a probability ensemble. It can be computed in the required time, because the i'th binary digit of $\mu^*_{|x|}(x)$ is 1 if and only if $w(|x|,i) \leq x$. It only remains to show that it is malign for $ALG(f)$. But for this we observe that if A_j is such an algorithm, $T(x,j) = T_{A_j}(x)$ for sufficiently large $|x|$, and for these x, $b(x,j) = 1$ if x is the lexicographically least worst case input of size $|x|$ for A_j. But then

$$T^a_{A_j}(\boldsymbol{\mu}, n) \geq 2^{-j} T^w_{A_j}(n)$$

for sufficiently large n.

□

Of course, if $f \in 2^{\Omega(n)}$, the factor $2^{|x|}$ can be deleted from the stated time, i.e. for classes of exponential time algorithms, we can compute the ensemble almost as fast as the algorithms run.

Corollary 4.1 *There exists an ensemble $\boldsymbol{\mu}$, computable in time $p(2^{|x|}, i)$, where p is a polynomial, so that $\boldsymbol{\mu}$ is malign for the class of polynomial time algorithms.*

Theorem 4.1 and the corollary reflect our intuition from the proof of theorem 3.1: Malignness can be obtained if we are willing to use exponential time. By using the same technique, we can provide a recursive measure that is malign for the classes of algorithms with some recursive upper bound on their running time. It won't provide us with a recursive measure for the class of all algorithms, and, as is mentioned below, no such thing exists. We now turn to a negative result, complementing theorem 4.1.

Theorem 4.2 *There is an $\epsilon > 0$ and a polynomial p, such that for all nondecreasing time constructible function f with $f \in \Omega(p)$, there is no ensemble $\boldsymbol{\mu}$, malign for $ALG(f)$ and computable in time $f(|x|)^\epsilon h(i)$, where h is any function.*

Proof The proof uses our ability to do a binary search for a string of low $\boldsymbol{\mu}$-measure. Given a nondecreasing, time constructible function g, $g \in \Omega(n)$ and any function h, consider an ensemble $\boldsymbol{\mu}$, computable in time $g(|x|)h(i)$. We may assume that h is recursive, since $h(i)$ otherwise can be replaced with $\max_n T_{\boldsymbol{\mu}}(n,i)$ which is recursive. We may furthermore assume that h is time constructible, strictly increasing and tends to infinity, since any general recursive function is dominated by such a function. Define

$$\tilde{h}(n) = \min(\max(1, \max\{j | h(j) < \log(n)\}), n).$$

By h's time constructibility, $\tilde{h}$ can be computed in polynomial time. Furthermore, the polynomial time bound does not depend upon h. Consider the following algorithm, B:

```
input x
y := Λ
for i := 1 to h̃(|x|) do
    v1 := a 2^{-i}-approximation to μ_{|x|}{z | z < y0^{|x|-i+1}}
    v2 := a 2^{-i}-approximation to μ*_{|x|}(y01^{|x|-i})
    v3 := a 2^{-i}-approximation to μ*_{|x|}(y1^{|x|-i+1})
    if v2 − v1 ≤ v3 − v2 then
        y := y0
    else
        y := y1
    fi
od
y := y0^{|x|-|y|}
if x = y then
    idle for q(log(|x|)g(|x|))^2 time steps (q being specified below)
fi
```

The function q should be a polynomial such that $q(\log(|x|)g(|x|))$ is an upper bound for the running time of the algorithm when $x \neq y$. Observe that q can be picked independently of $\boldsymbol{\mu}$, g and h. We may assume that q is of the form $q(t) = t^{\alpha}$. Put $\epsilon = \frac{1}{3\alpha}$. Putting $g(n) = f(n)^{\epsilon}$, we have that B halts on almost all inputs in time $q(\log(|x|)f(|x|)^{\epsilon})^2 = \log(|x|)^{2\alpha} f(|x|)^{\frac{2}{3}}$ which is less than $f(|x|)$ for almost all $|x|$. Thus, $B \in ALG(f)$. By an easy induction, the invariant

$$\mu_{|x|}\{yz|z \in \Sigma^{|x|-i}\} \leq (\frac{3}{4})^{i-3}$$

holds at the end of the i'th cycle of the for loop, so the y found by the for loop has

$$\mu_{|x|}(y) \leq (\frac{3}{4})^{\tilde{h}(|x|)-3}.$$

We then have

$$T_B^a(n) \leq (\frac{3}{4})^{\tilde{h}(n)-3} T_B^w(n) + \sqrt{T_B^w(n)}.$$

But this is smaller than $cT_B^w(n)$, for any $c > 0$, for sufficiently large n, i.e. $\boldsymbol{\mu}$ is not malign for B, which was to be proved.

□

For exponential time algorithms the lower bound matches the upper bound of theorem 4.1 within a polynomial. By a technique similar to the above proof, we can also prove that no recursive measure is malign for the class of all algorithms. This implies that the Solomonoff-Levin measure is not recursive (of course, we could have proven this in a more elementary way, see [4]). For polynomial time algorithms, we have

Corollary 4.2 *No ensemble $\boldsymbol{\mu} \in PE$ is malign for the class of polynomial time algorithms.*

5 Malign ensembles for classes of fast algorithms

Corollary 4.2 still leaves something to be desired. After all, most algorithms we are likely to run will be in e.g. $ALG(n^4)$. It still seems that exponential time is required to compute malign ensembles for such classes. However, we are not likely to be able to prove these intuitions correct, because the following theorem tells us that in order to show that superpolynomial time is necessary, we would have to prove $P \neq NP$. We are reusing the technique from the previous constructions.

Theorem 5.1 *For all k, an ensemble $\boldsymbol{\mu} \in PE^{\Sigma_2^p}$ exists, which is malign for $ALG(n^k)$.*

Proof Put $f(n) = n^k$ in the proof of theorem 4.1. This makes $T(x,i)$ polynomial time computable. Observe that the i'th binary digit of $\mu^*_{|x|}(x)$ is 1 if and only if

$$\exists y \in \Sigma^n \forall z \in \Sigma^n : y \leq x, T(z,i) \leq T(y,i), z < y \Rightarrow T(z,i) < T(y,i)$$

and this is a Σ^p_2-problem.

□

It thus seems that we will have to concentrate on merely making the existence of such a *PE*-ensemble unlikely, instead of trying to prove that it does not exist. We will indeed do this, by deriving from it a complexity-theoretic equality which seems unreasonable, although it is a lot weaker than $P = NP$. For this, we need a result on sampling.

Definition 5.1 *An ensemble $\boldsymbol{\mu}$ is polynomial time samplable if a polynomial time probabilistic Turing machine exists, which on input $< 1^n, 1^i >$ produces a string of length n, $M(n,i)$ such that for all $x \in \Sigma^n$*

$$|\Pr(M(n,i) = x) - \mu_n(x)| \leq 2^{-i}$$

A similar definition and a analogy to the following theorem appears in [8] (theorem 7). Note that we cannot demand that the strings are produced with the exact probability, since it is easy to see that no probabilistic machine with a worst case time bound can produce a string with an irrational probability. Thus, demanding exact sampling would make the following theorem false.

Theorem 5.2 *Every ensemble $\boldsymbol{\mu}$ in PE is polynomial time samplable.*

Proof The proof is similar to the one in [8]. First, we construct a probabilistic algorithm A with no time bound, whose outputs follow the distribution μ_n exactly. Given a finite string $b = b_1 b_2 \ldots b_j$ we denote by $\mathrm{val}(b)$ the number $\sum_{i=1}^{j} b_i 2^{-i}$. Given a one-way infinite tape $b = b_1 b_2 \ldots$, we denote by $\mathrm{val}(b)$ the number $\sum_{i=1}^{\infty} b_i 2^{-i}$. That is, $\mathrm{val}(b)$ is b read as the binary expansion of a real in the interval $[0,1]$. Clearly, if b is a random tape, the random variable $\mathrm{val}(b)$ is uniformly distributed in $[0,1]$. Given a random tape b and an input n, the following algorithm A determine the string x, such that

$$\mathrm{val}(b) \in (\mu^*_n(x-1), \mu^*_n(x)].$$

We will furthermore show that it will succeed with probability 1, i.e. for all tapes except for a set of measure 0. Then, clearly, the algorithm will sample μ.

```
input 1^n
r := Λ; x := Λ
for j := 1 to n do
    repeat
        r := r · 2 random bits (· means concatenation)
        α := a 2^{-|r|}-approximation to μ*_n(x01^{n-j})
        μ_max := α + 2^{-|r|}; μ_min := α − 2^{-|r|}
        v_min := val(r)
        v_max := val(r) + 2^{-|r|}
        { Invariant: v_min ≤ val(random tape) ≤ v_max }
    until μ_max < v_min or v_max ≤ μ_min
    if μ_max < v_min then
        x := x1
    else
        x := x0
    fi
    { Invariant: val(random tape) ∈ (μ_n{y|y < x0^{n-j}}, μ*_n(x1^{n-j})] }
od
output x
```

Since the invariants clearly hold, the algorithm produces the x with the stated property if it halts. It remains to show that it halts with probability 1. Observe that any time we enter the repeat loop, there are four equally likely possibilities for extending r. There are therefore also 4 equally likely possibilities for $[v_{\min}, v_{\max}]$, namely $[\mathrm{val}(r_{\mathrm{old}}), \mathrm{val}(r_{\mathrm{old}}) + j2^{-|r_{\mathrm{new}}|}]$, for $j \in \{1,2,3,4\}$. But since the values of $\mu_{\max}$ and $\mu_{\min}$ are calculated independent of this extension, at most two of the intervals can have $v_{\min} \leq \mu_{\max}$ and $v_{\max} > \mu_{\min}$. Therefore, before each cycle of the repeat loop there is a probability of at least $\frac{1}{2}$ of leaving the repeat loop after that cycle. We then have that at any time the probability of eventually leaving the current repeat loop is 1, and consequently the probability of halting is 1. We now find a p so that

$$\Pr(T_A(1^n) \geq p(i,n)) \leq 2^{-i}.$$

Let C_j, $j \in \{1, \ldots, n\}$ be the random variable denoting the actual number of cycles during the j'th execution of the repeat loop. Let N be the random variable denoting the total number of cycles of the repeat loop during the execution of the algorithm, i.e. $N = \sum C_j$. If $N \geq s$, we can conclude that $\exists j : C_j \geq \frac{s}{n}$. But we have argued that at any time the probability of leaving the repeat loop after its next cycle is at least $\frac{1}{2}$, i.e. for a fixed j:

$$\Pr(C_j \geq \frac{s}{n}) \leq 2^{-\frac{s}{n}+1}.$$

This means that

$$\Pr(N \geq s) \leq n2^{-\frac{s}{n}+1}$$

and

$$\Pr(N \geq (i+1)n^2) \leq n2^{-\frac{(i+1)n^2}{n}+1} =$$

$$2^{-(i+1)(n-\frac{\log n}{i+1})+1} \leq 2^{-(i+1)+1} = 2^{-i}.$$

But since both the number and the precision of the calculated μ_n^*-approximations are linear in N, there is a polynomial r so that $T_A(1^n) \leq r(n, N)$. We may assume without loss of generality that r is strictly increasing in each variable. Therefore

$$\Pr(T_A(1^n) \geq r(n, (i+1)n^2)) \leq \Pr(N \geq (i+1)n^2) \leq 2^{-i}.$$

Now consider a probabilistic Turing machine M which on input $< 1^n, 1^i >$ simulates A on input 1^n for $r(n, (i+1)n^2)$ time steps, outputs whatever A does if A halts during this period, and outputs 0^n otherwise. Clearly, for any $x \in \Sigma^n$,

$$|\Pr(M(< 1^n, 1^i >) = x) - \mu_n(x)| \leq 2^{-i}.$$

□

To get conditionally negative results on the samplability of malign ensembles, we need the following concept.

Definition 5.2 *Given a language L. An* expected polynomial time constructor *for L is a probabilistic algorithm which on input 1^n produces an $x \in L \cap \Sigma^n$ in expected polynomial time if one exists and otherwise fails to halt.*

This is a natural generalization of the deterministic constructors defined and studied by Sanchis and Fulk in [9]. A useful equivalence is

Lemma 5.1 *Every $L \in P$ has an expected polynomial time constructor if and only if every $L \in DTIME(n)$ has one.*

Proof Suppose every $L \in DTIME(n)$ has a constructor. Let J be a language in P. There is a constant c, such that $J \in DTIME(n^c)$. Define $\tilde{J} = \{x10^{|x|^c} | x \in J\}$. It is easy to see that $\tilde{J} \in DTIME(n)$, so, by assumption, $\tilde{J}$ has a constructor. Thus, the following algorithm is an expected polynomial time constructor for J.

input 1^n run $\tilde{J}$'s constructor on input 1^{n+1+n^c} if an output is produced, output its first n digits.

□

Theorem 5.3 *Suppose a polynomial time samplable ensemble $\boldsymbol{\mu}$ is malign for $ALG(f)$, where f is a time constructible function with $n \in o(f)$. Then every $L \in P$ has an expected polynomial time constructor.*

Proof By lemma 5.1, we need only show that every $L \in DTIME(n)$ has a constructor. Consider the following algorithm A:

input x Decide if $x \in L$. If it is not, halt immediately. If it is, wait $\frac{f(n)}{2}$ time-steps before halting.

A is an $ALG(f)$-algorithm, so there exists a c so that $T_A^a(\boldsymbol{\mu}, n) \geq cT_A^w(n)$. We will show that for sufficiently large n,

$$L \cap \Sigma^n \neq \emptyset \Rightarrow \mu_n(L \cap \Sigma^n) \geq \frac{c}{2}.$$

Assume not, i.e. $\mu_n(L \cap \Sigma^n) < \frac{c}{2}$. Then

$$\sum_{x \in L \cap \Sigma^n} \mu_n(x) T_A(x) < \frac{c}{2} T_A^w(n).$$

We also have that for sufficiently large n:

$$\sum_{x \in \Sigma^n \setminus L} \mu_n(x) T_A(x) \leq n.$$

But then

$$\frac{c}{2} T_A^w(n) + n \geq T_A^a(\boldsymbol{\mu}, n) \geq c T_A^w(n).$$

If $L \cap \Sigma^n \neq \emptyset$, the algorithm will run for $\frac{f(n)}{2}$ time steps for some value of x. Therefore

$$n \geq \frac{c}{2} T_A^w(n) \geq \frac{c f(n)}{4}$$

which is a contradiction for sufficiently large n. Let M be a polynomial time sampler for $\boldsymbol{\mu}$. Consider the algorithm A, which on input 1^n simulates M on input $< 1^n, 1^{\lceil -\log \frac{c}{4} \rceil + n} >$ and outputs whatever A does. By the definition of sampling,

$$L \cap \Sigma^n \neq \emptyset \Rightarrow \Pr(A(n) \in L \cap \Sigma^n) \geq \frac{c}{2} - \frac{c}{4} = \frac{c}{4}.$$

Now, the following algorithm is an expected polynomial time constructor for L.

```
input 1^n
repeat
    sample x of size n from μ̃_n
until x ∈ L
```

It runs in expected polynomial time because each cycle of the loop is polynomial time, and, by the above, the probability that each sampled x is in L is at least $\frac{c}{4}$.

□

The following theorem (an analog to proposition 4.1 in [9]) make expected polynomial time constructors for all languages in P unlikely.

Definition 5.3 *Let RE be the class of languages L for which there exist a probabilistic Turing machine, running in time $2^{c|x|}$ on input x, rejects if $x \notin L$, and accepts with probability at least $\frac{1}{2}$ if $x \in L$.*

Thus, RE is for $E = \cup_{c\geq 0} DTIME(2^{cn})$ what RP is for P. Hence, RE should be considered a rather small extension of E, and $RE = NE$, where $NE = \cup_{c\geq 0} NTIME(2^{cn})$ must be considered only slightly more plausible than $E = NE$. Actually, by a proof identical to the one used by Hartmanis in [10], $RE = NE$ if and only if there are no sparse languages in $NP - RP$.

Theorem 5.4 *If every $L \in P$ has an expected polynomial time constructor, then $RE = NE$.*

Proof Let L be a language in NE, and let M be a nondeterministic machine, running in time 2^{cn} which recognizes L. We can represent the nondeterministic computations of M on input x as binary strings of length $\leq 2^{c|x|}$, where the bits represent the nondeterministic choices of M. Denote by x_n^i the lexicographically i'th string of size n. Define $f : \Sigma^* \to \mathbf{N}$ by $f(x_n^i) = 2^{cn} + i$. The function f is clearly injective, provided $c \geq 1$, which we may assume. Now consider

$$\tilde{L} = \{y |\ \exists x : |y| = f(x) \text{ and } y \text{ codes accepting computation of } M \text{ on } x\}.$$

Clearly $\tilde{L} \in P$ and has therefore, by assumption, an expected polynomial time constructor, C. Let $p(n)$ be an upper bound on C's expected running time on inputs of size n. Consider the following algorithm:

Input x
Generate $1^{f(x)}$
Simulate C on $1^{f(x)}$ for $2p(f(x))$ time steps.
If an element has been produced by then, then accept, otherwise reject.

This obviously is a RE-acceptor for L.

□

Corollary 5.1 *If an ensemble $\boldsymbol{\mu} \in PE$ is malign for $ALG(f)$, where f is time constructible and $n \in o(f)$, then $RE = NE$.*

Acknowledgment

Thanks to Joan Boyar for her helpful supervision on the thesis on which this paper is based and to Sven Skyum for useful comments and suggestions.

References

[1] M. Li, P.M.B. Vitanyi. *A theory of learning simple concepts under simple distributions and average case complexity for the universal distribution (preliminary version),* ITLI Prepublication Series for Computation and Complexity Theory CT-89-07, University of Amsterdam (1989)[1].

[2] M. Li, P.M.B. Vitanyi. *Average case complexity under the universal distribution equals worst case complexity,* manuscript.

[3] M. Li, P.M.B. Vitanyi. *Inductive reasoning and Kolmogorov complexity (preliminary version),* Proceedings from the 4th IEEE Structure in Complexity Theory Conference (1989), 165-185.

[4] M. Li, P.M.B. Vitanyi. *Kolmogorov complexity and its applications,* Report CS-R8901, CWI, University of Amsterdam (1989).

[5] A.K. Zvonkin, L.A. Levin. *The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms,* Russ. Math. Surv. 25 (1970) pp. 83-124.

[6] L.A. Levin. *Average case complete problems,* Siam Journal of Computing, vol. 15 (1986) pp. 285-286.

[7] Y. Gurevich. *Complete and incomplete randomized NP problems,* Proceedings of the 28th IEEE FOCS (1987) pp. 111-117.

[8] S. Ben-David, B. Chor, O. Goldreich, M. Luby. *On the theory of average case complexity,* Proceedings of the 21st ACM STOC (1989) pp. 204-216.

[9] L.A. Sanchis, M.A. Fulk. *On the efficient generation of language instances,* Siam Journal of Computing, vol. 19 (1990) pp. 281-296.

[10] J. Hartmanis. *On sparse sets in NP - P,* Inform. Process. Lett., 16 (1983) pp. 55-60.

[1] An earlier version appeared at the 1989 FOCS.