Encoding Invariance in Average Case Complexity[*]

Nikolay Vereshchagin[†‡]

Abstract

When we represent a decision problem, like CIRCUIT-SAT, as a language over the binary alphabet, we usually do not specify how to encode instances by binary strings. This relies on the empirical observation that the truth of a statement of the form "CIRCUIT-SAT belongs to a complexity class C" does not depend on the encoding, provided both the encoding and the class C are "natural". In this sense most of the Complexity theory is "encoding invariant".

The notion of a polynomial time computable distribution from Average Case Complexity is one of the exceptions from this rule. It might happen that a distribution over some objects, like circuits, is polynomial time computable in one encoding and is not polynomial time computable in the other encoding. In this paper we suggest an encoding invariant generalization of a notion of a polynomial time computable distribution. The completeness proofs of known distributional problems, like Bounded Halting, are simpler for the new class than for polynomial time computable distributions.

This paper has no new technical contributions. All the statements are proved using the known techniques.

[*] This is the full version of the paper "An Encoding Invariant Version of Polynomial Time Computable Distributions" appeared in Proceedings of the 5th International Computer Science Symposium in Russia, CSR 2010, Kazan, Russia, June 16-20, 2010. Lecture Notes in Computer Science v. 6072, p. 371–383.

[†] The work was in part supported by a RFBR grant 09-01-00709

[‡] Moscow State University, Leninskie gory 1, Moscow 119992, Russia, E-mail: ver@mccme.ru, WWW home page: http://lpcs.math.msu.su/~ver

1 Polynomial time samplable and computable distributions

We are going to introduce the main notions from Average Case Complexity. We will follow the paper [1] of Bogdanov and Trevisan. The main goal of the theory is to show that certain NP problems are "hard on average". More specifically, we want to provide evidence that for certain NP problems L there is a "simple" probability distribution D on their instances such that every efficient algorithm errs with non-negligible probability on a randomly chosen instance. To this end we define reductions between pairs (problem, distribution) and show that the pair in question is complete in a large class of such pairs.

Definition 1 *([2, 1]).* A *probability distribution over* $\{0,1\}^*$ is a function D from $\{0,1\}^*$ to the non-negative reals such that $\sum_x D(x) = 1$. An *ensemble of distributions* $\mathcal{D}$ is a sequence

$$D_0, D_1, \ldots, D_n, \ldots$$

of probability distributions over $\{0,1\}^*$. (The parameter n is called the *security parameter*.) We say that ensemble $\mathcal{D}$ is *polynomial time samplable* if there is a randomized algorithm A that with an input n outputs a string in $\{0,1\}^*$ and:

- there is a polynomial p such that, on input n, A runs in time at most $p(n)$ regardless of its internal coin tosses;
- for every n and every $x \in \{0,1\}^*$, $\mathbf{Pr}[A(n) = x] = D_n(x)$.

We will call such an algorithm A a *sampler for* $\mathcal{D}$.

A *distributional problem* is a pair $(L, \mathcal{D})$ where L is a language over the binary alphabet and $\mathcal{D}$ is an ensemble of distributions. We say that a distributional problem $(L, \mathcal{D})$ is in $(\mathrm{NP}, \mathrm{PSamp})$ if $L \in \mathrm{NP}$ and $\mathcal{D}$ is polynomial time samplable.

Note that, in the defintion of polynomial time samplability, we do not require that the support of distribution D_n consist of strings of length exactly n. From the definition it follows only that it consists of strings of length at most $p(n)$.

The goal of Average Case Complexity is to show that certain distributional problems are $(\mathrm{NP}, \mathrm{PSamp})$ complete under reductions of certain kinds which preserve “simplicity on average”. We will not define here the type of reductions used in the definition of completeness, and refer to [2, 4, 1] for the definition. We will define simplified reductions (Definition 3.1 from [1]) that come back to [5]. These simplified reductions will suffice for the goal of this paper.

Definition 2. We say that ensemble $\mathcal{D}$ is *dominated* by an ensemble $\mathcal{D}'$ if there is a polynomial $p(n)$ such that for all n, x,

$$D_n(x) \leq p(n) D'_n(x).$$

We say that $(L, \mathcal{D})$ *reduces to* $(L', \mathcal{D}')$ if there is a function $f(x, n)$ that for every n and every x in the support of D_n can be computed in time polynomial in n and

- (Correctness) $x \in L$ if and only if $f(x, n) \in L'$;
- (Domination) There is a polynomial $q(n)$ such that the distribution

  $$D''_n(y) = \sum_{x : f(x,n) = y} D_n(x)$$

  is dominated by $D'_{q(n)}(y)$.

In particular $(L, \mathcal{D})$ always reduces to $(L, \mathcal{D}')$ provided $\mathcal{D}'$ dominates $\mathcal{D}$.

The following argument justifies this defintion. Assume that there is an algorithm A that on input x and parameter n solves decision problem L' with a negligible error probability ε_n for x randomly chosen with respect to D'_n. Then the algorithm $A(f(x, n), q(n))$ solves decision problem L with (negligible) error probability $p(n)\varepsilon_{q(n)}$ with respect to distribution D_n.

Remark 1. If the function f is injective (for every fixed n) then the domination condition is easier to understand. In this case $D''_n(f(x, n)) = D_n(x)$ and the domination condition boils down to requiring that

$$D_n(x) \leq p(n) D'_{q(n)}(f(x, n))$$

for some polynomials p, q.

Now we are going to define polynomial time computable ensembles of distributions. We will follow the exposition of [1]. Let $\preceq$ denote the lexicographic ordering between bit strings. If D is a distribution over $\{0, 1\}^*$ we define

$$f_D(x) = D(\{y \mid y \preceq x\}) = \sum_{y \preceq x} D(y).$$

The function f_D is called *cumulative* probability for distribution D.

Definition 3 *(Levin [5], Bogdanov–Trevisan [1]).* We say that ensemble $\mathcal{D}$ is *polynomial time computable* if there is an algorithm that given an integer n and a string $x \in \{0,1\}^*$, runs in time polynomial in n and computes $f_{D_n}(x)$. Let $(\mathrm{NP}, \mathrm{PComp})$ stand for the class of distributional problems $(L, \mathcal{D})$ where $L \in \mathrm{NP}$ and $\mathcal{D}$ is polynomial time computable.

Neither Levin, nor Bogdanov and Trevisan specify the meaning of polynomial time computability of a real valued function. To interpret Defintion 3 in a right way, we note that [1] claims that polynomial time computability implies polynomial time samplability. Notice that if $\mathcal{D}$ is a polynomial time samplable ensemble then $D_n(x)$ is always a dyadic rational. Therefore we will assume that, in the Definition 3, $f_{D_n}(x)$ is always a dyadic rational and the algorithm computing $f_{D_n}(x)$ outputs the numerator and denominator of $f_{D_n}(x)$ in binary notation. With this interpretation, every polynomial time computable ensemble is indeed polynomial time samplable, see Theorem 2 below.

This interpretation of Definition 3 has an obvious minor point: we restrict possible values of $f_{D_n}(x)$ to dyadic rationals. Therefore it is natural to consider the following relaxation of Definition 3.

Definition 4. An ensemble $\mathcal{D}$ is called *weakly polynomial time computable* if there is an algorithm that given integers n, m and a string $x \in \{0,1\}^*$, runs in time polynomial in $n+m$ and computes a rational number within a distance at most 2^{-m} from $f_{D_n}(x)$.

In this definition we allow $D_n(x)$ to be any non-negative real. It is not hard to see that every weakly polynomial time computable ensemble is dominated by a polynomial time computable ensemble (see Theorem 1 below) and thus both definitions are basically the same. In both definitions, the length of all strings in the support of D_n is bounded by a polynomial in n (otherwise it is not clear how a polynomial time algorithm can read x's).

Theorem 1 ([5]). *Every weakly polynomial time computable ensemble is dominated by a polynomial time computable ensemble.*

This theorem was essentially proven in [5] (although not explicitly stated there). For the sake of completeness we present the proof.

Proof. It suffices to prove the statement for weakly polynomial time computable ensembles that have the following property: there is a polynomial

$p(n)$ such the support of D_n is the set of all strings of length less than $p(n)$ and $D_n(x) \geq 2^{-p(n)-1}$ for all strings in the support of D_n. Indeed, let $p(n)$ stand for a polynomial such that the length of all strings in the support of D_n is strictly less than $p(n)$. Consider the distribution D'_n which is the arithmetic mean of D_n and the uniform distribution over all strings of length less than $p(n)$ (the latter assigns probability $1/(2^{p(n)+1} - 1)$ to all strings of length less than $p(n)$). Obviously, $\mathcal{D}'$ is weakly polynomial time computable, dominates D_n and $D'_n(x) > 2^{-p(n)-1}$ for all strings in the support of D'_n. As the domination relation is transitive, every ensemble dominating $\mathcal{D}'$ also dominates $\mathcal{D}$.

So assume that $\mathcal{D}$ is weakly polynomial time computed by an algorithm A and $D_n(x) \geq 2^{-p(n)-1}$ for some polynomial $p(n)$ and all x in the support of D_n, which consists of all strings of length less than $p(n)$. Let $g_n(x)$ stand for the dyadic rational number that is produced by A for the input triple $(n, p(n)+3, x)$ so that $|f_{D_n}(x) - g_n(x)| \leq 2^{-p(n)-3}$. Let, additionally, $g_n(x) = 1$ for the lexicographical largest string x of length less than $p(n)$. Finally, let $D'_n(x)$ denote the difference of $g_n(x)$ and $g_n(y)$ for the predecessor y of x. Then the values of $D'_n(x)$ sum up to 1. We claim that $D'_n(x)$ is always non-negative and dominates $D_n(x)$ and thus satisfies the theorem.

By construction we have

$$|f_{D_n}(z) - g_n(z)| \leq 2^{-p(n)-3}$$

for all z. Applying this inequality to x and its predecessor we conclude that

$$|D'_n(x) - D_n(x)| \leq 2^{-p(n)-2} \leq D_n(x)/2$$

and hence $D'_n(x) \geq D_n(x)/2$. □

The common scheme to prove (NP, PSamp) completeness is as follows. Assume that we want to show that $(L, \mathcal{D})$ is (NP, PSamp) complete. First we show that $(L, \mathcal{D})$ is (NP, PComp) complete with respect to reductions of Definition 2. Second, we use the result of [4, 2] stating that every distributional problem in (NP, PSamp) reduces to some distributional problem in (NP, PComp) (using reductions that are weaker than those of Definition 2).

The goal of this paper is to simplify this scheme. Namely, in place of the class (NP, PComp) we suggest to use a wider class, which we call (NP, PISamp). Ensembles of distributions from PISamp will be called "polynomial time invertibly samplable". Proving (NP, PISamp) completeness is easier than proving $(\mathrm{NP}, \mathrm{PComp})$ completeness. Besides the class PISamp is encoding invariant (in the sense that is explained in the next section), whereas the class PSamp might be not.

2 Encoding invariance

Let us specify what we mean by "encoding invariant" notions in Complexity theory. Fix a set X of "objects" (like boolean circuits). An *encoding of* X is an injective mapping g from X to the binary strings. To every decision problem L with instances from X and every encoding g of X we assign the language

$$L_g = \{g(x) \mid x \text{ is a YES-instance of } L\}.$$

We say that decision problem L in encoding g belongs to a complexity class C if $L_g \in C$.

It turns out that different "natural" encodings of boolean circuits are equivalent in the following sense. We call g_1 and g_2 *poly-time equivalent* if both functions $g_1(g_2^{-1}(x))$ and $g_2(g_1^{-1}(x))$ are computable in polynomial time. Note that these functions map strings to strings and thus the notion of polynomial time computability is meaningful in this context.

If encodings g_1 and g_2 are poly-time equivalent then $L_{g_1} = h(L_{g_2})$ for a polynomial time computable and invertible partial function (namely for $h(x) = g_1(g_2^{-1}(x))$). Let us call a class C of languages over a binary alphabet *encoding invariant* if for every $L \in C$ and every polynomial time computable and invertible partial function h we have $h(L) \in C$. Natural complexity classes above P, like NP or BPP, are encoding invariant.

We do we care about encoding invariance? The point is that natural encodings of the same natural set of objects X are poly-time equivalent (like in the case of boolean circuits). Thus we can freely say that a decision problem L is in an encoding invariant class C meaning that $L_g \in C$ without specifying the encoding g used. For encoding non-invariant class, like the class of languages having AC^0 circuits of depth 5 (say), we should carefully specify the encoding used, which is not convenient. For example, the question of whether the problem "Boolean circuit evaluation" can be solved by AC^0 circuits of depth 5 is meaningless.

The following straightforward lemma states that PSamp is an encoding invariant class. Assume that we are given computable injective mappings (encodings) g_1, g_2 from a set X of objects to the binary strings. Assume that D is a probability distribution over binary strings that represents a distribution over X in encoding g_1. Then $D(g_1(g_2^{-1}(x)))$ represents the same distribution in encoding g_2. Obviously $h(x) = g_1(g_2^{-1}(x))$ is a partial polynomial time computable and invertible injective function (i.e., both $h(x), h^{-1}(x)$ are computable in time polynomial in the length of x) and the support of D is included in the range of h.

Lemma 1. *Assume that $\mathcal{D}$ is polynomial time samplable and $h : \{0,1\}^* \to \{0,1\}^*$ is a partial polynomial time computable and invertible injective function and for all n the support of D_n is included in the range of h. Then the ensemble $\mathcal{D}^h$ where*

$$D_n^h(x) = \begin{cases} D_n(h(x)) & \textit{if } h(x) \textit{ is defined,} \\ 0 & \textit{otherwise} \end{cases}$$

is polynomial time samplable.

In this lemma, the condition on the support of D_n ensures that $D_n^h(x)$ is a probability distribution.

It is not clear whether Definitions 3 and 4 are encoding invariant[1]. Indeed, assume that an ensemble of distributions $\mathcal{D}$ is polynomial time computable and h is a polynomial time computable and invertible partial function. There is no guarantee that $\mathcal{D}^h$ is polynomial time computable: the function $h(x)$ might not preserve lexicographical order.

3 Polynomial time invertibly samplable distributions

Let A be a polynomial time probabilistic algorithm, as in the definition of a samplable distribution. Think of the source of randomness for A as a real number in the segment $[0, 1)$ with respect to the uniform distribution. More precisely, if a computation of $A(n)$ returns x where $r = r_1 \dots r_l$ is the sequence of outcomes of coin tosses made in that computation, we will think that $A(n)$ maps all reals in the half-interval $[0.r, 0.r+2^{-l})$ to x. In this way A defines a mapping from the set $\mathbb{N} \times [0, 1)$ to $\{0,1\}^*$ and we denote by $A(n, \alpha)$ the result of A for input n and randomness $\alpha \in [0, 1)$. Let

$$A^{-1}(n, x) = \{\alpha \in [0, 1) \mid A(n, \alpha) = x\}.$$

[1] And they are not provided one-way permutations exist, see Remark 1.

In general, $A^{-1}(n,x)$ is a union of a finite number of segments and each of them has the form $[k/2^l,(k+1)/2^l)$.

Definition 5. We call A, as above, *polynomial time invertible*, if for all n and all x in the support of D_n, the set $A^{-1}(n,x)$ is *one* subsegment of $[0,1)$ which can be computed from n,x in time polynomial in n. (Segments are represented by numerators and denominators of their end-points, written in binary notation.) We say that a polynomial time samplable ensemble $\mathcal{D}$ is *polynomial time invertibly samplable* if there is a polynomial time invertible sampler for $\mathcal{D}$.

Remark 2. If $\mathcal{D}$ is polynomial time invertibly samplable then $D_n(x)$ is a dyadic rational that can be computed from x in time polynomial in n. However this does not imply that the cumulative function of D_n is polynomial time computable.

It is easy to see that the class of polynomial time invertibly samplable distribution is encoding invariant (in the sense of Lemma 1). The next theorem shows that PComp $\subseteq$ PISamp and thus PISamp is an encoding invariant generalization of PComp.

Theorem 2 ([5]). *Every polynomial time computable ensemble of distributions is polynomial time invertibly samplable.*

This theorem was essentially proven in [5] (although not explicitly stated there). Actually the main idea of the proof comes back to Fano and Shannon (the so called Shannon-Fano codes). For the sake of completeness we present the proof.

Proof. As $f_{D_n}(x)$ is computable in time $p(n)$ (for some polynomial p), the values of $f_{D_n}(x)$ are always of the form $k/2^l$ where $l \le p(n)$ and $k \le 2^l$.

The sampler A for D_n is as follows:

> Choose at random a segment I of the form $\big[k/2^{p(n)},(k+1)/2^{p(n)}\big)$ with $k < 2^{p(n)}$ (tossing a coin $p(n)$ times).
>
> Using a binary search, find x such that
>
> $$I \subseteq \Big[\sum_{y\prec x} D_n(y), \sum_{y\prec x} D_n(y) + D_n(x)\Big).$$
>
> Output x and halt. Here we use the fact that the cumulative distribution is polynomial time computable. We use also that the

length of all x's in the support of D_n is bounded by a polynomial in n and thus binary search finishes in a polynomial number of steps.

The algorithm A is polynomial time invertible. Indeed,

$$A^{-1}(n,x) = \big[\sum_{y \prec x} D_n(y), \sum_{y \prec x} D_n(y) + D_n(x)\big)$$

for every x in the support of A. The endpoints of this interval are $f_{D_n}(x)$ and $f_{D_n}(y)$ for predecessor y of x, thus poly-time computable by the assumptions of the theorem. □

Now we will explain why proving $(\mathrm{NP}, \mathrm{PISamp})$ completeness is easier than proving $(\mathrm{NP}, \mathrm{PComp})$ completeness. Assume that we have to show that an arbitrary problem $(L', \mathcal{D}') \in (\mathrm{NP}, \mathrm{PComp})$ reduces to a fixed $(\mathrm{NP}, \mathrm{PComp})$ complete problem $(L, \mathcal{D})$. An analysis of the proof reveals that we use the assumption about polynomial time computability of $\mathcal{D}'$ to construct an invertible sampler for $\mathcal{D}'$. Thus to prove that $(L, \mathcal{D})$ is $(\mathrm{NP}, \mathrm{PISamp})$ complete is easier than to prove that it is $(\mathrm{NP}, \mathrm{PComp})$ complete: we can skip this step. On the other hand, for all known $(\mathrm{NP}, \mathrm{PComp})$ complete problem $(L, \mathcal{D})$ it is immediate that $\mathcal{D}$ is in PISamp. Thus the proof of $(\mathrm{NP}, \mathrm{PISamp})$ completeness of $(L, \mathcal{D})$ has one step less than that of its $(\mathrm{NP}, \mathrm{PComp})$ completeness. An example of $(\mathrm{NP}, \mathrm{PISamp})$ completeness proof is presented in Section 4.

The next theorem shows that under certain plausible assumptions PComp $\neq$ PISamp $\neq$ PSamp.

Theorem 3. *(a) If one-way functions exist, then there is a polynomial time samplable ensemble that is not dominated by any polynomial time invertibly samplable ensemble. (b) Assume that for some polynomial p there is a one-way function $f_n : \{0,1\}^n \to \{0,1\}^{p(n)}$ such that every string $y \in \{0,1\}^{p(n)}$ has at most poly(n) pre-images under f_n. Then there is a polynomial time invertibly samplable ensemble that is not dominated by any polynomial time computable ensemble.*

Item (a) implies that there is a polynomial time samplable ensemble that is not dominated by any polynomial time computable ensemble (provided one-way functions exist). The latter fact was proved in [2] using the techniques of (an earlier version of) [3]. Our proof will use just the result of [3] (the existence of Pseudorandom Generators).

Proof. (a) In [3], it is shown that if one-way functions exist, then there is a Pseudorandom Generator $G_n : \{0,1\}^n \to \{0,1\}^{2n}$. Consider G_n as a sampler and let D_n denote the sampled distribution:

$$D_n(x) = \mathbf{Pr}[G_n(s) = x],$$

where s denotes a randomly chosen string of length n. Assume, by way of contradiction, that $\mathcal{D}$ is dominated by a polynomial time invertibly samplable ensemble $\mathcal{D}'$ so that

$$D_n(x) \le p(n) D'_n(x) \tag{1}$$

for some polynomial $p(n)$. Using D'_n we will construct a polynomial time test that distinguishes the random variable $G_n(s)$ from the random variable which is uniformly distributed among all strings of length $2n$, which is a contradiction.

As such test consider the set

$$X_n = \{x : |x| = 2n,\ D'_n(x) \ge 2^{-n}/p(n)\}.$$

As $\mathcal{D}'$ is invertible samplable, the set X_n is polynomial time decidable. Indeed, $D'_n(x)$ is a dyadic rational that can be computed from x and n in time polynomial in n. We claim that $G_n(s)$ is in X_n for all s, whereas a random string of length $2n$ falls in X_n with negligible probability $p(n)2^{-n}$.

The first claim follows from (1). Indeed, as G_n is a sampler for D_n, for every $s \in \{0,1\}^*$ we have $D_n(G_n(s)) \ge 2^{-n}$ and hence $D'_n(G_n(s)) \ge 2^{-n}/p(n)$.

The second claim means that $|X_n| \le 2^n p(n)$. This fact follows from definition of X_n. Indeed, otherwise the cumulative probability of X_n w.r.t. D'_n would be larger than 1:

$$D'_n(X_n) = \sum_{x \in X_n} D'_n(x) \ge |X_n| 2^{-n}/p(n) > 1.$$

(b) Let f_n be the function from the assumption in item (b). We claim that the ensemble

$$D_n(z) = \begin{cases} 1/2^n, & \text{if } z = f_n(x)x,\ |x| = n, \\ 0, & \text{otherwise} \end{cases}$$

satisfies the conclusion of item (b).

$\mathcal{D}$ is polynomial time invertibly samplable: The following algorithm A is an invertible sampler for D_n: choose a string x of length n at random, return $f_n(x)x$ and halt. As f_n is polynomial time computable, A is indeed a polynomial time algorithm. Let us show that A is invertible in polynomial time. The following algorithm finds $A^{-1}(z,n)$: represent z in the form yx where $|x| = n$ (if $|z| < n$ then halt); apply f_n to x; if $y \neq f_n(x)$ then halt and otherwise output $[0.x, 0.x + 2^{-n})$ and halt.

$\mathcal{D}$ is not dominated by any polynomial time computable ensemble. Let us first prove a simpler statement: $\mathcal{D}$ is not polynomial time computable. By way of contradiction assume that this is not the case. How do we use this assumption? Under this assumption, we can find in polynomial time $D_n(\{zw \mid w \in \{0,1\}^{p(n)+n-|z|}\})$ for any given z of length at most $p(n)+n$. Indeed, it is equal to $f_{D_n}(z11\dots1) - f_{D_n}(z'11\dots1)$ for the predecessor z' of z w.r.t. lexicographical ordering. Here it is important that in the lexicographical ordering the first bits are the most significant ones.

We will show that given *any* y in the range of f_n we can find by binary search in polynomial time its lexicographical first pre-image. To this end we use the following simple fact: a string y has a pre-image with prefix z iff

$$D_n(\{yzw \mid w \in \{0,1\}^{n-|z|}\}) \geq 2^{-n}. \tag{2}$$

Recall that for any y, z we can decide in polynomial time whether (2) holds. Thus given any y of length $p(n)$ we can first find whether it has a pre-image at all, which happens iff (2) holds for the empty z. If this is the case we can find whether y has a pre-image starting with 0, which happens iff (2) holds for $z = 0$, and so on.

Now we will prove that $\mathcal{D}$ is not dominated by any polynomial time computable ensemble. By way a contradiction, assume that $\mathcal{D}'$ is polynomial time computable ensemble with

$$D'_n(z) \geq D_n(z)/q(n)$$

for a polynomial q. We will construct a polynomial time algorithm that inverts $f_n(x)$ for at least half of all x's of length n, which is a contradiction with non-invertibility of f_n.

Let us try first the same algorithm as before, but this time using D'_n instead of D_n. If y has a pre-image with prefix z then

$$D'_n(\{yzw \mid w \in \{0,1\}^{n-|z|}\}) \geq D_n(\{yzw \mid w \in \{0,1\}^{n-|z|}\})/q(n) \geq 1/2^n q(n),$$

but, unfortunately, not the other way around. Indeed, it may happen that

$$D'_n(\{yzw \mid w \in \{0,1\}^{n-|z|}\}) \geq 1/2^n q(n) \tag{3}$$

but y has no pre-image with prefix z.

How to fix this? Note that if we find the list of all x's with $D'_n(yx) \geq 1/2^n q(n)$ then we are done, as all pre-images of y are in that list and we thus can find one of them by probing all x's from the list. The problem is that the list can be super-polynomially big, and thus there is no hope to find it in polynomial time. The solution is as follows: we mentally divide all y's into "good" and "bad" ones. For good y's the set of x's with $D'_n(yx) \geq 1/2^n q(n)$ will have at most polynomial number of elements, and we will be able to find all of them by binary search, as before. For bad y's, we know nothing about the cardinality of this set. However, $f_n(x)$ will be bad for at most half of x of length n, and recall that we tolerate the probability 1/2 of failure.

Specifically, call y *good* if

$$D'_n(\{yx : |x| = n\}) \leq s(n)/2^{n-1} \tag{4}$$

where $s(n)$ stands for a polynomial upper bound for the number of pre-images under f_n.

We have to prove that there is a polynomial time algorithm that finds a pre-image of every given good y in the range of f_n and that $f_n(x)$ is bad for at most half of x of length n.

Assume that y is good, that is, Equation (4) holds. There are at most $2q(n)s(n)$ strings x with $D'_n(yx) \geq 1/2^n q(n)$, as otherwise the sum of their probabilities would exceed $s(n)/2^{n-1}$. Moreover, there are at most $2q(n)s(n)$ pairwise incomparable strings z such that (3) holds (we call z' and z'' incomparable if neither is a prefix of the other). Therefore we can find by binary search in polynomial time all maximal z's satisfying (3) (we call z maximal, if no its proper extension satisfies (3)). If a pre-image of y exists, the it is certainly among those z's.

Assume that $f_n(x)$ is bad. That is, x happens to be a pre-image of a bad y. The number of bad y's is at most $2^{n-1}/s(n)$. Indeed, otherwise the sum of $D'_n(\{yx : |x| = n\})$ over all bad y's would exceed 1. Every bad y has by assumption at most $s(n)$ pre-images. Hence the number of x such that $f_n(x)$ is bad is at most 2^{n-1}. □

Remark 3. Under the same hypothesis and by the same arguments, as in item (b), we can show that the class PComp is not encoding invariant. Indeed, using the notations from the proof, let h be cyclic shift by n bits on strings of length $n+p(n)$: $h(xy) = yx$. Obviously, h is polynomial time computable and invertible. The distribution $\mathcal{D}^h$ is then defined by the equation

$$D_n^h(z) = \begin{cases} 1/2^n, & \text{if } z = xf_n(x),\ |x| = n, \\ 0, & \text{otherwise.} \end{cases}$$

It is not hard to see that $\mathcal{D}^h$ is polynomial time computable.

4 Completeness proof

Let $(\mathrm{NP}, \mathrm{PISamp})$ denote the class of all distributional problem $(L, \mathcal{D})$ such that $L \in \mathrm{NP}$ and $\mathcal{D}$ is polynomial time invertibly samplable. We will prove that the "Bounded halting" distributional problem is $(\mathrm{NP}, \mathrm{PISamp})$ complete.

Definition 6. (Bounded halting distributional problem). Let

$$\begin{aligned} \mathrm{BH} = \{(M, x, 1^t) \mid\ & M \text{ is a program of a non-deterministic} \\ & \text{Turing machine that accepts } x \text{ in at most } t \text{ steps}\} \end{aligned}$$

We assume here that programs of Turing machines are encoded by binary strings in such a way that a universal Turing machine can simulate M with overhead $\mathrm{poly}(|M|)$: each step of machine M is simulated in $\mathrm{poly}(|M|)$ steps. We also assume that triples of strings are encoded by strings so that both encoding and decoding functions are polynomial time computable.

Define the probability distribution U_n on triples of binary strings as follows:

$$U_n(M, x, 1^t) = \begin{cases} 2^{-2l-|x|-|M|}, & \text{if } |x| < 2^l,\ |M| < 2^l,\ t = n, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $l = \lceil \log_2 n \rceil$.

Lemma 2. *Ensemble $\mathcal{U}$ is polynomial time invertibly samplable.*

Proof. The sampler $A(n)$ computes $l = \lceil \log_2 n \rceil$, tosses the coin $2l$ times and interprets the outcome as two integers a, b in the range $0, \dots, 2^l - 1$. Then it tosses a coin a times to obtain M, then b times to obtain x and finally returns the triple $M, x, 1^n$. It is straightforward that A is polynomial time invertible. □

Theorem 4 ([5]). *The distributional problem* $(BH,\mathcal{U})$ *is* $(NP, PISamp)$ *complete. That is, every distributional problem* $(L, \mathcal{D}) \in (NP, PISamp)$ *reduces to* $(BH,\mathcal{U})$ *in the sense of Definition 2.*

The proof of this theorem is not novice: basically, it is a part of the proof from [5] of $(\mathrm{NP}, \mathrm{PComp})$ completeness of $(\mathrm{BH},\mathcal{U})$. More specifically, to prove Theorem 4 we just skip in the latter proof its first step, which is basically proving that the given ensemble of distributions is in PISamp.

We start the proof with a lemma. We will call segments of the form $[0.r0, 0.r1)$ *standard segments*. The length of this segment is $2^{-|r|}$. The string r will be called the *name* of $[0.r0, 0.r1)$.

Lemma 3. *Let* $\mathcal{D}$ *be a polynomial time invertibly samplable ensemble and* A *an polynomial time invertible sampler for* $\mathcal{D}$*. Given* n *and any* x *in the support of* D_n *we can compute in time polynomial in* n *a standard segment* $I(x, n)$ *of length at least* $D_n(x)/4$ *with* $I(x, n) \subseteq A^{-1}(n, x)$*.*

Proof. Let $I(x, n)$ be a largest standard segment inside $A^{-1}(n, x)$. By assumption we can find $A^{-1}(n, x)$ and hence $I(x, n)$ in polynomial time from x, n.[2] It remains to show that its length is at most 4 times less than $D_n(x)$. Indeed, otherwise the segment $A^{-1}(n, x)$ contains three consecutive standard segments. Then a pair of them (the first and the second ones or the second and the third ones) can be united into a larger standard segment inside $A^{-1}(n, x)$. □

Proof of Theorem 4. Assume that $(L, \mathcal{D})$ is the given distributional problem to reduce to $(\mathrm{BH},\mathcal{U})$. Why not to do it in the standard way? Namely, fix a non-deterministic machine M accepting L in polynomial time $q(n)$ and consider the reduction

$$f(x, n) = (M, x, 1^{q(n)}). \tag{6}$$

This reduction certainly satisfies the correctness requirement. However it might violate the domination condition. As f is injective, the domination requirement is met iff

$$D_n(x) \le p(n) U_{q(n)}(M, x, 1^{q(n)})$$

[2] To find the largest standard segment inside a given segment, we can use binary search.

for some polynomial p. Up to a polynomial factor, $U_{q(n)}(M, x, 1^{q(n)})$ equals $2^{-|x|}$, which might be much smaller than $D_n(x)$. Thus to ensure the domination condition we need to replace x in the right hand side of (6) by its representation $\hat{x}$ in at most $-\log_2 D_n(x)+O(\log n)$ bits so that $2^{-|\hat{x}|} \geq D_n(x)/p(n)$ for a polynomial p. This is can be done using Lemma 3. Indeed, fix an invertible sampler A for $\mathcal{D}$ and let $I(x, n)$ be the mapping existing by Lemma 3. Let $r_{x,n}$ stand for the name of $I(x, n)$. The string $r_{x,n}$ together with n form a desired succinct representation of x (we need n, as the algorithm A cannot find x from $r_{x,n}$ alone). The number n should be represented by its "self-delimiting" description $\hat{n}$ in $O(\log n)$ bits . Indeed, we need to parse $\hat{n}r_{n,x}$ into $\hat{n}$ and $r_{n,x}$. For instance, we can set $\hat{n}$ to be the binary notation of n with all bits doubled and appended by 01 (e.g., $\hat{5} = 11001101$) so that $|\hat{n}| \leq 2\log_2 n + 2$.

So the reduction f is defined by

$$f(x, n) = (N, \hat{n}r_{x,n}, 1^{q(n)}) \tag{7}$$

where N is a non-deterministic (not necessarily polynomial time) Turing machine working as follows:

(1) on input z, reject if z is not of the form $\hat{n}r$ for any natural n,
(2) otherwise (when $z = \hat{n}r$) interpret r as the name of a standard segment $I \subseteq [0, 1)$,
(3) run $A(n)$ using any real in I as randomness needed for $A(n)$; let x be the output of $A(n)$,
(4) run $M(x)$ where M is the fixed non-deterministic polynomial time machine accepting L.

The running time of N for input $\hat{n}r_{x,n}$ is the sum of the running time of $A(n)$, which is polynomial in n, and the running time of $M(x)$, which is also polynomial in n, as so is the length of x. Thus N works in time $q(n)$ for all inputs $\hat{n}r_{x,n}$, where q is a polynomial. This polynomial should be used in (7). Then for every x in the support of D_n machine N accepts $\hat{n}r_{x,n}$ iff M accepts x. Therefore the reduction (7) satisfies the Correctness property. The domination condition holds by the choice of $r_{x,n}$. □

5 Conclusion

We have observed that the notion of a polynomial time computable ensemble of distributions is probably not encoding invariant. We have suggested a relaxation of PComp, called PISamp, so that PISamp is encoding invariant. We have provided an evidence that PISamp might be strictly larger than PComp. The notion of (NP, PISamp) completeness may be used instead of that of (NP, PComp) completeness in proofs of (NP,PSamp) completeness, which makes those proofs a little easier.

6 Acknowledgements

The author is grateful to Alexander Shen for bringing his attention to the fact that PComp might be not encoding invariant. The author is grateful to Dmitry Itsykson and to anonymous referees for a number of helpful suggestions.

References

[1] Andrej Bogdanov and Luca Trevisan, Average-Case Complexity, Foundations and Trends in Theoretical Computer Science 1(2), 2006: 1–106.

[2] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby, "On the Theory of Average Case Complexity", Journal of Computer and System Sciences, Vol 44, No. 2, 1992, 193–219.

[3] Johan Håstad, Russell Impagliazzo, Leonid A. Levin and Michael Luby, "A Pseudorandom Generator from any One-way Function", SIAM Journal on Computing, 28 (1999) 12–24.

[4] Russell Impagliazzo, Leonid A. Levin, No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. FOCS 1990: 812–821

[5] Leonid A. Levin, Average Case Complete Problems. SIAM J. Comput. 15(1): 285–286 (1986)