


Errorless Versus Error-Prone Average-Case Complexity

Shuichi Hirahara 

Principle of Informatics Research Division, National Institute of Informatics, Tokyo, Japan

Rahul Santhanam 

Department of Computer Science, University of Oxford, UK

Abstract

We consider the question of whether errorless and error-prone notions of average-case hardness are equivalent, and make several contributions.

First, we study this question in the context of hardness for NP, and connect it to the long-standing open question of whether there are instance checkers for NP. We show that there is an efficient non-uniform non-adaptive reduction from errorless to error-prone heuristics for NP if and only if there is an efficient non-uniform average-case non-adaptive instance-checker for NP. We also suggest an approach to proving equivalence of the two notions of average-case hardness for PH.

Second, we show unconditionally that error-prone average-case hardness is equivalent to errorless average-case hardness for P against NC^1 and for $UP \cap coUP$ against P.

Third, we apply our results about errorless and error-prone average-case hardness to get new equivalences between hitting set generators and pseudo-random generators.

2012 ACM Subject Classification Theory of computation \rightarrow Complexity classes; Theory of computation \rightarrow Interactive proof systems; Theory of computation \rightarrow Pseudorandomness and derandomization

Keywords and phrases average-case complexity, instance checker, pseudorandomness

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.84

Funding *Shuichi Hirahara*: Supported by JST, PRESTO Grant Number JPMJPR2024, Japan.

Rahul Santhanam: Supported by ESPRC New Horizons Grant No. EP/V048201/1 on “Structure vs Randomness in Algorithms and Computation”, UK.

Acknowledgements We thank anonymous reviewers for many insightful and inspiring comments.

1 Introduction

The theory of average-case complexity [22, 8] studies the complexity of distributional problems, i.e., pairs (L, D) where L is a language and D is a distribution over inputs. The key notion is what it means for L to be *easy on average* over D . Intuitively, we would like this to mean that there is an efficient algorithm A that solves L correctly with high probability over D . But there are at least two natural ways of capturing this, corresponding to whether A is *errorless* or *error-prone*. An errorless average-case algorithm solves L correctly with high probability over D , and *efficiently recognizes* when it is unable to solve L correctly on some input x , by outputting the “don’t know” character \perp on such an x . An error-prone¹ average-case algorithm solves L correctly with high probability over D , but may output the wrong answer on x for which it fails to solve L .

A fundamental question in average-case complexity is: how does error-prone average-case complexity relate to errorless average-case complexity? An efficient errorless average-case algorithm trivially implies an efficient error-prone average-case algorithm, but are there

¹ This notion is called “heuristic” in [8] and other works, but we use the term “error-prone” [19] to emphasize the distinction from the errorless notion.



© Shuichi Hirahara and Rahul Santhanam;

licensed under Creative Commons License CC-BY 4.0

13th Innovations in Theoretical Computer Science Conference (ITCS 2022).

Editor: Mark Braverman; Article No. 84; pp. 84:1–84:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

natural situations where the reverse implication holds as well? Indeed, Impagliazzo [19] poses this as an open question for NP in his early survey on average-case complexity.

This question has several motivations. An important line of research in complexity theory [3, 11] studies the relationship between worst-case complexity and average-case complexity, motivated by applications in cryptography [1] and derandomization [4, 28, 32]. It is known that “large” enough complexity classes, such as PSPACE and EXP have worst-case to average-case reductions, where the reduction works even if the average-case algorithm is error-prone. However, this is not known for classes such as P and NP, despite much effort. Indeed, for NP, there are negative results ruling out black-box reductions under standard complexity assumptions [11, 9]. Given the difficulty of proving worst-case to average-case reductions, it makes sense to ask if one can at least achieve the weaker goal of showing an error-prone to errorless average-case reduction for these classes. We show in this paper that this can indeed be achieved unconditionally for P, while for NP it relates closely to another long-standing open question in complexity theory – whether NP is *instance-checkable* [6].

A second motivation comes from recent work in the theory of *meta-complexity*. Meta-complexity studies the computational complexity of problems that are themselves about complexity, such as the Minimum Circuit Size Problem (MCSP) and the problem MINKT of computing polynomial time-bounded Kolmogorov complexity. Building on work of [10], Hirahara showed worst-case to average-case reductions for MCSP and MINKT [15]. However, unlike in the case of the worst-case to average-case reductions for PSPACE and EXP mentioned above, this reduction works only for *errorless* average-case complexity. More recently, [24] showed an equivalence between the existence of one-way functions and the error-prone average-case hardness of MINKT. Thus the only obstacle to basing the existence of one-way functions on the *worst-case* hardness of MINKT is the gap between errorless and error-prone average-case complexity. This gap becomes even more significant in the context of computing Levin’s notion Kt of Kolmogorov complexity – while the errorless average-case hardness of this problem is equivalent to separating EXP and BPP, the error-prone average-case hardness is equivalent to the existence of one-way functions [25, 29]!

A third motivation comes from pseudorandomness: trying to understand the relationship between *hitting set generators* and *pseudorandom generators*. In the context of complexity-theoretic generators where exponential time in the seed length is allowed to compute the generator, it is known that these two kinds of generators are equivalent [28, 2, 20], and the equivalence goes through a connection with average-case circuit lower bounds for EXP, which are known to be equivalent to worst-case circuit lower bounds for EXP. However, this equivalence is unclear for generators that are *polynomial-time computable*, as we do not know worst-case to average-case reductions for polynomial time.² Obtaining a reduction from errorless to error-prone average-case complexity is closely related to this question, since hitting set generators yield errorless average-case hardness in a natural way, and similarly pseudorandom generators yield error-prone average-case hardness in a natural way.

Obtaining an equivalence between hitting set generators and pseudorandom generators would be particularly interesting in a cryptographic setting where the adversary is stronger than the generator. As shown in [30], this would yield a remarkable set of equivalences between one-way functions, pseudorandomness, the non-existence of natural proofs and

² There are proof techniques that enable derandomizing two-sided-error randomized algorithms using a hitting set generator [12]; however, it is unknown if a hitting set generator can be converted into a pseudorandom generator using similar proof techniques.

hardness of learning. However, before this paper, such an equivalence was not known even in a complexity-theoretic setting where the generator is computable in polynomial time, and the adversary cannot simulate the generator.

In this paper, we initiate the study of relating error-prone and errorless average-case complexity, and prove several results, which we describe in more detail next.

1.1 Our Results

First we study the relationship between errorless and error-prone average-case complexity for NP. We connect the question of whether the two notions are equivalent for NP to another long-standing open question in complexity theory: is there an instance checker for NP-complete problems? This question was already raised in the seminal work of Blum and Kannan [6], which introduced the notion of instance checkability. Using the connection between interactive proof systems and instance checkers, this question is essentially equivalent to whether coNP can be proved by multi-prover interactive systems with honest provers being NP. The best upper bound is $P^{\#P}$, which follows from [26]. Improving this upper bound is a well-known open question, which is mentioned in, e.g., [3].

We show that these questions are almost equivalent to each other. Specifically, we consider a weaker notion of AvgBPP/poly-instance checker, which is an average-case (and non-uniform) version of an instance checker; see Definition 7 for the precise definition. We show that having an errorless-to-error-prone nonadaptive reduction is equivalent to the existence of a nonadaptive AvgBPP/poly-instance checker for NP.

► **Theorem 14.** *For every NP-complete language L , the following are equivalent:*

1. *For every $\mathcal{D} \in \text{PSamp/poly}$, there exists a nonadaptive AvgBPP/poly-instance checker for (L, \mathcal{D}) .*
2. *For every $\mathcal{D} \in \text{PSamp/poly}$, there exists an errorless-to-error-prone nonadaptive reduction computable in BPP/poly from (L, \mathcal{D}) to (L', \mathcal{D}') for some $L' \in \text{NP}$ and for some $\mathcal{D}' \in \text{PSamp/poly}$.*

Here, PSamp/poly denotes the class of distributions that can be sampled by randomized non-uniform algorithms. We consider non-uniform algorithms for a technical reason: The implication from Item 1 to 2 can be proved for uniform algorithms; however, to prove the implication from Item 2 to 1, we use a non-uniform tester of Mahmoody and Xiao [27], which takes as non-uniform advice the probability that yes instances are sampled from a distribution.

Theorem 14 suggests a new approach to the question of errorless vs. error-prone average-case hardness and enables us to employ the ideas developed in the rich literature on instance checking (e.g., [6, 7, 4, 33, 13, 27]). Although it is a long-standing open question to construct an instance checker for NP-complete problems, it may be feasible to construct an average-case-polynomial-time instance checker, which is an easier question whose importance is indicated by Theorem 14. In fact, we prove that every NP-complete problem admits a nonadaptive HeurBPP/poly-instance checker, which is an error-prone variant of an AvgBPP/poly-instance checker; see Theorem 17.

While we defer the formal definition of an errorless-to-error-prone reduction to Definition 4, it can be understood as follows:

► **Proposition 6.** *For distributional problems (L, \mathcal{D}) and (L', \mathcal{D}') , the following are equivalent:*

1. *There exists an errorless-to-error-prone BPP-reduction from (L, \mathcal{D}) to (L', \mathcal{D}') .*
2. *$(L', \mathcal{D}') \in \text{HeurBPP}^R$ implies $(L, \mathcal{D}) \in \text{AvgBPP}^R$ for every oracle R .*

Here, AvgBPP denotes the class of distributional problems solvable by errorless heuristic schemes; HeurBPP denotes the class of distributional problems solvable by error-prone heuristic schemes. A notion closely related to errorless-to-error-prone reduction is *worst-case-to-average-case* reduction. The former notion is weaker than the latter. Bogdanov and Trevisan [9] showed that if there exists a worst-case-to-average-case nonadaptive reduction from L to DistNP, then L is in $\text{NP/poly} \cap \text{coNP/poly}$; the implication of this result to errorless-to-error-prone reductions is left unexplored.

The implication from Item 1 to Item 2 in Theorem 14 is in fact generic and can be applied to an arbitrary language: any *nonadaptive* instance checker can be transformed into an errorless-to-error-prone reductions; see Theorem 8. Similarly, we show that interactive proof systems for PH, which make *adaptive* queries to provers, can be used to show the equivalence between errorless and error-prone average-case complexities of PH.

► **Theorem 21.** *If, for every language $L \in \text{PH}$, there exists a language $H \in \text{PH}$ such that L admits a PCP system with the honest oracle H , then $\text{DistPH} \subseteq \text{HeurBPP}$ implies $\text{DistPH} \subseteq \text{AvgBPP}$.*

Next, we turn our attention to errorless to error-prone reductions for P. We show *unconditional* errorless-to-error-prone reductions for P against NC^1 and for $\text{UP} \cap \text{coUP}$ against P. Below, we use DistP to refer to a pair consisting of a language $L \in \text{P}$ and an NC^1 -samplable distribution μ^3 , and $\text{Dist}(\text{UP} \cap \text{coUP})$ to refer to a pair consisting of a language $L \in \text{UP} \cap \text{coUP}$ and a P-samplable distribution μ .

► **Theorem 22.** *If $\text{DistP} \not\subseteq \text{AvgNC}^1$, then $\text{DistP} \not\subseteq \text{HeurNC}^1$.*

► **Theorem 25.** *If $\text{Dist}(\text{UP} \cap \text{coUP}) \not\subseteq \text{AvgP}$, then $\text{Dist}(\text{UP} \cap \text{coUP}) \not\subseteq \text{HeurP}$.*

Finally, we apply our unconditional errorless-to-error-prone reduction for P to obtain a new equivalence between hitting set generators and pseudorandom generators. Specifically, we consider *seed-extending* generators [21], which are generators whose output extends the seed. The well-known Nisan–Wigderson generator [28] is seed-extending, as is the direct product generator used in [16] to show the unexpected power of the Kolmogorov random strings. Note that seed-extending generators only make sense in a setting where the algorithm computing the generator is more powerful than the candidate distinguisher. If the candidate distinguisher can compute the generator, it can check whether its input is an output of the generator, and thereby break the generator.

► **Theorem 28.** *The following are equivalent:*

1. *There is a polynomial-time computable seed-extending hitting set generator with seed length $n - 1$ and error $o(1)$ against NC^1/poly .*
2. *For each $\delta > 0$, there is a polynomial-time computable seed-extending hitting set generator with seed length n^δ and error $1/n$ against NC^1/poly .*
3. *For each $\delta > 0$, there is a polynomial-time computable seed-extending pseudorandom generator with seed length n^δ and error $1/n$ against NC^1/poly .*

One corollary of the theorem above is that the stretch of polynomial-time computable seed-extending hitting set generators can be increased from 1 to any polynomial. Intriguingly, the only way we know how to prove this is using our connection between errorless and error-prone average case complexity for P.

³ When considering average-case complexity of problems in P, it makes sense to consider distributions sampled in a weaker class such as NC^1 [5].

1.2 Proof Techniques

First, we present an overview of the connection between instance checkers and errorless-to-error-prone reductions.

We start with recalling the definition of instance checker. A randomized oracle algorithm C is said to be an *instance checker* for L [6] if for every $x \in \{0, 1\}^*$,

1. $\Pr_C [C^L(x) \neq L(x)] \leq \frac{1}{4}$ and
2. For every oracle A , $\Pr_C [C^A(x) \notin \{L(x), \perp\}] \leq \frac{1}{4}$,

where the probabilities are taken over a coin flip sequence of C . We introduce an average-case-polynomial-time analogue of an instance checker: We say that an algorithm C is an *AvgBPP-instance checker* for a distributional problem (L, \mathcal{D}) if

Completeness $\Pr_{x \sim \mathcal{D}_n, C} [C^L(x; n, \delta) \neq L(x)] \leq \delta$ and

Soundness $\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta$ for every $n \in \mathbb{N}$, every $x \in \text{supp}(\mathcal{D})$, and every oracle A .

Next, we define an errorless-to-error-prone reduction: An algorithm M is said to be an *errorless-to-error-prone reduction from (L, \mathcal{D}) to (L', \mathcal{D}')* if for every oracle A that is “close”⁴ to L' with respect to \mathcal{D}' ,

Completeness $\Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta) \neq L(x)] \leq \delta$ and

Soundness $\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta$ for every $n \in \mathbb{N}$ and every $x \in \text{supp}(\mathcal{D})$.

Comparing these two definitions, we see that both completeness and soundness conditions are different. The completeness condition of an instance checker is weaker than that of an errorless-to-error-prone reduction. The soundness condition of an errorless-to-error-prone reduction is weaker than that of an instance checker. To show the equivalence, we need to close these gaps.

To convert an instance checker into an errorless-to-error-prone reduction, we need to strengthen the completeness condition: An instance checker works correctly only if a given oracle A is *exactly* equal to L , whereas we need the completeness property even if A is just *close* to L' . To close this gap, we use the fact that on any input x , the instance checker makes only polynomially many queries once its randomness is fixed. It suffices to ensure that all of these queries are answered correctly with high probability. This can be done by simply making sure that a given oracle A is sufficiently close to L' so that we can use a union bound to argue that all queries are answered correctly with high probability.

To convert an errorless-to-error-prone reduction into an instance checker, we need to strengthen the soundness property. An instance checker cannot output a wrong answer for *every* oracle A , whereas an errorless-to-error-prone reduction has the soundness property only when A is close to L' . While it seems to be difficult to close this gap in general, we observe that the existence of a *tester* enables us to bridge the gap. A tester for L' [6, 27] is an oracle algorithm that, given an oracle A , tests whether A is equal to L' or A is far from L' . Using a tester T for L' , we can guarantee that an oracle A is close to L' . Moreover, using the proof techniques of Feigenbaum and Fortnow [11], Mahmoody and Xiao [27] showed that every distributional NP search problem admits a non-uniform tester. Combining the non-uniform tester with an errorless-to-error-prone reduction, we can convert it to an instance checker. Details can be found in Section 3.

Next we explain the ideas behind the unconditional errorless-to-error-prone reductions for P against NC¹ and UP \cap coUP against P. In both cases, we use the same template, though the details are different. Given a language L that is errorless hard on average, we

⁴ More formally, we require that $\Pr_{y \sim \mathcal{D}'_m} [A(y; m, \epsilon) = L'(y)] \geq 1 - \epsilon$ for every $(m, \epsilon^{-1}) \in \mathbb{N}^2$. See Definition 4 for the precise definition.

first define a total search problem \mathcal{S}_L so that \mathcal{S}_L is errorless hard on average⁵. Then we use the totality of the search problem and the efficient verifiability of its solutions to argue that if \mathcal{S}_L is errorless hard on average, then \mathcal{S}_L is error-prone hard on average. Finally, we use a non-adaptive search-to-decision reduction to define an L' that is error-prone hard on average if \mathcal{S}_L is. The reason this approach fails when we consider NP against P is that it is unclear how to define an intermediate total search problem that preserves errorless hardness. The fact that P and $\text{UP} \cap \text{coUP}$ are closed under complement makes it easier for us to define such an intermediate total search problem.

Our proof template for the unconditional results can be interpreted as exploiting checkability properties that hold unconditionally for P and $\text{UP} \cap \text{coUP}$, but we find it more natural to present direct proofs rather than to use the lens of checkability.

Finally, we discuss the ideas behind our equivalence between seed-extending hitting set generators and seed-extending pseudorandom generators. Given a seed-extending hitting set generator computable in polynomial time (even a weak one that extends its seed by 1 bit suffices), we observe that there is a problem in P that is errorless average-case hard over the uniform distribution. Then we use our unconditional errorless to error-prone reduction for P to show the existence of a problem in P that is error-prone average-case hard over the uniform distribution. Here we exploit the fact that our reduction preserves the uniformity of the underlying distribution. Finally, we apply the Nisan–Wigderson generator [28] to derive a polynomial-time computable seed-extending pseudorandom generator with polynomial stretch.

2 Preliminaries

Notation

We often identify a language $L \subseteq \{0, 1\}^*$ with its characteristic function $L: \{0, 1\}^* \rightarrow \{0, 1\}$. $[n]$ denotes $\{1, 2, \dots, n\}$ for $n \in \mathbb{N}$. Let $\langle -, - \rangle: \mathbb{N}^2 \rightarrow \mathbb{N}$ be a bijection that is defined as, e.g., $\langle a, b \rangle = \sum_{i=0}^{a+b} i + a$. Similarly, let $\langle a, b, c \rangle := \langle \langle a, b \rangle, c \rangle$.

2.1 Average-Case Complexity

Here we review some standard notions of average-case complexity. We refer the readers to an excellent survey of Bogdanov and Trevisan [8] for a detailed exposition on the theory of average-case complexity.

► **Definition 1** (Polynomial-Time Samplable). *For an oracle A , we say that a family $\mathcal{D} = \{\mathcal{D}\}_{n \in \mathbb{N}}$ of distributions is polynomial-time samplable with oracle A if there exist an oracle polynomial-time algorithm M and a polynomial p such that, for every $n \in \mathbb{N}$ and every $x \in \{0, 1\}^*$,*

$$\Pr_{r \sim \{0,1\}^{p(n)}} [M^A(1^n, r) = x] = \mathcal{D}_n(x).$$

Let PSamp^A denote the class of polynomial-time samplable families of distributions with oracle A . We omit the superscript A if $A = \emptyset$. For a complexity class \mathfrak{C} , let $\text{PSamp}^{\mathfrak{C}}$ denote $\bigcup_{A \in \mathfrak{C}} \text{PSamp}^A$; let $\text{PSamp}/\text{poly} := \text{PSamp}^{\text{P}/\text{poly}}$. Let $\text{Dist}\mathfrak{C}$ denote $\mathfrak{C} \times \text{PSamp}$.

⁵ We need to define what it means for a search problem to be errorless hard on average, but this can be done in a natural way.

► **Definition 2** (Error-prone Heuristic Scheme [19, 8]). *An algorithm A is said to be an error-prone heuristic scheme for a distributional problem (L, \mathcal{D}) if for every $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$,*

$$\Pr_{x \sim \mathcal{D}_n, A} [A(x; 1^n, 1^{\delta^{-1}}) \neq L(x)] \leq \delta,$$

where the probability is taken over $x \sim \mathcal{D}_n$ as well as a coin flip sequence of A (if A is a randomized algorithm). For a class \mathcal{C} of algorithms, let $\text{Heur}\mathcal{C}$ denote the class of distributional problems that can be solved by an error-prone heuristic scheme from \mathcal{C} . For a function $\delta: \mathbb{N} \rightarrow [0, 1]$, let $\text{Heur}_\delta\mathcal{C}$ denote the class of distributional problems for which there exists an algorithm $A \in \mathcal{C}$ such that

$$\Pr_{x \sim \mathcal{D}_n, A} [A(x; 1^n) \neq L(x)] \leq \delta(n).$$

For notational simplicity, we often write $A(x; n, \delta)$ instead of $A(x; 1^n, 1^{\delta^{-1}})$.

The definition of randomized errorless heuristic scheme is given below.

► **Definition 3** (Randomized errorless heuristic scheme; AvgBPP [8]). *A randomized algorithm A is said to be a randomized errorless heuristic scheme for a distributional problem (L, \mathcal{D}) if the following hold for every $(n, \delta^{-1}) \in \mathbb{N}^2$:*

1. $\Pr_{x \sim \mathcal{D}_n, A} [A(x; n, \delta) \neq L(x)] \leq \delta$.
2. $\Pr_A [A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta$. for every $x \in \text{supp}(\mathcal{D}_n)$.
3. A halts in time $\text{poly}(n/\delta)$ on input $(x; n, \delta)$ for every $x \in \text{supp}(\mathcal{D}_n)$.

The class of distributional problems for which there exist randomized errorless heuristic schemes is denoted by AvgBPP.

It is known that AvgBPP can be characterized by the notion of *randomized average-case polynomial-time*; see [8] for details.

2.2 Nonadaptive Oracle Machine

A randomized polynomial-time *nonadaptive* oracle machine $M^{(\cdot)}$ is a polynomial-time oracle Turing machine M such that there exists a polynomial-time-computable function Q such that, for any input $x \in \{0, 1\}^*$, any coin flip sequence $r \in \{0, 1\}^*$ and every oracle O , any query made by M^O on input $(x; r)$ is in the list $Q(x; r)$ of strings. For a fixed input $x \in \{0, 1\}^*$ and $i \in \mathbb{N}$, let $Q(x; r)_i$ denote the i -th string in $Q(x; r)$. We may assume without loss of generality that the marginal distribution of $Q(x; r)_i$ is identical to that of $Q(x; r)_j$ for every $(i, j) \in \mathbb{N}^2$ over a random choice of r ; indeed, one can randomly permute the list $Q(x; r)$. The distribution of $Q(x; r)_1$ over a random choice of r is said to be the *query distribution* of M on input x . For a query distribution $Q(x)$ and a distribution \mathcal{D} , let $Q \circ \mathcal{D}$ denote the distribution from which a random sample q is generated by sampling $x \sim \mathcal{D}$ and $q \sim Q(x)$. For a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$, let $Q \circ \mathcal{D}$ denote $\{Q \circ \mathcal{D}_n\}_{n \in \mathbb{N}}$.

3 HeurBPP versus AvgBPP and Instance Checkers

In this section, we compare errorless and error-prone average-case complexities through the lens of average-polynomial-time instance checkers. In Section 3.1, we introduce the notion of errorless-to-error-prone reduction and show that this notion exactly characterizes the property that there is a connection from errorless to error-prone average-case complexities under any relativized world. In Section 3.2, we introduce the notion of average-polynomial-time instance

checker and show that it implies the existence of an errorless-to-error-prone reduction. The converse is given for NP in Section 3.3, where we also explore the connection between testers and HeurBPP-instance checkers. In Section 3.4, we construct an errorless-to-error-prone reduction for PH from (even adaptive) PCP systems whose honest oracles are in PH.

3.1 Errorless-To-Error-Prone Reductions

We introduce the notion of errorless-to-error-prone reduction.

► **Definition 4** (Errorless-To-Error-Prone Reduction). *An oracle algorithm M is said to be an errorless-to-error-prone reduction from (L, \mathcal{D}) to (L', \mathcal{D}') if for every oracle A such that for every $(m, \epsilon^{-1}) \in \mathbb{N}^2$,*

$$\Pr_{y \sim \mathcal{D}'_m} [A(y; m, \epsilon) = L'(y)] \geq 1 - \epsilon,$$

the following hold for every $(n, \delta^{-1}) \in \mathbb{N}^2$:

1. $\Pr_{x \sim \mathcal{D}_n, M} [M^A(x; n, \delta) \neq L(x)] \leq \delta$ and
2. $\Pr_M [M^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta$ for every $x \in \text{supp}(\mathcal{D}_n)$, where the probability is taken over a coin flip sequence of M .

We assume that the input $(x; n, \delta)$ is encoded as a binary string of length $\Theta(|x| + n + \delta^{-1})$. If the reduction M is computable in \mathfrak{C} , then we call it a \mathfrak{C} -reduction.

We observe that an errorless-to-error-prone reduction from (L, \mathcal{D}) to (L', \mathcal{D}') enables converting an error-prone heuristic scheme for (L', \mathcal{D}') into an errorless heuristic scheme for (L, \mathcal{D}) .

► **Proposition 5.** *Let $\mathfrak{C} \in \{\text{BPP}, \text{BPP/poly}\}$. If there exists an errorless-to-error-prone \mathfrak{C} -reduction from (L, \mathcal{D}) to $(L', \mathcal{D}') \in \text{Heur}\mathfrak{C}$, then $(L, \mathcal{D}) \in \text{Avg}\mathfrak{C}$.*

Proof. Let $M \in \mathfrak{C}$ be an errorless-to-error-prone reduction from (L, \mathcal{D}) to (L', \mathcal{D}') . Let $A \in \mathfrak{C}$ be an error-prone heuristic scheme for (L', \mathcal{D}') . Note that for every $(m, \epsilon^{-1}) \in \mathbb{N}^2$,

$$\Pr_{y \sim \mathcal{D}'_{m,r}} [A(y; m, \epsilon; r) \neq L'(y)] \leq \delta,$$

where $r \sim \{0, 1\}^{(m/\epsilon)^{O(1)}}$ denotes a coin flip sequence used by A . We define an algorithm B as follows: B takes $(x; n, \delta)$ as input and picks a coin flip sequence r of A . B simulates the reduction M on input $(x; n, \delta/2)$ by answering a query $(y; m, \epsilon)$ with $A(y; m, \epsilon^2/p(n/\delta); r)$, where $p(n/\delta)$ is a large polynomial chosen later.

We claim that the algorithm B is an errorless heuristic scheme for (L, \mathcal{D}) . Fix $(n, \delta^{-1}) \in \mathbb{N}^2$. For every coin flip sequence $r \in \{0, 1\}^*$, let A'_r denote an oracle such that $A'_r(y; m, \epsilon) := A(y; m, \epsilon^2/p(n/\delta); r)$ for every $(m, \epsilon^{-1}) \in \mathbb{N}^2$ and every $y \in \{0, 1\}^*$. By Markov's inequality, we obtain

$$\Pr_r \left[\Pr_{y \sim \mathcal{D}'_m} [A'_r(y; m, \epsilon) \neq L'(y)] \geq \epsilon \right] \leq \frac{1}{\epsilon} \cdot \Pr_{r,y} [A(y; m, \epsilon^2/p(n/\delta); r) \neq L'(y)] \leq \frac{\epsilon}{p(n/\delta)}.$$

The algorithm B fails if either

1. the oracle A'_r fails to satisfy $\Pr_{y \sim \mathcal{D}'_m} [A'_r(y; m, \epsilon) \neq L'(y)] \leq \epsilon$ for some $(m, \epsilon^{-1}) \in \mathbb{N}^2$ such that M queries $(y'; m, \epsilon^{-1})$ on input $(x; n, \delta^{-1})$ for some $y' \in \{0, 1\}^*$ and some $x \in \{0, 1\}^*$, or
2. the reduction M fails under the event that Item 1 does not happen.

By choosing the polynomial $p(n/\delta)$ large enough so that the number of queries made by M on input $(-; n, \delta^{-1})$ is at most $p(n/\delta)/2$, the first event happens with probability at most $\frac{\epsilon}{p(n/\delta)} \cdot \frac{p(n/\delta)}{2} = \frac{\delta}{2}$. The second event happens with probability at most $\frac{\delta}{2}$. Therefore, by a union bound, the failure probability of B is at most δ . \blacktriangleleft

In the case of uniform algorithms, it is possible to characterize the notion of errorless-to-error-prone reduction by the property of having a connection from HeurBPP to AvgBPP under any relativized world.

► **Proposition 6.** *For distributional problems (L, \mathcal{D}) and (L', \mathcal{D}') , the following are equivalent:*

1. *There exists an errorless-to-error-prone BPP-reduction from (L, \mathcal{D}) to (L', \mathcal{D}') .*
2. *$(L', \mathcal{D}') \in \text{HeurBPP}^R$ implies $(L, \mathcal{D}) \in \text{AvgBPP}^R$ for every oracle R .*

We mention in passing that similar characterizations can be given for most “black-box reductions” in other settings; see [14, 17] for similar results.

Proof. The implication from Item 1 to Item 2 follows from Proposition 5 by observing that the proof works under the presence of any oracle R .

We prove the contrapositive of Item 1 to Item 2. That is, assuming that there exists no errorless-to-error-prone BPP-reduction from (L, \mathcal{D}) to (L', \mathcal{D}') , we construct an oracle R such that $(L', \mathcal{D}') \in \text{HeurBPP}^R$ and $(L, \mathcal{D}) \notin \text{AvgBPP}^R$.

We enumerate all the randomized oracle polynomial-time Turing machines $\{M_e\}_{e \in \mathbb{N}}$. We construct an oracle $R_e: \{0, 1\}^* \rightarrow \{0, 1, \perp\}$ at stage $e \in \mathbb{N}$, where 0 and 1 indicate that the value is fixed and \perp indicates that the value is not yet defined. At stage $e \in \mathbb{N}$, we “extend” the oracle R_{e-1} . That is, $R_e^{-1}(\perp) \subseteq R_{e-1}^{-1}(\perp)$ and $R_{e-1}(x) = R_e(x)$ for every $x \in R_{e-1}^{-1}(\{0, 1\})$. Moreover, we assume that $R_e^{-1}(\perp)$ is finite. We define the oracle R such that for every $x \in \{0, 1\}^*$, $R(x) := R_{e(x)}(x)$, where $e(x)$ is the first stage $e(x)$ such that $R_{e(x)}(x) \neq \perp$. We also construct a finite set $P_e \subseteq \{(m, \epsilon) \mid m \in \mathbb{N}, \epsilon^{-1} \in \mathbb{N}\}$.

At each stage $e \in \mathbb{N}$, we construct (R_e, P_e) so that for any $(m, \epsilon) \in P_e$,

$$\Pr_{x \sim \mathcal{D}'_n} [L'(x) \neq R_e(x; m, \epsilon)] \leq \epsilon \tag{1}$$

and $R_e(x; m, \epsilon) \neq \perp$ for every $x \in \text{supp}(\mathcal{D}'_n)$. Since R extends R_e , this implies that

$$\Pr_{x \sim \mathcal{D}'_n} [L'(x) \neq R(x; m, \epsilon)] \leq \epsilon,$$

from which it is evident that $(L', \mathcal{D}') \in \text{HeurBPP}^R$.

We start with $R_0 \equiv \perp$ and $P_e = \emptyset$. At stage $e \geq 1$, we consider the randomized oracle Turing machine M_e . Fix n and $\delta^{-1} \in \mathbb{N}$ and an oracle O . We define an oracle \tilde{O} as follows: For every $q \in \{0, 1\}^*$, $\tilde{O}(q) := O(q)$ if $R_{e-1}(q) = \perp$; otherwise, $\tilde{O}(q) := R_{e-1}(q)$. Then consider an oracle algorithm $C^O(x; n, \delta) := M_e^{\tilde{O}}(x; n, \delta)$. To implement C^O in polynomial time, we hard-wire the finite set of $(q, R_{e-1}(q))$ for every $q \in R_{e-1}^{-1}(\{0, 1\})$. Since C^O is a polynomial-time algorithm, by assumption, C^O cannot be an errorless-to-error-prone BPP-reduction from (L, \mathcal{D}) to (L', \mathcal{D}') . Therefore, there exists an oracle A such that for every $(m, \epsilon^{-1}) \in \mathbb{N}^2$,

$$\Pr_{y \sim \mathcal{D}'_m} [A(y; m, \epsilon) \neq L'(y)] \leq \epsilon,$$

and for some $(n, \delta^{-1}) \in \mathbb{N}^2$, either

1. $\Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta) \neq L(x)] > \delta$, or

84:10 Errorless Versus Error-Prone Average-Case Complexity

2. there exists an input $x \in \text{supp}(\mathcal{D}_n)$ such that

$$\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] > \delta.$$

We argue that, in both cases, the oracle R_{e-1} can be extended to R_e so that Equation (1) is satisfied and M_e does not witness $(L, \mathcal{D}) \in \text{AvgBPP}^R$.

Consider the first case. In this case, we have

$$\Pr_{x \sim \mathcal{D}_n, M_e} [M_e^{\tilde{A}}(x; n, \delta) \neq L(x)] > \delta,$$

where $\tilde{A}(q) = A(q)$ if $R_{e-1}(q) = \perp$; otherwise, $\tilde{A}(q) = R_{e-1}(q)$. Let P denote the set of (m, ϵ) such that $(y; m, \epsilon)$ is queried by $M_e^{\tilde{A}}(x; n, \delta)$ for some $x \in \text{supp}(\mathcal{D}_n)$ and some $y \in \{0, 1\}^*$. Let $P_e := P_{e-1} \cup P$. We define R_e so that $R_e(y; m, \epsilon) := \tilde{A}(y; m, \epsilon)$ for every $y \in \text{supp}(\mathcal{D}'_m)$ and every $(m, \epsilon) \in P_e$ and other values are left undefined. Observe that Equation (1) is satisfied for R_e . Moreover, we have

$$\Pr_{x \sim \mathcal{D}_n, M_e} [M_e^{R_e}(x; n, \delta) \neq L(x)] > \delta,$$

which implies that M_e is not an AvgBPP^R algorithm for (L, \mathcal{D}) .

Consider the second case. In this case, there exists an input $x \in \text{supp}(\mathcal{D}_n)$ such that

$$\Pr_{M_e} [M_e^{\tilde{A}}(x; n, \delta) \notin \{L(x), \perp\}] > \delta.$$

Constructing an oracle R_e and P_e in a way similar to the first case, we can satisfy Equation (1) and have

$$\Pr_{M_e} [M_e^{R_e}(x; n, \delta) \notin \{L(x), \perp\}] > \delta,$$

which implies that M_e is not an AvgBPP^R algorithm for (L, \mathcal{D}) . ◀

3.2 Average-Polynomial-Time Instance Checker

We extend the standard definition of an instance checker [6] to an average-case-polynomial-time version.

► **Definition 7.** For a class \mathfrak{C} of oracle algorithms, a randomized oracle algorithm $C \in \mathfrak{C}$ is said to be an $\text{Avg}\mathfrak{C}$ -instance checker for a distributional problem (L, \mathcal{D}) if the following hold for every $(n, \delta^{-1}) \in \mathbb{N}^2$:

1. $\Pr_{x \sim \mathcal{D}_n, C} [C^L(x; n, \delta) \neq L(x)] \leq \delta$.
 2. $\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta$ for every $x \in \text{supp}(\mathcal{D}_n)$ and every oracle $A \subseteq \{0, 1\}^*$.
 3. C^A halts in time $\text{poly}(n/\delta)$ on input $(x; n, \delta)$ for every oracle A and every $x \in \text{supp}(\mathcal{D}_n)$.
- If Item 2 is weakened so that for every $n, \delta^{-1} \in \mathbb{N}$ and every oracle A ,

$$\Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta,$$

then C is said to be a $\text{Heur}\mathfrak{C}$ -instance checker.

We construct an errorless-to-error-prone reduction from an instance checker.

► **Theorem 8.** Let $(\mathfrak{C}, \mathfrak{D}) \in \{(\text{BPP}, \text{PSamp}), (\text{BPP}/\text{poly}, \text{PSamp}/\text{poly})\}$. If there exists a nonadaptive $\text{Avg}\mathfrak{C}$ -instance checker for (L, \mathcal{D}) , then there exists some distribution $\mathcal{D}' \in \mathfrak{D}$ such that there exists an errorless-to-error-prone nonadaptive \mathfrak{C} -reduction from (L, \mathcal{D}) to (L, \mathcal{D}') .

Proof. For simplicity, we present a proof only for $(\mathfrak{C}, \mathfrak{D}) = (\text{BPP}, \text{PSamp})$; a proof for $(\text{BPP/poly}, \text{PSamp/poly})$ is similar. Let C be a nonadaptive AvgBPP-instance checker for (L, \mathcal{D}) . Let Q be the query distribution of C . Let $\mathcal{D}' = \left\{ \mathcal{D}'_{\langle n, \delta^{-1} \rangle} \right\}_{\langle n, \delta^{-1} \rangle \in \mathbb{N}}$ be a family of distributions such that $\mathcal{D}'_{\langle n, \delta^{-1} \rangle}$ is the distribution of $Q(x; n, \delta/2)$ for $x \sim \mathcal{D}_n$. We define an oracle algorithm M^A so that for every oracle A , every $(n, \delta^{-1}) \in \mathbb{N}^2$ and every $x \in \{0, 1\}^*$,

$$M^A(x; n, \delta) := C^{A(\cdot; \langle n, \delta^{-1} \rangle, \delta')}(x; n, \delta/2),$$

where $\delta' := \frac{\delta}{2p(n/\delta)}$ and $p(n/\delta)$ is a polynomial that upper-bounds the number of queries made by C on input $(x; n, \delta^{-1})$ for any $x \in \text{supp}(\mathcal{D}_n)$.

We claim that $M^{(\cdot)}$ is an errorless-to-error-prone nonadaptive reduction from (L, \mathcal{D}) to (L, \mathcal{D}') . We first verify the soundness condition. For every oracle A , every $(n, \delta) \in \mathbb{N}^2$, and every $x \in \text{supp}(\mathcal{D}_n)$, we have

$$\Pr_M [M^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \Pr_{C, A} [C^{A(\cdot; \delta')}(x; n, \delta/2) \notin \{L(x), \perp\}] \leq \frac{\delta}{2},$$

where the last inequality follows from the soundness condition of the AvgBPP-instance checker C . This implies that $M^{(\cdot)}$ satisfies the soundness condition.

It remains to verify the completeness condition of $M^{(\cdot)}$. Since $M^A(x; n, \delta) \neq L(x)$ holds only if either A fails to solve L on the i -th query $Q(x; n, \delta/2)_i$ of C on input $(x; n, \delta/2)$ for some $i \in [p(n/\delta)]$, or C^L fails to solve L on input $(x; n, \delta/2)$, we have

$$\begin{aligned} & \Pr_{x \sim \mathcal{D}_n, M} [M^A(x; n, \delta) \neq L(x)] \\ & \leq \sum_{i=1}^{p(n/\delta)} \Pr_{x \sim \mathcal{D}_n} [A(Q(x; n, \delta/2)_i; \langle n, \delta^{-1} \rangle, \delta') \neq L(Q(x; n, \delta/2)_i)] + \Pr_C [C^L(x; n, \delta/2) \neq L(x)] \\ & \leq p(n/\delta) \cdot \Pr_{q \sim \mathcal{D}'_{\langle n, \delta^{-1} \rangle}} [A(q; \langle n, \delta^{-1} \rangle, \delta') \neq L(q)] + \delta/2 \\ & \leq p(n/\delta) \cdot \frac{\delta}{2p(n/\delta)} + \frac{\delta}{2} = \delta. \end{aligned} \quad \blacktriangleleft$$

3.3 Testers and HeurBPP-Instance Checkers

A *tester* for L is an algorithm that tests whether a given oracle is equal to L or far from L .

► **Definition 9** (Tester [6, 27]). *A randomized oracle algorithm T is said to be a tester for a distributional problem (L, \mathcal{D}) if for every $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$,*

1. $\Pr_T [T^L(n, \delta) = 1] \geq \frac{3}{4}$ and
2. *for every oracle A such that $\Pr_{x \sim \mathcal{D}_n} [A(x) \neq L(x)] > \delta$, it holds that $\Pr_T [T^A(n, \delta) = 0] \geq \frac{3}{4}$.*

We assume that the input (n, δ) is encoded as $(1^n, 1^{\delta^{-1}})$, which is of length $\Theta(n/\delta)$.

► **Remark 10.** By running a tester $\text{poly}(n/\delta)$ times independently, the constant $\frac{3}{4}$ of Definition 9 can be amplified to $1 - 2^{-p(n/\delta)}$ for any polynomial p .

We extend the definition of the tester for a language to a tester for an NP search problem.

► **Definition 11.** *A relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ is said to be an NP relation if R is decidable in polynomial time and $|y| \leq \text{poly}(|x|)$ for any $(x, y) \in R$. An NP search problem R is the task of finding y such that $(x, y) \in R$ on input x .*

84:12 Errorless Versus Error-Prone Average-Case Complexity

For a relation R , let $\mathcal{L}(R) := \{x \in \{0, 1\}^* \mid \exists y, R(x, y) = 1\}$. For a distribution \mathcal{D} , we say that a randomized oracle algorithm T is a tester for the distributional NP search problem (R, \mathcal{D}) if for every $(n, \delta^{-1}) \in \mathbb{N}^2$ and for every oracle $A: \{0, 1\}^* \rightarrow \{0, 1\}^*$,

1. $\Pr_T [T^A(n, \delta) = 1] \geq \frac{3}{4}$ if $R(x, A(x)) = \mathcal{L}(R)(x)$ for every $x \in \text{supp}(\mathcal{D}_n)$ and
2. $\Pr_T [T^A(n, \delta) = 0] \geq \frac{3}{4}$ if $\Pr_{x \sim \mathcal{D}_n} [R(x, A(x)) \neq \mathcal{L}(R)(x)] > \delta$.

► **Lemma 12** (Mahmoody and Xiao [27]). *For every NP search problem R and every $\mathcal{D} \in \text{PSamp/poly}$, there exists a nonadaptive BPP/poly-tester for (R, \mathcal{D}) .*

► **Theorem 13.** *Let L be an NP-complete language and $\mathcal{D} \in \text{PSamp/poly}$. If there exists an errorless-to-error-prone nonadaptive BPP/poly-reduction from (L, \mathcal{D}) to (L', \mathcal{D}') for some $L' \in \text{NP}$ and $\mathcal{D}' \in \text{PSamp/poly}$, then there exists a nonadaptive AvgBPP/poly-instance checker for (L, \mathcal{D})*

Proof. Let R be an NP search problem associated with L' , i.e., $\mathcal{L}(R) = L'$. By the search-to-decision reduction of [34, 5], there exists a randomized polynomial-time nonadaptive reduction from R to some language $L_1 \in \text{NP}$. Since L is NP-complete, L_1 is reducible to L . Combining these reductions, we obtain a randomized nonadaptive reduction B from the NP search problem R to L . Using Lemma 12, let T be a BPP/poly-tester for (R, \mathcal{D}') .

Let M be the errorless-to-error-prone nonadaptive reduction from (L, \mathcal{D}) to (L', \mathcal{D}') . Since R is the NP search problem associated with L' , the decision version (L', \mathcal{D}') is trivially reducible to (R, \mathcal{D}') . Combining this reduction with M , we may assume that M is an errorless-to-error-prone reduction from (L, \mathcal{D}) to (R, \mathcal{D}') .

We are now ready to describe an instance checker C for (L, \mathcal{D}) . The algorithm C takes $(x; n, \delta)$ as input and oracle access to an oracle A . C picks a coin flip sequence r used by the reduction B . We may assume without loss of generality that the success probability of B is at least $1 - 2^{-2p(n/\delta)}$, where $p(n/\delta)$ is a polynomial upper bound on the running time of M on input $(-; n, \delta)$. Let O be an oracle such that $O(y; m, \epsilon) := B^A(y; r)$ for every $y \in \{0, 1\}^*$, every $m \in \mathbb{N}$, and every $\epsilon^{-1} \in \mathbb{N}$, where $B^A(y; r)$ denotes the output of B on input y given a coin flip sequence r and oracle access to A . C simulates $M^O(x; n, \delta)$ and lets P be the set of (m, ϵ) such that $M^O(x; n, \delta)$ makes a query $(y; m, \epsilon)$ for some $y \in \{0, 1\}^*$. C runs the tester $T^O(m, \epsilon)$ for every $(m, \epsilon) \in P$. If $T^O(m, \epsilon) = 0$ for some $(m, \epsilon) \in P$, then C outputs \perp and halts. Otherwise, C outputs the output of $M^O(x; n, \delta)$.

We first verify the completeness of C . Let $A := L$. Fix integers $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$. Since B is a reduction from R to L , for every $q \in \{0, 1\}^*$, we have

$$\Pr_r [R(q, B^L(q; r)) = L'(q)] \geq 1 - 2^{-2p(n/\delta)}.$$

By a union bound for all $q \in \{0, 1\}^*$ such that $|q| \leq p(n/\delta)$, with probability at least $1 - 2^{-p(n/\delta)+1}$, $R(q, O(q; m, \epsilon)) = L'(q)$ for every $(q; m, \epsilon)$ queried by the reduction $M^O(x; n, \delta)$. Under this event, the oracle O solves the NP search problem R correctly; by the completeness of M , we have $\Pr_{x \sim \mathcal{D}_n, M} [M^O(x; n, \delta) \neq L(x)] \leq \delta$. Moreover, the completeness of the tester T implies that $T^O(m, \epsilon) = 1$ with high probability (say, with probability at least $1 - \delta$) for every $(m, \epsilon) \in P$. Overall, we obtain

$$\Pr_{x \sim \mathcal{D}_n, C} [C^L(x; n, \delta) \neq L(x)] \leq \delta + \delta + 2^{-p(n/\delta)+1} \leq 3\delta.$$

Thus, C satisfies the completeness.

It remains to verify the soundness of C . Consider an arbitrary oracle A . We consider two cases.

1. When

$$\Pr_{q \sim \mathcal{D}'_n} [R(q, O(q; m, \epsilon)) = L'(q)] \leq \epsilon \quad (2)$$

holds for every $(m, \epsilon) \in P$, by the soundness property of M , we obtain

$$\Pr_M [M^O(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta.$$

Therefore,

$$\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \delta.$$

2. When Equation (2) does not hold for some $(m, \epsilon) \in P$, the tester rejects with high probability: $T^O(m, \epsilon) = 0$ with probability at least $1 - \delta$. Therefore,

$$\Pr_C [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \Pr_C [C^A(x; n, \delta) \neq \perp] \leq \delta. \quad \blacktriangleleft$$

► **Theorem 14.** *For every NP-complete language L , the following are equivalent:*

1. *For every $\mathcal{D} \in \text{PSamp/poly}$, there exists a nonadaptive AvgBPP/poly-instance checker for (L, \mathcal{D}) .*
2. *For every $\mathcal{D} \in \text{PSamp/poly}$, there exists an errorless-to-error-prone nonadaptive reduction computable in BPP/poly from (L, \mathcal{D}) to (L, \mathcal{D}') for some $\mathcal{D}' \in \text{PSamp/poly}$.*
3. *For every $\mathcal{D} \in \text{PSamp/poly}$, there exists an errorless-to-error-prone nonadaptive reduction computable in BPP/poly from (L, \mathcal{D}) to (L', \mathcal{D}') for some $L' \in \text{NP}$ and for some $\mathcal{D}' \in \text{PSamp/poly}$.*

Proof.

(Item 1 \Rightarrow 2) This follows from Theorem 8.

(Item 2 \Rightarrow 3) Obvious.

(Item 3 \Rightarrow 1) This follows from Theorem 13. ◀

We observe that a tester is equivalent to a HeurBPP-instance checker.

► **Proposition 15.** *Let L be a language and $\mathcal{D} \in \text{PSamp}$ be a distribution. Then, there exists a tester for (L, \mathcal{D}) if and only if there exists a HeurBPP-instance checker for (L, \mathcal{D}) .*

Proof. We first prove the “only if” part. Let T be a tester for (L, \mathcal{D}) . We may assume without loss of generality that the error probability of T is at most δ on input (n, δ) . Let C^A be an oracle algorithm that takes $(x; n, \delta)$ as input and outputs \perp if $T^A(n, \delta) = 0$ and $A(x)$ otherwise. We claim that C^A is a HeurBPP-checker for (L, \mathcal{D}) . To see the completeness,

$$\Pr_{x \sim \mathcal{D}_n, C} [C^L(x; n, \delta) = L(x)] \leq \Pr_T [T^L(n, \delta) = 1] \leq \delta.$$

To see the soundness, consider an arbitrary oracle A . If $\Pr_{x \sim \mathcal{D}_n} [A(x) \neq L(x)] > \delta$, then we have

$$\Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \Pr_T [T^A(n, \delta) = 1] \leq \delta.$$

If $\Pr_{x \sim \mathcal{D}_n} [A(x) \neq L(x)] \leq \delta$, then we have

$$\Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta) \notin \{L(x), \perp\}] \leq \Pr_x [A(x) \neq L(x)] \leq \delta.$$

In both cases, C^A satisfies the soundness condition of a HeurBPP-instance checker.

84:14 Errorless Versus Error-Prone Average-Case Complexity

We now prove the “if” part. Let C^A be a HeurBPP-instance checker for (L, \mathcal{D}) . Let X be the indicator random variable that takes 1 if and only if $C^A(x; n, \delta/4) \neq A(x)$ for a random $x \sim \mathcal{D}_n$. If $A = L$, then we have $\mathbb{E}[X] = \Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta/4) \neq L(x)] \leq \delta/4 \ll \delta/2$. If $\Pr_{x \sim \mathcal{D}_n} [A(x) \neq L(x)] > \delta$, then we have

$$\begin{aligned} \mathbb{E}[X] &= \Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta/4) \neq A(x)] \\ &\geq \Pr_{x \sim \mathcal{D}_n} [A(x) \neq L(x)] - \Pr_{x \sim \mathcal{D}_n, C} [C^A(x; n, \delta/4) \notin \{L(x), \perp\}] \\ &> \delta - \delta/4 = 3\delta/4 \gg \delta/2. \end{aligned}$$

These inequalities motivate us to define the following tester T : An oracle algorithm T^A takes (n, δ) as input, estimates $\mathbb{E}[X]$ by randomly sampling $x \sim \mathcal{D}$ and checking whether $C^A(x; n, \delta/4) \neq A(x)$ several times, and outputs 1 if and only if $\mathbb{E}[X]$ is significantly less than $\delta/2$. Using a standard concentration inequality, it is easy to verify that T^A is a tester for (L, \mathcal{D}) . ◀

Beigel’s theorem [6] states that an instance checker for L_1 can be converted into an instance checker for L_2 if L_1 and L_2 are equivalent to each other under polynomial-time reductions. We prove an analogous result for testers:

► **Lemma 16.** *Let L_1 and L_2 be languages such that L_1 is reducible to L_2 via a many-one reduction R_2 and L_2 is reducible to L_1 via a many-one reduction R_1 . Let $\mathcal{D} \in \text{PSamp}$ be a distribution. If there exists a tester for $(L_1, R_1 \circ \mathcal{D})$, then there exists a tester for (L_2, \mathcal{D}) .*

Proof. By Proposition 15, it suffices to prove the result for a HeurBPP-instance checker. Let C be a HeurBPP-instance checker for $(L_1, R_1 \circ \mathcal{D})$. Define an algorithm C' so that $C'^A(x; n, \delta) := C^{A \circ R_2}(R_1(x); n, \delta)$. We prove that C' is a HeurBPP-instance checker for (L_2, \mathcal{D}_2) . Fix $n \in \mathbb{N}$. To see the completeness, since $L_2(x) = L_1(R_1(x))$, we have

$$\begin{aligned} &\Pr_{x \sim \mathcal{D}_n, C'} [C'^{L_2}(x; n, \delta) = L_2(x)] \\ &= \Pr_{x \sim \mathcal{D}_n, C} [C^{L_2 \circ R_2}(R_1(x); n, \delta) = L_2(x)] \\ &= \Pr_{x \sim \mathcal{D}_n, C} [C^{L_1}(R_1(x); n, \delta) = L_1(R_1(x))] \\ &= \Pr_{x' \sim R_1 \circ \mathcal{D}_n, C} [C^{L_1}(x'; n, \delta) = L_1(x')] \geq 1 - \delta. \end{aligned}$$

To see the soundness, observe that for every oracle A ,

$$\begin{aligned} &\Pr_{x \sim \mathcal{D}_n, C'} [C'^A(x; n, \delta) \notin \{L_2(x), \perp\}] \\ &= \Pr_{x \sim \mathcal{D}_n, C} [C^{A \circ R_2}(R_1(x); n, \delta) \notin \{L_1(R_1(x)), \perp\}] \\ &= \Pr_{x' \sim R_1 \circ \mathcal{D}_n, C} [C^{A \circ R_2}(x'; n, \delta) \notin \{L_1(x'), \perp\}] \leq \delta. \end{aligned} \quad \blacktriangleleft$$

We show that $\{L\} \times \text{PSamp}$ admits nonadaptive HeurBPP/poly-instance checkers for every NP-complete language L .

► **Theorem 17.** *Let L be an NP-complete language and $\mathcal{D} \in \text{PSamp}$ be a distribution. Then, there exists a nonadaptive HeurBPP/poly-instance checker.*

Proof Sketch. Let L be an NP-complete language. The idea is to convert the BPP/poly-tester of Lemma 12 into a HeurBPP/poly-instance checker.

Let V be an NP search problem such that $\mathcal{L}(V) = L$. By the search-to-decision reduction of [34, 5], there exists a randomized polynomial-time nonadaptive reduction from V to L . Moreover, there is a trivial reduction from L to V (that reduces an instance to itself). By Lemma 12, there exists a nonadaptive BPP/poly-tester for (V, \mathcal{D}) . Using (the proof ideas of) Lemma 16, a BPP/poly-tester for (V, \mathcal{D}) can be converted into a tester for (L, \mathcal{D}) . (While Lemma 16 is stated only for decision problems and many-one reductions, it is not hard to generalize it to search problems and randomized nonadaptive reductions.) By Proposition 15, we obtain a HeurBPP/poly-instance checker for (L, \mathcal{D}) . It is not hard to observe that the property of being nonadaptive is preserved. ◀

3.4 Errorless to Error-Prone Reductions from PCP Systems With Honest Oracles in PH

In this subsection, we show that interactive proof systems for PH with PH honest oracles enable constructing errorless-to-error-prone reductions. In fact, we consider a notion of PCP systems, which is weaker than interactive proof systems, so that our results are stronger:

► **Definition 18.** A randomized polynomial-time oracle algorithm V is said to be a PCP system for a language L with an honest oracle $H \subseteq \{0, 1\}^*$ if for every $x \in \{0, 1\}^*$,

1. if $x \in L$, then $\Pr_V [V^H(x) = 1] \geq \frac{2}{3}$ and
2. if $x \notin L$, then $\Pr_V [V^O(x) = 1] \leq \frac{1}{3}$ for every oracle O .

For a language L , let \bar{L} denote the complement of L , i.e., $\{0, 1\}^* \setminus L$.

► **Proposition 19.** If L and \bar{L} have PCP systems with honest oracles H_0 and H_1 , respectively, then there exists a randomized polynomial-time algorithm V such that for every $x \in \{0, 1\}^*$ and every $\delta^{-1} \in \mathbb{N}$,

1. $\Pr [V^{H_0, H_1}(x; \delta) \neq L(x)] \leq \delta$ and
2. $\Pr [V^{O_0, O_1}(x; \delta) \notin \{L(x), \perp\}] \leq \delta$ for any pair (O_0, O_1) of oracles.

We say that an algorithm V is a checker for L by (H_0, H_1) .

Proof. Let V_0 and V_1 be PCP systems for L and \bar{L} , respectively. Since the success probability can be amplified by repetition, we may assume that the error probability of V_0 and V_1 is at most $\delta/2$. We define V to be an oracle algorithm such that V^{O_0, O_1} takes $(x; \delta)$ as input and outputs 1 if $V_0^{O_0}(x) = 1$ and $V_1^{O_1}(x) = 0$, outputs 0 if $V_0^{O_0}(x) = 0$ and $V_1^{O_1}(x) = 1$, and outputs \perp otherwise.

We prove the correctness of V : It is easy to observe that $V^{H_0, H_1}(x; \delta)$ outputs $L(x)$ with probability at least $1 - \delta$. For example, if $x \in L$, then $V_0^{H_0}(x) = 1$ with probability at least $1 - \delta/2$ and $V_1^{H_1}(x) = 0$ with probability at least $1 - \delta/2$; thus, $V^{H_0, H_1}(x; \delta)$ outputs 1 with probability at least $1 - \delta$.

To verify the soundness property of V , consider an arbitrary pair (O_0, O_1) of oracles. $V^{O_0, O_1}(x; \delta)$ outputs the wrong answer $b := 1 - L(x)$ only if $V_b^{O_b}(x) = 0$, which happens with probability at most $\delta/2$. Therefore, the probability that $V^{O_0, O_1}(x; \delta) \notin \{L(x), \perp\}$ is at most $\delta/2$. ◀

Now, we show that a checker provides an errorless-to-error-prone reduction.

► **Theorem 20.** Let L and H be languages. If there exists a checker for L by H , then $\{H\} \times \text{PSamp}^H \subseteq \text{HeurBPP}$ implies $\{L\} \times \text{PSamp}^H \subseteq \text{AvgBPP}$.

Proof. Let C be a checker for L by H . Let $\mathcal{D} \in \text{PSamp}^H$. For each $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$, we define a distribution $\mathcal{D}'_{\langle n, \delta^{-1} \rangle}$ by the following sampling procedure with oracle access to H : Pick $x \sim \mathcal{D}_n$, simulate the checker C^H on input $(x; \delta)$ with oracle access to H , let

84:16 Errorless Versus Error-Prone Average-Case Complexity

q_1, \dots, q_m denote the queries made by the checker to the oracle during the simulation, and output q_i for a random choice of $i \sim [m]$. Let $\mathcal{D}' := \left\{ \mathcal{D}'_{\langle n, \delta^{-1} \rangle} \right\}_{\langle n, \delta^{-1} \rangle}$. It is easy to observe that $\mathcal{D}' \in \text{PSamp}^H$. By assumption, there exists a randomized heuristic scheme A for $(H, \mathcal{D}') \in \text{HeurBPP}$.

Now, we present a randomized errorless heuristic scheme B for (L, \mathcal{D}) . We define B so that $B(x; n, \delta) = C^{A(\cdot; \langle n, 2/\delta \rangle, \delta')}(x; \delta/2)$ for $\delta' := \delta/2p(n/\delta)$, where $p(n/\delta)$ is a polynomial upper bound on the number of queries made by C on input $(x; \delta/2)$.

To see the completeness of B , fix integers $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$. Observe that $B(x; n, \delta) \neq L(x)$ only if either $C^H(x; \delta/2)$ is incorrect, or there exists an integer $i \in [p(n/\delta)]$ such that i is the first index of the queries made by $C^H(x; \delta/2)$ that are incorrectly answered by A . Let $Q(x; n, \delta/2)_i$ denote the i -th query made by $C^H(x; \delta/2)$ to the oracle. By a union bound, we obtain

$$\begin{aligned} & \Pr_{x \sim \mathcal{D}_{n, B}} [B(x; n, \delta) \neq L(x)] \\ & \leq \sum_{i=1}^{p(n/\delta)} \Pr_{x \sim \mathcal{D}_n} [A(Q(x; n, \delta/2)_i; \langle n, 2/\delta \rangle, \delta') \neq H(Q(x; n, \delta/2)_i)] + \Pr_C [C^H(x; \delta/2) \neq L(x)] \\ & \leq p(n/\delta) \cdot \Pr_{q \sim \mathcal{D}'_{\langle n, 2/\delta \rangle}} [A(q; \langle n, 2/\delta \rangle, \delta') \neq H(q)] + \delta/2 \\ & \leq p(n/\delta) \cdot \frac{\delta}{2p(n/\delta)} + \frac{\delta}{2} = \delta. \end{aligned}$$

To verify the soundness of B , for every $n \in \mathbb{N}$, $\delta^{-1} \in \mathbb{N}$, and every $x \in \text{supp}(\mathcal{D}_n)$,

$$\begin{aligned} & \Pr_B [B(x; n, \delta) \notin \{L(x), \perp\}] \\ & \leq \Pr_{C, A} [C^{A(\cdot; \langle n, 2/\delta \rangle, \delta')}(x; \delta/2) \notin \{L(x), \perp\}] \\ & \leq \delta/2, \end{aligned}$$

where the last inequality holds because of the soundness property of the checker C . \blacktriangleleft

► **Theorem 21.** *If, for every language $L \in \text{PH}$, there exists a language $H \in \text{PH}$ such that L admits a PCP system with the honest oracle H , then $\text{DistPH} \subseteq \text{HeurBPP}$ implies $\text{DistPH} \subseteq \text{AvgBPP}$.*

Proof. Since PH is closed under complement, the assumption and Proposition 19 imply that every language in PH admits a checker by some oracle in PH . Using the proof techniques of Schuler and Watanabe [31] (see also [18, Lemma 18]) we obtain that $\text{PH} \times \text{PSamp}^{\text{PH}} \subseteq \text{HeurBPP}$ under the assumption that $\text{DistPH} \subseteq \text{HeurBPP}$. The result now follows from Theorem 20. \blacktriangleleft

4 Unconditional Errorless to Error-prone Reductions and an Application to Hitting vs Pseudorandomness

In this section, we first show unconditional errorless to error-prone reductions for P against NC^1 and for $\text{UP} \cap \text{coUP}$ against P . We then apply these unconditional results to derive an unconditional equivalence between seed-extending polynomial-time hitting set generators and seed-extending polynomial-time pseudo-random generators.

4.1 Unconditional Reductions

We first show an errorless to error-prone average-case reduction for P against NC^1 .

► **Theorem 22.** *If $\text{DistP} \not\subseteq \text{AvgNC}^1$, then $\text{DistP} \not\subseteq \text{HeurNC}^1$.*

Proof. Let $L \in \text{P}$ be a language and $\mu \in \text{NC1SAMP}$ be an NC^1 -samplable distribution such that $(L, \mu) \notin \text{AvgNC}^1$. We define a language $L' \in \text{P}$ and distribution $\mu' \in \text{NC1SAMP}$ such that $(L', \mu') \notin \text{HeurNC}^1$.

We do this in three stages. First we define a total polynomial-time search problem \mathcal{S}_L with solutions verifiable in AC^0 such that \mathcal{S}_L is errorless hard on average if L is. Then we observe that for a total search problem with solutions verifiable in AC^0 is errorless hard on average iff it is error-prone hard on average. Finally we reduce \mathcal{S}_L to a decision problem $L' \in \text{P}$ while maintaining error-prone average-case hardness with respect to some samplable distribution μ' .

Given a search problem \mathcal{S} and a distribution μ , we say that (\mathcal{S}, μ) is errorless hard on average against NC^1 if there is no NC^1 machine N and language $R_{\text{error}} \in \text{NC}^1$ such that N only fails to solve \mathcal{S} on inputs x such that $R_{\text{error}}(x) = 1$, and moreover the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible. Intuitively this means that for any candidate NC^1 algorithm claiming to solve \mathcal{S} , there is an NC^1 -decidable set that contains all the errors of the algorithms and has small measure according to the distribution μ .

We define \mathcal{S}_L as follows. Let M be a polynomial-time Turing machine deciding L and for any input x , let $T_M(x)$ denote the tableau of M on x , i.e., the sequence of configurations of M on x . \mathcal{S}_L maps x to the single valid output $T_M(x)$ for each x . Note that by the standard local checkability principle used in the proof of the Cook–Levin theorem, the check that $y = T_M(x)$ can be performed by an AC^0 machine V given x and y as inputs.

First, we show that if $(L, \mu) \notin \text{AvgNC}^1$, then (\mathcal{S}_L, μ) is errorless hard on average against NC^1 . Suppose to the contrary that there is an NC^1 machine N solving the search problem \mathcal{S}_L and language $R_{\text{error}} \in \text{NC}^1$ such that the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible, and moreover all instances x on which N fails to solve \mathcal{S}_L have $R_{\text{error}}(x) = 1$. Then we can solve $(L, \mu) \in \text{AvgNC}^1$ by first running the NC^1 machine for R_{error} on x and outputting \perp if $R_{\text{error}}(x) = 1$. If $R_{\text{error}}(x) = 0$, we run N on x , outputting 1 if $T_M(x)$ is an accepting tableau and 0 if it is a rejecting tableau. Since N is always correct on x when $R_{\text{error}}(x) = 0$, we never make an error. Also the probability of outputting \perp is negligible since the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible.

Next, we show that if (\mathcal{S}_L, μ) is errorless hard on average against NC^1 , then (\mathcal{S}_L, μ) is error-prone hard on average against NC^1 . Suppose that N is an NC^1 machine that solves \mathcal{S}_L over μ with all but negligible probability. We show how to solve \mathcal{S}_L over μ in an errorless sense. To do this, we define a language $R_{\text{error}} \in \text{NC}^1$ such that R_{error} has negligible measure and all errors of N are contained within R_{error} . Define $R_{\text{error}}(x) = 1$ iff $V(x, N(x)) = 0$, where V is the AC^0 verifier for \mathcal{S}_L . Since N is an NC^1 machine and V is an AC^0 machine, we have that $R_{\text{error}} \in \text{NC}^1$. If $V(x, N(x)) = 1$, then $N(x)$ is a correct solution to the search problem \mathcal{S}_L on x , and hence all errors of N are contained within R_{error} . Since N is correct with all but negligible probability over μ , we have that R_{error} has negligible measure.

Finally, we show that there is a language L' and a samplable distribution μ' such that if (\mathcal{S}_L, μ) is error-prone hard on average against NC^1 , then $(L', \mu') \notin \text{HeurNC}^1$. L' is defined as follows: $(x, i) \in L'$ if the i th bit of $\mathcal{S}_L(x)$ is 1. Since \mathcal{S}_L is a polynomial-time total search problem with unique solutions, L' is well-defined and belongs to P. μ' is defined by sampling x from μ and i uniformly at random from $[|\mathcal{S}_L(x)|]$. The NC^1 samplability of μ' follows from the NC^1 samplability of μ .

Suppose that $(L', \mu') \in \text{HeurNC}^1$, and let M' be an NC^1 machine solving L' with all but negligible probability over μ' . We define an NC^1 machine M solving $\mathcal{S}_{\mathcal{L}}$ with all but negligible probability over μ . Given x sampled from μ , M runs M' on (x, i) in parallel for each $i \in [|T_M(x)|]$, and forms a string y of length $|T_M(x)|$ by setting $y_i = 1$ iff M' accepts on (x, i) . M then outputs y . M is an NC^1 machine because M' is. We need to show that M solves $\mathcal{S}_{\mathcal{L}}$ with all but negligible probability over μ . Since M' solves L' with all but negligible probability over μ' , by a union bound over $i \in [|T_M(x)|]$, we have that with all but negligible probability over x chosen from μ , y_i is the i 'th bit of $T_M(x)$, and hence $y = T_M(x)$, as desired. ◀

We note that the errorless to error-prone average-case reduction of Theorem 22 works against any class closed under composition with uniform AC^0 circuits – this is the only property of NC^1 we use. Also, the proof gives that errorless average-case hardness over the uniform distribution for P implies error-prone average-case hardness over the uniform distribution for P . This is because μ' can be taken to be the uniform distribution when μ is uniform.

Next we show an errorless to error-prone average-case reduction for $\text{UP} \cap \text{coUP}$. We need a lemma characterizing languages in $\text{UP} \cap \text{coUP}$ using strong unambiguous machines.

► **Definition 23.** *A strong unambiguous machine M is a non-deterministic Turing machine such that for each input x , M outputs a Boolean value on exactly one computation path, and \perp on all other computation paths. We say a strong unambiguous machine M decides a language L if for each x , M outputs $L(x)$ on some computation path.*

► **Lemma 24.** *Let L be a language. $L \in \text{UP} \cap \text{coUP}$ iff there is a strong unambiguous polynomial-time machine M that decides L .*

Proof. We first prove the “if” direction. Suppose there is a strong unambiguous polynomial-time machine M that decides L . We define non-deterministic Turing machines N and N' as follows. On input x , N simulates M , accepting iff M outputs 1. On input x , N' simulates M , accepting iff M outputs 0. Since M is a strong unambiguous machine, N is an unambiguous machine deciding L , and N' is an unambiguous machine deciding \bar{L} .

For the “only if” direction, assume wlog that there are unambiguous machine N and N' deciding L and \bar{L} respectively, both running in polynomial time and using $n^c + c$ bits of non-determinism, for some constant $c > 0$. We can assume this without loss of generality, since an unambiguous machine N using T bits of non-determinism can be transformed into an equivalent one using $T' > T$ bits of non-determinism that accepts on computation path $y \in \{0, 1\}^{T'}$ iff $y = z0^{T'-T}$ and N accepts on computation path $z \in \{0, 1\}^T$.

Now we define a strong unambiguous polynomial-time machine M deciding L as follows. On input x of length n and computation path y of length $n^c + c$, M simulates N and N' on y . If N accepts and N' rejects, M outputs 1. If N rejects and N' accepts, M rejects 0. Otherwise M outputs \perp .

We need to show that M is a strong unambiguous machine and that it decides L . For any input x , either $x \in L$ or $x \in \bar{L}$, but not both. If $x \in L$, N accepts on exactly one computation path y and N' rejects on all computation path, hence there is exactly one computation path on which M outputs a Boolean value, and that value is $L(x)$. If $x \notin L$, N' accepts on exactly one computation path y and N rejects on all computation paths, hence there is exactly one computation path on which M outputs a Boolean value, and that value is $L(x)$, as desired. ◀

► **Theorem 25.** *If $\text{Dist}(\text{UP} \cap \text{coUP}) \not\subseteq \text{AvgP}$, then $\text{Dist}(\text{UP} \cap \text{coUP}) \not\subseteq \text{HeurP}$.*

Proof. Let $L \in \text{UP} \cap \text{coUP}$ be a language and $\mu \in \text{PSamp}$ be a samplable distribution such that $(L, \mu) \notin \text{AvgP}$. We define a language $L' \in \text{UP} \cap \text{coUP}$ and a distribution $\mu' \in \text{PSamp}$ such that $(L', \mu') \notin \text{HeurP}$.

The plan is the same as in the proof of Theorem 22, though the details are different. First we define a total unambiguous search problem \mathcal{S}_L with solutions verifiable in poly-time such that \mathcal{S}_L is errorless hard on average if L is. Then we observe that for a total search problem with solutions verifiable in poly-time is errorless hard on average against P iff it is error-prone hard on average against P . Finally we reduce \mathcal{S}_L to a decision problem $L' \in \text{UP} \cap \text{coUP}$ while maintaining error-prone average-case hardness with respect to some samplable distribution μ' .

Given a search problem \mathcal{S} and a distribution μ , we say that (\mathcal{S}, μ) is errorless hard on average against P if there is no poly-time machine N and language $R_{\text{error}} \in \text{P}$ such that N only fails to solve \mathcal{S} on inputs x such that $R_{\text{error}}(x) = 1$, and moreover the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible. Intuitively this means that for any candidate poly-time algorithm claiming to solve \mathcal{S} , there is a P -decidable set that contains all the errors of the algorithms and has small measure according to the distribution μ .

We define \mathcal{S}_L as follows. Let M be a polynomial-time strong unambiguous Turing machine deciding L and for any input x , let $P(x)$ denote unique computation path of M on which M outputs a Boolean value. \mathcal{S}_L maps x to the single valid output $P(x)$ for each x .

First, we show that if $(L, \mu) \notin \text{AvgP}$, then (\mathcal{S}_L, μ) is errorless hard on average against P . Suppose to the contrary that there is a P machine N solving the search problem \mathcal{S}_L and language $R_{\text{error}} \in \text{P}$ such that the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible, and moreover all instances x on which N fails to solve \mathcal{S}_L have $R_{\text{error}}(x) = 1$. Then we can solve $(L, \mu) \in \text{AvgP}$ by first running the P machine for R_{error} on x and outputting \perp if $R_{\text{error}}(x) = 1$. If $R_{\text{error}}(x) = 0$, we run N on x , outputting 1 if $P(x)$ is a computation path of M on which M outputs 1, and outputting 0 if $P(x)$ is a computation path of M on which M outputs 0. Since N is always correct on x when $R_{\text{error}}(x) = 0$, one of these two cases holds. Also the probability of outputting \perp is negligible since the probability over $x \sim \mu$ that $R_{\text{error}}(x) = 1$ is negligible.

Next, we show that if (\mathcal{S}_L, μ) is errorless hard on average against P , then (\mathcal{S}_L, μ) is error-prone hard on average against P . Suppose that N is a poly-time machine that solves \mathcal{S}_L over μ with all but negligible probability. We show how to solve \mathcal{S}_L over μ in an errorless sense. To do this, we define a language $R_{\text{error}} \in \text{P}$ such that R_{error} has negligible measure and all errors of N are contained within R_{error} . Define $R_{\text{error}}(x) = 1$ iff $N(x)$ outputs a string y encoding a computation path on which M outputs \perp . Since N is a poly-time machine and M can be simulated in poly-time on a given computation path y , we have that $R_{\text{error}} \in \text{P}$. If $N(x)$ outputs a computation path on which M outputs a Boolean value, then $N(x)$ is a correct solution to the search problem \mathcal{S}_L on x , and hence all errors of N are contained within R_{error} . Since N is correct with all but negligible probability over μ , we have that R_{error} has negligible measure.

Finally, we show that there is a language L' and a samplable distribution μ' such that if (\mathcal{S}_L, μ) is error-prone hard on average against NC^1 , then $(L', \mu') \notin \text{HeurNC}^1$. L' is defined as follows: $(x, i) \in L'$ if the i th bit of $\mathcal{S}_L(x)$ is 1. Since \mathcal{S}_L is an unambiguous polynomial-time total search problem, L' is well-defined and belongs to $\text{UP} \cap \text{coUP}$. μ' is defined by sampling x from μ and i uniformly at random from $[|P(x)|]$. The polynomial-time samplability of μ' follows from the polynomial-time samplability of μ .

Suppose that $(L', \mu') \in \text{HeurP}$, and let M' be a poly-time machine solving L' with all but negligible probability over μ' . We define a P machine M solving \mathcal{S}_L with all but negligible probability over μ . Given x sampled from μ , M runs M' on (x, i) in parallel for each

$i \in [|P(x)|]$, and forms a string y of length $|P(x)|$ by setting $y_i = 1$ iff M' accepts on (x, i) . M then outputs y . M is a P machine because M' is. We need to show that M solves $\mathcal{S}_{\mathcal{L}}$ with all but negligible probability over μ . Since M' solves L' with all but negligible probability over μ' , by a union bound over $i \in [|T_M(x)|]$, we have that with all but negligible probability over x chosen from μ , y_i is the i 'th bit of $T_M(x)$, and hence $y = T_M(x)$, as desired. \blacktriangleleft

4.2 An Application to Hitting vs Pseudorandomness

We now describe the application to connections between hitting-set generators and pseudorandom generators. We first need to define these objects.

► **Definition 26.** *Given a function $s : \mathbb{N} \rightarrow \mathbb{N}$ such that $s(n) < n$ for all $n \in \mathbb{N}$, an error function $\epsilon : \mathbb{N} \rightarrow [0, 1]$ and a circuit class \mathfrak{C} , a hitting set generator G with seed length s and error ϵ against \mathfrak{C} is a sequence of functions $\{G_n\}$, where for each n , $G_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^n$, such that for every sequence of circuits $\{C_n\}$ in \mathfrak{C} where C_n accepts at least an $\epsilon(n)$ fraction of inputs of length n , for each n there is $z_n \in \{0, 1\}^{s(n)}$ such that $C_n(G(z_n)) = 1$. We say that G is polynomial-time computable if G_n can be evaluated in polynomial time given n in unary. We say that G is seed-extending if for each $n \in \mathbb{N}$ and $z \in \{0, 1\}^{s(n)}$, z is a prefix of $G(z)$.*

Given a function $s : \mathbb{N} \rightarrow \mathbb{N}$ such that $s(n) < n$ for all $n \in \mathbb{N}$, an error function $\epsilon : \mathbb{N} \rightarrow [0, 1]$ and a circuit class \mathfrak{C} , a pseudorandom generator G with seed length s and error ϵ against \mathfrak{C} is a sequence of functions $\{G_n\}$, where for each n , $G_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^n$, such that for every sequence of circuits $\{C_n\}$ in \mathfrak{C} ,

$$\left| \Pr_{y \sim \{0, 1\}^n} [C_n(y) = 1] - \Pr_{z \sim \{0, 1\}^{s(n)}} [C_n(G_n(z)) = 1] \right| \leq \epsilon(n).$$

We say that G is polynomial-time computable if G_n can be evaluated in polynomial time given n in unary. We say that G is seed-extending if for each $n \in \mathbb{N}$ and $z \in \{0, 1\}^{s(n)}$, z is a prefix of $G(z)$.

The Nisan–Wigderson generator [28] is seed-extending, and implies the following construction of pseudorandom generators from error-prone average-case hardness.

► **Lemma 27.** *Suppose $L \in \mathsf{P}$ is a language such that $(L, U) \notin \text{HeurNC}^1/\text{poly}$. Then for each $\delta > 0$, there is a seed-extending polynomial-time computable pseudorandom generator with seed length n^δ and error $1/n$ against NC^1/poly .*

Lemma 27 follows from standard techniques, namely by applying Yao's XOR lemma [35, 23] to a language in P that is error-prone hard on average over the uniform distribution to obtain a language $L' \in \mathsf{P}$ that cannot be solved in NC^1/poly with probability noticeably better than $1/2$ over the uniform distribution, and then constructing the Nisan–Wigderson generator with seed length n^δ based on this hard predicate. Since $L' \in \mathsf{P}$, the Nisan–Wigderson generator with such a seed length is polynomial-time computable.

We now show the main result of this subsection – an equivalence between seed-extending hitting set generators and seed-extending pseudorandom generators. In order to prove this, we observe that seed-extending hitting set generators imply errorless average-case hardness for P. Then we apply Theorem 22 to transform this errorless hardness into error-prone hardness, and finally we apply Lemma 27 to derive a seed-extending pseudorandom generator with arbitrarily small polynomial seed length from the error-prone average-case hardness assumption.

► **Theorem 28.** *The following are equivalent:*

1. *There is a polynomial-time computable seed-extending hitting set generator with seed length $n - 1$ and error $o(1)$ against NC^1/poly .*
2. *For each $\delta > 0$, there is a polynomial-time computable seed-extending hitting set generator with seed length n^δ and error $1/n$ against NC^1/poly .*
3. *For each $\delta > 0$, there is a polynomial-time computable seed-extending pseudorandom generator with seed length n^δ and error $1/n$ against NC^1/poly .*

Proof. We show that the first item implies the third. The third item implies the second since every pseudorandom generator with error ϵ is also a hitting set generator with error ϵ , and the second item trivially implies the first.

Suppose that there is a polynomial-time computable seed-extending hitting set generator G with seed length $n - 1$ and negligible error against NC^1/poly . We show that this implies that there is a language $L \in \text{P}$ that is errorless hard on average against NC^1/poly over the uniform distribution. Indeed, let $x \in L$ iff x is not in the range of G . Since G is seed-extending, L is polynomial-time computable – we can decide whether $x \in L$ by checking if $G(z) = x$, where z is the $n - 1$ -bit prefix of x . To show that L is errorless hard on average, assume for the sake of contradiction that there is an errorless algorithm A in NC^1/poly solving L on average over the uniform distribution with all but negligible probability. Consider the NC^1 circuit C_n that accepts on x iff A accepts on x . A outputs an answer on all but negligible fraction of strings of length n , and whenever C_n accepts x , we have that $x \in L$, since A is errorless. Note that at least $1/2$ the strings of length n are in L since G stretches its seed, hence C_n accepts x for at least a $1/3$ fraction of strings of length L . Since G is a hitting set generator for NC^1/poly with error $o(1)$, there is some string x such that $C_n(x) = 1$ and x is in the range of G , but this is a contradiction.

Now, using the fact that Theorem 22 works over the uniform distribution, we have that there is a language $L' \in \text{P}$ that is error-prone hard on average over the uniform distribution against NC^1/poly . By Lemma 27, we have that the third item holds, as desired. ◀

We note that as with Theorem 22, there is nothing special about NC^1 in the proof of Theorem 28. Lemma 27 applies to any reasonable class closed under AC^0 reductions, and hence so does Theorem 28.

References

- 1 Martín Abadi, Joan Feigenbaum, and Joe Kilian. On Hiding Information from an Oracle. *J. Comput. Syst. Sci.*, 39(1):21–50, 1989. doi:10.1016/0022-0000(89)90018-4.
- 2 Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. A New General Derandomization Method. *J. ACM*, 45(1):179–213, 1998. doi:10.1145/273865.273933.
- 3 László Babai, Lance Fortnow, and Carsten Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056.
- 4 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
- 5 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the Theory of Average Case Complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.
- 6 Manuel Blum and Sampath Kannan. Designing Programs that Check Their Work. *J. ACM*, 42(1):269–291, 1995. doi:10.1145/200836.200880.

- 7 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-w.
- 8 Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.
- 9 Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
- 10 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.
- 11 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.
- 12 Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. Simplified Derandomization of BPP Using a Hitting Set Generator. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 59–67. Springer, 2011. doi:10.1007/978-3-642-22670-0_8.
- 13 Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 440–449, 2007. doi:10.1145/1250790.1250855.
- 14 Shuichi Hirahara. Identifying an Honest EXP^{NP} Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015. doi:10.4230/LIPIcs.CCC.2015.244.
- 15 Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.
- 16 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251.
- 17 Shuichi Hirahara. Unexpected Power of Random Strings. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 41:1–41:13, 2020. doi:10.4230/LIPIcs.ITCS.2020.41.
- 18 Shuichi Hirahara and Rahul Santhanam. Excluding PH Pessiland. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 78:1–78:25, 2022.
- 19 Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.
- 20 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997. doi:10.1145/258533.258590.
- 21 Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom Generators, Typically-Correct Derandomization, and Circuit Lower Bounds. *Computational Complexity*, 21(1):3–61, 2012. doi:10.1007/s00037-011-0019-z.
- 22 Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.
- 23 Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987. doi:10.1007/BF02579323.
- 24 Yanyi Liu and Rafael Pass. On One-way Functions and Kolmogorov Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020.
- 25 Yanyi Liu and Rafael Pass. On the Possibility of Basing Cryptography on $\text{EXP} \neq \text{BPP}$. In *Proceedings of the International Cryptology Conference (CRYPTO)*, pages 11–40, 2021. doi:10.1007/978-3-030-84242-0_2.

- 26 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.
- 27 Mohammad Mahmoody and David Xiao. On the Power of Randomized Reductions and the Checkability of SAT. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 64–75, 2010. doi:10.1109/CCC.2010.16.
- 28 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.
- 29 Hanlin Ren and Rahul Santhanam. Hardness of KT Characterizes Parallel Cryptography. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 35:1–35:58, 2021. doi:10.4230/LIPIcs.CCC.2021.35.
- 30 Rahul Santhanam. Pseudorandomness and the Minimum Circuit Size Problem. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 68:1–68:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.68.
- 31 Rainer Schuler and Osamu Watanabe. Towards Average-Case Complexity Analysis of NP Optimization Problems. In *Proceedings of the Structure in Complexity Theory Conference*, pages 148–159, 1995. doi:10.1109/SCT.1995.514854.
- 32 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.
- 33 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.
- 34 Leslie G. Valiant and Vijay V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.
- 35 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.