Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Information and Computation 182 (2003) 95–136

Information and Computation

www.elsevier.com/locate/ic

An oracle builder's toolkit[☆]

Stephen Fenner,[*,1] Lance Fortnow,[2] Stuart A. Kurtz, and Lide Li[3]

Department of Computer Science, University of Chicago, 1100 East 58th St., Chicago, IL 60637, USA

Received 17 May 2000; revised 22 August 2002

Abstract

We show how to use various notions of genericity as tools in oracle creation. In particular,

1. we give an abstract definition of genericity that encompasses a large collection of different generic notions;
2. we consider a new complexity class **AWPP**, which contains **BQP** (quantum polynomial time), and infer several strong collapses relative to $\mathcal{SP}$-generics;
3. we show that under additional assumptions these collapses also occur relative to Cohen generics;
4. we show that relative to $\mathcal{SP}$-generics, $\mathbf{ULIN} \cap \text{co-}\mathbf{ULIN} \not\subseteq \mathbf{DTIME}(n^k)$ for any k, where **ULIN** is unambiguous linear time, despite the fact that $\mathbf{UP} \cup (\mathbf{NP} \cap \text{co-}\mathbf{NP}) \subseteq \mathbf{P}$ relative to these generics;
5. we show that there is an oracle relative to which $\mathbf{NP}/\mathbf{1} \cap \text{co-}\mathbf{NP}/\mathbf{1} \not\subseteq (\mathbf{NP} \cap \text{co-}\mathbf{NP})/\mathbf{poly}$; and
6. we use a specialized notion of genericity to create an oracle relative to which

$$\mathbf{NP}^{\mathbf{BPP}} \not\supseteq \mathbf{MA}.$$

Keywords: Complexity classes; Relativization; Generic oracles; Genericity; Forcing

[☆] This paper is an expanded version of one that appeared in Proceedings of the Eighth Annual IEEE Conference on Structure in Complexity Theory, 1993, pp. 120–131.

[*] Corresponding author.

E-mail address: fenner@cse.sc.edu (S. Fenner).

[1] Partially supported by NSF grants CCR 92-09833, CCR 95-01794, and CCR 99-96310, and by ARO grant DAAD 190210048. Currently at University of South Carolina, Department of Computer Science and Engineering, Columbia, SC 29208, USA. Some work done while at University of Southern Maine.

[2] Partially supported by NSF grant CCR 92-53582. Currently at NEC Laboratories America, 4 Independence Way, Princeton, NJ 08540, USA.

[3] Partially supported by NSF grant CCR 92-53582.

1. Introduction

Many constructions in structural complexity involve combining diagonalization and coding requirements. Such constructions face three kinds of difficulties: those intrinsic to the diagonalizations, those intrinsic to the codings, and those arising out of interactions between coding and diagonalization techniques.

The problems intrinsic to either diagonalizations or codings are often easily solved, while the problems arising out of interactions usually prove to be the most challenging. In this paper, we show how various notions of genericity can be used to overcome interaction difficulties.

The general concept of genericity has proven quite useful in oracle building. There are in fact many different (usually mutually exclusive) ways to define when an oracle is generic. Each type of genericity isolates those required properties of an oracle that can be enforced at intermediate stages of an oracle construction. The best example of such requirements are those that can be satisfied by extending the partial oracle a finite amount, perhaps to diagonalize against some relativized computation. A *Cohen generic* oracle, defined in Section 4, is one that satisfies all definable requirements of this type.[4] There are, however, other types of diagonalization requirements that are satisfied only by extending an infinite amount, and still others that must cohabit with coding requirements to build the oracle we need. We will show that we can usually handle these other types of requirements in a uniform way by defining alternate notions of genericity.

We therefore note a correspondence between sets of requirements needed to construct an oracle on the one hand and notions of genericity on the other: a generic set is defined as one that meets all requirements in a particularly specified set. This paper gathers evidence for the following thesis: for any set of consistent requirements sufficient to construct an oracle with a desired property, one can find an appropriate definition of genericity where all the generic sets under this definition satisfy all the requirements and thus have the desired property. If the requirements in question include any sort of diagonalizations, even if they are mixed with coding requirements, the corresponding notion of genericity is nontrivial.

This correspondence has two useful implications.

First, explicit oracle constructions can be rephrased as proofs that any generic oracle (appropriately defined) can do the job. Rephrasing proofs in this way can lead to considerable conceptual simplification. An explicit construction often involves interleaving coding and diagonalization requirements, while worrying (a lot!) about how they might conflict with each other, all in one big stage-by-stage construction. A generic oracle proof, on the other hand, separates this task neatly into two independent parts:

- Choose an appropriate notion of genericity, and show that there exist sets which conform to this notion.
- Show that each requirement that the oracle must satisfy is *dense* with respect to our chosen notion of genericity, i.e., has the potential of being satisfied at any point.

[4] *Definability* is taken with respect to a countable formal language, such as first-order Peano arithmetic.

Taken together, these two facts automatically give an oracle that suffices for one's purposes; in fact, all the generic sets in question will suffice. One no longer needs to worry about how to interleave requirements explicitly.

The benefits of such simplification are well known in several cases involving Cohen generics, where the requirements can always be met by finite extension. For example, a minor observation regarding the construction of an oracle separating **P** from **NP** shows that any Cohen generic oracle separates these two classes. Another example is implicit in the observation that Yao's construction of an oracle that separates the polynomial hierarchy works by finite extension, and so any Cohen generic oracle must also separate the polynomial hierarchy. General techniques have been developed to obtain results about Cohen generics; we will investigate some of them in Section 4, where we show how several previously known oracle constructions can be replaced by results about Cohen generics.

What about including requirements that can only be met by an infinite extension, or by explicitly maintaining some invariant for an infinitely long time (coding requirements)? It is less obvious, but such constructions can also be turned into generic oracle proofs, using notions of genericity other than Cohen genericity. One such notion is $\mathcal{SP}$-genericity, which we define in Section 5, and another we will use in Section 7.2. In fact, there are a wide variety of different notions of genericity useful in complexity theory, each with its own particular properties (see Sections 3 and 8). We do not wish to imply that using genericity to prove the existence of oracles obviates the need for cleverness or ingenuity. We merely wish to assert that the added modularity in a generic oracle proof allows one to concentrate on the intricacies of the individual requirements themselves, rather than on how different requirements interact.

The second practical implication of the correspondence between requirements and genericity is that we often discover oracles with certain properties *without* explicitly constructing them, by simply considering various notions of genericity. Given some particular notion of genericity, it will be the case that any reasonable complexity theoretic property P either holds for all generics or none of them. Thus, it is a natural question to ask, "does a generic set satisfy P or not?" Answering such a question is considerably less daunting than trying to construct an oracle satisfying property P from scratch. To resolve the question, one can work incrementally, trying to prove the question true and false simultaneously, and gathering useful facts along the way. Hopefully, one proof strategy wins in the end. If the answer turns out to be affirmative, an explicit oracle construction is unnecessary.

This was exactly the line of attack used to discover an oracle for which the Berman–Hartmanis Isomorphism Conjecture holds [20]. The notion of $\mathcal{SP}$-genericity (see Section 5) was defined and many of its properties studied before its effect on the Conjecture was known. It was natural to try to determine the status of the Conjecture relative to $\mathcal{SP}$-generic oracles. We originally attempted to prove that the Isomorphism Conjecture *failed* relative to these oracles. A careful analysis of where the proof broke down led to the correct proof that it held relative to $\mathcal{SP}$-generics. Most importantly, this proof was built in a piecemeal fashion; it combines a number of smaller results about $\mathcal{SP}$-generics that were shown *independently*, and that bear no obvious relationship to each other. It is difficult for us to imagine how one could have the foresight to formulate all at once the widely different types of requirements needed to perform the construction from scratch, without knowing at the outset how such requirements needed to be combined. The success of applying the properties of $\mathcal{SP}$-generic sets to the Isomorphism Conjecture is to our knowledge the best illustration of the benefits of the genericity approach to oracle results in complexity theory.

In Section 3 we give a general framework for defining different types of generic sets in terms of arithmetic forcing. It is well known that there are often equivalent ways of defining genericity as a purely topological notion, using the ideas of Baire category. We mention these as well.

In Section 4, we systematically assemble a number of basic facts about Cohen generic sets, many of which are generalizations of known results [11]. We then use these results together with the technique of 'rerelativization' to produce a number of oracles that cause the Isomorphism Conjecture to fail in a variety of different ways.

We define $\mathcal{SP}$-generic sets in Section 5, and recount some of their basic properties. The definition of $\mathcal{SP}$-genericity we give there is simpler and more manageable than the original definition of $\mathcal{SP}$-genericity given in [20], where it was denoted "sp-genericity." The $\mathcal{SP}$-generic sets defined in the present paper, however, are easily seen to possess all the same useful properties of the original $\mathcal{SP}$-generics of [20]. We end Section 5 with a result that ties $\mathcal{SP}$-genericity with the construction of exact pairs in computability theory.

In Section 6 we show how generic oracles collapse complexity classes. We unify and extend results in [11,30,20] by showing that a large number of interesting complexity classes collapse to **P** relative to $\mathcal{SP}$-generic oracles. We also use the notion of *certificate complexity* and a new complexity class **AWPP** to present a general criterion for when such collapses occur. Recently, Fortnow and Rogers [24] have shown that **AWPP** contains **BQP**, the class of all languages decided by a quantum computer in bounded error probabilistic polynomial time [13]. Thus relative to $\mathcal{SP}$-generics, $\mathbf{P} = \mathbf{BQP}$ but **PH** separates. By similar proof techniques, all these collapses also occur relative to Cohen generics, given certain assumptions about unrelativized classes.

In contrast to Section 6, in Section 7 we give nontrivial separation results relative to generics. We show that there is an oracle relative to which

$$\mathbf{NP/1} \cap \mathbf{coNP/1} \not\subseteq (\mathbf{NP} \cap \mathbf{coNP})/\mathbf{poly}.$$

We also show that there is an oracle A relative to which $\mathbf{MA} \not\subseteq \mathbf{NP}^{\mathbf{BPP}}$. The proof introduces a new notion of genericity that is different from both Cohen and $\mathcal{SP}$-genericity. Finally, we show that $\mathcal{SP}$-generics separate various linear time promise classes from $\mathbf{DTIME}(n^k)$ for all k.

We describe further research and present some open problems in Section 8.

2. Preliminaries

This paper treads across a number of mathematical areas (logic, computability, computational complexity) with differing and sometimes conflicting notational conventions for basically the same concepts. We will use notation that is mostly idiomatic to the subject at hand.

We let $\Sigma = 2 = \{0, 1\}$ and let Σ^* be the set of all finite binary strings. For $n \geqslant 0$ we let Σ^n and $\Sigma^{\leqslant n}$ be the sets of binary strings of length equal to n and at most n, respectively. We use ω to denote the set $\{0, 1, 2, \ldots\}$ of natural numbers, which we identify with binary strings in the usual lexicographical way. We let $\mathbb{Z}$ denote the integers. Let f and g be partial functions on ω (or Σ^*). We say that *f extends g* ($f \succeq g$) if $\mathrm{dom}(g) \subseteq \mathrm{dom}(f)$ and f agrees with g on g's domain. If in addition $f \neq g$, then we may write $f \succ g$. This sense of the $\succeq$ relation is the one commonly found in the literature of computability theory and computer science, so we adopt it here even though it is the reverse of the standard partial order notation for forcing. We say that f and g are *compatible* if they share a common extension. We say f is a *partial characteristic function* or *partial oracle* if $\mathrm{range}(f) \subseteq 2$, and a *finite characteristic function* if, in addition, $\mathrm{dom}(f)$ is finite.

We let ϵ denote the empty string, and we let $|\sigma|$ denote the length of string σ. We may also regard string σ as a finite characteristic function whose domain is an initial segment of ω (or Σ^*). Thus $\mathrm{dom}(\sigma) = \{0, \ldots, |\sigma| - 1\}$. If x and y are strings, then xy is the concatenation of x followed by y, and for $n \in \omega$, x^n is the n-fold concatenation of x with itself. We use $A, B, C, \ldots$ to denote arbitrary subsets of ω, which of course are canonically identified with subsets of Σ^* ("languages") and with total functions $\omega \to 2$ or $\Sigma^* \to 2$.

Let M be a deterministic oracle machine, σ a partial oracle, and $x \in \Sigma^*$. We write $M^\sigma(x)\downarrow$ to mean M halts on input x where all of M's oracle queries are in $\mathrm{dom}(\sigma)$ and are answered according to σ.

If E is some expression then $\lambda x.E$ denotes the function that, on input y, outputs $[y/x]E$, that is, E with the value y substituted for all free occurrences of x in E.

We assume knowledge of standard complexity classes and other complexity theoretic concepts. These may be found in a number of places [6,7,47]. For example, when we say that a one-to-one (not necessarily onto) function $f : \Sigma^* \to \Sigma^*$ is invertible in ptime, we mean left-invertible in ptime.

2.1. Counting classes

In Section 6 we will refer to several classes that may not be widely known. These are mostly counting classes described in, for example [19,22,50]. For completeness, we define them here and give their basic properties.

Definition 2.1 [55]. A function $f : \Sigma^* \to \omega$ is in #**P** if there is a nondeterministic polynomial-time Turing machine M such that $f(x)$ is the number of accepting paths of M on input x, for all $x \in \Sigma^*$.

The rest of the information in this section is from [19], which can be consulted for more details.

Definition 2.2. A function $f : \Sigma^* \to \mathbb{Z}$ is in Gap**P** if there are two #**P** functions f_+ and f_- such that $f(x) = f_+(x) - f_-(x)$ for all x. Equivalently, $f \in$ Gap**P** if there is a nondeterministic polynomial-time Turing machine M such that $f(x)$ is the number of accepting paths minus the number of rejecting paths of M on input x (denoted $\mathrm{gap}_M(x)$).

It is easily shown that

$$\mathrm{Gap}\mathbf{P} = \{\lambda x.\big(2^{p(|x|)} - f(x)\big) \mid f \in \#\mathbf{P} \text{ and } p \text{ a polynomial}\},$$

and that Gap**P** has the following closure properties:
1. #**P** $\subseteq$ Gap**P**.
2. If $f \in$ Gap**P** then $-f \in$ Gap**P**.
3. If $f(\cdot,\cdot) \in$ Gap**P** and p is a polynomial, then $g(\cdot), h(\cdot) \in$ Gap**P**, where

$$g(x) = \sum_{y,|y| \leqslant p(|x|)} f(x,y)$$

and

$$h(x) = \prod_{n,n \leqslant p(|x|)} f(x, 0^n).$$

Several counting classes can be defined easily using Gap**P**.

Definition 2.3.

- A language L is in **PP** ("probabilistic polynomial time") if there is an $f \in \mathrm{Gap}\mathbf{P}$ such that for all $x \in \Sigma^*$,
  $$x \in L \iff f(x) > 0.$$
- A language L is in $\mathrm{C}_=\mathbf{P}$ ("exact counting polynomial time") if there is an $f \in \mathrm{Gap}\mathbf{P}$ such that for all $x \in \Sigma^*$,
  $$x \in L \iff f(x) = 0.$$
- A language L is in $\oplus\mathbf{P}$ ("parity polynomial time") if there is an $f \in \mathrm{Gap}\mathbf{P}$ such that for all $x \in \Sigma^*$,
  $$x \in L \iff f(x) \text{ is odd.}$$
- A language L is in **SPP** ("stoic **PP**") if there is an $f \in \mathrm{Gap}\mathbf{P}$ such that for all $x \in \Sigma^*$,
  $$x \in L \Rightarrow f(x) = 1,$$
  $$x \notin L \Rightarrow f(x) = 0.$$
- A language L is in **WPP** ("wide **PP**") if there is an $f \in \mathrm{Gap}\mathbf{P}$ and $g \in \mathbf{FP}$ such that, for all $x \in \Sigma^*$, $g(x) > 0$ and
  $$x \in L \Rightarrow f(x) = g(x),$$
  $$x \notin L \Rightarrow f(x) = 0.$$
- A language L is in **LWPP** ("length-dependent **WPP**") if there is an $f \in \mathrm{Gap}\mathbf{P}$ and $g \in \mathbf{FP}$ such that, for all $x \in \Sigma^*$, $g(x) > 0$ and
  $$x \in L \Rightarrow f(x) = g(0^{|x|}),$$
  $$x \notin L \Rightarrow f(x) = 0.$$

The following inclusions are known [19]:

$$\mathbf{P} \subseteq \mathbf{UP} \subseteq \mathbf{FewP} \subseteq \mathbf{Few} \subseteq \mathbf{SPP} \subseteq \mathbf{LWPP} \subseteq \mathbf{WPP}$$
$$\subseteq \mathrm{C}_=\mathbf{P} \cap \text{co-}\mathrm{C}_=\mathbf{P} \subseteq \mathrm{C}_=\mathbf{P} \cup \text{co-}\mathrm{C}_=\mathbf{P} \subseteq \mathbf{PP}.$$

It is known that the Graph Isomorphism problem is in **LWPP**, and Graph Automorphism is in **SPP** [36]. More recently Graph Isomorphism has been shown to be in **SPP** [1]. The class **AWPP** ("approximate **WPP**"), defined in Section 6, contains **WPP**, but probably does not contain $\mathrm{C}_=\mathbf{P}$ or co-$\mathrm{C}_=\mathbf{P}$. All the definitions above relativize in the usual way.

3. Abstract genericity

This section gives a detailed treatment of the theory of generic oracles. A full understanding of this section is *not* required for understanding the remaining parts of the paper. Corollary 3.16 and Lemma 3.17 are the most important facts, and the latter does not mention the forcing relation.

3.1. Basic definitions and lemmas

In this section, we will discuss forcing in arithmetic in greater generality than usual. The results given here are standard in the literature, with some occasional minor variations. This section is meant primarily as a primer on forcing in arithmetic. Let $\langle \omega, +, \times, 0, S \rangle$ be the standard model of Peano Arithmetic (S is the successor function). We let 2^ω be the space of infinite binary sequences endowed with the Cantor topology, so that each basic open set corresponds to a finite characteristic function σ and consists of all infinite binary sequences extending σ. We identify subsets of ω with infinite binary sequences via their characteristic functions.

Definition 3.1. A *condition* is a nonempty perfect subset of 2^ω.

Recall that the perfect sets are closed sets without isolated points. A perfect subset of 2^ω is identified uniquely with a complete (fully branching) binary subtree of the full binary tree $\{0, 1\}^*$, and vice versa, so it is sometimes better to view a condition as a tree. This allows us to speak unambiguously about the 0-branch or 1-branch of a condition.

Definition 3.2. Let γ be a condition and let $b \in \{0, 1\}$. The *b-branch* of γ is the condition

$$\{A \in \gamma \mid A(m) = b\},$$

where $m \in \omega$ is least such that $(\exists A, B \in \gamma)\, A(m) \neq B(m)$.

For $k > 1$ and $b_1, \ldots, b_k \in \{0, 1\}$, we inductively define the *$b_1 \cdots b_k$-branch* of γ to be the $b_2 \cdots b_k$-branch of the b_1-branch of γ.

Note that the 0- and 1-branches of a condition are also conditions.

From our point of view, conditions are approximations to subsets of ω (single infinite binary sequences), where the latter are ultimately identified with subsets of $2^{<\omega} = \Sigma^*$ for the purposes of computation. An important special case of a condition is the set of total characteristic functions extending some given partial function with coinfinite domain; in such cases we will identify the condition with the partial function itself.

Definition 3.3. A *notion of genericity* is a nonempty set $\mathcal{G}$ of conditions such that, for all $\gamma \in \mathcal{G}$, all $G \in \gamma$, and all $n \in \omega$, there is a condition $\gamma' \in \mathcal{G}$ such that $\gamma' \subseteq \gamma$ and $(\forall G' \in \gamma')[G'(n) = G(n)]$.

The conditions of $\mathcal{G}$ are called $\mathcal{G}$-conditions.

The closure property of Definition 3.3 simply enables us to refine any approximation ($\mathcal{G}$-condition) γ to completely determine the value of the oracle being approximated at any particular input. It is also exactly the restriction necessary to prove one of the basis cases of Lemma 3.14 below. Note that Definition 3.3 implies that both the 0- and 1-branches of any $\mathcal{G}$-condition contain other $\mathcal{G}$-conditions as subsets. Further, Definition 3.3 holds if *all* branches of any $\mathcal{G}$-condition contain other $\mathcal{G}$-conditions as subsets, i.e., if for all $\gamma \in \mathcal{G}$ and all $s \in 2^{<\omega}$, there is a $\gamma' \in \mathcal{G}$ which is a subset of the s-branch of γ.

Remark. A more traditional and more general approach to forcing and genericity would be to start with an arbitrary partial order $\langle P, \leqslant \rangle$ and let the conditions be the elements of P. Here we restrict our partial orders to be particular families of perfect subsets of 2^{ω}, partially ordered by $\subseteq$ and satisfying Definition 3.3. Our restricted approach has the benefit of concreteness and is more directly suited to constructing oracles. Traditionally, one defines forcing on $\langle P, \leqslant \rangle$ in order to obtain generic subsets of P. The information in these subsets must then be extracted to produce oracles according to some explicit translation scheme that depends on the situation at hand. Our approach makes this last translation step trivial and uniform over all the notions of genericity we consider—we simply take the intersection of all elements of the generic subset (see Definitions 3.6 and 3.7 below). Furthermore, the added generality of the traditional approach does not really buy us anything here; to our knowledge, all types of generic oracles used in complexity theory can be built using notions of genericity obeying Definition 3.3.

For the rest of this section, we will assume that $\mathcal{G}$ is some fixed but arbitrary notion of genericity.

Definition 3.4. Let $\mathcal{L}_{\mathrm{PA}}[X]$ be the language of Peano Arithmetic, augmented by a unary predicate symbol X. Let $\mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$ be the set of sentences (formulas with no free variables) in $\mathcal{L}_{\mathrm{PA}}[X]$.

We assume that the only logical operators in $\mathcal{L}_{\mathrm{PA}}[X]$ are $\neg$, $\vee$, and $\exists$; the other standard operators are defined in terms of these three. The $\mathcal{G}$-forcing relation $\Vdash_{\mathcal{G}}$ on $\mathcal{G} \times \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$ is defined by a simple recursion on the structure of formulas, which is essentially Tarski's definition of truth except in the case of negation. If $n \in \omega$, we let $\bar{n}$ denote the formal term $\underbrace{SS\cdots S}_{n}0$. The variables γ and τ range over $\mathcal{G}$.

$$\gamma \Vdash_{\mathcal{G}} \varphi \iff \varphi \text{ is true in } \omega \text{ (where } \varphi \text{ is atomic and } X \text{ does not occur in } \varphi), \tag{1}$$

$$\gamma \Vdash_{\mathcal{G}} X(\bar{n}) \iff (\forall A \in \gamma)\, A(n) = 1 \text{ (where } n \in \omega), \tag{2}$$

$$\gamma \Vdash_{\mathcal{G}} \varphi \vee \psi \iff \gamma \Vdash_{\mathcal{G}} \varphi \text{ or } \gamma \Vdash_{\mathcal{G}} \psi, \tag{3}$$

$$\gamma \Vdash_{\mathcal{G}} (\exists x)\varphi \iff (\exists a \in \omega)\, \gamma \Vdash_{\mathcal{G}} [\bar{a}/x]\varphi, \tag{4}$$

$$\gamma \Vdash_{\mathcal{G}} \neg\varphi \iff (\forall \tau \subseteq \gamma)\, \tau \nVdash_{\mathcal{G}} \varphi. \tag{5}$$

A condition τ *extends* a condition γ if $\tau \subseteq \gamma$. The intuition behind the term "extends" is that τ more completely specifies the oracle than γ. Thus, clause (5) says roughly that $\neg\varphi$ is forced iff we can never force φ by refining our approximation of the oracle. Note that if $\gamma \Vdash_{\mathcal{G}} \varphi$ and $\delta \subseteq \gamma$, then $\delta \Vdash_{\mathcal{G}} \varphi$ via a straightforward induction. Note also that $\gamma \Vdash_{\mathcal{G}} \neg X(\bar{n})$ iff $(\forall A \in \gamma) A(n) = 0$, by the closure property in Definition 3.3. We will usually drop the subscript from $\Vdash$ if the underlying notion of genericity is clear from the context. In this section it will always be $\mathcal{G}$.

One more clarification: when we identify partial characteristic functions with conditions, we have $\sigma \preceq \tau$ as partial functions if and only if $\sigma \supseteq \tau$ as conditions. In either sense, we still say that τ extends σ. Likewise, if $A \subseteq \omega$, then saying $\sigma \prec A$ assumes that we identify A with its total characteristic function and σ with a partial function; if we view σ as a condition and A an element of 2^ω, we would say $A \in \sigma$ to mean the same thing.

For any set S of conditions, we say that A *meets* S if $A \in \sigma$ ($\sigma \prec A$) for some $\sigma \in S$. For $S \subseteq \mathcal{G}$, we say that S is *dense* (in $\mathcal{G}$) if for every $\sigma \in \mathcal{G}$ there is a $\tau \in S$ with $\sigma \supseteq \tau$. For any $Y \subseteq 2^\omega$, we say Y is *dense* in $\mathcal{G}$ if $\sigma \cap Y \neq \emptyset$ for all $\sigma \in \mathcal{G}$.

The following lemmas and definitions are standard.

Lemma 3.5. *For every sentence φ of $\mathcal{L}_{\mathrm{PA}}[X]$, the set $\{\alpha : \alpha \Vdash \varphi \vee \neg\varphi\}$ is dense in $\mathcal{G}$, i.e., for all $\gamma \in \mathcal{G}$, there exists a $\delta \in \mathcal{G}$ such that $\delta \subseteq \gamma$ and either $\delta \Vdash \varphi$ or $\delta \Vdash \neg\varphi$.*

Proof. Given γ suppose there is no $\delta \subseteq \gamma$ such that $\delta \Vdash \varphi$. Then by definition, $\gamma \Vdash \neg\varphi$, so we take $\delta = \gamma$. $\square$

The peculiar definition of forcing negations was chosen by Cohen precisely to facilitate the proof of Lemma 3.5. Our next goal is to extend the concept of forcing by elements of $\mathcal{G}$ (which approximate oracles) to forcing by the oracles they approximate. However, we must first introduce generic filters, which one may alternatively call "consistent approximation schemes." These objects come from a standard way of defining genericity in set theory (see [37] or [31] for example, and the Remark above). Each generic filter will uniquely determine an oracle.

Definition 3.6. A *generic filter over $\mathcal{G}$* is a subset $\mathbb{G} \subseteq \mathcal{G}$ such that
1. $(\forall \sigma, \tau \in \mathcal{G})[\tau \in \mathbb{G}\ \&\ \tau \subseteq \sigma \rightarrow \sigma \in \mathbb{G}]$,
2. $(\forall \sigma_1, \sigma_2 \in \mathbb{G})(\exists \tau \in \mathbb{G})\tau \subseteq \sigma_1 \cap \sigma_2$ (i.e., τ extends both σ_1 and σ_2), and
3. For each $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, there is a $\gamma \in \mathbb{G}$ such that $\gamma \Vdash \varphi \vee \neg\varphi$.

Condition (3) implies that $\mathbb{G} \neq \emptyset$. Moreover, since (i) 2^ω is compact (with respect to the Cantor topology), (ii) all $\mathcal{G}$-conditions are perfect sets and hence closed, and (iii) the intersection of any finite number of $\mathcal{G}$-conditions in $\mathbb{G}$ is nonempty by Condition (2), it follows that $\bigcap \mathbb{G} \neq \emptyset$. Furthermore, for each $n \in \omega$, there is a $\gamma \in \mathbb{G}$ such that $\gamma \Vdash X(\bar{n}) \vee \neg X(\bar{n})$, which implies that $A(n)$ is the same for all $A \in \gamma$. It follows that $\bigcap \mathbb{G}$ is a singleton.

Definition 3.7. A set $G \subseteq \omega$ is *$\mathcal{G}$-generic* if there is a generic filter $\mathbb{G}$ over $\mathcal{G}$ with $\bigcap \mathbb{G} = \{G\}$. In such a case, we say that $\mathbb{G}$ *builds G*.

We can now define forcing for generic filters over $\mathcal{G}$ and for $\mathcal{G}$-generic sets.

Definition 3.8. Let $\mathbb{G}$ be a generic filter over $\mathcal{G}$, and let $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$ be some sentence. We say that $\mathbb{G} \Vdash \varphi$ if there is a $\gamma \in \mathbb{G}$ such that $\gamma \Vdash \varphi$.

We will see shortly (Lemma 3.14) that, given φ as above, the question of whether or not $\mathbb{G} \Vdash \varphi$ depends only on the $\mathcal{G}$-generic set built by $\mathbb{G}$, and thus the following definition makes sense:

Definition 3.9. Let G be a $\mathcal{G}$-generic set, and let $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$. We say that $G \Vdash \varphi$ if some (equivalently, every) generic filter over $\mathcal{G}$ that builds G also forces φ.[5]

In the meantime, we consider an important special case that applies to most of our notions of genericity.

Definition 3.10. We say that $\mathcal{G}$ is *basic* if for every $\sigma_1, \sigma_2 \in \mathcal{G}$ and every $A \in \sigma_1 \cap \sigma_2$, there exists a $\tau \in \mathcal{G}$ with $A \in \tau \subseteq \sigma_1 \cap \sigma_2$.

Most of the notions of genericity that we will work with are basic. In particular, all of the notions we consider that are made up of conditions corresponding to partial characteristic functions are basic. One chief advantage of basic notions is that they allow us to dispense with talk of generic filters. For example,

Lemma 3.11. *Suppose $\mathcal{G}$ is basic. Let G be $\mathcal{G}$-generic and $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$. Then $G \Vdash \varphi$ if and only if there exists a $\gamma \in \mathcal{G}$ such that $G \in \gamma$ and $\gamma \Vdash \varphi$.*

Proof. Use Definition 3.9 and the fact that if $\mathcal{G}$ is basic, then $\{\gamma \in \mathcal{G} \mid G \in \gamma\}$ is a generic filter over $\mathcal{G}$ (the biggest) that builds G. $\square$

The reason for all of this machinery is, of course, to obtain $\mathcal{G}$-generic sets. This would be a lot of work for naught unless $\mathcal{G}$-generics exist. Fortunately, they do, and in abundance.

Lemma 3.12 (Existence of generic sets). *For every notion of genericity $\mathcal{G}$, the set of $\mathcal{G}$-generic sets is dense in $\mathcal{G}$, i.e., for every $\gamma \in \mathcal{G}$, there is a $\mathcal{G}$-generic set $G \in \gamma$ (in fact, there are precisely $2^{\aleph_0}$ (continuum) many generic sets in γ).*

Proof. We first prove density. Let $\{\varphi_i\}_{i\in\omega}$ be an enumeration of all sentences of $\mathcal{L}_{\mathrm{PA}}[X]$. Fix $\gamma \in \mathcal{G}$, and let $\gamma_{-1} = \gamma$. For all $i \geqslant 0$, given γ_{i-1}, choose γ_i extending γ_{i-1} such that $\gamma_i \Vdash \varphi_i \vee \neg\varphi_i$ (cf. Lemma 3.5). Let $\mathbb{G} = \{\delta \in \mathcal{G} | (\exists i)\gamma_i \subseteq \delta\}$. It is easy to check that $\mathbb{G}$ is a generic filter over $\mathcal{G}$ which builds some element of γ.

To show abundance, we modify the construction above slightly as follows: Let $\gamma_\epsilon = \gamma$ (ϵ is the empty string). For all $w \in \Sigma^*$ and $b \in \{0,1\}$, given γ_w, choose γ_{wb} to extend the b-branch of γ_w such that $\gamma_{wb} \Vdash \varphi_{|w|} \vee \neg\varphi_{|w|}$. For any infinite binary sequence $A \in 2^\omega$, define $\mathbb{G}_A = \{\delta \mid (\exists w \in \Sigma^*) w \prec A \;\&\; \gamma_w \subseteq \delta\}$. The $\mathbb{G}_A$ are all generic filters over $\mathcal{G}$ building distinct generic sets. $\square$

Finally, we need to make the formal connection between forcing and truth, i.e., between our approximations to truth (forced sentences), and the theory of the objects being approximated (subsets of ω, i.e., oracles). Given $A \subseteq \omega$, let $\omega[A]$ be the expansion of the standard model of Peano Arithmetic to the language $\mathcal{L}_{\mathrm{PA}}[X]$ in which X is interpreted as A, that is, $X^{\omega[A]} = A$. For

[5] Often, there will be only one unique filter building G.

$\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, the expression $\omega[A] \models \varphi$ means that φ is true (in the standard model) when X is interpreted as A. We may write φ^A to stand for $\omega[A] \models \varphi$. An easy connection between truth and forcing is the following, which will be used in later sections.

Proposition 3.13. *For all $\sigma \in \mathcal{G}$ and $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, either Σ_1 or Π_1, if $(\forall A \in \sigma)\ \omega[A] \models \varphi$, then $\sigma \Vdash \neg\neg\varphi$, and if $\sigma \Vdash \neg\neg\varphi$, then $(\exists A \in \sigma)\ \omega[A] \models \varphi$. (By Σ_1 we mean that φ consists of zero or more '$\exists$' quantifiers before a quantifier-free formula. By Π_1 we mean the negation of a Σ_1 sentence.)*

Proof. Induction on the syntax of φ. The '$\exists$' case uses the compactness of σ in the Cantor topology. $\square$

The next connection is more significant.

Lemma 3.14 (Forcing is truth). *Let $\mathbb{G}$ be any generic filter over $\mathcal{G}$ and let G be the $\mathcal{G}$-generic set built by $\mathbb{G}$. Then for all $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, we have $\mathbb{G} \Vdash \varphi$ if and only if $\omega[G] \models \varphi$.*

Lemma 3.14 justifies Definition 3.9 by showing that $G \Vdash \varphi$ is independent of the generic filter building G. Through the lens of Definition 3.9, it just shows that $G \Vdash \varphi$ if and only if $\omega[G] \models \varphi$.

Proof of Lemma 3.14. We use a straightforward induction on the syntactical structure of φ.

- The lemma is clear if φ is atomic and does not mention X.
- If $\varphi = X(\bar{n})$, then we have

$$\begin{aligned}\mathbb{G} \Vdash X(\bar{n}) &\iff (\exists \gamma \in \mathbb{G})\gamma \Vdash X(\bar{n})\\ &\iff (\exists \gamma \in \mathbb{G})(\forall A \in \gamma)A(n) = 1\\ &\iff G(n) = 1\\ &\iff \omega[G] \models X(\bar{n}).\end{aligned}$$

The first two equivalences follow from the definition of forcing. The third follows from the fact that, for any $\mathbb{G}'$ building G, there is a $\gamma \in \mathbb{G}'$ that forces $X(\bar{n}) \vee \neg X(\bar{n})$, and if $\gamma \Vdash \neg X(\bar{n})$ then $G(n)$ must be 0.

- The cases for disjunctions and existentials are obvious.
- For negations, we have

$$\begin{aligned}\mathbb{G} \Vdash \neg\varphi &\iff (\exists \gamma \in \mathbb{G})\gamma \Vdash \neg\varphi\\ &\iff (\forall \gamma' \in \mathbb{G})\gamma' \nVdash \varphi\\ &\iff \mathbb{G} \nVdash \varphi\\ &\iff \omega[G] \not\models \varphi\\ &\iff \omega[G] \models \neg\varphi.\end{aligned}$$

To see the second equivalence, suppose some $\gamma' \in \mathbb{G}$ forces φ; then γ and γ' have a mutual extension in $\mathbb{G}$ that forces both φ and $\neg\varphi$—a contradiction. The fourth equivalence follows from the inductive hypothesis. $\square$

The following lemma gives another connection between truth and forcing, this time with forcing negations. Here we have a nice extensional characterization of forcing negations in the case where $\mathcal{G}$ is basic.

Lemma 3.15. *Suppose $\mathcal{G}$ is basic. Then, for every $\gamma \in \mathcal{G}$ and every $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$,*

$$\gamma \Vdash \neg\varphi \iff (\forall \mathcal{G}\text{-generic } G \in \gamma)\omega[G] \nvDash \varphi.$$

That is, $\gamma \Vdash \neg\varphi$ iff φ is false for all generic elements of γ.

Proof. Suppose $\gamma \Vdash \neg\varphi$. Then no extension of γ forces φ. Suppose $\omega[G] \models \varphi$ for some $\mathcal{G}$-generic $G \in \gamma$. Then by Lemmas 3.11 and 3.14, $G \in \sigma$ for some $\sigma \in \mathcal{G}$ that forces φ. Since $\mathcal{G}$ is basic, γ and σ have a common extension, which also forces φ—a contradiction.

Conversely, suppose $\gamma \nVdash \neg\varphi$. Fix a $\mathcal{G}$-condition $\sigma \subseteq \gamma$ such that $\sigma \Vdash \varphi$. By Lemma 3.12, there is a $\mathcal{G}$-generic $G \in \sigma$. By Lemma 3.11, $G \Vdash \varphi$ and hence $\omega[G] \models \varphi$ by Lemma 3.14. □

Corollary 3.16. *If $\mathcal{G}$ is basic and $\gamma \in \mathcal{G}$, then $\gamma \Vdash \neg\neg\varphi$ iff $\omega[G] \models \varphi$ for all $\mathcal{G}$-generic $G \in \gamma$.*

(Statements similar to Lemma 3.15 and Corollary 3.16, involving generic filters, hold for arbitrary $\mathcal{G}$ (not necessarily basic).)

Forcing the double negation is sometimes referred to as *weak forcing*, although some authors prefer simply to call it forcing, and refer to our definition as "strong forcing." Strong forcing clearly implies weak forcing. Corollary 3.16 points to two conceptual advantages of weak forcing over strong forcing: first, it gives a clean "extensional" characterization of forcing, without referring to the syntactic structure of φ; as a consequence, weak forcing respects equivalent formulas, i.e., if φ_1 and φ_2 are equivalent in $\omega[A]$ for all A, and σ is any condition, then $\sigma \Vdash \neg\neg\varphi_1$ iff $\sigma \Vdash \neg\neg\varphi_2$. We'll mention weak forcing particularly in the proof of Lemma 6.8.

Lemmas 3.11, 3.14, and 3.15, when taken together, have strong intuitive and practical appeal. They imply (at least for basic $\mathcal{G}$) that a given generic set G satisfies a given arithmetical property P if and only if all generic sets in some $\mathcal{G}$-condition containing G also satisfy P. In a rough sense, G satisfies P iff P is ensured at some "finite" stage of G's "construction;" if one had to work forever to preserve P (say if P were some kind of coding requirement), then G would simply not satisfy P. This fact alone is useful enough, and it captures the essence of forcing without explicitly mentioning forcing or generic filters at all.

Lemma 3.17. *If $\mathcal{G}$ is basic, G is $\mathcal{G}$-generic, and $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, then*

$$\omega[G] \models \varphi \text{ iff } (\exists \gamma \in \mathcal{G})[G \in \gamma \ \& \ (\forall \mathcal{G}\text{-generic } G' \in \gamma)\omega[G'] \models \varphi].$$

Proof. By Lemmas 3.11, 3.14, and Corollary 3.16. □

We are aware of seven previous notions of forcing in arithmetic, all of which are special cases of our Definition 3.3 above. Cohen [14,15] introduced the notion of forcing in set theory to establish the independence of the continuum hypothesis from ZFC. His ideas were transferred to arithmetic in the guise of finite function forcing by Feferman [16]. Finite function forcing (and the derived notion of Cohen genericity over ω) has been extensively studied in computability theory (see [32] for a slightly dated but very useful survey) and computer science (e.g. [43,11,2] and others). Spector [52] implicitly used forcing with computable trees to construct a minimal Turing degree. Sacks [49] considered forcing with arithmetical and pointed perfect sets.[6] See [45] for a detailed technical development of Feferman's and Sacks's ideas in a context different from that presented in the present paper. Two notions of genericity—similar to each other—were used by Slaman and Woodin [53,54] to get results on definability in the Turing and enumeration degrees. The notion of $\mathcal{SP}$-genericity, defined in Section 5, was used by Fenner et al. [20] to establish the existence of an oracle relative to which the Isomorphism Conjecture holds. $\mathcal{SP}$-genericity is really just a variant of what one might call "exact pair forcing" (à la [35], see Section 5.1), whose definition is calibrated for complexity theory.

Finally, random sets, studied extensively by many people and first used as oracles in complexity theory by Bennett and Gill [9], also fall under our scheme as $\mathcal{R}$-generic sets for a particular notion of genericity $\mathcal{R}$. The fact that randomness may be viewed as a particular form of genericity has been known for some time [51]. The set $\mathcal{R}$ consists of those conditions, all of whose branches have positive Lebesgue measure, i.e., $\rho \in \mathcal{R}$ iff ρ, its 0- and 1-branches, the 0- and 1-branches of its 0- and 1-branches, etc. all have positive measure. More precisely, $\mathcal{R}$ is the largest notion of genericity satisfying: (1) all $\mathcal{R}$-conditions have positive Lebesgue measure, and (2) both the 0- and 1-branches of any $\mathcal{R}$-condition are $\mathcal{R}$-conditions. It can be easily shown that all $\mathcal{R}$-generic sets are arithmetically random,[7] and $\mu(\{R \mid R \text{ is } \mathcal{R}\text{-generic}\}) = 1$. $\mathcal{R}$ is not basic.

3.2. Relativized genericity

All the preceding concepts, definitions, and results can be relativized in a straightforward way. We will need relativized genericity in Section 4 when we discuss the rerelativization technique, starting with Theorem 4.10.

Fix a set $B \subseteq \omega$. Forcing and truth relative to B is defined just as in the unrelativized case, except that we expand both the language $\mathcal{L}_{\mathrm{PA}}[X]$ and the standard model ω of Peano Arithmetic to include B as an extra unary predicate. Consequently we obtain the notion of "$\mathcal{G}$-generic in B," or "$\mathcal{G}$-generic relative to B," or "$\mathcal{G}$-generic with respect to B," et cetera. All the results of this section relativize to B in this manner.

Finally, two unary predicates can be collapsed to one in arithmetic formulas by using the join operator. That is, given any formula φ of $\mathcal{L}_{\mathrm{PA}}[X, Y]$ mentioning the two unary predicates X and Y, we can effectively find a formula ψ of $\mathcal{L}_{\mathrm{PA}}[X]$ such that for all $A, B \subseteq \omega$,

$$\omega[A, B] \models \varphi \text{ iff } \omega[A \oplus B] \models \psi.$$

This comes up first in the proof of Theorem 4.10, where we relativize to an oracle R, then *rerelativize* to another oracle G. This is tantamount to relativizing once to the oracle $R \oplus G$.

[6] A pointed perfect set is a subtree of 2^{ω} that can be computed given any of its branches. For example, all conditions corresponding to partial characteristic functions with computable domains are pointed perfect sets.

[7] A set is *arithmetically random* if it is contained in all (first order) arithmetically definable subclasses of 2^{ω} with Lebesgue measure 1.

3.3. Further remarks

We conclude this section with some technical remarks of possible interest to specialists.

The notions of genericity studied in logic generally require that individual forcing conditions be arithmetically definable (or arithmetically definable in any generic they contain, as in Sacks's pointed perfect forcing). Such a restriction is often useful, because it makes it possible to reason formally about the forcing process in ω (or $\omega[G]$ for pointed perfect forcing). Nevertheless, no such hypothesis is needed for the crucial Lemmas 3.12 and 3.14.

It is also possible to formalize forcing over structures other than ω. Indeed, the first—and historically most important—application of forcing was in the context of set theory. We would very much prefer to formalize forcing over a suitable theory of Σ^* rather than ω, simply because then our generics would be oracles, rather than *encodings* of oracles. While it is clear in principle that such a theory must exist (and in the sequel, we generally describe our forcing conditions as though they were subsets of 2^{Σ^*} rather than 2^ω), there are significant technical difficulties that remain to be surmounted. It is clear that if we view Σ^* as the theory of two successors—much as Peano Arithmetic is the theory of one successor—we can arrive at an appropriate theory. The quickest approach is simply to add countably many new function symbols, one for each primitive recursive function over Σ^*; along with axioms that give the corresponding definitions. The defect of such an approach is that it requires a *countable*, rather than simply a *finite* language. For a logician, the addition of countably many new symbols presents no essential difficulties. For us as computer scientists, it is important that our languages ultimately have finitely based representations, for otherwise they have no conceivable connection to the practice of computing. Much of the attractiveness of the Incompleteness Theorem for Peano Arithmetic (and the metamathematical encoding upon which the Incompleteness Theorem is based) for us comes from the fact that a finite language ($\{0, +, \times, S\}$) suffices. Obviously there is a theory of Σ^* with finite signature that is sufficiently powerful for our purposes—one needs only those symbols necessary to define the Kleene T-predicate—but whether or not we can identify some simple set of functions and relations, with interpretations over Σ^* as natural as the interpretations of $+$ and $\times$ are over ω, remains to be seen.

It may also be helpful to reflect on topological issues in forcing. These issues are clearest in the case of Cohen (finite function) forcing, since the forcing conditions themselves are merely the basic clopen sets of the Cantor topology on 2^ω. Such topological issues are helpful in any case, however, inasmuch as our forcing conditions are necessarily compact. From such a perspective, the generic existence theorem is readily reducible to the well-known theorem that in a Hausdorff space the intersection of any nested sequence of compact sets is nonempty. For Cohen forcing, the Baire category theorem is an even more relevant idea from topology. The set of Cohen generics can serve as the set of primitive elements ("Ur-elements") of the arithmetical comeager sets: the set of Cohen generics is comeager, it is a (proper) subset of every arithmetical comeager set, and in fact is precisely the intersection of all arithmetical comeager sets.[8]

[8] This last fact follows from Cohen conditions being encodable as finite objects and the arithmetic definability of the set $\{\sigma \mid \sigma \Vdash \varphi\}$ for any fixed φ.

The topological properties of Cohen forcing mentioned above can be generalized to basic notions of genericity. For $\mathcal{G}$ to be basic simply means that $\mathcal{G}$ is a basis for a topology $\mathcal{T}$ on $\cup\mathcal{G}$. The essential properties of $\mathcal{G}$ are now reducible to the following two simple facts about $\mathcal{T}$:

- $\langle \cup\mathcal{G}, \mathcal{T}\rangle$ is a Baire space, by essentially the same proof that $\mathcal{G}$-generic sets exist, and
- For any $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$, the set $S_\varphi \overset{\mathrm{df}}{=} \cup\{\sigma \in \mathcal{G} \mid \sigma \Vdash_{\mathcal{G}} \varphi \vee \neg\varphi\}$ is open and dense in $\cup\mathcal{G}$, whence the comeager set
$$\bigcap_{\varphi\in\mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])} S_\varphi$$
consists of exactly the $\mathcal{G}$-generic elements of $\cup\mathcal{G}$.

In the case of Cohen forcing (following section), $\mathcal{T}$ is the Cantor topology.

Having laid out the connections with topology, we must caution that there is more going on here than simple compactness. We have phrased forcing and genericity in terms of a unary relation X over ω. There is nothing about the idea of forcing, however, that restricts us to unary relations. The adaptation of forcing to relations of higher arity, or even to *functions* presents no interesting difficulties. The point is that even though 2^ω (or even 2^{ω^n}) is compact, ω^ω is not. The existence theorem for generics still goes through, albeit by a simple tree pruning argument that does not seem to have an equally simple topological expression.

We also note that the extension of forcing and genericity from an individual relation or function symbol to a countable set of relations and/or functions symbols is also routine. Although none of the particular examples has survived to this paper, an earlier proof of Theorem 7.1 relied on the simultaneous forcing of a relation (oracle) and function (advice function).

4. Cohen genericity

We will first consider finite function forcing, i.e., Cohen forcing, where the conditions are all the partial characteristic functions on ω with finite domain. The resulting notion of genericity is clearly basic (see Definition 3.10), and any Cohen generic set G is built by exactly one generic filter: $\{\gamma \mid \gamma \prec G\}$ (see Definition 3.6). In this section, "generic" always means Cohen generic, and "forces" is forcing with finite functions. We assume the usual Cantor topology on 2^ω, of which the finite functions form a basis. We identify in the obvious way (binary) strings (elements of Σ^*) with finite characteristic functions whose domains are initial segments of ω. The following lemma is well known (see [32]):

Lemma 4.1. *A set G is Cohen generic (in our sense) if and only if G meets every dense set of strings that is (coded as) an arithmetically definable element of* 2^ω.

Definition 4.2. Let $\varphi_0(X;x), \varphi_1(X;x), \ldots$ be a sequence of formulas in $\mathcal{L}_{\mathrm{PA}}[X]$, each with at most x free. For all $A \subseteq \omega$ and $i \in \omega$, let

$$L_i(A) = \{x \mid \omega[A] \models \varphi_i(X;x)\},$$

and let $\mathcal{C}^A = \{L_i(A) \mid i \in \omega\}$. Let $\mathcal{C}$ be the operator that takes A to $\mathcal{C}^A$. We call $\mathcal{C}$ a *relativizable complexity class* if

1. for all i, the operator $\lambda X.L_i(X)$ is continuous, i.e., for each $x \in \omega$, and each $A \subseteq \omega$, $L_i(A)(x)$ depends on only finitely much of A, and
2. for all $A, B \subseteq \omega$, if $A \triangle B$ is finite then $\mathcal{C}^A = \mathcal{C}^B = \mathcal{C}^{A \oplus \emptyset} = \mathcal{C}^{\emptyset \oplus A}$.

Most relativizable machine-based complexity classes studied fall under this definition. For example, to get $\mathbf{NP}^A$, we let $\varphi_i(A;x)$ say "the ith polynomial-time nondeterministic oracle TM (NOTM) with oracle A accepts x." To get $\mathbf{NP}^A \cap \text{co-}\mathbf{NP}^A$, we let $\varphi_{\langle i,j\rangle}(A;x)$ say "with oracle A, the ith ptime NOTM accepts x, and for all $y < x$, the ith ptime NOTM accepts y iff the jth ptime NOTM rejects y." In this latter case, either $L_{\langle i,j\rangle}(A)$ is the language accepted by the ith machine (when the ith and jth machines accept complementary languages), or $L_{\langle i,j\rangle}(A)$ is finite. The classes **UP**, **BPP**, etc. can be captured in a similar way. Function classes such as **FP** are also captured under this definition: by identifying a function f with its graph $\{\langle x,y\rangle \mid f(x) = y\}$, we may treat a function class as a special type of language class.

The following technical lemma extends a result of Blum and Impagliazzo [11]. It says that a generic oracle helps in computing a language only if the language is nonarithmetic. The proof is a routine use of Lemma 3.17.

Lemma 4.3. *Let $\mathcal{C}$ be a relativizable complexity class by the definition above, let G be a generic set, and let L be an arbitrary arithmetically definable language. If $L \in \mathcal{C}^G$, then $L \in \mathcal{C}^{\emptyset}$.*

Proof. Let L be arithmetic, defined by some $\varphi_L \in \mathcal{L}_{\mathrm{PA}}$, and let $L = L_i(G)$ for some i. In what follows, γ ranges over all finite functions. We have

$$\begin{aligned}
&L = L_i(G)\\
&\quad\Rightarrow (\exists\gamma)(\forall \text{generic } G' \succ \gamma)L = L_i(G') \qquad (\text{Lemma 3.17 with } \varphi = \forall x[\varphi_L(x) \leftrightarrow \varphi_i(X;x)])\\
&\quad\Rightarrow (\exists\gamma)(\forall A \succ \gamma)L = L_i(A) \qquad (\text{Lemma 3.12 and } L_i \text{ is continuous; see below})\\
&\quad\Rightarrow (\exists\gamma)L = L_i(\gamma^{-1}(1)) \qquad (\text{because } \gamma \prec \gamma^{-1}(1))\\
&\quad\Rightarrow (\exists\gamma)L \in \mathcal{C}^{\gamma^{-1}(1)}\\
&\quad\Rightarrow L \in \mathcal{C}^{\emptyset}(\gamma^{-1}(1) \text{ is finite}).
\end{aligned}$$

The second implication above holds because the Cohen generics in $S = \{A \mid A \succ \gamma\}$ form a dense subset of S with respect to the Cantor topology. Since the function L_i is continuous and constant on a dense subset of S, it must be constant on all of S. (This fact is crucial, because we need to know $L_i(A)$ for nongeneric A.) □

It is straightforward to relativize Lemma 4.3 to any oracle B.

Lemma 4.4. *Let $\mathcal{C}$ be a relativizable complexity class, let $B \subseteq \omega$ be arbitrary, and let G be generic in B. Suppose L is a language arithmetic in B. If $L \in \mathcal{C}^{B \oplus G}$, then $L \in \mathcal{C}^B$.*

From Lemma 4.3, we can easily derive a number of simple, useful results.

Lemma 4.5. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be relativizable complexity classes. Suppose that for any set $A \subseteq \omega$ we have $\mathcal{C}_1^{\emptyset} \subseteq \mathcal{C}_1^A$. Then for any generic G, $\mathcal{C}_1^{\emptyset} - \mathcal{C}_2^{\emptyset} \subseteq \mathcal{C}_1^G - \mathcal{C}_2^G$.*

Proof. Suppose $L \in \mathcal{C}_1^{\emptyset} - \mathcal{C}_2^{\emptyset}$. Then L is arithmetic and $L \in \mathcal{C}_1^G$. But if $L \in \mathcal{C}_2^G$ then by Lemma 4.3, $L \in \mathcal{C}_2^{\emptyset}$, contradicting the assumption. Thus $L \in \mathcal{C}_1^G - \mathcal{C}_2^G$. $\square$

Corollary 4.6. *If $\mathcal{C}_1^{\emptyset} \neq \mathcal{C}_2^{\emptyset}$, then for any generic G, $\mathcal{C}_1^G \neq \mathcal{C}_2^G$.*

Corollary 4.7. *If $\mathbf{P}^G = (\mathbf{NP} \cap \mathbf{coNP})^G$ for generic G, then $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ (relative to $\emptyset$). Similarly, if $\mathbf{P}^G = \mathbf{UP}^G$ for generic G, then $\mathbf{P} = \mathbf{UP}$.*

Let $\mathcal{C}$ be a class and let A and B be disjoint sets. A *$\mathcal{C}$-separator* for A and B is a set $S \in \mathcal{C}$ such that $A \subseteq S \subseteq \overline{B}$. We say that A and B are *$\mathcal{C}$-inseparable* if they have no $\mathcal{C}$-separator.

Proposition 4.8. *If G is generic and $\mathbf{NP}^G$ has no $\mathbf{P}^G$-inseparable sets, then $\mathbf{NP}$ has no $\mathbf{P}$-inseparable sets.*

Proof. Assume the hypothesis, and let L_1 and L_2 be disjoint $\mathbf{NP}$ languages. Then $L_1, L_2 \in \mathbf{NP}^G$, and we have a language $L \in \mathbf{P}^G$ such that $L_1 \subseteq L \subseteq \overline{L_2}$. Let M be a deterministic polynomial time OTM such that $L = L(M^G)$. Since G is generic and L_1 and L_2 are arithmetic, there is a finite function $\gamma \prec G$ such that

$$\gamma \Vdash L_1 \subseteq L(M^G) \subseteq \overline{L_2}.$$

This implies that for no extension $\delta \succeq \gamma$ and for no x do we have

$$(x \in L_1 \ \&\ M^{\delta}(x) \downarrow = 0) \text{ or } (x \in L_2 \ \&\ M^{\delta}\ (x) \downarrow = 1).$$

Thus, for any $A \succ \gamma$, we have $L_1 \subseteq L(M^A) \subseteq \overline{L_2}$. Taking A to be the finite set $\gamma^{-1}(1)$ makes $L(M^A)$ a $\mathbf{P}$-separator for L_1 and L_2. $\square$

Corollary 4.6 leads to a most intriguing technique: rerelativizing by a generic oracle. The rest of this section illustrates the technique. For background on the notions of “honest,” “paddable,” “1-li degrees,” etc., see for example Kurtz et al. [34] or [33].

Definition 4.9 [34]. A one-to-one and honest function $f \in \mathbf{FP}$ is *annihilating* if every $\mathbf{P}$ subset of range(f) is sparse; f is *scrambling* if range(f) does not contain a paddable set.

If scrambling functions exist, then the complete 1-li degree of $\mathbf{NP}$ does not collapse to a polynomial-time isomorphism type, and hence the Isomorphism Conjecture fails [34]. Annihilating functions exist relative to a random oracle [34], and it is easy to see that every annihilating function is a scrambling function, which in turn is a one-way function. We can show, however, that the existence of one-way functions does not necessarily entail the existence of annihilating functions.

Theorem 4.10. *There is an oracle relative to which one-way functions exist, but annihilating functions do not.*

To prove Theorem 4.10, we use the following lemma, whose proof is a modification of an earlier proof showing that the Isomorphism Conjecture fails relative to a generic oracle [38].

Lemma 4.11. *If G is generic, then there are no annihilating functions relative to G.*

Proof. Fix a deterministic ptime oracle transducer T, and for any oracle A, let $f^A \in \mathbf{FP}^A$ be the function computed by T^A. Suppose f^G is one-to-one and honest. We show that f^G cannot be annihilating relative to G. The idea is that G will code infinitely often the pre-images under f^G of many elements in its range at a given length. This coding allows us to recognize a nonsparse subset of range(f^G).

It is clear that range(f^A) cannot be sparse for any oracle A, i.e., for every polynomial $q(n)$, there are infinitely many n such that $|\text{range}(f^A) \cap \Sigma^n| > q(n)$. Let $p(n)$ be a nondecreasing polynomial bounding both the running time and honesty condition of f^G. Consider the following decision procedure P^G, which runs in polynomial time relative to G:
On input y: for all $i \leqslant p(|y|)$, let

$$x_i \overset{\text{df}}{=} G(y10^{p(i)+1})G(y10^{p(i)+2}) \cdots G(y10^{p(i)+i}).$$

If $f^G(x_i) = y$ for some x_i then accept, else reject.

Note that $|x_i| = i$ for all i. Clearly, $L(P^G) \subseteq \text{range}(f^G)$. It remains to show that $L(P^G)$ is not sparse. Let $q(n)$ be a polynomial, and let S_q be the set of strings σ with the following property: there exists a length n such that, for at least $q(n) + 1$ many y of length n, $y \in \text{range}(f^\sigma)$ with pre-image $x = \sigma(y10^{p(|x|)+1})\sigma(y10^{p(|x|)+2}) \cdots \sigma(y10^{p(|x|)+|x|})$. In other words, σ codes the pre-image x of each y in just the right spot for P to find, above the use of the computation of $f^\sigma(x)$.

The set S_q is clearly arithmetically definable for all polynomials q. It is also not hard to see that S_q is dense when viewed as a set of Cohen conditions: we can extend any finite function σ to a string $\tau \in S_q$ because there are infinitely many lengths n with more than $q(n)$ elements of length n in the range of f^A for any oracle A extending σ. We simply take that portion of A bounding the use of these computations of range elements, then extend further by coding all the pre-images, which does not disturb the computations. Since S_q is definable and dense, G must meet S_q by Lemma 4.1. Thus, for every q there is a length n such that $|L(P^G) \cap \Sigma^n| > q(n)$, and so range$(f^G)$ contains a nonsparse subset in $\mathbf{P}^G$. □

Proof of Theorem 4.10. Let R be random, and let G be (Cohen) generic with respect to R (see Section 3.2). We show that $R \oplus G$ satisfies the theorem. Let f be an annihilating function relative to R [34].[9] As $\mathbf{FP}^R \subseteq \mathbf{FP}^{R\oplus G}$, f is in $\mathbf{FP}^{R\oplus G}$. On the other hand, f^{-1} is arithmetic in R, but is not in $\mathbf{FP}^R$ since f is one-way relative to R, therefore f^{-1} cannot be in $\mathbf{FP}^{R\oplus G}$ by Lemma 4.4 with $B = R$, and so f is a one-way function relative to $R \oplus G$.

Now Lemma 4.11 clearly relativizes. Relativizing it to R, we obtain that there can be no annihilating functions in $\mathbf{FP}^{R\oplus G}$, as desired. □

Remark. Part of Theorem 4.10 can also be proved with language classes: the existence of one-way functions relative to R is equivalent to $\mathbf{P}^R \neq \mathbf{UP}^R$, by relativizing results in [27]. Relativizing Corollary 4.6 to R gives us $\mathbf{P}^{R\oplus G} \neq \mathbf{UP}^{R\oplus G}$, hence there exist one-way functions relative to $R \oplus G$ [27].

[9] The fact that annihilating functions do exist relative to R is, strangely enough, not relevant to the proof.

Much of structural complexity, including this work, grew out of the Berman–Hartmanis Isomorphism Conjecture. The rerelativization technique can be used to give several novel failures of this conjecture.

The following theorem was proven by Hartmanis and Hemachandra [28] to refute a conjecture by Kurtz et al. that the Isomorphism Conjecture might be equivalent to the nonexistence of one-way functions. Their original proof combined two difficult constructions: Kurtz's original proof of an oracle relative to which the Isomorphism Conjecture fails, with Rackoff's construction of an oracle relative to which $\mathbf{P} = \mathbf{UP}$.

Theorem 4.12 [28]. *There is an oracle relative to which the Isomorphism Conjecture fails, but relative to which there are no one-way functions. Relative to this oracle, the complete 1-li degree of* **NP** *is an isomorphism type.*

Proof. Consider $A = B \oplus G$ for **PSPACE**-complete B and generic G. The Isomorphism Conjecture fails relative to A, because G is generic with respect to B (Kurtz's result [38] relative to B). On the other hand, Blum and Impagliazzo [11] showed that if $\mathbf{P} = \mathbf{NP}$, then $\mathbf{P}^G = \mathbf{UP}^G$. Since $\mathbf{P}^B = \mathbf{NP}^B$, rerelativizing by G yields $\mathbf{P}^A = \mathbf{UP}^A$. By Grollmann and Selman [27], we see that there can be no one-way functions relative to A. Our final comment in the theorem is justified by Berman and Hartmanis's theorem [10] that if two sets are equivalent by one-one, length-increasing, invertible functions, then they must be isomorphic. □

This oracle A collapses several other classes to **P** simultaneously. See Section 6.5.

A more recent result of Rogers now gives us oracles for all four possible combinations of truth values for the two statements: (1) the Isomorphism Conjecture holds; (2) one-way functions exist [28,34,20,48].

Finally we know of two degree-theoretic ways for the Isomorphism Conjecture to fail: the complete m-degree can fail to be a 1-degree, as relative to a generic oracle [38]; or the complete 1-li degree can fail to be an isomorphism type, as relative to a random oracle [34]. It is interesting to note that we can combine these failures.

Theorem 4.13. *There is an oracle relative to which both the complete m-degree of* **NP** *fails to be a 1-degree, and the complete 1-li degree fails to be an isomorphism type.*

Sketch. Consider $R \oplus G$ as in Theorem 4.10. Since G was chosen to be generic with respect to R, we know that the complete m-degree of **NP** is not a 1-degree.

Now, it would be tempting to get annihilating functions by arguing that R must be random relative to G, but this cannot be the case. As we've already seen, there are no annihilating functions relative to $R \oplus G$, so such an argument must fail. On the other hand, if f is annihilating with respect to R, it is easily seen that f must be *scrambling* with respect to $R \oplus G$ (essentially because G is infinitely often empty for arbitrarily computably long stretches, and so f "looks like" an annihilating function for enough lengths to make it a scrambling function), and therefore the complete 1-li degree of **NP** cannot be an isomorphism type relative to $R \oplus G$.

It is interesting to contrast all these results with the fact that the Isomorphism Conjecture holds relative to $\mathcal{SP}$-generics [20].

5. Symmetric perfect generic sets

The study of symmetric perfect genericity was initiated by Fenner et al. [20], who showed that the Isomorphism Conjecture holds relative to symmetric perfect generic sets. In that paper, a symmetric perfect forcing condition (sp-condition) was associated with some increasing polynomial p and was defined to be a partial characteristic function on Σ^* whose domain consisted of all strings whose lengths were not in the set $\{a_0 < a_1 < a_2 < \cdots\}$, where $a_0 = 2$ and $a_{i+1} = p(a_i)$ for $i \geqslant 0$. The definition below is different and simpler than the one given there, and is a slight variant of a definition suggested by Böttcher [12]. It is easy to show, however, that the new definition suffices for all our results regarding $\mathcal{SP}$-generics, both here and in the previous paper [20].

Definition 5.1. Let c be a positive integer. Define $A_c = \{2^{2^{cn}} | n \geqslant 1\}$. A *c-condition* is a partial characteristic function $\tau : \Sigma^* \to \{0, 1\}$ such that

$$\mathrm{dom}(\tau) = \overline{\bigcup_{m \in A_c} \Sigma^m}.$$

In other words $\tau(x)$ is undefined for all x such that $|x| = 2^{2^{cn}}$ for some $n \in \omega$ with $n \geqslant 1$, and $\tau(x)$ is defined otherwise. A condition τ is a *symmetric perfect forcing condition ($\mathcal{SP}$-condition)* if τ is a c-condition for some $c \geqslant 1$. We let $\mathcal{SP}$ denote the class of all $\mathcal{SP}$-conditions.

Remark. By our present definition, $\mathcal{SP}$ is basic (which was not true with the old definition) and covers 2^ω. Specifically, if σ_1 and σ_2 are compatible c_1- and c_2-conditions respectively, then their least common extension (their intersection as perfect sets) is an $\mathrm{lcm}(c_1, c_2)$-condition. The resulting topology (cf. Section 3.3) is homeomorphic to the product of ω many copies of $\langle 2^\omega, \mathcal{D}\rangle$, where $\mathcal{D} = \mathcal{P}(2^\omega)$ is the discrete topology on 2^ω. □

In keeping with our current definition and naming conventions, we will call the resulting generic sets $\mathcal{SP}$-generic rather than sp-generic as in the older definition [20].

Unlike Cohen forcing conditions, $\mathcal{SP}$-conditions cannot be coded by finite objects. Further, $\mathcal{SP}$-generics are in fact very different from ordinary (Cohen) generics. For example, Lemma 4.3 shows that relativizing to Cohen generic oracles adds only nonarithmetic languages to the standard complexity classes. Relative to $\mathcal{SP}$-generics, the situation is almost the complete opposite:

Lemma 5.2 [20]. *For any language L and any $\mathcal{SP}$-condition σ, there is an $\mathcal{SP}$-condition τ extending σ such that $L \in \mathbf{P}^G$ for all $G \in \tau$.*

Of course, Lemma 5.2 does not imply that there is an $\mathcal{SP}$-generic set G such that for every set L, L is reducible to G. For example, the halting problem relative to a $\mathcal{SP}$-generic set G cannot be reduced to G. Lemma 5.2 only implies that those L such that the predicate "$L \in \mathbf{P}^X$" is expressible in $\mathcal{L}_{\mathrm{PA}}[X]$ are encoded into $\mathcal{SP}$-generics. We have:

Corollary 5.3. *If L is arithmetical and G is $\mathcal{SP}$-generic, then $L \in \mathbf{P}^G$.*

We now summarize some of the complexity theoretic facts known to hold relative to $\mathcal{SP}$-generic sets.

Proposition 5.4. *If G is any $\mathcal{SP}$-generic oracle, the following are true relative to G:*
1. *Any two* **NP***-complete sets (under Karp reductions, i.e., ptime m-reductions) are ptime isomorphic (the Berman–Hartmanis Isomorphism Conjecture) [20].*
2. **PH** *(the polynomial-time hierarchy) is infinite.*
3. $\mathbf{P} = \mathbf{UP} = \mathbf{FewP} = \mathbf{NP} \cap \text{co-}\mathbf{NP}$ [20].

We note here that subsequent to the original Isomorphism oracle result [20], Beigel, Buhrman, and Fortnow constructed an oracle making the Isomorphism Conjecture hold using an entirely different construction employing coding via polynomials [5]. Relative to their oracle, $\mathbf{NP} = \mathbf{EXP}$ and $\mathbf{P} = \mathbf{UP} = \oplus\mathbf{P}$ (see Definition 2.3 for a definition of $\oplus\mathbf{P}$). It is known that the first two identities together relativizably imply the Isomorphism Conjecture [29].

We will show collapses to **P** of a variety of other complexity classes in Section 6. In Section 7.3 we show that these collapses are optimal in some sense: we cannot stratify these collapses in time classes smaller than **P**. The techniques we use there will quickly give us another oracle where none of these classes collapse even to **P** (Section 7.3.1).

5.1. Exact pair forcing

This section will be of interest primarily to computability theorists. It describes the type of computability theoretic forcing that $\mathcal{SP}$-generic sets are capable of.

A pair $\mathbf{a}, \mathbf{b}$ of Turing degrees is an *exact pair* if $\mathbf{a} \cap \mathbf{b}$ does not exist, that is, there is an infinite, strictly increasing sequence of degrees $\mathbf{c}_0 < \mathbf{c}_1 < \cdots$ below both $\mathbf{a}$ and $\mathbf{b}$ such that, for any $\mathbf{d}$ with $\mathbf{d} \leqslant \mathbf{a}$ and $\mathbf{d} \leqslant \mathbf{b}$, there is an n such that $\mathbf{d} \leqslant \mathbf{c}_n$.

Exact pairs were first constructed [35,39,52] to show that the Turing degrees do not form a lattice (see [46, V.4] for example). Their construction can be easily cast as a forcing argument with $\mathcal{SP}$ as the notion of genericity.

Definition 5.5. For any set $A \subseteq \Sigma^*$ and $i \in \{0, 1\}$, define $A^{[i]} \subseteq \Sigma^*$ by $A^{[i]}(x) = A(xi)$. We extend the notation to partial characteristic functions in the obvious way.

We say that a class $Y \subseteq 2^\omega$ is an *arithmetically closed ideal* if (i) Y is closed under finite joins, and (ii) for all $A \in Y$ and B arithmetical in A we have $B \in Y$.

Let $\leqslant_r$ be some type of reducibility, and let $A \subseteq \omega$. We use $\leqslant_r(A)$ to denote the lower $\leqslant_r$-cone of A, that is, $\leqslant_r(A) = \{B | B \leqslant_r A\}$.

We have the following connection between $\mathcal{SP}$-generic sets and arithmetically closed ideals. It shows that the two "halves" of $\mathcal{SP}$-generic sets form exact pairs in a very strong sense: the intersections of the lower $\leqslant_\mathrm{T}$-cones of the two halves of $\mathcal{SP}$-generic sets are arithmetically closed ideals.

Theorem 5.6. *If G* is *$\mathcal{SP}$-generic, then $\leqslant_\mathrm{T}(G^{[0]}) \cap \leqslant_\mathrm{T}(G^{[1]})$ is an arithmetically closed ideal.*

Proof. Let G be $\mathcal{SP}$-generic and let $I = \leqslant_{\mathrm{T}}(G^{[0]}) \cap \leqslant_{\mathrm{T}}(G^{[1]})$. Clearly, I is closed under finite joins. Let $A \in I$ be arbitrary, and fix e_0, e_1 such that

$$A = \{e_0\}^{G^{[0]}} = \{e_1\}^{G^{[1]}}.$$

It suffices to show that $A' \in I$, where $A' = \{e \mid \{e\}^A(e)\downarrow\}$ is the Turing jump of A. Let φ be a sentence in $\mathcal{L}_{\mathrm{PA}}[X]$ expressing that $\{e_0\}^{X^{[0]}} = \{e_1\}^{X^{[1]}}$ and that both are total characteristic functions. Then $\omega[G] \models \varphi$, and so by Lemma 3.14, G extends some $\mathcal{SP}$-condition σ with $\sigma \Vdash_{\mathcal{SP}} \varphi$. By Lemma 3.15, every $\mathcal{SP}$-generic extending σ satisfies φ. It follows that outputs of $\{e_0\}^{G^{[0]}}$ or $\{e_1\}^{G^{[1]}}$ cannot vary depending on queries made outside of $\mathrm{dom}(\sigma^{[0]})$ or $\mathrm{dom}(\sigma^{[1]})$, respectively, since otherwise σ extends to a τ that preserves an inequality between the two functions. Thus A $(= \{e_0\}^{G^{[0]}} = \{e_1\}^{G^{[1]}})$ is computable in σ. (This is the well-known trick to building an exact pair.) Let $e \in \omega$ be such that $A = \{e\}^\sigma$ and $\{e\}$ makes all its oracle queries in $\mathrm{dom}(\sigma)$. Let $\xi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$ say "$\{e\}^X$ is total and its jump is computable both in $X^{[0]}$ and in $X^{[1]}$." We claim that $\omega[G] \models \xi$ and hence $A' \in I$. Suppose $\omega[G] \models \neg\xi$. Then by Lemma 3.17 there is an $\mathcal{SP}$-condition τ extended by G such that all $\mathcal{SP}$-generic sets extending τ satisfy $\neg\xi$. Let ρ be a common extension of σ and τ. We can clearly extend ρ to another condition π for which $\pi^{[0]}$ and $\pi^{[1]}$ each code A'. But then for any $\mathcal{SP}$-generic H extending π we have $A = \{e\}^H$ as well as $A' \leqslant_{\mathrm{T}} H^{[0]}$ and $A' \leqslant_{\mathrm{T}} H^{[1]}$. Thus $\omega[H] \models \xi$, contradicting our choice of τ. □

A partial converse to Theorem 5.6 holds. If $Y = \{A_0, A_1, \ldots\}$ is any countable, arithmetically closed ideal, then we can build a set G such that $Y = \leqslant_{\mathrm{T}}(G^{[0]}) \cap \leqslant_{\mathrm{T}}(G^{[1]})$ and G extends $\mathcal{SP}$-conditions that force $\varphi \vee \neg\varphi$ for all Σ_1 sentences φ. This follows from the fact, provable by induction on the syntax of φ, that if σ is an $\mathcal{SP}$-condition and φ is Σ_1, then either $\sigma \Vdash_{\mathcal{SP}} \neg\varphi$ or there is an extension τ of σ such that $\tau \Vdash_{\mathcal{SP}} \varphi$ and τ is *arithmetic* in σ. Using this fact, we build G by extending $\mathcal{SP}$-conditions in stages: at stage $2i$ we extend a condition γ just enough to code A_i on each side $\gamma^{[0]}$ and $\gamma^{[1]}$; at stage $2i+1$ we extend γ arithmetically to force the ith Σ_1 sentence or its negation. It is not clear whether G can be made to be $\mathcal{SP}$-generic, however, as this construction may not work for sentences that are not Σ_1.

6. Collapsing classes

We have seen that generic oracles do not only separate classes, but in some cases they in fact collapse them. In this section we will extend the results of Blum and Impagliazzo [11], Impagliazzo and Naor [30], and Fenner et al. [20] to show general collapses for classes into **P**. We will show that the following complexity classes equal **P** relative to $\mathcal{SP}$-generic oracles: **UP**, **FewP**, **SPP**, **BPP**, **BQP**,[10] **WPP** and **NP** ∩ co-**NP**. Under certain complexity assumptions we have these collapses for Cohen generics as well.

In this section, we first define a new class **AWPP** that contains several well-known classes, including all those mentioned in the previous paragraph with the possible exception of

[10] The class **BQP** consists of languages accepted by polynomial-time quantum TMs, or equivalently, uniform families of polynomial-size quantum circuits [13,56].

NP $\cap$ co-**NP**. Next, we develop a general framework for proving collapses for generic oracles, involving the notion of certificate complexity. We then apply this framework to show collapsing results for **AWPP** and **NP** $\cap$ co-**NP**, which together imply all the other collapses mentioned above.

For a summary of the known inclusions between classes, and for definitions of Gap**P** and counting classes, see Section 2.

6.1. AWPP

In this section, we introduce the class **AWPP**, and show that both **BPP** and **WPP** are subclasses of **AWPP**. The **BPP** $\subseteq$ **AWPP** inclusion is a simple observation and is not optimal; Fortnow and Rogers [24] have shown that **BQP** $\subseteq$ **AWPP**, and it is well known [13] that **BPP** $\subseteq$ **BQP**.

Definition 6.1. The class **AWPP** consists of all languages L such that, for all polynomials r, there exist $g \in \text{Gap}\mathbf{P}$ and a polynomial q such that

$$x \in L \Rightarrow (1 - 2^{-r(n)})2^{q(n)} \leqslant g(x) \leqslant 2^{q(n)},$$
$$x \notin L \Rightarrow \quad 0 \leqslant g(x) \leqslant 2^{-r(n)}2^{q(n)},$$

where $n = |x|$.

Li [40] essentially showed that **AWPP** is low for the class **PP**, that is, $\mathbf{PP}^L = \mathbf{PP}$ for every $L \in \mathbf{AWPP}$ (for a published proof, see [17]). Fenner [18] also showed that one can "amplify" the ratio $g(x)/2^{q(n)}$ towards zero or one, so that $2^{-r(n)}$ may be replaced with a constant, say $1/3$, wherever it occurs in the definition above, without changing the class.

Since #**P** is a subclass of Gap**P**, it is not difficult to see that **AWPP** contains **BPP**. However, it is not clear from the definition whether **WPP** is contained in this class. To show that it indeed is, we introduce a new class that we shall prove is equal to **AWPP**.

Definition 6.2. The class **AWPP**$'$ consists of all languages L such that for all polynomials r, there exist $f \in \mathbf{FP}$ and $g \in \text{Gap}\mathbf{P}$ such that for all x, $f(x) > 0$ and

$$x \in L \Rightarrow (1 - 2^{-r(n)})f(x) \leqslant g(x) \leqslant f(x),$$
$$x \notin L \Rightarrow \quad 0 \leqslant g(x) \leqslant 2^{-r(n)}f(x),$$

where $n = |x|$.

It is clear that **WPP** is a subclass of **AWPP**$'$.

Lemma 6.3. AWPP = AWPP$'$.

Proof. Since for every polynomial q, $2^{q(n)} \in \mathbf{FP}$, we have **AWPP** $\subseteq$ **AWPP**$'$.

Conversely, let $L \in \mathbf{AWPP}'$ as in Definition 6.2. Given a polynomial r', we need to find a $g' \in \text{Gap}\mathbf{P}$ and a polynomial q such that

$$x \in L \Rightarrow (1 - 2^{-r'(n)})2^{q(n)} \leqslant g'(x) \leqslant 2^{q(n)},$$
$$x \notin L \Rightarrow \quad 0 \leqslant g'(x) \leqslant 2^{-r'(n)}2^{q(n)}.$$

For any $g \in \mathbf{GapP}$, $f \in \mathbf{FP}$, and polynomial r, let $g'(x) = \lfloor 2^{q(n)}/f(x) \rfloor g(x)$ where q is a polynomial that will be specified later. Clearly $g' \in \mathbf{GapP}$. Suppose $0 \leqslant g(x), f(x) \leqslant 2^{p(n)}$ for some polynomial p. If $x \in L$,

$$(1 - 2^{-r(n)})f(x) \leqslant g(x) \leqslant f(x),$$
$$\left\lfloor \frac{2^{q(n)}}{f(x)} \right\rfloor (1 - 2^{-r(n)})f(x) \leqslant g'(x) \leqslant \left\lfloor \frac{2^{q(n)}}{f(x)} \right\rfloor f(x),$$
$$(2^{q(n)} - f(x))(1 - 2^{-r(n)}) \leqslant g'(x) \leqslant 2^{q(n)},$$
$$(2^{q(n)} - 2^{p(n)})(1 - 2^{-r(n)}) \leqslant g'(x) \leqslant 2^{q(n)}.$$

Now setting $r = r' + 1$ and $q = p + r' + 1$, we have $(1 - 2^{-r'})2^q \leqslant (2^q - 2^p)(1 - 2^{-r})$. That is, if $x \in L$,

$$(1 - 2^{-r'(n)})2^{q(n)} \leqslant g'(x) \leqslant 2^{q(n)}.$$

On the other hand, if $x \notin L$,

$$0 \leqslant g(x) \leqslant 2^{-r(n)}f(x),$$
$$0 \leqslant g'(x) \leqslant \left\lfloor \frac{2^{q(n)}}{f(x)} \right\rfloor 2^{-r(n)}f(x) \leqslant 2^{q(n)-r(n)} \leqslant 2^{-r'(n)}2^{q(n)}.$$

This proves that $L \in \mathbf{AWPP}$. □

Corollary 6.4. BPP, WPP $\subseteq$ AWPP.

6.2. General framework

In this section we give a definition of complexity classes—different from Definition 4.2—that emphasizes the accept/reject criterion of the machine.

Definition 6.5. Let $L(M, A)$ be some definable (in $\mathcal{L}_{\mathrm{PA}}[A]$) partial function, mapping time-bounded oracle machines M and languages $A \in 2^{\Sigma^*}$ to languages, such that, for any M the function $\lambda A.L(M, A)$ is a continuous[11] partial mapping from 2^{Σ^*} to 2^{Σ^*} with closed domain. We say that $L(\cdot, \cdot)$ is a *language acceptance criterion*, and we interpret $L(M, A)$ as "the language accepted by M with oracle A," according to this criterion. We will abuse notation and write $L(M^A)$ for $L(M, A)$. If $L(M^A)$ is defined we say that M^A is *proper*. Let $M_1, M_2, \ldots$ be some computable enumeration of time-bounded oracle machines. We say a complexity class $\mathcal{C}$ is *defined by this enumeration and acceptance criterion* if, for any oracle A,

$$\mathcal{C}^A = \{L(M_i^A) \mid M_i^A \text{ is proper}\}.$$

[11] We assume the usual Cantor topology on 2^{Σ^*}.

It is straightforward to check that $\mathcal{C}$, as defined above, is a relativizable complexity class according to Definition 4.2.

Here and in the following section, we will fix a language acceptance criterion $L(\cdot,\cdot)$ and say that $M^A(x)$ *accepts* if $L(M^A)(x) = 1$ and $M^A(x)$ *rejects* if $L(M^A)(x) = 0$.

Note that if M^A is not proper, then there is a finite partial function $\alpha \prec A$ such that $L(M^B)$ is undefined for all $B \succ \alpha$. This is just a restatement of the condition that the domain of $L(M,\cdot)$ be closed.

For example, $\mathbf{BPP}^A$ is defined by an enumeration of probabilistic polynomial-time Turing machines where M_i^A is proper iff for all $x \in \Sigma^*$, either $M_i^A(x)$ accepts with probability at least two-thirds or accepts with probability at most one-third. It is important to note that this acceptance criterion and those with other probability thresholds appearing later are all expressible in $\mathcal{L}_{\mathrm{PA}}[X]$. If M_i^A is not proper and x is an input string that exhibits that fact, then M_i^B is not proper for any oracle B that agrees with A on all strings of length at most $p(|x|)$, where p is the polynomial running time bound on M_i.

6.3. Certificate complexity

In this section, we'll have occasion to regard finite partial characteristic functions from Σ^* to $\{0,1\}$ as inputs to time-bounded computations. Therefore, for any such function ρ, we will define the *size* of ρ to be

$$\mathrm{size}(\rho) = \sum_{y \in \mathrm{dom}(\rho)} (1 + |y|),$$

which reasonably approximates the amount of space needed to represent ρ as a list of ordered pairs.

In this section, "M" will always refer to a Turing machine.

Definition 6.6. Let σ be a partial characteristic function from Σ^* to $\{0,1\}$, not necessarily finite. We say machine M is *categorical over* σ if M^A is proper for all A extending σ. M is *categorical* if M is proper for all A.

Fix a partial function σ, an $x \in \Sigma^*$, an oracle A extending σ, and a machine M categorical over σ. Let $\mathrm{certificate}_A(x)$ be the lexicographically least partial function $\beta \preceq A$ of smallest size such that $L(M^A)(x) = L(M^B)(x)$ for all B extending $\sigma \cup \beta$. By continuity, $\mathrm{dom}(\mathrm{certificate}_A(x))$ must be finite.

Definition 6.7. Assume machine M is categorical over a partial oracle σ. The *certificate complexity* of $M(x)$ over σ is

$$\max_{A \succeq \sigma} (\mathrm{size}(\mathrm{certificate}_A(x))).$$

A class $\mathcal{C}$ defined by machines $\{M_i\}_{i\in\omega}$ and acceptance criterion $L(\cdot,\cdot)$ has *polynomial certificate complexity* if, for any machine M_i there is a polynomial p such that, for all partial functions σ, if M_i is categorical over σ then for all $x \in \Sigma^*$ the certificate complexity of $M_i(x)$ over σ is bounded by $p(|x|)$.

Informally, certificate complexity measures how much of any oracle $A \succeq \sigma$ we must commit (beyond σ itself) to fix the value of M on input x to be that of $L(M^A)(x)$. If M is categorical over σ, then the certificate complexity of $M(x)$ always exists and is finite for any fixed x. This can be seen from purely topological considerations. Clearly, the set $S = \{A \in 2^{\Sigma^*} \mid \sigma \preceq A\}$ is closed and hence compact. Since every certificate is finite, we have that for every $A \in S$, the set $O_A = \{B \in 2^{\Sigma^*} \mid \mathrm{certificate}_A(x) \prec B\}$ is open and contains A. Thus $\{O_A\}_{A \in S}$ is an open covering of S from which we may take a finite subcovering.

The following lemma connects certificate complexity with generic collapses.

Lemma 6.8. *If $\mathcal{C}$ has polynomial certificate complexity, then $\mathcal{C}^G = \mathbf{P}^G$ for any $\mathcal{SP}$-generic G.*

The proof of this lemma uses techniques from [11,20,28,30,38]. Any forcing in the proof is assumed to be $\mathcal{SP}$-forcing.

Proof. Let $R_i \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}[X])$ be the requirement: "Either M_i^X is not proper or $L(M_i^X) \in \mathbf{P}^X$."

Clearly, if an oracle A satisfies R_i for all i, then $\mathbf{P}^A = \mathcal{C}^A$.

Fix i. We will show that the set of $\mathcal{SP}$-conditions that force $\neg\neg R_i$ is dense (see the discussion of weak forcing following Corollary 3.16). This immediately implies that $\gamma \Vdash \neg\neg\neg\neg R_i$ for any $\mathcal{SP}$-condition γ, and so by Corollary 3.16 any $\mathcal{SP}$-generic G will satisfy R_i. We will show this set of $\mathcal{SP}$-conditions is dense by showing how to extend any $\mathcal{SP}$-condition σ to another condition τ that forces $\neg\neg R_i$.

Let $M = M_i$. Suppose σ does not force $\neg\neg$("M^X is proper"). Then by Corollary 3.16, M^A is not proper for some A extending σ, and thus there is some $\alpha \prec A$ of finite domain such that for all B extending α, M^B is not proper. Let τ be some $\mathcal{SP}$-condition extending $\sigma \cup \alpha$. (τ is a c-condition for some c such that 2^{2^c} is greater than the length of any string in $\mathrm{dom}(\alpha)$.) Clearly, $\tau \Vdash \neg$("M^X is proper") by Lemma 3.15, and so $\tau \Vdash R_i$.

For the rest of the proof we will assume $\sigma \Vdash \neg\neg$("M^X is proper"). This implies that M is categorical over σ: otherwise, there is a finite α compatible with σ such that M^B is not proper for any $B \succ \sigma \cup \alpha$, and hence M^B is not proper for at least one $\mathcal{SP}$-generic $B \succ \sigma$, which is a contradiction by Corollary 3.16.

Let $q(n)$ be a polynomial bound on the certificate complexity of $M(x)$ over σ' for any $\sigma' \succeq \sigma$ and $|x| = n$. Let α be any finite characteristic function compatible with σ. There are three possibilities for the behavior of M:

1. $M^B(x)$ accepts for every $B \succ \sigma \cup \alpha$,
2. $M^B(x)$ rejects for every $B \succ \sigma \cup \alpha$,
3. Neither (1) nor (2), in which case there are β_0 and β_1, each with size at most $q(n)$ and compatible with $\sigma \cup \alpha$, such that $M^B(x)$ rejects for all $B \succ \sigma \cup \alpha \cup \beta_0$ and $M^B(x)$ accepts for all $B \succ \sigma \cup \alpha \cup \beta_1$.

Define

$$f^\sigma(x, \alpha) = \begin{cases} 1 & \text{if case 1 holds,} \\ 0 & \text{if case 2 holds,} \\ \langle \beta_0, \beta_1 \rangle & \text{if case 3 holds,} \end{cases} \tag{6}$$

where β_0 and β_1 are lexicographically least of smallest size satisfying the description in case 3.

Let $\mathrm{Code}(f^\sigma) = \{\langle x, \alpha, i\rangle \mid \text{the } i\text{th bit of } f^\sigma(x,\alpha) \text{ is one}\}$. By Lemma 5.2 there is an $\mathcal{SP}$-condition τ extending σ such that $\mathrm{Code}(f^\sigma) \in \mathbf{P}^G$ (and thus $f^\sigma \in \mathbf{FP}^G$) for all $\mathcal{SP}$-generic G extending τ.

Now fix any $\mathcal{SP}$-generic G that extends τ. We will show that $L(M^G) \in \mathbf{P}^G$ using the algorithm in Fig. 1, which we hereafter refer to as the Standard Algorithm.

Note that since α is always extended by G, if the Standard Algorithm halts then it will accept if and only if $x \in L(M^G)$. Lemma 6.8 will follow from the following lemma:

Lemma 6.9. *The Standard Algorithm runs in polynomial time relative to f^σ and thus relative to G.*

Proof. Let $n = |x|$. Suppose $M^G(x)$ accepts. Since $M(x)$ has certificate complexity at most $q(n)$ over σ, there must be a $\rho \prec G$ with size at most $q(n)$ such that $M^B(x)$ accepts for any B extending $\sigma \cup \rho$. Consider a single nonhalting step of the algorithm with its corresponding α, β_0 and β_1, just after the latter two are defined. Now β_0 and ρ must be incompatible, for otherwise there would be a single oracle C extending $\sigma \cup \alpha \cup \beta_0 \cup \rho$ such that $M^C(x)$ both accepts and rejects, a manifest contradiction. Thus $\mathrm{dom}(\beta_0) \cap \mathrm{dom}(\rho) \neq \emptyset$, and it is clear by the minimum size of β_0 that $\mathrm{dom}(\beta_0) \cap \mathrm{dom}(\alpha) = \emptyset$. This means that $|\mathrm{dom}(\rho) - \mathrm{dom}(\alpha)|$ must decrease at every step. Since $|\mathrm{dom}(\rho)| \leqslant \mathrm{size}(\rho) \leqslant q(n)$, there can be at most $q(n)$ iterations of the while loop before $\alpha \succeq \rho$ and hence $f^\sigma(x,\alpha) = 1$. Further, $\mathrm{size}(\alpha)$ increases by at most $\mathrm{size}(\beta_0) + \mathrm{size}(\beta_1) \leqslant 2q(n)$ at each step, so α always has polynomial size.

A similar argument works if $M^G(x)$ rejects, with the roles of β_0 and β_1 transposed. □

In terms of forcing, the Standard Algorithm shows that $\tau \Vdash \neg\neg(L(M_i^X) \in \mathbf{P}^X)$. Since weak forcing respects equivalent formulas, we have $\tau \Vdash \neg\neg R_i$. This ends the proof of Lemma 6.8. □

6.4. Collapses for $\mathcal{SP}$-generics

We can use Lemma 6.8 to show that many complexity classes collapse to $\mathbf{P}$ relative to $\mathcal{SP}$-generics. We will start with an easy one:

```
BEGIN  ALGORITHM $M^G(x)$
  $\alpha := \emptyset$;
  WHILE true
    /* Invariant: $\alpha \prec G$ */
    If $f^\sigma(x,\alpha) = 1$ then accept;
    If $f^\sigma(x,\alpha) = 0$ then reject;
    Set $\langle\beta_0,\beta_1\rangle = f^\sigma(x,\alpha)$;
    $\alpha := \alpha \cup (G$ restricted to $\mathrm{dom}(\beta_0) \cup \mathrm{dom}(\beta_1))$;
  ENDWHILE
END  ALGORITHM.
```

Fig. 1. The Standard Algorithm.

Theorem 6.10. *For $\mathcal{SP}$-generic G, $\mathbf{NP}^G \cap \text{co-}\mathbf{NP}^G = \mathbf{P}^G$.*

Proof. $\mathbf{NP} \cap \text{co-}\mathbf{NP}$ is defined by the enumeration $M_1, M_2, \ldots$, where M_i with $i = \langle i_1, i_2 \rangle$ is the pair of nondeterministic oracle Turing machines N_{i_1} and N_{i_2} such that both these machines run in time n^i. The corresponding language acceptance criterion for M_i^A is $L(N_{i_1}^A)$ if $L(N_{i_1}^A) = \overline{L(N_{i_2}^A)}$ and undefined otherwise ($L(\cdot,\cdot)$ is the standard **NP** language acceptance criterion). Fix i and let $M = M_i$. If $M^A(x)$ accepts, then $\text{certificate}_A(x)$ must be no larger than A restricted to the oracle queries along a single accepting computation path of $N_{i_1}^A(x)$. If $M^A(x)$ rejects, then $\sigma_A(x)$ must be no larger than A restricted to the oracle queries along a single accepting computation path of $N_{i_2}^A(x)$. Thus $\mathbf{NP} \cap \text{co-}\mathbf{NP}$ has polynomial certificate complexity, and Theorem 6.10 follows immediately from Lemma 6.8. $\square$

For the remaining collapses we will use the class **AWPP** described in Section 6.1 and a powerful theorem from [44].

For a Boolean function $f(x_1, \ldots, x_N)$ and an input $y \in \{0,1\}^N$ let $S_y \subseteq \{1, \ldots, N\}$ be the lexicographically least set of minimum size such that $f(z) = f(y)$ for all z agreeing with y on the variables with indices in S_y. We define the *certificate complexity of f* as

$$\max_{y \in \{0,1\}^N} |S_y|.$$

Theorem 6.11 (Nisan–Szegedy). *There is a fixed polynomial c such that, if*
1. *$f(x_1, \ldots, x_N)$ is a Boolean function,*
2. *$p(x_1, \ldots, x_N)$ is a polynomial with real coefficients of degree d, and*
3. *for all $\langle x_1, \ldots, x_N \rangle \in \{0,1\}^N$,*

$$|f(x_1, \ldots, x_N) - p(x_1, \ldots, x_N)| \leqslant \frac{1}{3},$$

then f has certificate complexity bounded by $c(d)$.

Let M be a nondeterministic oracle Turing machine running in time n^k for all inputs of length n. Clearly the length of each oracle query is also bounded by n^k. Let $y = \langle y_\epsilon, \ldots, y_{1^{n^k}} \rangle$ be a (0,1)-vector indexed by all binary strings of lengths up through n^k. Such a y certainly defines a partial oracle in an obvious way. Thus for all x of length n we define $g^y(x) = \text{gap}_{M^y}(x)$, the finite $\text{Gap}\mathbf{P}^y$ function defined by M on inputs of length n with "oracle" y (see Definition 2.2). On the other hand, we may fix x and define a function $g_x : \{0,1\}^{2^{n^k+1}-1} \to \mathbb{Z}$ so that $g_x(y) = g^y(x)$.

Lemma 6.12. *For every $\text{Gap}\mathbf{P}^y$ function g defined by machine M running in time n^k and string x of length n, there is a polynomial h with integer coefficients of degree bounded by n^k such that $g_x(y) = h(y)$ for all $y = \langle y_\epsilon, \ldots, y_{1^{n^k}} \rangle \in \{0,1\}^{2^{n^k+1}-1}$.*

Proof. Modify M so that it guesses the query answers before its computation and then verifies its answers at the end. For any $1 \leqslant i \leqslant n^k$ and any path p of M, let $t_{i,p}$ be a polynomial over the variables y_w for $w \in \{\epsilon, \ldots, 1^{n^k}\}$ defined as

$$t_{i,p} = \begin{cases} y_w & \text{if } w \text{ is the } i\text{th query along } p \text{ and is guessed to be in the oracle,} \\ 1 - y_w & \text{if } w \text{ is the } i\text{th query along } p \text{ and is guessed to not be in the oracle,} \\ 1 & \text{if } M \text{ uses fewer than } i \text{ queries on path } p. \end{cases}$$

We then define the polynomial t_p over the variables y_w as follows:

$$t_p = \begin{cases} \prod_{1 \leqslant i \leqslant n^k} t_{i,p} & \text{if } M(x) \text{ accepts along path } p; \\ -\prod_{1 \leqslant i \leqslant n^k} t_{i,p} & \text{if } M(x) \text{ rejects along path } p. \end{cases}$$

It is easy to verify that $h = \sum_p t_p$ fulfills the requirements of Lemma 6.12. □

Theorem 6.13. **AWPP** *has polynomial certificate complexity.*

Proof. Let M be a nondeterministic oracle Turing machine that runs in time n^k and let g be the Gap$\mathbf{P}^y$ function defined by that machine. For any oracle A, let $L(M^A)$ be defined according to the conditions of Definition 6.1 when $r(n) = 2$, provided those conditions are fulfilled for every x, that is, provided there is a polynomial q such that either $3/4 \leqslant g(x)/2^{q(n)} \leqslant 1$ or $0 \leqslant g(x)/2^{q(n)} \leqslant 1/4$, where $n = |x|$.

Suppose M is categorical over a partial function σ. Fix an input x of length n and let $s = |\Sigma^{\leqslant n^k}| = 2^{n^k+1} - 1$. For all $y = \langle y_\epsilon, \ldots, y_{1^{n^k}} \rangle \in \{0,1\}^s$ extending σ (i.e., $\forall w \in \mathrm{dom}(\sigma)$, $y_w = \sigma(w)$ is fixed), define the function $f(y)$ to be 1 when $M^y(x)$ accepts and 0 when $M^y(x)$ rejects. Let $q(n)$ be from Definition 6.1 and $h(y)$ be from Lemma 6.12. Define $p(y)$, for $y = \langle y_\epsilon, \ldots, y_{1^{n^k}} \rangle \in \{0,1\}^s$ extending σ, as $h(y)/2^{q(n)}$.

The degree of $p(y)$ is bounded by n^k by Lemma 6.12, and by the definitions of $f(y)$, $p(y)$, and **AWPP**, we have $|f(y) - p(y)| \leqslant 1/3$. So we can apply Theorem 6.11 with $N = |\Sigma^{\leqslant n^k} - \mathrm{dom}\,\sigma|$ to get the certificate complexity of f to be polynomial in n. That is, if we fix a y extending σ, there is a set S_y (of size at most $c(n^k)$) of strings of length less than or equal to n^k such that $f(y) = f(z)$ for all z agreeing with y on S_y. Particularly, let y be induced by any oracle A, i.e., $\forall w, y_w = \chi_A(w)$. We then see that the certificate complexity of $M(x)$ over σ is bounded by $c(n^k)(1 + n^k)$, a polynomial independent of σ; therefore, **AWPP** has polynomial certificate complexity. □

Corollary 6.14. *For $\mathcal{SP}$-generic G,* $\mathbf{AWPP}^G = \mathbf{P}^G$.

Proof. From Theorem 6.13 and Lemma 6.8. □

Corollary 6.15. *For $\mathcal{SP}$-generic G,* $\mathbf{P}^G = \mathbf{UP}^G = \mathbf{FewP}^G = \mathbf{SPP}^G = \mathbf{WPP}^G = \mathbf{BPP}^G = \mathbf{BQP}^G$.

The techniques in this section imply lower bounds on the sizes of quantum circuits computing properties of black-box functions. This idea was developed and refined independently by Beals et al. [4].

6.5. Collapses for Cohen generics

How can we use Section 6.3 to show collapses for Cohen generics? Cohen generics do not allow us to encode a function f as in the proof of Lemma 6.8, but we can still get collapses if we allow a little help or use assumptions. We will give an informal description of the process in this section.

To simplify matters, we will make one additional assumption about the complexity class $\mathcal{C}$ we wish to collapse. Suppose M_i is categorical over a *finite* partial function σ. We will require that we can easily find—given M_i and σ—another machine M_j categorical (over ϵ) such that $L(M_i^A) = L(M_j^A)$ for all A extending σ. This assumption certainly holds for all the classes we discuss below. Note that the first part of the proof of Lemma 6.8 still goes through if we replace $\mathcal{SP}$ forcing with Cohen forcing, that is, if a machine is not categorical over some (finite!) Cohen condition, we can force it not to be proper and hence discard it from consideration. Then by our assumption we need only deal with categorical machines.

Now reconsider the function f^σ defined (Eq. (6)) for the Standard Algorithm in Fig. 1 of the proof of Lemma 6.8, where σ was an $\mathcal{SP}$-condition. For our current effort, we can just as well have σ be a finite (Cohen) condition, and by the discussion above, we can restrict attention to machines M_j that are categorical over ϵ, and thus we may further assume that $\sigma = \epsilon$. In essence, our function does not depend on the particular Cohen generic G we create. For all j, let $F_j(x, \alpha)$ be the function $f^\epsilon(x, \alpha)$ created in the proof of Lemma 6.8 for machine M_j, given that M_j is a categorical machine (if M_j is not categorical, then $F_j(x, \alpha)$ could be anything). We now have the following lemma, similar to Lemma 6.8, but for Cohen generics:

Lemma 6.16. *If $\mathcal{C}$ has polynomial certificate complexity, then $\mathcal{C}^G \subseteq \mathbf{P}^{G\oplus F_1\oplus F_2\oplus F_3\oplus\cdots}$ for all Cohen generic G. In fact, for any categorical M_j, $L(M_j^G) \in \mathbf{P}^{G\oplus F_j}$.*

For a given class $\mathcal{C}^A$ with polynomial certificate complexity, and for a fixed j such that M_j is categorical, how hard is it to compute the corresponding function $F_j(x, \alpha)$, and thus run the Standard Algorithm? Blum and Impagliazzo [11] show that $F_j \in \mathbf{FP}^{\mathbf{NP}}$ for $\mathcal{C}^A$ either $\mathbf{UP}^A$ or $\mathbf{NP}^A \cap \text{co-}\mathbf{NP}^A$. Impagliazzo and Naor [30] show that $F_j \in \mathbf{FP}^{\Sigma_2^p}$ for $\mathcal{C}^A = \mathbf{BPP}^A$. Fenner et al. [20] show that $F_j \in \mathbf{FP}^{\mathbf{NP}}$ for $\mathcal{C}^A = \mathbf{FewP}^A$. In general, we have the following lemma, whose proof is a straightforward adaptation of [30].

Lemma 6.17 (Impagliazzo, Naor). *Under the conditions of Lemma 6.16, F_j is computable in $\mathbf{FP}^{(\Sigma_2^p)^{\mathcal{C}}}$, where $\mathcal{C}$ here stands for the unrelativized complexity class $\mathcal{C}^\emptyset$.*

Sketch. Let $q(n)$ be a polynomial bound on the certificate complexity for $M_j(x)$ over any finite partial oracle and any input x, letting $n = |x|$. For any given finite partial oracle α, a *1-certificate* is a polynomial-size (in n) partial oracle β compatible with α such that $\alpha \cup \beta$ ensures that $M_j(x)$ accepts, and a *0-certificate* is such a β which ensures that $M_j(x)$ rejects.

Fix a finite partial oracle α and x as input to F_j. Given any polynomial-size β compatible with α, we decide whether or not it is a 1-certificate as follows: if β is not a 1-certificate, then there is an oracle extending $\alpha \cup \beta$ that makes $M_j(x)$ reject, hence there is a polynomial-size 0-certificate γ compatible with $\alpha \cup \beta$; on the other hand, if β is a 1-certificate, then there is no extension at all that makes $M_j(x)$ reject. Thus, β is a 1-certificate iff, for all γ compatible with $\alpha \cup \beta$ and with size at most $q(n)$, $M^{\alpha\cup\beta\cup\gamma}(x)$ accepts, where all queries outside $\mathrm{dom}(\alpha \cup \beta \cup \gamma)$ are answered negatively.

This test is in co-$\mathbf{NP}^{\mathcal{C}}$. Similarly, there is a co-$\mathbf{NP}^{\mathcal{C}}$ test for a 0-certificate. We can now find either kind of certificate using a standard prefix search algorithm, asking $\mathbf{NP}$ questions relative to the tests. $\square$

Given Lemma 6.17 it is easy to see that $F_j \in \mathbf{FPSPACE}$ for $\mathcal{C}^A = \mathbf{AWPP}^A$.

Clearly if $F_j \in \mathbf{FP}$ then Lemma 6.8 goes through for Cohen generics. Using the fact that $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{P} = \Sigma_2^p$ we get the following theorem:

Theorem 6.18. *Let G be Cohen generic.*
1. *If* $\mathbf{P} = \mathbf{NP}$ *then* $\mathbf{UP}^G = \mathbf{NP}^G \cap \text{co-}\mathbf{NP}^G = \mathbf{FewP}^G = \mathbf{BPP}^G = \mathbf{P}^G$.
2. *If* $\mathbf{P} = \mathbf{PSPACE}$ *then* $\mathbf{UP}^G = \mathbf{NP}^G \cap \text{co-}\mathbf{NP}^G = \mathbf{FewP}^G = \mathbf{BPP}^G = \mathbf{SPP}^G = \mathbf{AWPP}^G = \mathbf{P}^G$.

Lemma 4.5 tells us that we can't do without at least some assumptions in Theorem 6.18.

The rerelativization technique can be combined with Theorem 6.18 to get Cohen generic oracles to collapse these classes. We use the same oracle as in Theorem 4.12. Let B be some **PSPACE**-complete set. Then $\mathbf{P} = \mathbf{PSPACE}$ relative to B, and since Theorem 6.18 also holds relative to B, rerelativizing with a Cohen generic G gives us all the collapses above relative to $B \oplus G$ with no assumptions. In this and other papers [23,24], when trying to show that an oracle exists for a certain property, we often assume that $\mathbf{P} = \mathbf{PSPACE}$ unrelativized before we define the oracle. If our oracle construction is relativizable (and it always is), then this assumption costs us nothing, since it can be discharged by rerelativization.

7. Separating classes

In this section, we will give some generic oracle separations that are less obvious than those given in previous sections.

7.1. Separating nonuniform classes

Theorem 6.18 shows that generic oracles may collapse classes like **UP** and $\mathbf{NP} \cap \text{co-}\mathbf{NP}$. However, various generic requirements will actually separate these classes from **P** for many different input lengths. Machines witnessing this separation will fail to be categorical on other input lengths. If we had access to nonuniform computation, we could pick out these hard input lengths.

We can use genericity to separate various nonuniform classes in this manner. In particular, we can use the theory of genericity to exhibit the difference in the two different ways of defining nonuniform $\mathbf{NP} \cap \text{co-}\mathbf{NP}$:

Theorem 7.1. *If G is a Cohen generic and* $\mathbf{P} = \mathbf{NP}$, *then* $\mathbf{NP}^G/\mathbf{1} \cap \text{co-}\mathbf{NP}^G/\mathbf{1}$ *is not contained in* $(\mathbf{NP}^G \cap \text{co-}\mathbf{NP}^G)/\mathbf{poly}$.

Corollary 7.2. *There is an oracle A such that* $\mathbf{NP}^A/\mathbf{1} \cap \text{co-}\mathbf{NP}^A/\mathbf{1}$ *is not contained in* $(\mathbf{NP}^A \cap \text{co-}\mathbf{NP}^A)/\mathbf{poly}$.

Proof Sketch of Theorem 7.1. Assume $\mathbf{P} = \mathbf{NP}$. By Theorem 6.18 we have that $\mathbf{NP}^G \cap \text{co-}\mathbf{NP}^G = \mathbf{P}^G$. Thus we need only show that $\mathbf{NP}^G/\mathbf{1} \cap \text{co-}\mathbf{NP}^G/\mathbf{1}$ is not contained in $\mathbf{P}^G/\mathbf{poly}$. For each n, define

$$L_n^G = \{x \mid \exists y |x| = |y| = n \wedge \langle x, y, 0\rangle \in G\}.$$

We say n is *nice* relative to G if for all x of length n, there is a y of length n such that $\langle x, y, 0\rangle \in G$ if and only if there is no y of length n such that $\langle x, y, 1\rangle \in G$.

Let requirement R_i be "There exists an n such that n is nice and L_n^G is not accepted by deterministic Turing machine M_i running in time n^i with *any* advice string of length n^i." A straightforward combinatorical argument shows that the set of Cohen conditions forcing R_i is dense.

Define $L^G = \bigcup_{\{n|n \text{ nice}\}} L_n^G$. Then L^G is in $\mathbf{NP}^G/\mathbf{1} \cap \text{co-}\mathbf{NP}^G/\mathbf{1}$ by letting the one bit of advice $a_n = 1$ iff n is nice. However, L^G is not in $\mathbf{P}^G/\mathbf{poly}$ since each requirement R_i is met. □

7.2. Separating $\mathbf{NP}^{\mathbf{BPP}}$ from $\mathbf{MA}$

We can use similar techniques to prove the following theorem. One interesting aspect of this theorem is that the basic construction consists of specifying a new notion of genericity which ensures that **MA** coding requirements are met. This new notion still fits into our general scheme described in Section 3, that is, **MA**-conditions are (or at least can be identified with) perfect subsets of 2^ω that collectively satisfy Definition 3.3.

Theorem 7.3. *There is an oracle G such that*

$$\mathbf{NP}^{\mathbf{BPP}^G} \not\supseteq \mathbf{MA}^G.$$

Proof. By first relativizing with a **PSPACE**-complete set, we can assume $\mathbf{P} = \mathbf{NP}$ unrelativized (see the last paragraph in Section 6).

Define $L(A)$ as follows:

$$L(A) = \{1^n \mid \exists y \in \Sigma^n, \text{ for most } z \in \Sigma^n, yz \in A\}.$$

We say A is **MA**-*proper* for length n if either there exists a $y \in \Sigma^n$ such that $yz \in A$ for at least two-thirds of the z's in Σ^n, or for all $y \in \Sigma^n$, $yz \in A$ for at most one-third of the z's in Σ^n. We say that A is **MA**-*proper* if A is **MA**-proper for all lengths.

Definition 7.4. An **MA**-*condition* σ is a partial characteristic function with domain $\Sigma^{\leqslant n}$ for some n, such that there is some **MA**-proper A extending σ.

The **MA**-conditions form a notion of genericity that is basic, as in Definition 3.10. In this section, all our conditions will be **MA**-conditions and our forcing will be **MA**-forcing. Clearly, all **MA**-generic oracles are **MA**-proper. We will show that any **MA**-generic G fulfills the following three properties:
1. $L(G) \in \mathbf{MA}^G$,
2. $L(G) \notin \mathbf{NP}^G$, and
3. $\mathbf{BPP}^G = \mathbf{P}^G$.

If G fulfills these conditions then $\mathbf{NP}^{\mathbf{BPP}^G} = \mathbf{NP}^G$ which does not contain $L(G) \in \mathbf{MA}^G$.

The first property holds by the **MA**-propriety of **MA**-generic oracles.

Let $N_1, \ldots$ be an enumeration of nondeterministic oracle Turing machines with an n^i clock on N_i. Let $P_1, \ldots$ be an enumeration of probabilistic oracle Turing machines with an n^i clock on P_i.

To prove the second property we need to show that for every i, the following set of conditions is dense:

$$S_i = \{\tau \mid \tau \Vdash L(X) \neq L(N_i^X)\}.$$

If S_i is dense, then by definition, the empty string ϵ forces $\neg\neg(L(X) \neq L(N_i^X))$. The second property will then follow from Corollary 3.16.

Fix i. Let σ be a condition. Pick n larger than the length of any string in $\mathrm{dom}(\sigma)$ and also such that $2^n > 3n^i$. If there is an A extending σ such that $N_i^A(1^n)$ accepts, then let γ be A restricted to queries made on an accepting path of $N_i^A(1^n)$. Let $\tau(x)$ be defined as

$$\tau(x) = \begin{cases} \sigma(x) & \text{if } x \in \mathrm{dom}(\sigma) \\ \gamma(x) & \text{if } x \in \mathrm{dom}(\gamma) \\ 0 & \text{if } x \in \Sigma^{\leqslant 2n+n^i} - \mathrm{dom}(\sigma) - \mathrm{dom}(\gamma) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

If there is no such oracle A, then let τ be σ extended by defining $\tau(x) = 1$ for all $x \in \Sigma^{\leqslant 2n} - \mathrm{dom}(\sigma)$.

In either case, τ is a condition and $\tau \Vdash L(X) \neq L(N_i^X)$ by Proposition 3.13, and so τ is in S_i and extends σ. Hence, S_i is dense.

For the third property we show that for every i, the following set of conditions is dense:

$$T_i = \{\tau \mid \tau \Vdash \big(L(P_i^X) \text{ defined} \to L(P_i^X) \in \mathbf{P}^X\big)\}.$$

Recall that for defining **BPP**, P_i^A is proper just in the case that its acceptance probability for any input is either at least $2/3$ or at most $1/3$.

Fix i and let σ be a condition. We will find a condition $\tau \in T_i$ extending σ. Pick n larger than the length of any element of $\mathrm{dom}(\sigma)$ and also such that $2^n >> n^i$. We can assume that σ is defined on exactly the strings of length less than n.

First, suppose that there exists some x with $m = |x|$ and $3i \log m \geqslant n$ and there exists an oracle A that extends σ and is **MA**-proper for all lengths less than $3i \log m$, such that the probability that $P_i^A(x)$ accepts is between $2/5$ and $3/5$. We would now like to extend σ according to A to a τ defined on all strings of length up through m^i, thus preventing P_i^τ from being a proper **BPP** machine. Unfortunately, we can't quite do this, because although A is **MA**-proper for lengths less than $3i \log m$, it may not be proper for lengths between $3i \log m$ and $m^i/2$, and so τ would not be an **MA**-condition. In this case, however, we can tweak τ a little bit so that it becomes an **MA**-condition without changing the acceptance probability of $P_i^\tau(x)$ too much.

For any $y \in \Sigma^*$ let $S_y = \{yz : |y| = |z|\}$.

Lemma 7.5. *For every $t \geqslant 3i \log m$, there exists a y_t* with $|y_t| = t$ *such that the strings in S_{y_t} only appear in less than a $1/(15m^i)$ fraction of the computation paths of $P_i^A(x)$.*

Proof. Fix t in the given range. Let Q_p be the set of queries made on computation path p of $P_i^A(x)$. Note $|Q_p| \leqslant m^i$.

Choose y at random of length t. We have, for any fixed p,

$$\Pr(S_y \cap Q_p \neq \emptyset) \leqslant \frac{m^i}{2^t} \leqslant \frac{m^i}{m^{3i}} < \frac{1}{15m^i}.$$

Of course, if we choose both y and p at random the same inequality holds. Thus there must be some y such that if we choose p at random the same inequality holds. $\square$

Define $\tau(w)$ as

$$\tau(w) = \begin{cases} \sigma(w) & \text{if } |w| < n \\ 1 & \text{if } w \in S_{y_t} \text{ for some } t, 3i \log m \leqslant t \leqslant m^i/2 \\ A(w) & \text{otherwise, if } |w| \leqslant m^i \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Clearly, τ is an **MA**-condition extending σ. Further, τ differs from A at most on strings in S_{y_t} for $n \leqslant t \leqslant m^i/2$, which appear as queries on a combined total of less than 1/15 of the paths of $P_i^A(x)$. Thus $P_i^\tau(x)$ accepts with probability strictly between 1/3 and 2/3, which implies that $\tau \Vdash (L(P_i^X)$ is undefined) by Proposition 3.13, and so $\tau \in T_i$.

Now suppose there is no such x and A, that is, for every x such that $3i \log |x| \geqslant n$, and for every A extending σ that is **MA**-proper for all lengths less than $3i \log |x|$, the probability that $P_i^A(x)$ accepts is either less than 2/5 or greater than 3/5.

Fix any x with $|x| = m$ and such that $\ell = 3i \log m \geqslant n$, and let G be any **MA**-proper oracle extending σ. We will show that $L(P_i^G) \in \mathbf{P}^G$ and thus $\sigma \in T_i$. Let σ' be G restricted to strings of length less than ℓ. (For the moment, we will allow **BPP** machines to have error up to 2/5; this does not affect the rest of the proof.)

Now P_i may not be categorical over σ'; σ' alone does not tell us about the behavior of P_i on inputs bigger than m, so we cannot define the certificate complexity of $P_i(y)$ over σ' for $|y| > m$ in accordance with Definition 6.7. However, we do know that $P_i(x)$ behaves in a **BPP**-proper way for all oracles extending σ', and the results of Impagliazzo and Naor [30] show us that polynomial-sized (in m) certificates for $P_i(x)$ over σ' can be computed in $\mathbf{FP}^{\Sigma_2^p}$ (see the discussion following Lemma 6.16), and hence in **FP**. Therefore, once we compute σ' explicitly (using m^{3i} queries to G), we can run the Standard Algorithm, simulating the computation of $f^{\sigma'}(x, \alpha)$ corresponding to the machine $P_i(x)$ in polynomial time. This approach works for all x with $|x| \geqslant 2^{n/3i}$, where n and i were fixed. This shows that $L(P_i^G) \in \mathbf{P}^G$. This concludes the proof of Theorem 7.3. $\square$

Definition 6.7 can be loosened to accommodate this last point. We could have defined categoricity of M over σ for a particular input x in the expected way. Then, to define the certificate complexity of $M(x)$ over σ, we only require that M be categorical over σ for input x. The function f of Eq. (6) is then defined arbitrarily whenever M is not categorical for x. But when M *is* categorical for x, the Standard Algorithm can be run.

7.3. Separations with $\mathcal{SP}$-generics

Theorem 6.10 and Corollaries 6.14 and 6.15 show dramatic collapses relative to $\mathcal{SP}$-generic oracles. A natural question to ask is whether these collapses are tight with regard to the time hierarchy. For example, $\mathbf{P}^G = \mathbf{UP}^G$ for $\mathcal{SP}$-generic G, but this would also follow from a stronger collapse: $(\forall i)(\exists j)[\mathbf{UTIME}^G(n^i) \subseteq \mathbf{DTIME}^G(n^j)]$. Such stronger collapses occur when the class in question has a complete set. For example, if $\mathbf{P} = \mathbf{NP}$, then the "standard" complete set (cf. [6])

$$K = \{\langle i, x, 1^t\rangle \mid \text{the } i\text{th NTM accepts } x \text{ within time } t\}$$

would be computable in $\mathbf{DTIME}(n^a)$ for some $a \in \omega$, and so for all $b \in \omega$,

$$\mathbf{NTIME}(n^b) \subseteq \mathbf{DTIME}^K(n^b) \subseteq \mathbf{DTIME}(n^{a+b}).$$

The following theorem shows that no such stronger collapse occurs relative to G for any of the classes we have been discussing. Another motivation for the theorem is that the proof easily scales up to obtain a generic oracle that *separates* all these classes from **P**. We'll discuss this briefly in Section 7.3.1.

Theorem 7.6. *For any $\mathcal{SP}$-generic G and any $k \in \omega$,*

$$\mathbf{ULIN}^G \cap \text{co-}\mathbf{ULIN}^G \cap \mathbf{ZPLIN}^G \not\subseteq \mathbf{DTIME}^G(n^k).$$

[Here, **ULIN** stands for unambiguous nondeterministic linear time, and **ZPLIN** stands for zero-error probabilistic linear time.]

Proof. For simplicity, we only prove that $\mathbf{ULIN}^G \cap \text{co-}\mathbf{ULIN}^G \not\subseteq \mathbf{DTIME}^G(n^k)$. Combining this with similar techniques proves the theorem. The basic idea is simple: when building the oracle G, we extend some $\mathcal{SP}$-condition σ to an $\mathcal{SP}$-condition τ by filling in some of the gaps in $\mathrm{dom}(\sigma)$ with hidden witnesses or cowitnesses, in order to put some standard test language L^G out of $\mathbf{DTIME}(n^k)^G$ while maintaining some $\mathbf{ULIN} \cap \text{co-}\mathbf{ULIN}$ promise for L^G. We can make the gaps remaining in $\mathrm{dom}(\tau)$ to be too far apart to interfere with the diagonalizations against $\mathbf{DTIME}(n^k)$ oracle machines.

Unfortunately, the situation is more complicated than this and requires greater care. There are infinitely many lengths where τ is completely undefined, and even though a $\mathbf{DTIME}(n^k)$ machine running on a test input of length n does not have time to make queries in the next bigger gap in $\mathrm{dom}(\tau)$, it can still make queries outside of $\mathrm{dom}(\tau)$ at lengths *shorter* than n. We have no control over how the oracle G is eventually defined at these lengths, and G may perhaps tell the machine where to find witnesses and so defeat the diagonalization. The trick to the construction is to attempt diagonalization on many inputs of length n at once. The deterministic machine may be able to find *some* witnesses, but computing the right answer on *all* test inputs would require more information than can fit in the oracle at the shorter lengths.

Let σ be any c-condition for $c \geqslant 1$, undefined on strings with lengths in A_c (cf. Definition 5.1.) We show how to extend σ to a d-condition τ with $d = 2c$ such that the test language

$$L^G = \{x : 4|x| \in (A_c - A_d) \& (\exists y)[|y| = 3|x| - 1 \ \&\ xy1 \in G]\}$$

is a member of $(\mathbf{ULIN}^G \cap \text{co-}\mathbf{ULIN}^G) - \mathbf{DTIME}(n^k)^G$ for any oracle G extending τ. We can assume without loss of generality that $2^c > k$ (otherwise, we just extend σ in some arbitrary way first), which means that for any element $n \in A_c$, the next bigger element of A_c is $n^{2^c} > n^k$. The extension τ is defined by giving its values on strings with lengths in $A_c - A_d$ as follows: For every $n \in A_c - A_d$, let $m = n/4$ and let $x_1, \ldots, x_{2^m}$ be the lexicographical enumeration of all strings of length m. Choose a Kolmogorov random (relative to σ) string $y = y_1 y_2 \cdots y_{2^m}$ of length $3m2^m$, cut into blocks y_i of length $3m$ each. We let $\tau(x_i y_i) = 1$ for $1 \leqslant i \leqslant 2^m$, and let $\tau(z) = 0$ for all other strings z of length n. Doing the above for all $n \in A_c - A_d$ defines τ. This clearly puts L^G into $\mathbf{ULIN}^G \cap \text{co-}\mathbf{ULIN}^G$ for all G extending τ.

Fix a deterministic oracle machine M running in time n^k. Let n_0 be a sufficiently large element of $A_c - A_d$, and let $m_0 = n_0/4$. Let $x_1, \ldots, x_{2^{m_0}}$ be the lexicographic enumeration of the strings of length m_0, and let $y = y_1 \cdots y_{2^{m_0}}$ be the Kolmogorov random string as described above, for $n = n_0$. The next higher (than n_0) gap in $\mathrm{dom}(\sigma)$ comes at a length above n_0^k, and so does not affect any computation of $M(x_i)$. The gaps at lengths smaller than n_0 come (at worst) at lengths $n_0^{1/2}, n_0^{1/4}, n_0^{1/8}, \ldots$, for a total of at most $2^{n_0^{1/2}+1} < 2^{m_0/4}$ strings shorter than n_0 outside $\mathrm{dom}(\sigma)$. Call this set of strings the *spoiler set*, and let G be any oracle extending τ.

We restrict our attention to computations of $M^G(x_j)$ for $1 \leqslant j \leqslant 2^{m_0}$. We claim first that the total number N of strings $x_i y_i$ queried by M over all these computations together is less than $2^{m_0/4}$. Suppose not, i.e., $N \geqslant 2^{m_0/4}$. We could give a short description (relative to σ; recall that the Kolmogorov randomness of y is relative to σ) of y consisting of the following four self-terminating[12] strings in sequence:

- m_0 in binary,
- all the bits of G restricted to the spoiler set,
- a concatenation in increasing order of i of all the y_i such that $x_i y_i$ is not queried by M, and
- a concatenation of strings $x_j r$ such that $M^G(x_j)$ makes its first query to some string $x_i y_i$ as its rth query. (Each r has length $k \log m_0$ bits, and each queried $x_i y_i$ is counted exactly once in the string.)

It is clear that y can be effectively recovered, using σ as an oracle, from this description and a program for M. By assumption, $N \geqslant 2^{m_0/4}$, and if m_0 is large enough then $m_0 \geqslant k \log m_0$, and so the entire description has length

$$\begin{aligned}
&2^{m_0/4} + 3m_0(2^{m_0} - N) + N(m_0 + k \log m_0) + \mathrm{O}(m_0) \\
&\quad = 2^{m_0/4} + 3m_0 2^{m_0} + N(k \log m_0 - 2m_0) + \mathrm{O}(m_0) \\
&\quad \leqslant 2^{m_0/4} + 3m_0 2^{m_0} - N m_0 + \mathrm{O}(m_0) \leqslant 2^{m_0/4} \\
&\qquad + 3m_0 2^{m_0} - 2^{m_0/4} m_0 + \mathrm{O}(m_0) \\
&\quad < m_0\big(3 \cdot 2^{m_0} - (1/2)2^{m_0/4} + \mathrm{O}(1)\big),
\end{aligned}$$

which is shorter than $|y|$ by more than a constant. This contradicts our choice of Kolmogorov random y, and thus less than $2^{m_0/4}$ of the $x_i y_i$ are queried by M.

Now we chop up the set $\{1, \ldots, 2^{m_0}\}$ into blocks

$$B_j = \{j2^{3m_0/4} + 1, j2^{3m_0/4} + 2, \ldots, (j+1)2^{3m_0/4}\}$$

for $0 \leqslant j < 2^{m_0/4}$. From the claim above, there is a block B_j such that no $x_i y_i$ for $i \in B_j$ is ever queried by M. We now claim that there must be an x_i with $i \in B_j$ such that $M^G(x_i) \neq L^G(x_i)$. Suppose not. Noticing that $L^G(x_i)$ corresponds to the rightmost bit of $x_i y_i$, we see that, for all $i \in B_j$, $M^G(x_i)$ correctly computes the rightmost bit of $x_i y_i$ without querying any $x_{i'} y_{i'}$ for $i' \in B_j$. We therefore get a short description of y as a concatenation of the following strings:

- j in binary, using $m_0/4$ bits,
- all the bits of G restricted to the spoiler set, padded out to $2^{m_0/4}$ bits,

[12] To be self-terminating, string s is preceded by some information to let a machine know where s ends, assuming it is concatenated with more bits to its right. The length of this prefix is $\mathrm{O}(\log |s|)$. See [42] for details.

- a concatenation of all y_i without the rightmost bit of each, for all $i \in B_j$, and
- a concatenation of all y_i for $i \notin B_j$, in order of increasing i.

Again, y can be effectively generated relative to σ from this description (note that m_0 can be computed from its length). To find the rightmost bit of some y_i with $i \in B_j$, we run $M^G(x_i)$, using the spoiler set bits when we need them. If $M^G(x_i)$ queries any $x_{i'}z$ where $|z| = 3m_0$, then the answer is 0 if $i' \in B_j$ since $M^G(x_i)$ does not query $x_{i'}y_{i'}$ by assumption. Otherwise if $i' \notin B_j$, we can determine the answer by checking whether z occurs in the last part of the description at the position corresponding to i'.

The length of the description above is

$$\begin{aligned} &m_0/4 + 2^{m_0/4} + (3m_0 - 1)2^{3m_0/4} + 3m_0\left(2^{m_0} - 2^{3m_0/4}\right) \\ &\quad = 3m_0 2^{m_0} - 2^{3m_0/4} + 2^{m_0/4} + m_0/4 < |y| - \mathrm{O}(1), \end{aligned}$$

which again contradicts the fact that y is incompressible.

Since we chose M arbitrarily, we've established that $L(M^G) \neq L^G$ for any G extending the $\mathcal{SP}$-condition τ. Hence, $\tau \Vdash_{\mathcal{SP}} \neg\neg\varphi$, where $\varphi \in \mathrm{sent}(\mathcal{L}_{\mathrm{PA}}^X)$ is the sentence "$(\forall k)[\mathbf{ULIN}^X \cap \text{co-}\mathbf{ULIN}^X \nsubseteq \mathbf{DTIME}(n^k)^X]$." Thus $\sigma \Vdash_{\mathcal{SP}} \neg\neg\varphi$, and since σ was chosen arbitrarily, we have that $\omega[G] \models \varphi$ for any $\mathcal{SP}$-generic G. □

7.3.1. $\mathcal{SP}$-Conditions with larger gaps

In this section we generalize $\mathcal{SP}$ to a family $\{\mathcal{SP}_i\}_{i\in\omega}$ of notions of genericity with different complexity theoretic properties. Compare the following with Definition 5.1.

Definition 7.7. For $i, n \in \omega$, define $T_i(n)$ to be an exponential tower of i 2's below n, that is,

$$T_0(n) = n,$$
$$T_{j+1}(n) = 2^{T_j(n)} \quad \text{for } j \in \omega.$$

For c a positive integer, let $A_{i,c} = \{T_i(cn) \mid n \in \omega\}$. An *i-c-condition* is a partial characteristic function $\tau : \Sigma^* \to \{0, 1\}$ such that

$$\mathrm{dom}(\tau) = \bigcup_{m\in\omega - A_{i,c}} \Sigma^m.$$

A condition τ is an $\mathcal{SP}_i$-condition if τ is an i-c-condition for some $c \geqslant 1$. We let $\mathcal{SP}_i$ denote the class of all $\mathcal{SP}_i$-conditions.

Clearly, $\mathcal{SP} = \mathcal{SP}_2$. As i increases, the gaps in the domains of $\mathcal{SP}_i$ conditions grow further and further apart. For instance, each successive gap in an $\mathcal{SP}_3$-condition comes at strings of length superpolynomial in lengths of strings in the previous gap. It is easy to see that $\mathcal{SP}_i$ is basic for all i. Indeed, the notions $\mathcal{SP}_i$ all have identical topological and computability theoretic properties; in particular, Theorem 5.6 holds for $\mathcal{SP}_i$-generics by the same proof. The notions have different complexity theoretic properties, however—a difference that we will see especially in the polynomial case between $\mathcal{SP}_2$ and $\mathcal{SP}_3$, underscoring how much more sensitive complexity theory is compared to computability theory in the face of relativization.

Definition 7.8. For all integers $c > 0$ and real $x > 0$, we define inductively

$$g_{0,c}(x) = x + c,$$

and

$$g_{i+1,c}(x) = 2^{g_{i,c}(\log_2 x)}$$

for all $i \in \omega$.

The functions $g_{i,c}$ are defined so that, for any i-c-condition σ, if dom(σ) has a gap at length ℓ, then the next bigger gap will be at length $g_{i,c}(\ell)$. Note that $g_{1,c}(x) = x2^c$ and $g_{2,c}(x) = x^{2^c}$, and for fixed c, $g_{3,c}(x) = 2^{(\log x)^{2^c}}$ grows superpolynomially in x. The $g_{i,c}$ form a natural hierarchy of subexponentially growing functions (see, for example, Lutz [41]). The proof of Theorem 7.6 "scales up" in a straightforward way to prove

Theorem 7.9. *For $i, j, k \in \omega$ with $2 \leqslant i < j$, relative to any $\mathcal{SP}_j$-generic set,*

$$\mathbf{ULIN} \cap \text{co-}\mathbf{ULIN} \cap \mathbf{ZPLIN} \nsubseteq \mathbf{DTIME}(g_{i,k}).$$

In particular, for polynomially bounded classes we have

Corollary 7.10. *Relative to any $\mathcal{SP}_3$ generic set,*

$$\mathbf{P} \neq \mathbf{UP} \cap \text{co-}\mathbf{UP} \cap \mathbf{ZPP}.$$

Thus $\mathcal{SP}_3$-generic oracles are almost the opposite of $\mathcal{SP}$-generics in how they treat these promise classes. On the other hand, all the collapsing results for $\mathcal{SP}$-generics also scale up to show that

Proposition 7.11. *Relative to any $\mathcal{SP}_i$-generic set with $i \geqslant 2$,*

$$\bigcup_k \mathbf{DTIME}(g_{i,k}) = \bigcup_k \mathbf{UTIME}(g_{i,k}) = \bigcup_k \mathbf{BPTIME}(g_{i,k}) = \cdots$$

8. Further work and open problems

Despite the almost complete and coherent relativized world view generic oracles provide, there are still some open questions regarding the collapse of certain complexity classes relative to generics. For $\mathcal{SP}$-generics, or for Cohen generics under suitably strong unrelativized assumptions, we have $\mathbf{P} = \mathbf{AWPP}$ and $\mathbf{P} = \mathbf{NP} \cap \text{co-}\mathbf{NP}$. Are stronger collapses possible? For example, is it the case that $\mathbf{P} = \mathrm{C}_=\mathbf{P} \cap \text{co-}\mathrm{C}_=\mathbf{P}$ with respect to $\mathcal{SP}$-generics, or with respect to Cohen generics with sufficient unrelativized "help"? This would be the case if a certain strengthening of Nisan-Szegedy [44] held, i.e., if we weakened the hypothesis by allowing the Boolean formula to be represented by any rational function rather than just a polynomial. Such a collapse would imply at once both the known collapses $\mathbf{P} = \mathbf{WPP}$ and $\mathbf{P} = \mathbf{NP} \cap \text{co-}\mathbf{NP}$ with respect to generics, currently achieved by different proofs. Is there any reasonable class containing both **NP** $\cap$ co-**NP** and **AWPP** that would collapse to **P**?

The collapse of **AWPP** to **P** relative to $\mathcal{SP}$-generics, while **PH** remains infinite (Proposition 5.4), has special significance to the quantum computing model. The relativizable inclusion **BQP** $\subseteq$ **AWPP** shown by Fortnow and Rogers [24] implies that $\mathcal{SP}$-generics provide a relativized world where quantum computers are no more powerful than classical deterministic ones, and **NP**-complete problems are beyond the reach of both. This bolsters an earlier result of Bennett *et al.* that **NP** $\nsubseteq$ **BQP** relative to a random oracle [3], and suggests severe limitations on quantum computation.

Several questions that were open when the proceedings version of this paper appeared [21] have been answered, and further results have been obtained. In Section 7.2, we created a specialized form of generic, an **MA**-generic, to solve a specific problem. Other specialized generic sets, including **UP**-generic and (**NP** $\cap$ co-**NP**)-generic sets, have been applied by Fortnow and Rogers [23] to get simultaneous collapses and separations of various subclasses of **NP**. Buhrman and Fortnow [8] also used a **UP**-generic set to find an oracle where **NP** $\neq$ co-**NP** but $\mathbf{P}^{\mathbf{NP}[1]} = \mathbf{P}^{\mathbf{NP}[2]} =$ **PSPACE**. In a clever use of the rerelativization technique, Rogers [48] defined a generic oracle where both **P** $\neq$ **UP** and the Isomorphism Conjecture holds. This oracle is a **UP**-generic oracle built "on top of" an $\mathcal{SP}$-generic. Perhaps other specialized forms of generics may have applications to other oracle constructions.

So far all our collapsing results collapse classes down to **P**. Until recently, a long standing open question was what happens one level up in the polynomial hierarchy. With respect to $\mathcal{SP}$-generics (or Cohen generics with help), is it the case that $\Sigma_2^p \cap \Pi_2^p$ collapses to $\mathbf{P}^{\mathbf{NP}}$? What happens at higher levels? Both these questions were resolved by Fortnow and Yamakami [26], who showed that $\Sigma_2^p \cap \Pi_2^p$ does *not* collapse to $\mathbf{P}^{\mathbf{NP}}$, and that separations also occur at higher levels as well. Whether their techniques apply to hierarchies involving counting classes is a natural question. For example, is $\mathbf{SPP}^{\mathbf{NP}} = \mathbf{P}^{\mathbf{NP}}$ relative to an $\mathcal{SP}$-generic oracle?

Finally, there are notions of forcing/genericity in the literature—used in constructing oracles in complexity theory—that fall under the general scheme described in Section 3, but that did not appear to originally. Particular examples are in Fortnow and Rogers [23] and Fenner and Schaefer [25, Proof of Lemma 8.5]. These notions of genericity involve extra "promises" about the structure of the generic set being built. For instance, Fortnow and Rogers define *size-bounded generic sets* as follows: Define $t(0) = 1$ and $t(n+1) = 2^{t(n)}$, and call the range of t the set of *allowed lengths*. A *condition* is a pair (σ, k) such that k is a positive integer, $\sigma : \Sigma^{\leqslant n} \to \{0, 1\}$ for some $n > 0$, and $\sigma(x) = 0$ for all $x \in \mathrm{dom}(\sigma)$ when $|x|$ is not an allowed length. One condition (τ, ℓ) *extends* another condition (σ, k) iff

- $\sigma \preceq \tau$,
- $k \leqslant \ell$, and
- for all n such that there are strings of length n in $\mathrm{dom}(\tau) - \mathrm{dom}(\sigma)$, we have $\tau(x) = 1$ for at most $n/2^k$ strings x of length n.

A dense set of conditions is defined in the usual way, and given a prespecified countable collection of dense sets of conditions, there is a countable chain $c_1 \preceq c_2 \preceq \cdots$ of conditions that intersects each dense set in the collection. The *sb-generic* oracle is then the union of the first components of the conditions in the chain. The presence of the integer k in a condition encodes the promise that future extensions of σ will be sufficiently sparse. Fortnow and Rogers show that relative to a sufficiently sb-generic oracle, $\mathbf{P} = \mathbf{UP} = \mathbf{NP} \cap \text{co-}\mathbf{NP}$ and there are no pairs of **P**-inseparable sets either in **NP** or in co-**NP**. (There are **P**-inseparable sets in co-**NP** relative to Cohen generics.)

The notion of size-bounded genericity is actually an instance of Definition 3.3 when we identify a condition (σ, k) with the correct perfect subset $\gamma_{\sigma,k}$ of 2^ω. The set $\gamma_{\sigma,k}$ consists of all $A \in 2^\omega = 2^{\Sigma^*}$ such that

- A extends σ,
- $A(x) = 0$ for all $x \in \Sigma^*$ such that $|x|$ is not an allowed length, and
- for every allowed length n such that σ is undefined on strings of length n, we have $A(x) = 1$ for at most $n/2^k$ strings x of length n.

Such a $\gamma_{\sigma,k}$ is easily seen to be a perfect set. Moreover, for any two conditions (σ, k) and (τ, ℓ), we have $(\sigma, k) \preceq (\tau, \ell)$ if and only if $\gamma_{\sigma,k} \supseteq \gamma_{\tau,\ell}$. Finally, it is clear that the set of all $\gamma_{\sigma,k}$ is a notion of genericity according to Definition 3.3.

A similar identification works for the (unnamed) notion of genericity defined in [25]. Thus to our knowledge, all notions of genericity used in complexity theory (including randomness) fit the framework described in Section 3. Any useful notion that violates this framework would be highly interesting.

Acknowledgments

We would like to take this opportunity to repay through thanks a number of specific debts we incurred in preparing this paper. We have benefited from helpful discussions with Martin Böttcher, Carl Jockusch, John Rogers, Jim Royer, and Yong Zhang. We have come to rely on Noam Nisan and Mario Szegedy, who have demonstrated a remarkable ability to publish combinatorial results just before we need them.

References

[1] V. Arvind, P.P. Kurur, Graph Isomorphism is in SPP, in: Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science, IEEE, New York, 2002, pp. 743–750.

[2] K. Ambos-Spies, Resource-bounded genericity, in: S.B. Cooper, T.A. Slaman, S.S. Wainer (Eds.), Computability, Enumerability, Unsolvability: Directions in Recursion Theory, Cambridge University Press, Cambridge, 1996, pp. 1–59.

[3] C.H. Bennett, E. Bernstein, G. Brassard, U. Vazirani, Strengths and weaknesses of quantum computation, SIAM J. Comput. 26 (5) (1997) 1510–1523, quant-ph/9701001.

[4] R. Beals, H. Buhrman, R. Cleve, M. Mosca, R. de Wolf, Quantum lower bounds by polynomials, in: Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, IEEE, New York, 1998, pp. 352–353, quant-ph/9802049.

[5] R. Beigel, H. Buhrman, L. Fortnow, NP might not be as easy as detecting unique solutions, in: Proceedings of the 30th ACM Symposium on the Theory of Computing, ACM, New York, 1998, pp. 203–208.

[6] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity I, EATCS Monographs on Theoretical Computer Science, vol. 11, Springer-Verlag, Berlin, 1988.

[7] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity II, EATCS Monographs on Theoretical Computer Science, vol. 22, Springer-Verlag, Berlin, 1990.

[8] H. Buhrman, L. Fortnow, Two queries, J. Comput. Syst. Sci. 59 (2) (1999) 182–194.

[9] C.H. Bennett, J. Gill, Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1, SIAM J. Comput. 10 (1981) 96–113.

[10] L. Berman, J. Hartmanis, On isomorphism and density of NP and other complete sets, SIAM J. Comput. 1 (1977) 305–322.
[11] M. Blum, R. Impagliazzo, Generic oracles and oracle classes, in: Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, IEEE, New York, 1987, pp. 188–196.
[12] M. Böttcher, Private communication, 1993.
[13] E. Bernstein, U. Vazirani, Quantum complexity theory, SIAM J. Comput. 26 (5) (1997) 1411–1473.
[14] P.J. Cohen, The independence of the continuum hypothesis, I, in: Proceedings of the National Academy of Science, vol. 50, 1963, pp. 1143–1148.
[15] P.J. Cohen, The independence of the continuum hypothesis, II, in: Proceedings of the National Academy of Science, vol. 51, 1964, pp. 105–110.
[16] S. Feferman, Some applications of the notions of forcing and generic sets, Fundam. Math. 56 (1965) 325–345.
[17] S.A. Fenner, Counting complexity and quantum computation, in: R.K. Brylinski, G. Chen (Eds.), Mathematics of Quantum Computation, CRC Press, Boca Raton, 2002, pp. 171–219 (Chapter 8).
[18] S.A. Fenner, PP-lowness and a simple definition of AWPP, Theory of Computing Systems 36 (2003) 199–212. Also available as ECCC Report TR02-036.
[19] S. Fenner, L. Fortnow, S. Kurtz, Gap-definable counting classes, J. Comput. Syst. Sci. 48 (1) (1994) 116–148.
[20] S.A. Fenner, L.J. Fortnow, S.A. Kurtz, The isomorphism conjecture holds relative to an oracle, SIAM J. Comput. 25 (1) (1996) 193–206.
[21] S. Fenner, L. Fortnow, S. Kurtz, L. Li, An oracle builder's toolkit, in: Proceedings of the 8th IEEE Structure in Complexity Theory Conference, 1993, pp. 120–131.
[22] L. Fortnow, Counting complexity, in: L.A. Hemaspaandra, A.L. Selman (Eds.), Complexity Theory Retrospective II, Springer-Verlag, Berlin, 1997.
[23] L. Fortnow, J. Rogers, Separability and one-way functions, in: Proceedings of the 5th Annual International Symposium on Algorithms and Computation, Lecture Notes in Computer Science, vol. 834, Springer-Verlag, Berlin, 1994, pp. 396–404.
[24] L. Fortnow, J. Rogers, Complexity limitations on quantum computation, J. Comput. Syst. Sci. 59 (2) (1999) 240–252, cs.CC/9811023.
[25] S. Fenner, M. Schaefer, Simplicity and strong reductions, Unpublished manuscript, 1997. Available from <http://www.cse.sc.edu/~fenner/papers/simplicity.ps>.
[26] L. Fortnow, T. Yamakami, Generic separations, J. Comput. Syst. Sci. 52 (1) (1996) 191–197.
[27] J. Grollmann, A. Selman, Complexity measures for public-key cryptosystems, SIAM J. Comput. 17 (1988) 309–335.
[28] J. Hartmanis, L. Hemachandra, One-way functions and the nonisomorphism of NP-complete sets, Theoret. Comput. Sci. 81 (1) (1991) 155–163.
[29] S. Homer, A. Selman, Oracles for structural properties: the isomorphism problem and public-key cryptography, J. Comput. Syst. Sci. 44 (2) (1992) 287–301.
[30] R. Impagliazzo, M. Naor, Decision trees and downward closures, in: Proceedings of the 3rd IEEE Structure in Complexity Theory Conference, IEEE, New York, 1988, pp. 29–38.
[31] T. Jech, Set Theory, Academic Press, New York, 1978.
[32] C.G. Jockusch, Degrees of generic sets, in: F.R. Drake, S.S. Wainer (Eds.), Recursion Theory: Its Generalizations and Applications, Cambridge University Press, Cambridge, 1980, pp. 110–139.
[33] S.A. Kurtz, S.R. Mahaney, J.S. Royer, The structure of complete degrees, in: A.L. Selman (Ed.), Complexity Theory Retrospective, Springer-Verlag, Berlin, 1990, pp. 108–146 (Chapter 6).
[34] S. Kurtz, S. Mahaney, J. Royer, The isomorphism conjecture fails relative to a random oracle, J. ACM 42 (2) (1995) 401–420.
[35] S. Kleene, E. Post, The uppersemilattice of degrees of recursive unsolvability, Ann. Math. 59 (1954) 379–407.
[36] J. Köbler, U. Schöning, J. Torán, Graph Isomorphism is low for PP, Comput. Complexity 2 (4) (1992) 301–330.
[37] K. Kunen, Set Theory: An Introduction to Independence Proofs, Studies in Logic and the Foundations of Mathematics, vol. 102, North-Holland, Amsterdam, 1980.
[38] S. Kurtz, The isomorphism conjecture fails relative to a generic oracle, Technical Report 88-018, Department of Computer Science, University of Chicago, 1988.

[39] D. Lacombe, Sur le semi-réseau constitué par les degrès d'indécidabilité ré récursive, Comptes rendus hebdomadaires des séances de l'Académie des Sciences (Paris) 239 (1954) 1108–1109.
[40] L. Li, On the counting functions. Technical Report TR-93-12, The University of Chicago, 1993. PhD Thesis. Available from <http://www.cs.uchicago.edu/research/publications/techreports/TR-93-12>.
[41] J.H. Lutz, Almost everywhere high nonuniform complexity, J. Comput. Syst. Sci. 44 (1992) 220–258.
[42] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, Texts and Monographs in Computer Science, Springer-Verlag, Berlin, 1993.
[43] K. Mehlhorn, On the size of sets of computable functions, in: Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory, IEEE Computer Society, Silver Spring, 1973, pp. 190–199.
[44] N. Nisan, M. Szegedy, On the degree of boolean functions as real polynomials, Comput. Complexity 4 (1994) 301–313.
[45] P. Odifreddi, Forcing and reducibilities, J. Symbol. Logic 48 (2) (1983) 288–310.
[46] P. Odifreddi, in: Classical Recursion Theory, Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1989.
[47] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[48] J. Rogers, The isomorphism conjecture holds and one-way functions exist relative to an oracle, J. Comput. Syst. Sci. 54 (3) (1997) 412–423.
[49] G.E. Sacks, Forcing with perfect closed sets, in: D.S. Scott (Ed.), Axiomatic Set Theory, American Mathematical Society, Providence, 1971, pp. 331–355.
[50] U. Schöning, The power of counting, in: A.L. Selman (Ed.), Complexity Theory Retrospective, Springer-Verlag, Berlin, 1990.
[51] R.M. Solovay, A model of set theory in which every set of reals is Lebesgue measurable, Ann. Math. 92 (1970) 1–56.
[52] C. Spector, On degrees of recursive unsolvability, Ann. Math. 64 (2) (1956) 581–592.
[53] T.A. Slaman, W.H. Woodin, Definability in the Turing degrees, Illinois J. Math. 30 (1986) 320–334.
[54] T.A. Slaman, W.H. Woodin, Definability in the enumeration degrees, Arch. Math. Logic 36 (1997) 255–267.
[55] L. Valiant, The complexity of computing the permanent, Theoret. Comput. Sci. 8 (1979) 189–201.
[56] A. Yao, Quantum circuit complexity, in: Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, 1993, pp. 352–361.