Complexity Classes of Equivalence Problems Revisited

Lance Fortnow and Joshua A. Grochow

Abstract

To determine if two lists of numbers are the same set, we sort both lists and see if we get the same result. The sorted list is a *canonical form* for the equivalence relation of set equality. Other canonical forms arise in graph isomorphism algorithms. To determine if two graphs are cospectral (have the same eigenvalues), we compute their characteristic polynomials and see if they are equal; the characteristic polynomial is a *complete invariant* for cospectrality. Finally, an equivalence relation may be decidable in P without either a complete invariant or canonical form. Blass and Gurevich (SIAM J. Comput., 1984) ask whether these conditions on equivalence relations—having an FP canonical form, having an FP complete invariant, and being in P—are distinct. They showed that this question requires non-relativizing techniques to resolve. We extend their results, and give new connections to probabilistic and quantum computation.

Keywords: Computational complexity; complexity class; oracle; probabilistic computation; quantum computation; equivalence relation; isomorphism problem; normal form; canonical form

1. Introduction

Equivalence relations and their associated algorithmic problems arise throughout mathematics and computer science. Examples run the gamut from trivial—decide whether two lists contain the same set of elements—to undecidable—decide whether two finitely presented groups are isomorphic [57, 20]. Some examples are of great mathematical importance, and some are of great interest to complexity theorists, such as graph isomorphism (GI).

Complete invariants are a common tool for finding algorithmic solutions to equivalence problems. Normal or canonical forms—where a unique representative is chosen from each equivalence class as the invariant of that class—are also quite common, particularly in algorithms for GI and its variants [40, 41, 13, 33, 55, 11]. More recently, Agrawal and Thierauf [4, 74] used a randomized canonical form to show that Boolean formula non-isomorphism ($\overline{FI}$) is in $\mathsf{AM}^{\mathsf{NP}}$. More generally, the

monograph by Thierauf [74] gives an excellent overview of equivalence and isomorphism problems in complexity theory.

Many efficient algorithms for special cases of *GI* have been upgraded to canonical forms or complete invariants. Are these techniques necessary for an efficient algorithm? Are these techniques distinct? Gary Miller [55] pointed out that *GI* has a polynomial-time complete invariant if and only if it has a polynomial-time canonical form (see also [37]). The general form of this question is central both in Blass and Gurevich [18, 19] and here: are canonical forms or complete invariants necessary for the efficient solution of equivalence problems?

In 1984, Blass and Gurevich [18, 19] introduced complexity classes to study these algorithmic approaches to equivalence problems. Although we came to the same definitions and many of the same results independently, this work can be viewed partially as an update and a follow-up to their papers in light of the intervening 25 years of complexity theory. The classes UP, RP, and BQP, the function classes NPMV (multi-valued functions computed by NP machines) and NPSV (single-valued functions computed by NP machines), and generic oracle (forcing) methods feature prominently in this work.

Blass and Gurevich [18, 19] introduced the following four problems and the associated complexity classes. Where they use "normal form" we say "canonical form," though the terms are synonymous and the choice is immaterial. We also introduce new notation for these complexity classes that makes the distinction between language classes and function classes more explicit. For an equivalence relation $R \subseteq \Sigma^* \times \Sigma^*$, they defined:

The *recognition problem*: given $x, y \in \Sigma^*$, decide whether $x \sim_R y$.

The *invariant problem*: for $x \in \Sigma^*$, calculate a complete invariant $f(x) \in \Sigma^*$ for R, that is, a function such that $x \sim_R y$ if and only if $f(x) = f(y)$.

The *canonical form problem*: for $x \in \Sigma^*$ calculate a canonical form $f(x) \in \Sigma^*$ for R, that is, a function such that $x \sim_R f(x)$ for all $x \in \Sigma^*$, and $x \sim_R y$ implies $f(x) = f(y)$.

The *first canonical form problem*: for $x \in \Sigma^*$, calculate the first $y \in \Sigma^*$ such that $y \sim_R x$. Here, "first" refers to the standard length-lexicographic ordering on Σ^*, though any ordering that can be computed easily enough would suffice.

The corresponding polynomial-time complexity classes are defined as follows:

Definition 1.1. PEq consists of those equivalence relations whose recognition problem has a polynomial-time solution. Ker(FP) consists of those equivalence relations that have a polynomial-

time computable complete invariant. CF(FP) consists of those equivalence relations that have a polynomial-time canonical form. LexEqFP consists of those equivalence relations whose first canonical form is computable in polynomial time.

We occasionally omit the "FP" from the latter three classes. It is obvious that

$$\mathsf{LexEq} \subseteq \mathsf{CF} \subseteq \mathsf{Ker} \subseteq \mathsf{PEq},$$

and our first guiding question is: which of these inclusions is tight?

1.1. Examples

To get a better feel for these complexity classes and help motivate them, we begin with several examples, especially including those that potentially witness the separation of these classes. Some of these will be discussed in more depth in Section 4.2. We also rephrase some of the examples we have already mentioned using these classes.

Example 1.2. Graph isomorphism is in NPEq (equivalence problems decidable in NP), and is in Ker(FP) if and only if it is in CF(FP) [55] (see also [37]). In fact, this result also holds for any function class that is closed under FP reductions such as $\mathsf{FP}^{\mathsf{NP}\cap\mathsf{coNP}}$.

Example 1.3. Boolean formula equivalence (do two Boolean formulae compute the same function) is in coNPEq, and is coNP-complete (to check if φ is a tautology, see if it is equivalent to the constant-true formula 1).

Example 1.4. Sorting a list is a first canonical form for set equality. Set equality is thus in LexEqFP.

Example 1.5. The characteristic polynomial is a polynomial-time complete invariant for graph cospectrality. No polynomial-time canonical form is known for this problem, so graph cospectrality is a potential witness to $\mathsf{CF} \neq \mathsf{Ker}$.

Example 1.6. The *subgroup equality problem* is: given two subsets $\{g_1, \ldots, g_t\}$, $\{h_1, \ldots, h_s\}$ of a group G determine if they generate the same subgroup. For permutation groups on $\{1, \ldots, n\}$, this problem lies in CF(FP), via a simple modification [10] of the classic techniques of Sims [68, 69], whose analysis was completed by Furst, Hopcroft, and Luks [34] and Knuth [47]. However, the subgroup equality problem for other groups is a potential source of witnesses to $\mathsf{Ker} \neq \mathsf{PEq}$.

Although factoring integers is not an equivalence problem, its hardness would imply $\mathsf{CF} \neq \mathsf{Ker}$, as the next proposition shows. In Section 4.2.1, we show a similar result based on the hardness of collision-free hash functions that can be computed deterministically. The proof of this proposition highlights what seems to be an essential difference between CF and Ker.

Proposition 1.7. *If* $\mathsf{CF} = \mathsf{Ker}$ *then integers can be factored in probabilistic polynomial time.*

Proof. Suppose we wish to factor an integer N. We may assume N is not prime, since primality can be determined in polynomial time [3], but even much weaker machinery lets us do so in probabilistic polynomial time [72, 58], which is sufficient here. By hypothesis, the kernel of the Rabin function $x \mapsto x^2 \pmod N$:

$$R_N = \{(x, y) : x^2 \equiv y^2 \pmod N\}$$

has a canonical form $f \in \mathsf{FP}$.

Randomly choose $x \in \mathbb{Z}/N\mathbb{Z}$ and let $y = f(x)$. Then $x^2 \equiv y^2 \pmod N$; equivalently, $(x - y)(x + y) \equiv 0 \pmod N$. If $y \not\equiv \pm x \pmod N$, then since neither $x - y$ nor $x + y$ is $\equiv 0 \pmod N$, $\gcd(N, x - y)$ is a nontrivial factor z of N. Let $r(N)$ be the least number of distinct square roots modulo N. Then $\Pr_x[y \not\equiv \pm x] \geq 1 - \frac{2}{r(N)}$. Since N is composite and odd without loss of generality, $r(N) \geq 4$. Thus $\Pr_x[y \not\equiv \pm x] = \Pr_x[\text{the algorithm finds a factor of } N] \geq \frac{1}{2}$. Recursively call the algorithm on N/z. □

1.2. Main results

Blass and Gurevich showed that none of the four problems above polynomial-time Turing-reduces (Cook-reduces) to the next in line. We extend their results using generic oracles, and we also give further complexity-theoretic evidence for the separation of these classes, giving new connections to probabilistic and quantum computing. Our main results in this regard are:

Proposition 1.7. *If* $\mathsf{CF} = \mathsf{Ker}$ *then integers can be factored in probabilistic polynomial time.*

Proposition 4.12. *If* $\mathsf{CF} = \mathsf{Ker}$ *then collision-free hash functions that can be evaluated in deterministic polynomial time do not exist.*

Theorem 4.3. *If* $\mathsf{Ker} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{BQP}$. *If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{RP}$.

Theorem 4.6. *If* $\mathsf{PromiseKer} = \mathsf{PromisePEq}$ *then* $\mathsf{NP} \subseteq \mathsf{BQP} \cap \mathsf{SZK}$, *and in particular* $\mathsf{PH} = \mathsf{AM}$.

We give the definitions of $\mathsf{PromisePEq}$ and $\mathsf{PromiseKer}$ in Section 4.1.1. We also show the following two related results:

Corollary 4.2. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{NP} = \mathsf{UP}$ *and* $\mathsf{PH} \subseteq \mathsf{S}_2[\mathsf{NP} \cap \mathsf{coNP}] \subseteq \mathsf{ZPP}^{\mathsf{NP}}$.

Corollary 4.4. *If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{NP} = \mathsf{UP} = \mathsf{RP}$ *and in particular,* $\mathsf{PH} = \mathsf{BPP}$.

Corollary 4.2 follows from the slightly stronger Theorem 4.1, but we do not give the statement here as it requires further definitions.

1.3. Organization

The remainder of the paper is organized as follows. In Section 2 we give preliminary definitions and background. In Section 3 we review the original results of Blass and Gurevich [18, 19]. We also combine their results with other results that have appeared in the past 25 years to yield

some immediate extensions. In Section 4.1 we prove new results connecting these classes with probabilistic and quantum computation. In Section 4.1.1 we introduce the promise versions of PEq and Ker and prove Theorem 4.6. In Section 4.1.2, we introduce a group-like condition on the witness sets of NP-complete problems that would allow us to extend the first half of Theorem 4.3 from UP to NP, giving much stronger evidence that Ker $\neq$ PEq. We believe the question of whether any NP-complete sets have this property is of independent interest: a positive answer would provide nontrivial quantum algorithms for NP problems, and a negative answer would provide further concrete evidence for the lack of structure in NP-complete problems. In Section 4.2 we discuss collision-free hash functions, the subgroup equality problem and Boolean function congruence (not isomorphism) as potential witnesses to the separation of these classes. We also introduce a notion of reduction between equivalence relations and the corresponding notion of completeness. In Section 5, we update and extend some of the oracle results of Blass and Gurevich [18, 19] using generic oracles. In the final section we mention several directions for further research, in addition to the several open questions scattered throughout the paper.

2. Preliminaries

We assume the reader is familiar with standard complexity classes such as P, NP, BPP, and the polynomial hierarchy $\mathsf{PH} = \bigcup \Sigma_k \mathsf{P} = \bigcup \Pi_k \mathsf{P} = \bigcup \Delta_k \mathsf{P}$. We refer the reader to the textbook by Arora and Barak [7] and the Complexity Zoo at `http://qwiki.stanford.edu/index.php/Complexity_Zoo` for more details.

A language L is in the class UP if there is a nondeterministic machine deciding L that has at most one accepting path on each input.

The class BQP consists of those languages that can be decided on a quantum computer in polynomial time with error strictly bounded away from 1/2. For more details on quantum computing, we recommend the book by Nielson and Chuang [56].

For any class $\mathcal{C}$, the class $\mathsf{S}_2[\mathcal{C}]$ is defined as follows. A language L is in $\mathsf{S}_2[\mathcal{C}]$ if there is a language $V \in \mathcal{C}$ and a polynomial p such that

$$\begin{aligned} x \in L &\implies (\exists y : |y| \leq p(|x|))(\forall z : |z| \leq p(|x|))[V(x,y,z) = 1] \\ x \notin L &\implies (\exists z : |z| \leq p(|x|))(\forall y : |y| \leq p(|x|))[V(x,y,z) = 0]. \end{aligned}$$

The class $\mathsf{S_2P}$ was defined independently by Russell and Sundaram [61] and Canetti [26]. Cai [24] showed that $\mathsf{S_2[NP \cap coNP]} \subseteq \mathsf{ZPP^{NP}}$.

2.1. Function Classes

Complexity-bounded function classes are defined in terms of Turing transducers. A transducer only outputs a value if it enters an accepting state. In general, then, a nondeterministic transducer can be partial and/or multi-valued. For such a function f, we write

$$\textit{set-}f(x) = \{y : \text{ some accepting computation of } f(x) \text{ outputs } y\}$$

The *domain* of a partial multi-valued function is the set

$$\mathrm{dom}(f) = \{x : \textit{set-}f(x) \neq \emptyset\}.$$

The *graph* of a partial multi-valued function is the set

$$\mathrm{graph}(f) = \{(x, y) : y \in \textit{set-}f(x)\}.$$

The class FP is the class of all total functions computable in deterministic polynomial time. The class PF is the class of all partial functions computable in deterministic polynomial time. Note that machines computing a PF function must halt in polynomial time even when they make no output.

The class FL is the class of all total functions computable by deterministic logarithmic-space transducers, that is, the length of the output and the i-th bit of the output of the function can be computed in logarithmic-space.

The class NPSV consists of all single-valued partial functions computable by a nondeterministic polynomial-time transducer. Note that multiple branches of an NPSV transducer may accept, but they must all have the same output. The class NPMV consists of all multi-valued partial functions computable by a nondeterministic polynomial-time transducer. The classes $\mathsf{NPSV_t}$ and $\mathsf{NPMV_t}$ are the subclasses of NPSV and NPMV, respectively, consisting of the total functions in those classes. The classes $\mathsf{NPSV_g}$ and $\mathsf{NPMV_g}$ are the subclasses of NPSV and NPMV, respectively, whose graphs are in P.

A *refinement* of a multi-valued partial function f is a multi-valued partial function g such that $\mathrm{dom}(g) = \mathrm{dom}(f)$ and $\textit{set-}g(x) \subseteq \textit{set-}f(x)$ for all x. In particular, if $\textit{set-}f(x)$ is nonempty then so

is *set-g*(x). If $\mathcal{F}_1$ and $\mathcal{F}_2$ are two classes of partial multi-valued functions, then

$$\mathcal{F}_1 \subseteq_c \mathcal{F}_2$$

means that every function in $\mathcal{F}_1$ has a refinement in $\mathcal{F}_2$.

It is known that $\mathsf{NPMV} \subseteq_c \mathsf{PF}$ if and only if $\mathsf{P} = \mathsf{NP}$ [62] if and only if $\mathsf{NPSV} \subseteq \mathsf{PF}$ [64]. Selman [63] is one of the classic works in this area, and gives many more results regarding these function classes.

2.2. Equivalence Relations

For an equivalence relation $R \subseteq \Sigma^* \times \Sigma^*$, we write $x \sim_R y$ if $(x, y) \in R$. We write $[x]_R$ for the R-equivalence class of x. The *kernel* of a function f is the equivalence relation $\mathrm{Ker}(f) = \{(x, y) : f(x) = f(y)\}$. For an equivalence relation R, if $R = \mathrm{Ker}(f)$, we say that f is a *complete invariant* for R. If, furthermore, $x \sim_R f(x)$ for every x, then f is a *canonical form* for R. If, further still, $f(x)$ is the first member of $[x]_R$ under lexicographic order, we say that f is the *first canonical form* for R. The *trivial relation* is all of $\Sigma^* \times \Sigma^*$, that is, all strings are equivalent under the trivial relation, or equivalently $[x] = \Sigma^*$ for all x.

An equivalence relation is *length-restricted* if $x \sim y$ implies $|x| = |y|$. An equivalence relation is *polynomially bounded* if there is a polynomial p such that $x \sim y$ implies $|x| \leq p(|y|)$. Note that the first canonical form for a polynomially bounded equivalence relation is a polynomially honest function. If $\mathcal{C}$ is a class of equivalence relations, we write $\mathcal{C}_=$ for the class of length-restricted equivalence relations in $\mathcal{C}$, and $\mathcal{C}_p$ for the class of polynomially bounded equivalence relations in $\mathcal{C}$.

Let $\langle \cdot, \cdot \rangle \colon \Sigma^* \times \Sigma^* \to \Sigma^*$ be a polynomial-time computable and polynomial-time invertible pairing function such that $|\langle x, y \rangle|$ depends only on $|x|$ and $|y|$. By polynomial-time invertible we mean that the projection functions $\pi_i(\langle x_1, x_2 \rangle) = x_i$ for $i = 1, 2$ are computable in polynomial time.

3. Previous Results

Here we recall the previous results most relevant to our work. Most of the results in this section are from Blass and Gurevich [18, 19]. We are not aware of any other prior work in this area. However, results in other areas of computational complexity that have been obtained since 1984 can be used as black boxes to extend their results, which we do here.

We mention that analogues of these classes for finite-state machines have been studied, and nearly all their interrelationships completely determined [44]. For the class of computable functions

or the class of primitive recursive functions, Blass and Gurevich [18] already noted that all four classes of equivalence relations are equal.

If $R \in \mathsf{PEq}$, then the language $R' = \{(x, y) : (\exists z)[z \leq_{lex} y \text{ and } (x, z) \in R]\}$ is in NP, and can be used to perform a binary search for the first canonical form for R. Hence, $\mathsf{PEq} \subseteq \mathsf{LexEqFP}^{\mathsf{NP}}$. The first result shows that this containment is tight:

Theorem 3.1 ([18] Theorem 1)**.** *There is an equivalence relation $R \in \mathsf{CF}$ whose first canonical form problem is essentially $\Delta_2\mathsf{P}$-complete, that is, it is in $\mathsf{FP}^{\mathsf{NP}} = \mathsf{F}\Delta_2\mathsf{P}$ and is $\Delta_2\mathsf{P}$-hard.*

Note that the above proof that $\mathsf{PEq} \subseteq \mathsf{LexEqFP}^{\mathsf{NP}}$ relativizes, so all four polynomial-time classes of equivalence relations are equal in any world where $\mathsf{P} = \mathsf{NP}$, in particular, relative to any PSPACE-complete oracle. The next result gives relativized worlds in which $\mathsf{Ker} \neq \mathsf{PEq}$, $\mathsf{CF} \neq \mathsf{Ker}$, and $\mathsf{LexEq} \neq \mathsf{CF}$, though these worlds cannot obviously be combined.

Theorem 3.2 (Blass & Gurevich [18] Theorem 2)**.** *Of the four equivalence problems defined above, none is Cook reducible to the next in line. In particular:*

a. *There is an equivalence relation $R \notin \mathsf{Ker}(\mathsf{FP}^R)$, i. e., $\mathsf{Ker}(\mathsf{FP}^R) \neq \mathsf{P}^R\mathsf{Eq}$.*
b. *There is a function f such that $\mathrm{Ker}(f) \notin \mathsf{CF}(\mathsf{FP}^f)$, i. e., $\mathsf{CF}(\mathsf{FP}^f) \neq \mathsf{Ker}(\mathsf{FP}^f)$.*
c. *There is an idempotent function f such that $\mathrm{Ker}(f) \notin \mathsf{LexEqFP}^f$, i. e., $\mathsf{LexEqFP}^f \neq \mathsf{CF}(\mathsf{FP}^f)$.*

Furthermore, there is an equivalence relation $R \notin \mathsf{Ker}(\mathsf{NPSV}_{\mathsf{t}}^R)$, i. e., $\mathsf{P}^R\mathsf{Eq} \not\subseteq \mathsf{Ker}(\mathsf{NPSV}_{\mathsf{t}}^R)$ [19, Thm. 5].

In addition to several extensions of these results, Blass and Gurevich [18, 19] also show that collapses between certain classes of equivalence problems are equivalent to more standard complexity-theoretic hypotheses. Here we collect some of their main results:

Theorem 3.3. 1. $\mathsf{CF}(\mathsf{FP}) \subseteq \mathsf{LexEqNPSV}_{\mathsf{t}} \iff \mathsf{NPEq} \subseteq \mathsf{coNPEq} \iff \mathsf{coNPEq} \subseteq \mathsf{NPEq} \iff \mathsf{NP} = \mathsf{coNP}$ *[19, Thm. 1].*
2. $\mathsf{LexEqNPSV}_{\mathsf{t}} \subseteq \mathsf{PEq} \iff \mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$ *[19, Thm. 2].*

Note that NPEq consists of those equivalence relations decidable in NP, and is distinct from $\mathsf{P}^{\mathsf{NP}}\mathsf{Eq}$ assuming $\mathsf{NP} \neq \mathsf{P}^{\mathsf{NP}}$. This follows from the observation that, for any set A there is an equivalence relation R that is polynomial-time equivalent to A, namely the equivalence relation generated by $\{(0x, 1x) : x \in A\}$ (if A is neither empty nor Σ^*, then $A \equiv_m^p R$; in any case, $A \equiv_{1-tt}^p R$).

We think the following result is one of their most surprising:

Theorem 3.4 (Blass & Gurevich [19] Theorem 3)**.** *The following statements are equivalent:*

1. $\mathsf{Ker}(\mathsf{FP})_{=} \subseteq \mathsf{CF}(\mathsf{NPSV_t})$.
2. NP *has the shrinking property (see Glaßer, Reitwießner, and Selivanov [35]): if* $A, B \in \mathsf{NP}$*, then there are disjoint* $A', B' \in \mathsf{NP}$ *such that* $A' \subseteq A$*,* $B' \subseteq B$*, and* $A \cup B = A' \cup B'$*.*
3. $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$*, i. e., the uniformization principle holds for* NP*.*

Hemaspaandra, Naik, Ogihara, and Selman [39] showed that if $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ then $\mathsf{SAT} \in (\mathsf{NP} \cap \mathsf{coNP})/poly$. At the time, the strongest known consequence of $\mathsf{SAT} \in (\mathsf{NP} \cap \mathsf{coNP})/poly$ was $\mathsf{PH} = \Sigma_2\mathsf{P}$ [45]. Shortly thereafter Köbler and Watanabe [49] improved the collapse to $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$, and in the early 2000's Cai, Chakaravarthy, Hemaspaandra, and Ogihara [25] further improved the collapse to $\mathsf{PH} = \mathsf{S}_2[\mathsf{NP} \cap \mathsf{coNP}]$. Combined with Theorem 3.4, this immediately implies a result that has not been announced previously:

Corollary 3.5. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{PH} \subseteq \mathsf{S}_2[\mathsf{NP} \cap \mathsf{coNP}] \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. □

4. Evidence for Separation

4.1. New Collapses

Blass and Gurevich's [19] proof that $\mathsf{Ker}(\mathsf{FP})_{=} \subseteq \mathsf{CF}(\mathsf{NPSV_t}) \implies \mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ essentially shows the following slightly stronger result. However, as $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ is not known to imply $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$, our result does not directly follow from their *result*, but only from its proof, the core of which is reproduced here:

Theorem 4.1. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$.

Proof. Let $f \in \mathsf{NPMV_g}$, let M be a nondeterministic polynomial-time transducer computing f, and let V be a polynomial-time decider for $\mathrm{graph}(f)$. If $\mathsf{CF} = \mathsf{Ker}$, then the equivalence relation

$$\{((x,y),(x,y')) : V(x,y) = V(x,y')\} = \mathrm{Ker}((x,y) \mapsto (x, V(x,y)))$$

has a canonical form $c \in \mathsf{FP}$. Then the following algorithm computes a refinement of f in $\mathsf{NPSV_g}$: simulate $M(x)$. On each branch, if the output would be y, accept if and only if $c(x,y) = (x,y)$. Hence $f \in_c \mathsf{NPSV_g}$. □

Similar to the original result [19], we can weaken the assumption of this theorem to $\mathsf{Ker}_p \subseteq \mathsf{CF}$, without modifying the proof. By padding, we can further weaken the assumption to $\mathsf{Ker}_{=} \subseteq \mathsf{CF}$.

Corollary 4.2. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{NP} = \mathsf{UP}$ *and* $\mathsf{PH} \subseteq \mathsf{S}_2[\mathsf{NP} \cap \mathsf{coNP}] \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. □

Note that Corollary 3.5 alone does not imply Corollary 4.2, as neither of the statements $\mathsf{PH} = \mathsf{S_2[NP \cap coNP]}$ and $\mathsf{NP} = \mathsf{UP}$ is known to imply the other. Indeed, it is still an open question as to whether $\mathsf{NP} = \mathsf{UP}$ implies any collapse of PH whatsoever.

The next new result we present gives a new connection between complexity classes of equivalence problems and quantum and probabilistic computation:

Theorem 4.3. *If* $\mathsf{Ker} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{BQP}$. *If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{RP}$.

Proof. Suppose $\mathsf{Ker} = \mathsf{PEq}$. Let L be a language in UP, let V be a UP verifier for L, let p be a polynomial bounding the size of V-witnesses for L. Consider the relation

$$R_L = \{((a,x),(a,y)) : x = y \text{ or } |x| = |y| \text{ and } V(a, x \oplus y) = 1\}$$

where $\oplus$ denotes bit-wise exclusive-or. Clearly $R_L \in \mathsf{PEq}$, so by hypothesis R_L has a complete invariant $f \in \mathsf{FP}$. Since $L \in \mathsf{UP}$, for each $a \in L$ there is a unique string w_a such that $V(a, w_a) = 1$. Define $f_a(x) = f(a,x)$. Then for all distinct x and x', $f_a(x) = f_a(x')$ if and only if $x \oplus x' = w_a$. Given a and f_a, and the promise that f_a is either injective or two-to-one in the manner described, finding w_a or determining that there is no such string is exactly Daniel Simon's problem, which is in BQP [66].

Now suppose further that $\mathsf{CF} = \mathsf{PEq}$. Then we may take f to be not only a complete invariant but further a canonical form for R_L. On input a, the following algorithm decides L in polynomial time with bounded error: for each length $\ell \leq p(|a|)$, pick a string x of length ℓ at random, compute $f((a,x)) = (a,y)$, and compute $V(a, x \oplus y)$. If $V(a, x \oplus y) = 1$ for any length ℓ, output 1. Otherwise, output 0. If $a \notin L$ then this algorithm always returns 0. If $a \in L$ and 0^ℓ is a's witness, then the algorithm always returns 1. If $a \in L$ and 0^ℓ is not a's witness, then $y \neq x$, and hence the answer is correct, with probability 1/2. □

We would like to extend the first half of Theorem 4.3 from UP to NP to give stronger evidence that $\mathsf{Ker} \neq \mathsf{PEq}$, but the techniques do not obviously apply. We pose two approaches to this problem in Sections 4.1.1 and 4.1.2.

Corollary 4.4. *If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{NP} = \mathsf{UP} = \mathsf{RP}$ *and in particular,* $\mathsf{PH} = \mathsf{BPP}$.

Proof. If $\mathsf{CF} = \mathsf{PEq}$ then it follows directly from Theorems 4.1 and 4.3 that $\mathsf{NP} = \mathsf{UP} \subseteq \mathsf{RP}$. Thus $\mathsf{NP} = \mathsf{RP}$, since $\mathsf{RP} \subseteq \mathsf{NP}$ without any assumptions. Furthermore, it follows that $\mathsf{PH} \subseteq \mathsf{BPP}$ [77], and since $\mathsf{BPP} \subseteq \mathsf{PH}$ [52, 70], the two are equal. □

The collapse inferred here is stronger than that of Corollary 3.5, since $\mathsf{BPP} \subseteq \mathsf{S_2P} \subseteq \mathsf{S_2[NP \cap coNP]}$ [61, 26]. However, this result is incomparable to Corollary 3.5 since it also makes the stronger assumption $\mathsf{CF} = \mathsf{PEq}$, rather than only assuming $\mathsf{CF} = \mathsf{Ker}$.

4.1.1. Promise classes

One way to extend the first half of Theorem 4.3 from UP to NP, suggested to us by Scott Aaronson [2], involves promise versions of PEq and Ker.

Definition 4.5. A language R of triples is in PromisePEq if there is a polynomial-time algorithm A such that, whenever $R_a = \{(x, y) : (a, x, y) \in R\}$ is an equivalence relation, $A(a, x, y) = R(a, x, y)$ for all $x, y \in \Sigma^*$.

Similarly, R is in PromiseKer if there is a polynomial-time function f such that, whenever R_a is an equivalence relation, $f(a, x) = f(a, y) \iff (a, x, y) \in R$ for all $x, y \in \Sigma^*$. We call such f a *promise complete invariant* for R.

As usual for promise classes, if R_a is not an equivalence relation, we do not restrict the output of $A(a, x, y)$ or $f(a, x)$ in any way.

Theorem 4.6. *If* PromiseKer = PromisePEq *then* NP $\subseteq$ BQP $\cap$ SZK*, and in particular* PH = AM*.*

Proof. The first part of the proof follows that of Theorem 4.3, treating the promises with care. Suppose PromiseKer = PromisePEq. Let L be a language in PromiseUP, let V be a PromiseUP verifier for L, let p be a polynomial bounding the size of V-witnesses for L. That is, if $\#V(x) = \#\{y : V(x, y) = 1\} \leq 1$ then $x \in L \iff (\exists y)[|y| \leq p(|x|)$ and $V(x, y) = 1]$. Consider the relation

$$R_L = \{((a, x), (a, y)) : x = y \text{ or } |x| = |y| \text{ and } V(a, x \oplus y) = 1\}$$

(the same relation as in Theorem 4.3). Clearly $R_L \in$ PromisePEq, so by hypothesis R_L has a promise complete invariant $f \in$ FP. Since $L \in$ PromiseUP, for each $a \in L$ such that $\#V(x) = 1$, there is a unique string w_a such that $V(a, w_a) = 1$. Define $f_a(x) = f(a, x)$. Then for all distinct x and x', $f_a(x) = f_a(x')$ if and only if $x \oplus x' = w_a$. As in Theorem 4.3, given a and f_a, finding w_a or determining that there is no such string is exactly Simon's problem, which is in BQP [66]. Here, of course, we have reduced to the *promise version* of Simon's problem.

To show NP $\subseteq$ BQP, we use the technique of Valiant and Vazirani [76]: given a Boolean formula φ, they randomly produce a formula φ' such that if φ is unsatisfiable, then so is φ', and if φ is satisfiable, then φ' has a *unique* satisfying assignment with probability at least $1/p(|\varphi|)$ for some polynomial p. In this case, $(\varphi', f_{\varphi'})$ satisfies the promise of Simon's problem, and the BQP algorithm for Simon's problem either finds the satisfying assignment to φ' or correctly reports that none exists. Since the initial randomized construction of φ' from φ can also be carried out in BQP, this whole algorithm puts $SAT \in$ BQP.

Next we show NP $\subseteq$ SZK. As above, we randomly transform a Boolean formula φ into a formula φ' which has at most one satisfying assignment, with probability at least $1/p(|\varphi|)$. Then we run the SZK protocol for Simon's problem on φ', which we reproduce here for completeness. If $\varphi'(00\cdots0) = 1$, then the verifier accepts immediately. Otherwise, the verifier randomly picks x and sends $f_{\varphi'}(x) = f(\varphi', x)$ to the prover; the prover must try to recover x. If φ' has no satisfying assignments, then $f_{\varphi'}$ is one-to-one, and the prover always succeeds. If φ' has a (unique, not-all-zero) satisfying assignment, then $f_{\varphi'}$ is two-to-one, and the prover fails with probability at least $1/2$. It is clear that this is an SZK protocol.

Since the construction of φ' from φ does not require any interaction between the prover and verifier, it can be prepended to the above protocol to give a statistical zero-knowledge protocol for SAT.

Finally, we have $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ [31, 5], and $\mathsf{NP} \subseteq \mathsf{coAM}$ implies $\mathsf{PH} = \mathsf{AM}$ [9, 21]. □

The two conclusions of the above theorem (that is, "$\mathsf{NP} \subseteq \mathsf{BQP}$" and "$\mathsf{PH} = \mathsf{AM}$") are not known to be related by implication in either direction. Even $\mathsf{NP} \subseteq \mathsf{BQP}$ and $\mathsf{NP} \subseteq \mathsf{SZK}$ are not known to be related by implication. Indeed, there is an oracle relative to which SZK is not contained in BQP [1], and there is an oracle relative to which BQP is not contained in SZK [27].

4.1.2. Groupy witnesses for NP *problems*

The technique of the first half of Theorem 4.3 does not apply to arbitrary problems in NP. However, if an NP problem's witnesses satisfy a certain group-like condition, then Theorem 4.3 may be extended to that problem.

Let $L \in \mathsf{NP}$ and let V be a polynomial-time verifier for L. By padding if necessary, we may suppose that for each $a \in L$, a's witnesses all have the same length. Suppose there is a polynomial-time length-restricted group structure on Σ^*, that is, a function $f \in \mathsf{FP}$ such that for each length n, Σ^n is given a group structure defined by $xy^{-1} \stackrel{def}{=} f(x, y)$. Then

$$R_L = \{((a, x), (a, y)) : x = y \text{ or } V(a, xy^{-1}) = 1\}$$

is an equivalence relation if and only if a's witnesses are a subgroup of this group structure, or a subgroup less the identity. The technique of Theorem 4.3 then reduces L to the hidden subgroup problem over the family of groups defined by f.

The *hidden subgroup problem*, or HSP, for a group G is: given generators for G, an oracle computing the operation $(x, y) \mapsto xy^{-1}$, a set X, and a function $f\colon G \to X$ such that $\mathrm{Ker}(f)$ is the partition given by the right cosets of some subgroup $H \leq G$, find a generating set for H [46]. Hidden subgroup problems have played a central role in the study of quantum algorithms. Integer factoring and the discrete logarithm problem both easily reduce to abelian HSPs. The first polynomial-time quantum algorithm for these problems was discovered by Shor [65]; Kitaev [46] then noticed that Shor's algorithm in fact solves all abelian HSPs. The unique shortest vector problem for lattices reduces to the dihedral HSP [59], which is solvable in subexponential quantum time [50]. The graph isomorphism problem reduces to the HSP for the symmetric group [16] or the wreath product $S_n \wr S_2$ [29], but it is still unknown whether any nontrivial quantum algorithm exists for GI.

The proof of Theorem 4.3 showed that if $\mathsf{Ker} = \mathsf{PEq}$ then every language in UP reduces to Daniel Simon's problem. We can now see that Simon's problem is in fact the HSP for $(\mathbb{Z}/2\mathbb{Z})^n$, where the hidden subgroup has order 2. Simon [66] gave a zero-error expected polynomial-time quantum algorithm for this problem, putting it in $\mathsf{ZQP} \subseteq \mathsf{BQP}$. This result was later improved by Brassard and Høyer [22] to a worst-case polynomial time quantum algorithm, that is, in the class EQP (sometimes referred to as just QP).

This discussion motivates the following definition, results, and open question:

Definition 4.7. Let $L \in \mathsf{NP}$. For each a let $W(a)$ denote the set of a's witnesses; without loss of generality, by padding if necessary, assume that $W(a) \subseteq \Sigma^n$ for some n. The language L has *groupy witnesses* if there are functions $\mathrm{mul}, \mathrm{gen}, \mathrm{dec} \in \mathsf{FP}$ such that for each $a \in L$:

1. let $G(a) = \{x \in \Sigma^n : \mathrm{dec}(a, x) = 1\}$; then for all $x, y \in G(a)$, defining $xy^{-1} \stackrel{def}{=} \mathrm{mul}(a, x, y)$ gives a group structure to $G(a)$;
2. $\mathrm{gen}(a) = (g_1, g_2, \ldots, g_k)$ is a generating set for $G(a)$; and
3. $W(a)$ is a subgroup of $G(a)$, or a subgroup less the identity.

The following results are corollaries to the proof, rather than to the result, of Theorem 4.3.

Corollary 4.8. *If* $\mathsf{Ker} = \mathsf{PEq}$ *and a language* $L \in \mathsf{NP}$ *has groupy witnesses in a family* $\mathcal{G}$ *of groups, then* L *Cook-reduces to the hidden subgroup problem for the family* $\mathcal{G}$*. Briefly:* $L \leq_T^P HSP(\mathcal{G})$*.*

Proof. Let $L \in \mathsf{NP}$, let W, G, dec, mul, and gen be as in the definition of groupy witnesses, and let V be a polynomial-time verifier for L such that the witnesses accepted by V on input a are exactly the strings in $W(a)$. Then the equivalence relation

$$R_L = \{((a, x), (a, y)) : x = y, \text{ or } \mathrm{dec}(a, x) = \mathrm{dec}(a, y) \text{ and } [\mathrm{dec}(a, x) = 1 \implies V(a, xy^{-1}) = 1]\}$$

is in PEq, since xy^{-1} can be computed by the polynomial-time algorithm mul guaranteed in the definition of groupy witnesses. By hypothesis, R_L has a complete invariant f. The function f, the function mul, and the generating set $\mathrm{gen}(a)$ are a valid instance of the hidden subgroup problem. If $a \notin L$, then f is injective, and the hidden subgroup is trivial. If $a \in L$, then the hidden subgroup is $W(a)$. Conversely, if the hidden subgroup is trivial, then either $a \notin L$ or the identity of the group is a witness that $a \in L$, which can be easily checked. Hence L reduces to the hidden subgroup problem. □

Corollary 4.9. *If* $\mathsf{Ker} = \mathsf{PEq}$ *and the language* L *has abelian groupy witnesses, then* $L \in \mathsf{BQP}$*.* □

Lemma 4.10. *Every language in* UP *has abelian groupy witnesses.* □

Open Question 4.11. Are there NP-complete problems with abelian groupy witnesses? Assuming $\mathsf{P} \neq \mathsf{NP}$, are there any problems in $\mathsf{NP} \backslash \mathsf{UP}$ with abelian groupy witnesses?

Our definition of having groupy witnesses is similar but not identical to Arvind and Vinodchandran's definition of group-definability [8]. If a set $A \in \mathsf{NP}$ has abelian groupy witnesses, then in

general the function $a \mapsto |G(a)|$ is in #P. If it so happens that this function is in FP, then Arvind and Vinodchandran's techniques are sufficient to show that A is low for PP. This may or may not be taken as evidence that such an A is unlikely to be NP-complete: on the one hand, Beigel [17] gives an oracle relative to which NP is *not* low for PP, and hence A could not be NP-complete. On the other hand, Toda and Ogiwara [75] show that $\mathsf{PP}^{\mathsf{PH}} \subseteq \mathsf{BP} \cdot \mathsf{PP}$ (Tarui [73], independently but using similar methods, strengthens this to $\mathsf{ZP} \cdot \mathsf{PP}$). Hence, under a derandomization assumption, NP *is* in fact low for PP, and so the lowness of A for PP is no obstruction to its being NP-complete.

However, even if $|G(a)|$ is computable in polynomial time, it may yet be possible to use Corollary 4.8 to show that $\mathsf{Ker} = \mathsf{PEq} \implies \mathsf{NP} \subseteq \mathsf{BQP}$, as there are several classes of non-abelian, and even non-solvable, groups for which the HSP is known to be in BQP (see, e. g., [36, 32, 42]).

4.2. Hardness

4.2.1. Collision-free hash functions

Collision-free hash functions are a useful cryptographic primitive (see, e. g., [15]). Proposition 1.7 suggests a more general connection between the collapse $\mathsf{CF} = \mathsf{Ker}$ and the existence of collision-free hash functions.

A *collection of collision-free hash functions* is a collection of functions $\{h_i : i \in I\}$ for some $I \subseteq \Sigma^*$ where $h_i : \Sigma^{|i|+1} \to \Sigma^{|i|}$ are

1. Easily accessible: there is a probabilistic polynomial-time algorithm G such that $G(1^n) \in \Sigma^n \cap I$;
2. Easy to evaluate: there is a probabilistic polynomial-time algorithm E such that $E(i, w) = h_i(w)$; and
3. Collision-free: for all probabilistic polynomial-time algorithms A and all polynomials p there is a length N such that $n > N$ implies:

$$\Pr_{\substack{i=G(1^n) \\ (x,y)=A(i)}} [x \neq y \text{ and } h_i(x) = h_i(y)] < \frac{1}{p(n)}.$$

It is not known whether collections of collision-free hash functions exist, though their existence is known to follow from other cryptographic assumptions (see, e. g., [28]). Many proposed collections of collision-free hash functions, such as MD5 or SHA, can be evaluated deterministically, that is, $E \in \mathsf{FP}$.

Proposition 4.12. *If* $\mathsf{CF} = \mathsf{Ker}$ *then collision-free hash functions that can be evaluated in deterministic polynomial time do not exist.*

Proof. The equivalence relation $\{((i,x),(i,y)) : E(i,x) = E(i,y)\}$ has a canonical form $f \in \mathsf{FP}$ by hypothesis. As in the proof of Proposition 1.7, the canonical form f can be used by a randomized algorithm to find collisions in h_i with non-negligible probability: choose x at random, and if $f(x) \neq x$ then a collision has been found.

Since h_i maps $\Sigma^{|i|+1} \to \Sigma^{|i|}$, there are at most $2^{|i|} - 1$ singleton classes in $R = \mathrm{Ker}(h_i)$. If x lies in an equivalence class of size at least 2, then $\Pr_x[f(x) \neq x | \#[x]_R \geq 2] \geq \frac{1}{2}$. Thus $\Pr_x[f(x) \neq x] = \Pr_x[f(x) \neq x | \#[x]_R \geq 2] \Pr_x[\#[x]_R \geq 2] \geq \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2^{|i|+1}}\right) > \frac{1}{4}$. □

4.2.2. Subgroup equality

The *subgroup equality problem* is: given two subsets $\{g_1, \ldots, g_t\}$, $\{h_1, \ldots, h_s\}$ of a group G determine if they generate the same subgroup. The *group membership problem* is: given a group G and group elements $g_1, \ldots, g_t, x$, determine whether or not $x \in \langle g_1, \ldots, g_t \rangle$. A solution to the group membership problem yields a solution to the subgroup equality problem, by determining whether each h_i lies in $\langle g_1, \ldots, g_t \rangle$ and vice versa. However, a solution to the group membership problem does *not* obviously yield a complete invariant for the subgroup equality problem. Thus subgroup equality problems are a potential source of candidates for problems in $\mathsf{PEq} \backslash \mathsf{Ker}$.

Note that the complexity of these problems still makes sense for non-finite groups, so long as group elements can be specified by finite strings and the group operations are computable.

Fortunately or unfortunately, the subgroup equality problem for permutation groups on $\{1, \ldots, n\}$ has a polynomial-time canonical form, via a simple modification [10] of classicial techniques [68, 69, 34, 47] (see Example 1.6 for more of the history).

4.2.3. Boolean function congruence

Two Boolean functions f and g are *congruent* if the inputs to f can be permuted and possibly negated to make f equivalent to g. If f and g are given by formulae φ and ψ, respectively, deciding whether φ and ψ define congruent functions is Karp equivalent to FI. If f and g are given by their truth tables, however, Luks [54] gives a polynomial-time algorithm for deciding whether or not they are congruent. Yet no polynomial-time complete invariant for Boolean function congruence is known. Hence function congruence may be in $\mathsf{PEq} \backslash \mathsf{Ker}$.

4.2.4. Complete problems?

Equivalence problems that are P-complete under NC or L reductions may lie in $\mathsf{PEq} \backslash \mathsf{Ker}$ due to their inherent difficulty. However, we currently have no reason to believe that P-completeness

is related to complexity classes of equivalence problems. Towards this end, we introduce a natural notion of reduction for equivalence problems:

Definition 4.13. An equivalence relation R *kernel-reduces* to an equivalence relation S, denoted $R \leq^P_{ker} S$, if there is a function $f \in \mathsf{FP}$ such that

$$x \sim_R y \iff f(x) \sim_S f(y).$$

Note that $R \in \mathsf{Ker}$ if and only if R kernel-reduces to the relation of equality. Also note that if $R \leq^P_{ker} S$ via f, then $R \leq^P_m S$ via $(x, y) \mapsto (f(x), f(y))$, leading to the question:

Open Question 4.14. Are kernel reduction and Karp reduction different? Are they different on PEq? In other words, are there two equivalence relations R and S (in PEq?) such that $R \leq^P_m S$ but $R \not\leq^P_{ker} S$?

An equivalence relation $R \in \mathsf{PEq}$ is PEq-*complete* if every $S \in \mathsf{PEq}$ kernel-reduces to R. For any PEq-complete R, $R \in \mathsf{Ker}$ if and only if $\mathsf{Ker} = \mathsf{PEq}$ if and only if the relation of equality is PEq-complete.

Unlike NP-completeness, however, the notion of PEq-completeness does not become trivial if $\mathsf{Ker} = \mathsf{PEq}$: the relation of equality does not kernel-reduce to the trivial relation simply because equality has infinitely many equivalence classes but the trivial relation has only one. In particular, if $\mathsf{P} = \mathsf{NP}$ then kernel reduction and Karp reduction are distinct on PEq, albeit in a rather trivial way. The question becomes more interesting if we ask for languages R and S in PEq *of the same densities* on which kernel reduction and Karp reduction differ.

Open Question 4.15. Are there PEq-complete equivalence problems?

5. Oracles

In order to combine the oracles from Blass and Gurevich [18] into a single oracle, as well as construct new oracles that simultaneously separate some classes of equivalence relations and collapse others, we introduce two notions of generic oracle. Generic oracles maintain some of the key advantages of random oracles, but allow us much greater flexibility—much of the power of finite injury arguments—in their construction[1]. For example, it is often possible to show that some property (complexity class collapse or separation) holds relative to *every* generic oracle, so that it

[1]Indeed, there is a notion of genericity $\mathcal{R}$ such that results regarding $\mathcal{R}$-generic oracles are completely equivalent to results regarding random oracles [71] (see also [30], the paragraph just prior to Section 3.2), so generic oracle constructions can be viewed as an extension of random oracle constructions.

becomes much easier to construct oracles satisfying multiple properties at once. We begin with a review of generic oracle constructions; for a more in-depth discussion, see Fenner, Fortnow, Kurtz, and Li [30].

For those not interested in the technical details of generic oracles, the main result we will need from the next section is Lemma 5.4, but we have attempted to keep the technicalities to a minimum. We only use fairly restricted versions of genericity[2] and all the associated concepts in this paper, allowing us to greatly simplify their discussion. Much more general versions and their uses are presented in Fenner, Fortnow, Kurtz, and Li [30].

5.1. Preliminaries on Generic Oracles

Throughout this section we will use the first construction of an oracle separating P from NP [14] as a canonical example.

Many oracle constructions proceed by finite extensions: at each stage of the construction, some requirement is to be satisfied (e.g. "the i-th polynomial-time machine does not accept some fixed relativizable language L^O"), and we satisfy it by specifying the oracle on finitely many more strings, leaving those strings we have previously specified untouched. In this paper, a generic oracle is one built by finite extensions which also satisfies Murphy's law: "anything which can happen will happen." More prosaically, a generic oracle is built by interleaving all finite extension arguments that are "interleavable." In the remainder of this section we make these ideas precise.

A *condition* is a partial characteristic function whose domain is finite, that is, a partial function $\sigma\colon \Sigma^* \to \{0,1\}$ with $\mathrm{dom}(\sigma)$ finite. In more general discussions of genericity, such conditions are called *Cohen conditions*. We say that an oracle O *extends* σ if the characteristic function of O agrees with σ on $\mathrm{dom}(\sigma)$. Two conditions σ_1, σ_2 are *consistent* if for every $a \in \mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2)$ we have $\sigma_1(a) = \sigma_2(a)$.

Terminologically we treat a partial characteristic function as a partial oracle/set: we write $a \in \sigma$ and say "a is in σ" if $\sigma(a) = 1$, and similarly we write $a \notin \sigma$ and "a is not in σ" if $\sigma(a) = 0$. We are careful not to use either terminology if $a \notin \mathrm{dom}(\sigma)$.

Definition 5.1. A *notion of genericity* is a nonempty set $\mathcal{G}$ of conditions such that

[2]For the initiated: rather than treat conditions in general as perfect collections of oracles, we define a condition as a partial characteristic function with finite domain. We also require a strong form of basicness: the union of any two consistent $\mathcal{G}$-conditions (union as partial characteristic functions) must also be a $\mathcal{G}$-condition.

0. (branching) for all $\sigma \in \mathcal{G}$, there are at least two distinct conditions $\tau_1, \tau_2 \in \mathcal{G}$ extending σ;
1. (generic) for all $\sigma \in \mathcal{G}$ and all $a \in \Sigma^* \setminus \mathrm{dom}(\sigma)$ there is a condition $\sigma' \in \mathcal{G}$ extending σ such that $a \in \mathrm{dom}(\sigma')$; and
2. (basic) if $\sigma_1, \sigma_2 \in \mathcal{G}$ are consistent, then $\sigma_1 \cup \sigma_2 \in \mathcal{G}$.

Note that the collection of all (Cohen) conditions is a notion of genericity, typically referred to as Cohen genericity. Less trivial is the notion of UP-genericity. A UP condition is a condition which has at most one string of each length, and only has strings at lengths $tower(k)$, where the *tower* function is defined by $tower(0) = 1$ and $tower(n+1) = 2^{tower(n)}$. The collection of all UP conditions yields the notion of UP-genericity.

A $\mathcal{G}$-generic oracle is simply one built by further and further specification by $\mathcal{G}$-conditions which satisfies an additional constraint, namely, the formal version of "Murphy's law" which we now present.

Throughout this section we fix a logical system that is strong enough to express all the sentences we care about; for example, Peano Arithmetic with an additional unary predicate X, corresponding to the oracle, will suffice. If φ is a sentence in such a system, then an oracle O satisfies φ if φ is true upon replacing the predicate X by the characteristic function for O. We assume, without loss of generality from the point of view of our constructions, that the logical system has only countably many sentences.

We say that a condition σ *forces* the truth of a sentence φ if φ is true of every oracle O extending σ. For example, φ might be the sentence

$$(\exists n)[M(1^n) = 0 \iff (\exists x)[|x| = n \text{ and } X(x)]]. \tag{1}$$

The classic argument of Baker, Gill, and Solovay [14] shows how to construct a Cohen condition forcing φ. That is, we only need to specify a finite amount of the oracle to ensure that φ is true, regardless of how we construct the rest of the oracle.

We say that a notion of genericity $\mathcal{G}$ is strong enough to force a sentence φ if φ can *always* eventually be forced, that is, for every $\mathcal{G}$-condition σ there is another $\mathcal{G}$-condition σ' extending σ such that σ' forces φ. We say, equivalently, that $\{\sigma \in \mathcal{G} : \sigma \text{ forces } \varphi\}$ is *dense* in $\mathcal{G}$. In fact Baker, Gill, and Solovay essentially showed that Cohen genericity is strong enough to force (1).

Finally, "Murphy's law," which we require of generic oracles, is that a $\mathcal{G}$-generic oracle must force every sentence φ that $\mathcal{G}$ is strong enough to force.

Definition 5.2 (Generic Oracle)**.** Let $\mathcal{G}$ be a notion of genericity. An oracle O is $\mathcal{G}$-generic if there is a consistent collection of $\mathcal{G}$-conditions $\{\sigma_1, \sigma_2, \dots\}$ such that O extends every σ_i, the σ_i fully specify O (that is, $\bigcup_i \mathrm{dom}(\sigma_i) = \Sigma^*$), and every sentence φ that $\mathcal{G}$ is strong enough to force is forced by some σ_i.

We see that this definition essentially captures the idea of simultaneously interleaving all constructions that "can be interleaved," that is, that $\mathcal{G}$ is strong enough to force.

Lemma 5.3 (Existence of $\mathcal{G}$-generic oracles)**.** *For every notion of genericity $\mathcal{G}$, $\mathcal{G}$-generic oracles exist. Furthermore, the $\mathcal{G}$-generics are dense in $\mathcal{G}$, that is, for every $\mathcal{G}$-condition σ there is a $\mathcal{G}$-generic oracle extending σ.*

Proof. This is essentially Lemma 3.12 of Fenner, Fortnow, Kurtz, and Li [30], and their proof goes through *mutatis mutandis*, despite our restricted definitions. □

Putting this all together, the way we construct generic oracles in practice is captured by the following lemma:

Lemma 5.4. *Let $\mathcal{G}$ be a notion of genericity and φ a sentence. If $\mathcal{G}$ is strong enough to force φ—that is, if every $\sigma \in \mathcal{G}$ can be extended to a $\sigma' \in \mathcal{G}$ forcing φ—then* every *$\mathcal{G}$-generic oracle satisfies φ.*

Finally, this entire discussion relativizes. When we relativize to an oracle A, our formal system includes a new unary predicate which is the characteristic function of A, in addition to the previous unary predicate X corresponding to the generic oracle. We then speak of $\mathcal{G}$-generics relative to A.

5.2. Oracles for PEq*,* Ker*, and* CF

In this section we introduce and use two new notions of genericity. A *one-sided transitive* condition is a (Cohen) condition τ such that

1. (Length restriction on the 1-side): $1\langle x, y\rangle \in \tau$ implies $|x| = |y|$, and
2. (Transitivity on the 1-side): $1\langle x, y\rangle \in \tau$ and $1\langle y, z\rangle \in \tau$ implies $1\langle x, z\rangle \in \tau$.

We refer to the set of strings starting with the bit b as "the b-side" of an oracle or condition. Note that in a one-sided transitive condition, all we require of the 0-side is that $\mathrm{dom}(\sigma)$ is finite there. It is easily verified that one-sided transitive conditions form a notion of genericity, so by Lemma 5.3, one-sided transitive generics exist, and furthermore Lemma 5.4 applies to them.

A UP*-transitive condition* is a condition τ such that

1. ("UP") For each length n, there is at most one string of length n in σ;

2. (gappy) σ is only nonempty at lengths $tower(k)$ for some k. The *tower* function is defined by $tower(0) = 1$ and $tower(n) = 2^{tower(n-1)}$;
3. (length-restricted) $\langle x, y\rangle \in \sigma$ implies $|x| = |y|$.

Note that transitivity—$\langle x, y\rangle \in \tau$ and $\langle y, z\rangle \in \tau$ implies $\langle x, z\rangle \in \tau$—follows from the UP restriction (1) and the length restriction (3). Again it is easily verified that UP-transitive conditions form a notion of genericity, so UP-transitive generics exist, and Lemma 5.4 applies to them.

Theorem 5.5. *There are oracles A and B relative to which* $\mathsf{P} \neq \mathsf{NP}$ *and*

$$\mathsf{CF}(\mathsf{FP}^A) \neq \mathsf{Ker}(\mathsf{FP}^A) \neq \mathsf{P}^A\mathsf{Eq}, \tag{1}$$

$$\mathsf{CF}(\mathsf{FP}^B)_p = \mathsf{Ker}(\mathsf{FP}^B)_p \text{ and } \mathsf{Ker}(\mathsf{FP}^B) \neq \mathsf{P}^B\mathsf{Eq}. \tag{2}$$

In fact, (1) holds relative to any one-sided transitive generic oracle and (2) holds relative to $O \oplus G$ whenever O is PSPACE*-complete and G is* UP*-transitive generic relative to O.*

We break most of the proof into three lemmas. The proofs of Lemmas 5.7 and 5.8 are adaptations of the proofs of Blass and Gurevich [18] to generic oracles. The proof of Lemma 5.9 is new.

We start by restating a useful combinatorial lemma:

Lemma 5.6 (Blass & Gurevich [18] Lemma 1)**.** *Let G be a directed graph on $2k$ vertices such that the out-degree of each vertex is strictly less than k. Then there are two nonadjacent vertices in G.*

Lemma 5.6 can be proved by a simple counting argument.

For UP-transitive conditions σ (or oracles O) we denote by $\sim_\sigma$ the corresponding equivalence relation, that is, the reflexive, symmetric closure of $\{(x, y) : \langle x, y\rangle \in \sigma\}$. If σ is only a partial function, we take care to only ever write $x \sim_\sigma y$ if $\langle x, y\rangle \in \mathrm{dom}(\sigma)$. For one-sided transitive conditions τ, we use the same notation $\sim_\tau$ to denote the equivalence relation corresponding to the 1-side, that is, the reflexive, symmetric closure of $\{(x, y) : 1\langle x, y\rangle \in \tau\}$.

Lemma 5.7. *Relative to any one-sided transitive generic oracle or any* UP*-transitive generic oracle,* $\mathsf{Ker} \neq \mathsf{PEq}$.

Proof. The proofs for the two types of genericity are essentially identical. Let $\mathcal{G}$ be "one-sided transitive" or "UP-transitive" throughout. We give the proof for one-sided transitive genericity, in which all the diagonalization happens on the 1-side; for UP-transitive genericity, drop the prefixed 1's throughout and only add strings at lengths $n = tower(k)$ for some k.

For each polynomial-time oracle Turing machine M, let φ_M denote the sentence (often called a requirement):

$$\varphi_M \stackrel{def}{=} (\exists n)[\mathrm{Ker}(M^X) \neq\!\sim_X \text{ on strings of length } n]$$

By Lemma 5.4, it suffices to show that any $\mathcal{G}$-condition τ can be extended to a $\mathcal{G}$-condition τ' such that τ' forces φ_M. For then φ_M will hold for every $\mathcal{G}$-generic oracle and for every M, separating Ker from PEq.

Let M be a polynomial-time oracle transducer running in time $p(|x|)$. Let τ be any $\mathcal{G}$-condition. Let $\overline{\tau}$ denote the minimal (under inclusion) extension of τ to a complete characteristic function (i. e., oracle). We show how to extend τ to another $\mathcal{G}$-condition τ' that forces φ_M, i. e., such that $\mathrm{Ker}(M^O) \neq \sim_O$ for any O extending τ'.

Let n be a length such that $p(n) < 2^{n-1}$ and τ is not defined on $1\langle a, b\rangle$ for any strings a and b of length $\geq n$. Let τ' be the extension of τ to length $p(n)$ that is equal to $\overline{\tau}$ to length $p(n)$. If there are distinct strings x and y of length n such that $M^{\overline{\tau}}(x) = M^{\overline{\tau}}(y)$, then $x \not\sim_{\tau'} y$ but $M^{\tau'}(x) = M^{\tau'}(y)$, and this clearly holds for any O extending τ'.

Otherwise, $M^{\overline{\tau}}(x) \neq M^{\overline{\tau}}(y)$ for every two distinct strings x and y. Say that x *affects* y if M queries $\overline{\tau}$ about $1\langle x, y\rangle$ or $1\langle y, x\rangle$ in the computation of $M^{\overline{\tau}}(y)$. Let G be a digraph on the strings of length n, in which there is a directed edge from y to x if x affects y. The out-degree of each vertex is at most $p(n)$, which is strictly less than 2^{n-1} by the choice of n. Since there are 2^n vertices, Lemma 5.6 implies that there are two strings x and y of length n such that neither affects the other. Put $1\langle x, y\rangle$ into τ'. Then $M^{\tau'}(x) \neq M^{\tau'}(y)$ but $x \sim_{\tau'} y$, and this holds for any oracle O extending τ'.

Thus $\mathsf{Ker}^O \neq \mathsf{PEq}^O$ relative to any $\mathcal{G}$-generic oracle O, for $\mathcal{G}$ either "one-sided transitive" or "UP-transitive." □

Lemma 5.8. *Relative to any one-sided transitive generic oracle,* $\mathsf{CF} \neq \mathsf{Ker}$.

Proof. For this proof, all the diagonalization is performed on the 0-side.

We describe our oracles O and conditions τ with values in the alphabet $\{0, 1, 2\}$ for simplicity (that is, $\tau\colon \Sigma^* \to \{0, 1, 2\}$). Let $read^O\colon \Sigma^* \to \Sigma^*$ denote the oracle function

$$read^O(x) = O(0x01)O(0x011)\cdots O(0x01^{k-1})$$

where k is the least value such that $O(0x01^k) = 2$. Note that the bits used by $read^O$ on input x are disjoint from those used by $read^O$ on any input $y \neq x$. Also note that $read^O$ only queries the oracle regarding strings on the 0-side. Let $R^O = \mathrm{Ker}(read^O)$.

Let f be any polynomial-time oracle transducer, and define

$$\psi_f \stackrel{def}{=} (\exists n)[f^X \text{ is not a canonical form for } R^X \text{ on strings of length } n].$$

As in Lemma 5.7, it suffices to show that any one-sided transitive condition τ can be extended to a one-sided transitive condition τ' forcing ψ_f, by Lemma 5.4.

Let f be a polynomial-time oracle transducer running in time $p(|x|)$. Let τ be a one-sided transitive condition, and let $\overline{\tau}$ denote the oracle extending τ which has value 2 on strings of the form $0x$ that are not in $\mathrm{dom}(\tau)$ and value 0 on all other strings not in $\mathrm{dom}(\tau)$. We show how to extend τ to a one-sided transitive condition τ' such that f^O does not compute a canonical form for R^O for any O extending τ'.

Let n be a length such that $p(n) < 2^{n-1}$ and such that τ is not defined for any strings $0x$ with $|x| \geq n$. For a string x of length n, let τ_x denote the minimal extension of τ such that $read^{\overline{\tau_x}}$ is the identity on all strings of length n, except $read^{\overline{\tau_x}}(x) = 1^{n+1}$. Since the $read$ function only queries strings on the 0-side, τ_x differs from τ only on the 0-side, and we do not need to worry

about violating transitivity on the 1-side. Note that $read^{\overline{\tau_x}}$ is injective on strings of length n, so its kernel at length n is the relation of equality. In particular, any canonical form for $R^{\overline{\tau_x}}$ must be the identity on strings of length n.

If there is an x of length n such that $f^{\overline{\tau_x}}(x) \neq x$, then $f^{\overline{\tau_x}}(x)$ is not the identity on strings of length n, so $f^{\overline{\tau_x}}$ is not a canonical form for $R^{\overline{\tau_x}}$. Let the extension τ' be $\overline{\tau_x}$ up to length $p(n)$.

Otherwise, $f^{\overline{\tau_x}}(x) = x$ for all x of length n. We say that $f^O(x)$ queries the oracle about y if $f^O(x)$ queries any of the strings that $read^O(y)$ queries. Find x and y of length n such that $f^{\overline{\tau_x}}(x)$ does not query the oracle about y and $f^{\overline{\tau_y}}(y)$ does not query the oracle about x. This is possible by Lemma 5.6, as in the proof of Lemma 5.7. Let τ' be the minimal oracle extending τ such that $read^{\tau'}$ is the identity on strings of length n, except $read^{\tau'}(x) = read^{\tau'}(y) = 1^{n+1}$. Then τ' differs from $\overline{\tau_x}$ only on those strings in its domain queried by $read^{\tau'}(y)$ and τ' differs from $\overline{\tau_y}$ only on those strings in its domain queried by $read^{\tau'}(x)$. Since $f^{\overline{\tau_x}}(x)$ does not query the oracle about y we have $f^{\overline{\tau_x}}(x) = f^{\tau'}(x) = x$ and similarly $f^{\overline{\tau_y}}(y) = f^{\tau'}(y) = y$. So relative to any oracle O extending τ', we have $(x, y) \notin \mathrm{Ker}(f^O)$ but $read^O(x) = read^O(y) = 1^{n+1}$. Again, τ' forces that $f^{\tau'}$ is not a canonical form for $R^{\tau'}$.

Thus $\mathsf{CF}^O \neq \mathsf{Ker}^O$ relative to any one-sided transitive generic oracle O. □

Lemma 5.9. *If* $\mathsf{P} = \mathsf{PSPACE}$*, and* O *has at most one string of each length* $tower(k)$ *and no other strings, then* $\mathsf{CF}(\mathsf{FP}^O)_p = \mathsf{Ker}(\mathsf{FP}^O)_p$*. Furthermore, this result relativizes.*

Proof. Let O have at most one string of each length $tower(k)$, and no other strings. Let f be an oracle transducer running in polynomial time $p(|x|)$, let $R = \mathrm{Ker}(f^O)$, and suppose that $\langle x, y\rangle \in R$ implies $|x| \leq q(|y|)$ for some polynomial q. For any input x of sufficient length, all elements of O except possibly one have length either $\leq \log p(|x|)$, in which case they can be found rapidly, or $> p(q(|x|))$ in which case they cannot be queried by f on any input $y \sim_R x$. Following a technique used in [23], we call this one element the "cookie" for this equivalence class.

For the remainder of this proof, "minimum," "least," etc. will be taken with respect to the standard length-lexicographic ordering.

We show how to efficiently compute a canonical form for R. Let R_y denote the inverse image of y under f^O, which is an R-equivalence class. Let

$$B_y \quad = \quad \{x : f^O(x) = y \text{ and } f^O(x) \text{ does not query the cookie}\},$$

$r_y = \min R_y$, and $b_y = \min B_y$. A canonical form for R is

$$g(x) = \begin{cases} b_y & \text{if } B_y \neq \emptyset \\ r_y & \text{otherwise,} \end{cases}$$

where $y = f^O(x)$. Now we show that g is in fact in FP^O. On input x, the computation of g proceeds as follows:

1. Find all elements of O of length at most $\log p(|x|)$. Any further queries to O of length $\leq \log p(|x|)$ will be simulated without queries by using this data.
2. Compute $y = f^O(x)$.
3. If the cookie was queried, then *all* further queries to O will be simulated without queries using this data. Using the power of PSPACE, determine whether or not $B_y = \emptyset$. If $B_y = \emptyset$, find and output r_y. If $B_y \neq \emptyset$, find and output b_y.

4. If the cookie was not queried, then $x \in B_y$, so $B_y \neq \emptyset$. Use the power of PSPACE to find the least z such that $f(z) = y$, answering 0 to any queries made by f to strings of length ℓ between $\log p(|x|) < \ell \leq p(q(|x|))$.
5. Run $f^O(z)$. If $f^O(z)$ did not query the cookie, then $f^O(z) = f(z) = y$ and $z = b_y$, so output z. Otherwise, $f^O(z)$ queried the cookie, so no further oracle queries need be made. Using the power of PSPACE, find and output b_y.

□

Proof of Theorem 5.5. ($\mathsf{CF} \neq \mathsf{Ker} \neq \mathsf{PEq}$) By Lemmas 5.7 and 5.8, $\mathsf{CF} \neq \mathsf{Ker} \neq \mathsf{PEq}$ relative to any one-sided transitive generic oracle.

($\mathsf{CF_p} = \mathsf{Ker_p}$ and $\mathsf{Ker} \neq \mathsf{PEq}$) Relativize to any PSPACE-complete set C, let O be any UP-transitive generic oracle relative to C, and *re*relativize to O. Note that Lemma 5.7 relativizes, so relative to C and O combined, $\mathsf{Ker} \neq \mathsf{PEq}$. Since $\mathsf{P} = \mathsf{PSPACE}$ relative to C, and O has at most one string of each length $tower(k)$ and no other strings, and Lemma 5.9 relativizes, we also have $\mathsf{CF}_p = \mathsf{Ker}_p$ relative to C and O combined. □

Open Question 5.10. Does $\mathsf{CF} = \mathsf{Ker}$ imply $\mathsf{P} = \mathsf{NP}$? Or is there an oracle relative to which $\mathsf{CF} = \mathsf{Ker}$ but nonetheless $\mathsf{P} \neq \mathsf{NP}$? Further, is there an oracle relative to which $\mathsf{P} \neq \mathsf{NP}$ but $\mathsf{CF} = \mathsf{Ker} = \mathsf{PEq}$?

Open Question 5.11. Is there an oracle relative to which $\mathsf{CF} \neq \mathsf{Ker} = \mathsf{PEq}$?

6. Future Work

Here we present several directions for future work, in addition to the open problems mentioned throughout the paper.

6.1. Logarithmic Space

It would also be interesting to study equivalence relations decidable in logarithmic space.

For example, it has been shown that the word equality problem (given two words in the generators of a group, do they represent the same group element?) for a finitely generated linear group is decidable in logarithmic space [53, 67]. (A group is linear if it is isomorphic to a group of matrices over some field.) In fact, implicit in the proofs is a log-space complete invariant: essentially the matrix corresponding to a word in the generators. But it seems unlikely that, in general, one can get from the matrix a corresponding canonical form, that is, a canonical word in the group generators representing each group element. Hence the word problem in finitely generated linear groups is a potential witness to $\mathsf{Ker}(\mathsf{FL}) \neq \mathsf{CF}(\mathsf{FL})$. One open problem is to explicitly construct a linear group with no log-space canonical form for its word equality problem.

Analogues of many of the results in this paper for logarithmic space are intriguing open questions:

- Is LEq contained in $\mathsf{CF}(\mathsf{FL}^{\mathsf{NL}})$? Is it contained in $\mathsf{CF}(\mathsf{FP})$? In $\mathsf{Ker}(\mathsf{FP})$? We note that the straightforward binary search technique used to show $\mathsf{PEq} \subseteq \mathsf{LexEqFP}^{\mathsf{NP}}$ does not work in logarithmic space. Jenner and Torán [43] showed that the lexicographically minimal (or maximal—in this case the same technique works) solution of any NL search problem can be computed in $\mathsf{FL}^{\mathsf{NL}}$. However, the notion of an NL search problem is based on the following characterization of NL due to Lange [51]: a language A is in NL if and only if there is a a polynomial p and a log-space machine $M(x, \vec{y})$ that reads its second input in one direction only, indicated by "$\vec{y}$", such that

$$x \in A \iff (\exists y : |y| \leq p(|x|))[M(x, \vec{y}) = 1].$$

  Without the one-way restriction, this definition would give a characterization of NP rather than NL. An NL search problem is then: given such a machine M and input x, find a y such that $M(x, \vec{y}) = 1$. Any equivalence relation that can be decided by such a machine—that is, where $x \sim y$ if and only if $M(x, \vec{y}) = 1$—is in $\mathsf{LexEqFL}^{\mathsf{NL}}$, but it is not clear that this captures all of LEq.

- Does $\mathsf{CF}(\mathsf{FL}) = \mathsf{Ker}(\mathsf{FL})$ imply $\mathsf{NL} = \mathsf{UL}$? Note that $\mathsf{NL} = \mathsf{UL}$ if and only if $\mathsf{FL}^{\mathsf{NL}} \subseteq \#\mathsf{L}$ [6].

- Does $\mathsf{CF}(\mathsf{FL}) = \mathsf{LEq}$ imply $\mathsf{UL} \subseteq \mathsf{RL}$? A positive answer to this question and the previous one would give very strong evidence that $\mathsf{CF}(\mathsf{FL}) \neq \mathsf{LEq}$, as significant progress has been made towards showing $\mathsf{L} = \mathsf{RL}$ [60].

6.2. Additional Questions

In no particular order:

- In Example 1.3 we observed that Boolean formula equivalence is a natural equivalence relation that is coNP-complete. The equivalence relation generated by $0x \sim_R 1x$ if and only if $x \in SAT$ is clearly NP-complete, but is not particularly natural as an equivalence relation. Are there *natural* NP-complete equivalence relations?

- Study expected polynomial-time canonical forms. If every $R \in \mathsf{Ker}(\mathsf{FP})$ has an expected polynomial-time canonical form, does PH collapse? An interesting example of an expected polynomial-time canonical form is that for graph isomorphism [12].

- Find a class of groups for which the group membership problem is in P but no efficient complete invariant is known for the subgroup equality problem (see Section 4.2.2).
- If $\mathsf{Ker} = \mathsf{PEq}$, does PH collapse?
- $\mathsf{LexEqFP}^{\Sigma_i\mathsf{P}} \stackrel{?}{=} \mathsf{CF}(\mathsf{FP}^{\Sigma_i\mathsf{P}}) \stackrel{?}{=} \mathsf{Ker}(\mathsf{FP}^{\Sigma_i\mathsf{P}}) \stackrel{?}{=} \mathsf{P}^{\Sigma_i\mathsf{P}}\mathsf{Eq}$. If $\mathsf{Ker}(\mathsf{FP}^{\Sigma_i\mathsf{P}}) = \mathsf{P}^{\Sigma_i\mathsf{P}}\mathsf{Eq}$ does PH collapse?
- Study counting classes of equivalence relations. For an equivalence relation R, the associated counting function is $f(x) = \#\{y : y \sim_R x\}$.
- Preorders have been studied in the context of p-selectivity and semifeasible sets [48], and partial orders have been studied in the context of $\#\mathsf{P}$ and acceptance mechanisms for nondeterministic machines [38]. It would be interesting to develop these further, as well as to study complexity classes of lattices and total orders.

Acknowledgments

The authors thank Stuart Kurtz and Laci Babai for several useful discussions. In particular, Stuart suggested the use of the equivalence relation R_L, which led us to Theorem 4.3, and Laci pointed out the canonical form for subgroup equality of permutation groups [10]. We thank Scott Aaronson for the observations leading to Section 4.1.1. We thank Andreas Blass for pointing us to the original two papers he co-authored with Gurevich [18, 19]. We thank Paolo Codenotti for useful comments on a draft. Finally, we thank the editor, Lane Hemaspaandra, and two anonymous reviewers for suggestions that significantly improved the clarity and the organization of the paper. In particular, one of the reviewers suggested that we define some sort of hybrid notion of Cohen and transitive genericity, as well as suggested the notion of UP-transitive genericity.

References

[1] S. Aaronson, Quantum lower bound for the collision problem, in: STOC '02: 34th Annual ACM Symposium on Theory of Computing, ACM, 2002, pp. 635–642.

[2] S. Aaronson, 2009. Personal communication.

[3] M. Agrawal, N. Kayal, N. Saxena, PRIMES is in P, Ann. of Math. (2) 160 (2004) 781–793.

[4] M. Agrawal, T. Thierauf, The formula isomorphism problem, SIAM J. Comput. 30 (2000) 990–1009.

[5] W. Aiello, J. Håstad, Statistical zero-knowledge languages can be recognized in two rounds, J. Comput. System Sci. 42 (1991) 327–345. FOCS '87: 28th Annual IEEE Symposium on Foundations of Computer Science.

[6] C. Álvarez, B. Jenner, A very hard log-space counting class, Theoret. Comput. Sci. 107 (1993) 3–30.

[7] S. Arora, B. Barak, Computational complexity: a modern approach, Cambridge University Press, Cambridge, 2009. Draft available online at `http://www.cs.princeton.edu/theory/complexity/`.

[8] V. Arvind, N.V. Vinodchandran, The counting complexity of group-definable languages, Theoret. Comput. Sci. 242 (2000) 199–218.

[9] L. Babai, Trading group theory for randomness, in: STOC '85: 17th Annual ACM Symposium on Theory of Computing, ACM, 1985, pp. 421–429.

[10] L. Babai, 2008. Personal communication.

[11] L. Babai, D.Y. Grigoryev, D.M. Mount, Isomorphism of graphs with bounded eigenvalue multiplicity, in: STOC '82: 14th Annual ACM Symposium on Theory of Computing, ACM, 1982, pp. 310–324.

[12] L. Babai, L. Kučera, Canonical labelling of graphs in linear average time, in: FOCS '79: 20th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1979, pp. 39–46.

[13] L. Babai, E.M. Luks, Canonical labeling of graphs, in: STOC '83: 15th Annual ACM Symposium on Theory of Computing, ACM, 1983, pp. 171–183.

[14] T. Baker, J. Gill, R. Solovay, Relativizations of the P =? NP question, SIAM J. Comput. 4 (1975) 431–442.

[15] S. Bakhtiari, R. Safavi-naini, J. Pieprzyk, Cryptographic hash functions: a survey, Technical Report, Department of Computer Science, University of Wollongong, 1995.

[16] R. Beals, Quantum computation of Fourier transforms over symmetric groups, in: STOC '97: 29th Annual ACM Symposium on Theory of Computing, ACM, 1997, pp. 48–53.

[17] R. Beigel, Perceptrons, PP, and the polynomial hierarchy, Comput. Complexity 4 (1994) 339–349. Special issue on circuit complexity (Barbados, 1992).

[18] A. Blass, Y. Gurevich, Equivalence relations, invariants, and normal forms, SIAM J. Comput. 13 (1984) 682–689.

[19] A. Blass, Y. Gurevich, Equivalence relations, invariants, and normal forms, II, in: Logic and Machines: Decision Problems and Complexity, volume 171 of *Lecture Notes in Computer Science*, Springer, 1984, pp. 24–42.

[20] W.W. Boone, Certain simple, unsolvable problems of group theory. V, VI, Nederl. Akad. Wetensch. Proc. Ser. A. 60 = Indag. Math. 19 (1957) 22–27, 227–232.

[21] R. Boppana, J. Håstad, S. Zachos, Does co-NP have short interactive proofs?, Inform. Process. Lett. 25 (1987) 27–32.

[22] G. Brassard, P. Høyer, An exact quantum polynomial-time algorithm for Simon's problem, in: Proc. 5th Israeli Symp. on Theory of Computing Systems, IEEE Computer Society, 1997, pp. 12–23.

[23] H. Buhrman, L. Fortnow, Two queries, J. Comput. System Sci. 59 (1999) 182–194. 13th Annual IEEE Conference on Computation Complexity (Buffalo, NY, 1998).

[24] J.Y. Cai, $S_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$, J. Comput. System Sci. 73 (2007) 25–35.

[25] J.Y. Cai, V.T. Chakaravarthy, L.A. Hemaspaandra, M. Ogihara, Competing provers yield improved Karp-Lipton collapse results, Inform. and Comput. 198 (2005) 1–23.

[26] R. Canetti, More on BPP and the polynomial-time hierarchy, Inform. Process. Lett. 57 (1996) 237–241.

[27] A.M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, D.A. Spielman, Exponential algorithmic speedup by a quantum walk, in: STOC '03: 35th Annual ACM Symposium on Theory of Computing, ACM, 2003, pp. 59–68 (electronic).

[28] I. Damgård, Collision free hash functions and public key signature schemes, in: EuroCrypt87, volume 304 of *Lecture Notes in Computer Science*, Springer, 1988, pp. 203–216.

[29] M. Ettinger, P. Høyer, A quantum observable for the graph isomorphism problem, arXiv:quant-ph/9901029, 1999.

[30] S.A. Fenner, L. Fortnow, S.A. Kurtz, L. Li, An oracle builder's toolkit, Inform. and Comput. 182 (2003) 95–136.

[31] L. Fortnow, The complexity of perfect zero-knowledge, in: STOC '87: 19th Annual ACM Symposium on Theory of Computing, ACM, 1987, pp. 204–209.

[32] K. Friedl, G. Ivanyos, F. Magniez, M. Santha, P. Sen, Hidden translation and orbit coset in quantum computing, in: STOC '03: 35th Annual ACM Symposium on Theory of Computing, ACM, 2003, pp. 1–9.

[33] M. Fürer, W. Schnyder, E. Specker, Normal forms for trivalent graphs and graphs of bounded valence, in: STOC '83: 15th Annual ACM Symposium on Theory of Computing, ACM, 1983, pp. 161–170.

[34] M. Furst, J. Hopcroft, E. Luks, Polynomial-time algorithms for permutation groups, in: FOCS '80: 21st Annual IEEE Symposium on Foundations of Computer Science, IEEE, 1980, pp. 36–41.

[35] C. Glaßer, C. Reitwießner, V. Selivanov, The Shrinking Property for NP and coNP, Technical Report TR08-029, Electronic Colloquium on Computational Complexity, 2008.

[36] M. Grigni, L.J. Schulman, M. Vazirani, U. Vazirani, Quantum mechanical algorithms for the nonabelian hidden subgroup problem, Combinatorica 24 (2004) 137–154.

[37] Y. Gurevich, From invariants to canonization, Bulletin of the EATCS 63 (1997) 115–119.

[38] L.A. Hemaspaandra, C.M. Homan, S. Kosub, K.W. Wagner, The complexity of computing the size of an interval, SIAM J. Comput. 36 (2006) 1264–1300.

[39] L.A. Hemaspaandra, A.V. Naik, M. Ogihara, A.L. Selman, Computing solutions uniquely collapses the polynomial hierarchy, SIAM J. Comput. 25 (1996) 697–708.

[40] J. Hopcroft, J.K. Wong, Linear time algorithm for isomorphism of planar graphs (preliminary report), in: STOC '74: 6th Annual ACM Symposium on Theory of Computing, ACM, 1974, pp. 172–184.

[41] J.E. Hopcroft, R.E. Tarjan, Isomorphism of planar graphs, in: Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N. Y., 1972), Plenum, New York, 1972, pp. 131–152, 187–212.

[42] G. Ivanyos, F. Magniez, M. Santha, Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem, Internat. J. Found. Comput. Sci. 14 (2003) 723–739.

[43] B. Jenner, J. Torán, The complexity of obtaining solutions for problems in NP and NL, in: Complexity theory retrospective, II, Springer, New York, 1997, pp. 155–178.

[44] J.H. Johnson, Rational equivalence relations, in: L. Kott (Ed.), ICALP '86: Proceedings of the 13nd International Colloquium on Automata, Languages and Programming, volume 226 of *Lecture Notes in Computer Science*, Springer, 1986, pp. 167–176.

[45] R.M. Karp, R.J. Lipton, Turing machines that take advice, Enseign. Math. (2) 28 (1982) 191–209.

[46] A. Kitaev, Quantum measurements and the abelian stabilizer problem, arXiv:quant-ph/9511026, 1995.

[47] D.E. Knuth, Efficient representation of perm groups, Combinatorica 11 (1991) 33–43.

[48] K.I. Ko, On self-reducibility and weak P-selectivity, J. Comput. System Sci. 26 (1983) 209–221.

[49] J. Köbler, O. Watanabe, New collapse consequences of NP having small circuits, SIAM J. Comput. 28 (1999) 311–324.

[50] G. Kuperberg, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, SIAM J. Comput. 35 (2005) 170–188.

[51] K.J. Lange, Two characterizations of the logarithmic alternation hierarchy, in: Proceedings of the 12th Symposium on Mathematical Foundations of Computer Science 1986, volume 233 of *Lecture Notes in Computer Science*, Springer-Verlag, 1986, pp. 518–526.

[52] C. Lautemann, BPP and the polynomial hierarchy, Inform. Process. Lett. 17 (1983) 215–217.

[53] R.J. Lipton, Y. Zalcstein, Word problems solvable in logspace, J. ACM 24 (1977) 522–526.

[54] E.M. Luks, Hypergraph isomorphism and structural equivalence of Boolean functions, in: STOC '99: 31st Annual ACM Symposium on Theory of Computing, ACM, 1999, pp. 652–658.

[55] G. Miller, Isomorphism testing for graphs of bounded genus, in: STOC '80: 12th Annual ACM Symposium on Theory of Computing, ACM, 1980, pp. 225–235.

[56] M.A. Nielson, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[57] P.S. Novikov, Ob algoritmičeskoĭ nerazrešimosti problemy toždestva slov v teorii grupp, Trudy Mat. Inst. im. Steklov. no. 44, Izdat. Akad. Nauk SSSR, Moscow, 1955. English translation: On the algorithmic insolvability of the word problem in group theory, in: American Mathematical Society Translations, Ser. 2, Vol. 9, AMS, 1958, pp. 1–122.

[58] M.O. Rabin, Probabilistic algorithm for testing primality, J. Number Theory 12 (1980) 128–138.

[59] O. Regev, Quantum computation and lattice problems, SIAM J. Comput. 33 (2004) 738–760 (electronic).

[60] O. Reingold, L. Trevisan, S. Vadhan, Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem, in: STOC '06: 38th Annual ACM Symposium on Theory of Computing, ACM, 2006, pp. 457–466.

[61] A. Russell, R. Sundaram, Symmetric alternation captures BPP, Comput. Complexity 2 (1995) 152–162.

[62] A.L. Selman, A survey of one-way functions in complexity theory, Math. Systems Theory 25 (1992) 203–221.

[63] A.L. Selman, A taxonomy of complexity classes of functions, J. Comput. System Sci. 48 (1994) 357–381.

[64] A.L. Selman, M.R. Xu, R.V. Book, Positive relativizations of complexity classes, SIAM J. Comput. 12 (1983) 565–579.

[65] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (1997) 1484–1509.

[66] D.R. Simon, On the power of quantum computation, SIAM J. Comput 26 (1997) 1474–1483.

[67] H.U. Simon, Word problems for groups and contextfree recognition, in: Fundamentals of computation theory (Proc. Conf. Algebraic, Arith. and Categorical Methods in Comput. Theory, Berlin/Wendisch-Rietz, 1979), volume 2 of *Math. Res.*, Akademie-Verlag, Berlin, 1979, pp. 417–422.

[68] C.C. Sims, Computational methods in the study of permutation groups, in: Computational Problems in Abstract Algebra (Oxford, 1967), Pergamon, Oxford, 1970, pp. 169–183.

[69] C.C. Sims, Computation with permutation groups, in: SYMSAC '71: Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation, ACM, 1971, pp. 23–28.

[70] M. Sipser, A complexity theoretic approach to randomness, in: STOC '83: 15th Annual ACM Symposium on Theory of Computing, ACM, 1983, pp. 330–335.

[71] R.M. Solovay, A model of set-theory in which every set of reals is Lebesgue measurable, Ann. of Math. (2) 92 (1970) 1–56.

[72] R.M. Solovay, V. Strassen, A fast Monte-Carlo test for primality, SIAM J. Comput. 6 (1977) 84–85.

[73] J. Tarui, Randomized polynomials, threshold circuits, and the polynomial hierarchy, in: STACS '91: Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, 1991, pp. 238–250.

[74] T. Thierauf, The Computational Complexity of Equivalence and Isomorphism Problems, volume 1852 of *Lecture Notes in Computer Science*, Springer, New York, 2000.

[75] S. Toda, M. Ogiwara, Counting classes are at least as hard as the polynomial-time hierarchy, SIAM J. Comput. 21 (1992) 316–328.

[76] L.G. Valiant, V.V. Vazirani, NP is as easy as detecting unique solutions, Theoret. Comput. Sci. 47 (1986) 85–93.

[77] S. Zachos, Probabilistic quantifiers and games, J. Comput. System Sci. 36 (1988) 433–451. Structure in Complexity Theory Conference 1986.