

COMPLEXITY CLASSES OF EQUIVALENCE PROBLEMS REVISITED

LANCE FORTNOW AND JOSHUA A. GROCHOW

ABSTRACT. To determine if two given lists of numbers are the same set, we would 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 for equivalences arise in graph isomorphism and its variants, and the equality of permutation groups given by generators. To determine if two given graphs are cospectral (have the same eigenvalues), however, we compute their characteristic polynomials and see if they are the same; the characteristic polynomial is a *complete invariant* for the equivalence relation of cospectrality. This is weaker than a canonical form, and it is not known whether a polynomial-time canonical form for cospectrality exists. Note that it is a priori possible for an equivalence relation to be decidable in polynomial time without either a complete invariant or canonical form.

Blass and Gurevich (“Equivalence relations, invariants, and normal forms, I and II”, 1984) ask whether these conditions on equivalence relations – having an FP canonical form, having an FP complete invariant, and simply being in P – are in fact different. They showed that this question requires non-relativizing techniques to resolve. Here we extend their results, and give new connections to probabilistic and quantum computation.

We denote the class of equivalence problems in P by PEq , the class of problems with complete FP invariants Ker , and the class with FP canonical forms CF ; $\text{CF} \subseteq \text{Ker} \subseteq \text{PEq}$, and we ask whether these inclusions are proper. If $x \sim y$ implies $|y| \leq \text{poly}(|x|)$, we say that \sim is polynomially bounded; we denote the corresponding classes of equivalence relation CF_p , Ker_p , and PEq_p . Our main results are:

- If $\text{CF} = \text{PEq}$ then $\text{NP} = \text{UP} = \text{RP}$ and thus $\text{PH} = \text{BPP}$;
- If $\text{CF} = \text{Ker}$ then $\text{NP} = \text{UP}$, $\text{PH} = \text{ZPP}^{\text{NP}}$, integers can be factored in probabilistic polynomial time, and deterministic collision-free hash functions do not exist;
- If $\text{Ker} = \text{PEq}$ then $\text{UP} \subseteq \text{BQP}$;
- There is an oracle relative to which $\text{CF} \neq \text{Ker} \neq \text{PEq}$; and
- There is an oracle relative to which $\text{CF}_p = \text{Ker}_p$ and $\text{Ker} \neq \text{PEq}$.

Attempting to generalize the third result above from UP to NP leads to a new open question about the structure of witness sets for NP problems (roughly: can the witness sets for an NP-complete problem form an Abelian group?). We also introduce a natural notion of reduction between equivalence problems, and present several open questions about generalizations of these concepts to the polynomial hierarchy, to logarithmic space, and to counting problems.

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 [Nov55, Boo57]. 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 [HT72, BL83, FSS83, Mil80, BGM82]. More recently, Agrawal and Thierauf [AT00, Thi00] used a randomized canonical form to show that Boolean formula non-isomorphism (\overline{FI}) is

in AM^{NP} . More generally, the book by Thierauf [Thi00] 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 [Mil80] pointed out that GI has a polynomial-time complete invariant iff it has a polynomial-time canonical form (see also [Gur97]). The general form of this question is central both in [BG84a, BG84b] and here: are canonical forms or complete invariants necessary for the efficient solution of equivalence problems?

In 1984, Blass and Gurevich [BG84a, BG84b] 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 [BG84a, BG84b] 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 $f(x)$ for R , that is, a function such that $x \sim_R y$ iff $f(x) = f(y)$.

The *canonical form problem*: for $x \in \Sigma^*$ calculate a canonical form $f(x)$ 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

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

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

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 forcing, 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 4.10. *If $\text{CF} = \text{Ker}$ then integers can be factored in probabilistic polynomial time.*

Proposition 4.11. *If $\text{CF} = \text{Ker}$ then collision-free hash functions that can be evaluated in deterministic polynomial time do not exist.*

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

We also show the following two related results:

Corollary 4.2. *If $CF = Ker$ then $NP = UP$ and $PH = ZPP^{NP}$.*

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

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

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 [BG84a, BG84b]. 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 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 Sections 4.2.1 and 4.2.2 we discuss connections with integer factoring and collision-free hash functions, respectively. In Section 4.2.5 we 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 [BG84a, BG84b] using forcing techniques. 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 $PH = \bigcup \Sigma_k P = \bigcup \Pi_k P = \bigcup \Delta_k P$. We refer the reader to the Complexity Zoo at http://qwiki.stanford.edu/wiki/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 [NC00].

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

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

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

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

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

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

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

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 $NPSV_t$ and $NPMV_t$ are the subclasses of NPSV and NPMV, respectively, consisting of the total functions in those classes.

The classes $\text{NPSV}_{\mathbf{g}}$ and $\text{NPMV}_{\mathbf{g}}$ are the subclasses of NPSV and NPMV , respectively, whose graphs are in \mathbf{P} .

A *refinement* of a multi-valued partial function f is a multi-valued partial function g such that $\text{dom}(g) = \text{dom}(f)$ and $\text{set-}g(x) \subseteq \text{set-}f(x)$ for all x . In particular, if $\text{set-}f(x)$ is nonempty then so is $\text{set-}g(x)$. 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 $\text{NPMV} \subseteq_c \text{PF}$ iff $\mathbf{P} = \mathbf{NP}$ [Sel92] iff $\text{NPSV} \subseteq \text{PF}$ [SXB83]. Selman [Sel94] 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 $\text{Ker}(f) = \{(x, y) : f(x) = f(y)\}$. For an equivalence relation R , if $R = \text{Ker}(f)$, 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} .

2.3. Generic Oracles and Forcing. In Section 5 we will use generic oracles for various notions of genericity. Here we give a brief overview of the definitions and basic results on generic oracles; for a more in-depth discussion, see [FFKL03].

Throughout this section we will use a to represent a string in Σ^* , O or X to represent oracles, σ , τ , and γ to represent conditions (collections of oracles), and \mathcal{G} to represent a notion of genericity (a collection of conditions). All these terms are defined below.

Informally, a *condition* is a collection γ of oracles in which it is possible to carry out any oracle construction that proceeds by finite extensions. If an oracle $O \in \gamma$, we may think of O as “satisfying γ .” Formally:

Definition 2.1. A *condition* is a nonempty perfect subset γ of 2^{Σ^*} . In other words, it is a collection of oracles such that if $O \in \gamma$, and σ is any finite initial segment of O , then there are two (and hence infinitely many) distinct oracles in γ extending σ .

If γ and τ are two conditions, and $\gamma \subseteq \tau$, we say that γ *extends* τ . Although this terminology at first may seem backwards, it is used because a smaller set of conditions more fully specifies an oracle. For example, if τ is the set of all oracles with initial segment 010 and γ is the set of all oracles with initial segment 010001011, then γ extends τ , and γ more fully specifies an oracle. Moreover, as in the preceding example, this terminology is consistent with the usage for finite strings: we say that 010001011 extends 010 and that γ extends τ .

Informally, a *notion of genericity* is a collection \mathcal{G} of conditions in which it is possible to carry out a forcing construction (which we’ll explain shortly). Formally:

Definition 2.2. A *notion of genericity* is a nonempty set \mathcal{G} of conditions such that, for all $\gamma \in \mathcal{G}$, all $O \in \gamma$, and all $a \in \Sigma^*$, there is a condition $\gamma' \in \mathcal{G}$ such that $\gamma' \subseteq \gamma$ and $(\forall O' \in \gamma')[O'(a) = O(a)]$. The conditions of \mathcal{G} are called *\mathcal{G} -conditions*.

Here we define a very restricted notion of forcing, as it suffices for our purposes. A more general definition of forcing is given in [FFKL03].

Definition 2.3 (Forcing). A condition γ *forces* $X(a) = 0$ if $X(a) = 0$ for all $X \in \gamma$. In this case, we write $\gamma \Vdash X(a) = 0$. Similarly for $X(a) = 1$. If γ forces either $X(a) = 0$ or $X(a) = 1$, we say that γ forces $X(a)$, or forces the value of a .

A generic oracle is simply one built by a forcing construction. This notion is formalized in the definition of a generic filter:

Definition 2.4 (Generic Filter). If \mathcal{G} is a notion of genericity, a *generic filter over \mathcal{G}* is a subset $\mathbb{G} \subseteq \mathcal{G}$ such that

- (1) For all $\sigma, \tau \in \mathcal{G}$, if $\tau \in \mathbb{G}$ and $\tau \subseteq \sigma$ then $\sigma \in \mathbb{G}$ (i. e., upward closure),
- (2) For all $\sigma_1, \sigma_2 \in \mathbb{G}$, there is a $\tau \in \mathbb{G}$ such that $\tau \subseteq \sigma_1 \cap \sigma_2$ (i. e., every two condition in \mathbb{G} have a common extension)
- (3) For each $a \in \Sigma^*$, there is a $\gamma \in \mathbb{G}$ such that γ forces the value of a .

Conditions (1) and (2) are the definition of a filter (which is used more broadly in logic and topology).

In particular, condition (2) implies that any two conditions in a filter \mathbb{G} have nonempty intersection. By the compactness of Cantor space 2^{Σ^*} , this means that the intersection of all conditions in \mathbb{G} is nonempty: $\bigcap \mathbb{G} \neq \emptyset$. Furthermore, condition (3), the forcing condition, implies that for every $a \in \Sigma^*$, there is a condition $\gamma \in \mathbb{G}$ forcing the value of a , i. e., forcing $X(a) = b$ where $b \in \{0, 1\}$. If $G \in \bigcap \mathbb{G}$, then $G \in \gamma$, so $G(a) = b$. Since the value of $G(a)$ is forced for all a , G is uniquely determined by \mathbb{G} . Hence $\bigcap \mathbb{G}$ consists of a single oracle.

Definition 2.5 (Generic Oracle). Let \mathcal{G} be a notion of genericity, and \mathbb{G} is a generic filter over \mathcal{G} , and let $\bigcap \mathbb{G} = \{O\}$. Then we say that \mathbb{G} *builds* O , and that O is a \mathcal{G} -*generic oracle*.

Lemma 2.6 (Existence of \mathcal{G} -generic oracles). *For every notion of genericity \mathcal{G} , the set of \mathcal{G} -generic oracles is dense in \mathcal{G} . In other words, for every \mathcal{G} -condition γ , there is a \mathcal{G} -generic oracle $O \in \gamma$.*

A *Cohen condition* is a condition γ specified by a partial characteristic function with finite domain. In other words, if A and B are disjoint finite sets of strings, then the set of all oracles including A and excluding B , i. e., $\{X : (\forall a \in A)[X(a) = 1] \text{ and } (\forall b \in B)[X(b) = 0]\}$, is a Cohen condition. This gives rise to the notion of Cohen genericity.

A *UP condition* is a Cohen condition γ that includes at most one string of each length, and only includes strings of length $\text{tower}(k)$, for any k . The tower function is defined inductively by $\text{tower}(0) = 1$ and $\text{tower}(n) = 2^{\text{tower}(n-1)}$. This gives rise to the notion of UP-generics. We use both Cohen generics and UP-generics in Section 5, as well as defining a new notion of genericity.

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 [BG84a, BG84b]. 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.

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

Theorem 3.1 ([BG84a] Theorem 1). *There is an equivalence relation $R \in \text{CF}$ whose first canonical form problem is essentially $\Delta_2\text{P}$ -complete, that is, it is in $\text{FP}^{\text{NP}} = \text{F}\Delta_2\text{P}$ and is $\Delta_2\text{P}$ -hard.* \square

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

Theorem 3.2 ([BG84a] 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 $R \notin \text{Ker}(\text{FP}^R)$, i. e., $\text{Ker}(\text{FP}^R) \neq \text{P}^R\text{Eq}$.*
- (b) *There is a function f such that $\text{Ker}(f) \notin \text{CF}(\text{FP}^f)$, i. e., $\text{CF}(\text{FP}^f) \neq \text{Ker}(\text{FP}^f)$.*
- (c) *There is an idempotent function f such that $\text{Ker}(f) \notin \text{LexEqFP}^f$, i. e., $\text{LexEqFP}^f \neq \text{CF}(\text{FP}^f)$.*

□

In addition to several extensions of these results, Blass and Gurevich [BG84a, BG84b] also show that collapses between certain classes of equivalence problems are equivalent to more standard complexity-theoretic hypotheses. Perhaps the most surprising of these are the equivalence between $\text{CF}(\text{FP}) \subseteq \text{LexEqNPSV}_t$ and $\text{NP} = \text{coNP}$ ([BG84b] Thm. 1), and the following result.

Theorem 3.3 ([BG84b] Theorem 3). *The following statements are equivalent:*

- (1) $\text{Ker}(\text{FP}) = \subseteq \text{CF}(\text{NPSV}_t)$
- (2) *NP has the shrinking property*
- (3) $\text{NPMV} \subseteq_c \text{NPSV}$, i. e., *the uniformization principle holds for NP*

We refer to [BG84b] for the definition of the shrinking property.

Hemaspaandra, Naik, Ogihara, and Selman [HNOS94] showed that if $\text{NPMV} \subseteq_c \text{NPSV}$ then $\text{SAT} \in (\text{NP} \cap \text{coNP})/\text{poly}$. At the time, they showed that this implied $\text{PH} = \Sigma_2\text{P}$; shortly thereafter Köbler and Watanabe [KW95] improved the collapse to $\text{PH} = \text{ZPP}^{\text{NP}}$. Combined with Theorem 3.3, this immediately implies a result that has not been announced previously:

Corollary 3.4. *If $\text{CF} = \text{Ker}$ then $\text{PH} = \text{ZPP}^{\text{NP}}$.*

4. EVIDENCE FOR SEPARATION

4.1. New Collapses. Blass and Gurevich's [BG84b] proof that $\text{Ker}(\text{FP}) = \subseteq \text{CF}(\text{NPSV}_t) \implies \text{NPMV} \subseteq_c \text{NPSV}$ essentially shows the following slightly stronger result. However, as $\text{NPMV} \subseteq_c \text{NPSV}$ is not known to imply $\text{NPMV}_g \subseteq_c \text{NPSV}_g$, 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 $\text{CF} = \text{Ker}$ then $\text{NPMV}_g \subseteq_c \text{NPSV}_g$.*

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

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

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

Similar to the original result [BG84b], we can weaken the assumption of this theorem to $\text{Ker}(\text{FP})_p \subseteq \text{CF}(\text{NPSV}_t)$, without modifying the proof. By padding, we can further weaken the assumption to $\text{Ker}(\text{FP}) = \subseteq \text{CF}(\text{NPSV}_t)$.

Corollary 4.2. *If $\text{CF} = \text{Ker}$ then $\text{NP} = \text{UP}$ and $\text{PH} = \text{ZPP}^{\text{NP}}$.*

Note that Corollary 3.4 alone does not imply Corollary 4.2, as neither of the statements $\text{PH} = \text{ZPP}^{\text{NP}}$ and $\text{NP} = \text{UP}$ is known to imply the other. Indeed, it is still an open question as to whether $\text{NP} = \text{UP}$ 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 $\text{Ker} = \text{PEq}$ then $\text{UP} \subseteq \text{BQP}$. If $\text{CF} = \text{PEq}$ then $\text{UP} \subseteq \text{RP}$.*

Proof. Suppose $\text{Ker} = \text{PEq}$. Let L be a language in UP and let V be a nondeterministic polynomial-time machine with at most one accepting path for each input, such that $x \in L \iff (\exists y)[|y| \leq p(|x|) \text{ and } V(x, y) = 1]$ for some polynomial p . 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 bitwise XOR. Clearly $R_L \in \text{PEq}$, so by hypothesis R_L has a complete invariant $f \in \text{FP}$. Since $L \in \text{UP}$, 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' , $f_a(x) = f_a(x')$ iff $x \oplus x' = w_a$. 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 [Sim94].

Now suppose further that $\text{CF} = \text{PEq}$. 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 $1/2$. \square

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

Proof. If $\text{CF} = \text{PEq}$ then it follows directly from Theorems 4.1 and 4.3 that $\text{NP} = \text{UP} \subseteq \text{RP}$. Thus $\text{NP} = \text{RP}$, since $\text{RP} \subseteq \text{NP}$ without any assumptions. Furthermore, it follows that $\text{PH} \subseteq \text{BPP}$ [Zac88], and since $\text{BPP} \subseteq \text{PH}$ [Lau83, Sip83], the two are equal. \square

The collapse inferred here is stronger than that of Corollary 3.4, since $\text{BPP} \subseteq \text{ZPP}^{\text{NP}}$ [Sip83, ZH86]. However, this result is incomparable to Corollary 3.4 since it also makes the stronger assumption $\text{CF} = \text{PEq}$, rather than only assuming $\text{CF} = \text{Ker}$.

4.1.1. *Groupy witnesses for NP problems.* We would like to extend the first half of Theorem 4.3 from UP to NP to give stronger evidence that $\text{Ker} \neq \text{PEq}$, but the technique 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 $L \in \text{NP}$ 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 $f \in \text{FP}$ such that for each length n , Σ^n is given a group structure defined by $xy^{-1} \stackrel{\text{def}}{=} f(x, y)$. Then

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

is an equivalence relation iff 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 $f: G \rightarrow X$ such that $\text{Ker}(f)$ is the partition given by the cosets of some subgroup $H \leq G$, find a generating set for H [Kit95]. 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 [Sho94]; Kitaev [Kit95] then noticed that Shor's algorithm in fact solves all Abelian HSPs. The unique shortest vector problem for lattices reduces to the dihedral HSP [Reg02], which is solvable in subexponential quantum time [Kup05]. The graph isomorphism problem reduces to the HSP for the symmetric group [Bea97]

or the wreath product $S_n \wr S_2$ [EH99], but it is still unknown whether any nontrivial quantum algorithm exists for GI .

In addition to the HSP for Abelian groups, the HSPs for several families of non-Abelian groups are also in BQP, for example: the class of so-called “almost Abelian groups”, which consists of groups G where the index of the intersection of all normalizers $\bigcap_{H \leq G} N_G(H)$ is small [GSVV04], the class of “smoothly solvable” groups, which consists of solvable groups of constant commutator length whose Abelian factors G_i can each be expressed as the direct product of a group H_i of constant exponent and a group K_i of size $\log^{O(1)}(|G_i|)$ [FIM⁺03], and the class of groups whose commutator subgroup has polylogarithmic size [IMS03].

The proof of Theorem 4.3 showed that if $\text{Ker} = \text{PEq}$ 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 [Sim94] gave a zero-error expected polynomial time quantum algorithm for this problem, putting it in $\text{ZQP} \subseteq \text{BQP}$. This result was later improved by Brassard and Høyer [BH97] 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 4.5. Let $L \in \text{NP}$. 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 $\text{mul}, \text{gen}, \text{dec} \in \text{FP}$ such that for each $a \in L$:

- (1) let $G(a) = \{x \in \Sigma^n : \text{dec}(a, x) = 1\}$; then for all $x, y \in G(a)$, defining $xy^{-1} \stackrel{\text{def}}{=} \text{mul}(a, x, y)$ gives a group structure to $G(a)$;
- (2) $\text{gen}(a) = (g_1, g_2, \dots, g_k)$ 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 4.6. *If $\text{Ker} = \text{PEq}$ and a language $L \in \text{NP}$ 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: $L \leq_T^P \text{HSP}(\mathcal{G})$.*

Proof. Let $L \in \text{NP}$, let W and G 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

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

is in PEq, since membership in $G(a)$ can be decided in polynomial time by the algorithm dec guaranteed in the definition of groupy witnesses, and 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 $\text{gen}(a)$ 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 $W(a)$. Therefore L reduces to the hidden subgroup problem. \square

Corollary 4.7. *If $\text{Ker} = \text{PEq}$ and the language L has Abelian groupy witnesses, then $L \in \text{BQP}$.*

Corollary 4.8. *Every language in UP has Abelian groupy witnesses.*

Open Question 4.9. Are there NP-complete problems with Abelian groupy witnesses? Assuming $\text{P} \neq \text{NP}$, are there any problems in $\text{NP} \setminus \text{UP}$ with Abelian groupy witnesses?

Our definition of having groupy witnesses is similar but not identical to Arvind and Vinodchandran’s definition of group-definability [AV00]. In fact, if a set $A \in \text{NP}$ has Abelian groupy witnesses and $|G(a)|$ is computable in time $\text{poly}(|a|)$, then their techniques are sufficient to show that A is

low for PP, and hence unlikely to be NP-complete. Despite this, it may yet be possible to use Corollary 4.6 to show that $\text{Ker} = \text{PEq} \implies \text{NP} \subseteq \text{BQP}$, as there are several classes of non-Abelian groups for which the HSP is known to be in BQP (see, e.g., [GSVV04, FIM⁺03, IMS03]).

4.2. Hardness.

4.2.1. Factoring integers.

Proposition 4.10. *If $\text{CF} = \text{Ker}$ 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 [AKS04], but even much weaker machinery lets us do so in probabilistic polynomial time [SS77, Rab80], 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 $f \in \text{FP}$.

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 . \square

4.2.2. *Collision-free hash functions.* Collision-free hash functions are a useful cryptographic primitive (see, e.g., [BSnP95]). Proposition 4.10 suggests a more general connection between the collapse $\text{CF} = \text{Ker}$ 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 $h_i : \Sigma^{|i|+1} \rightarrow \Sigma^{|i|}$ are

1. Easily accessible: there is an efficient, i.e., probabilistic polynomial-time, algorithm G such that $G(1^n) \in \Sigma^n \cap I$;
2. Easy to evaluate: there is an efficient algorithm E such that $E(i, w) = h_i(w)$; and
3. Collision-free: for all efficient 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., [Dam88]). Many proposed collections of collision-free hash functions, such as MD5 or SHA, can be evaluated deterministically, that is, $E \in \text{FP}$.

Proposition 4.11. *If $\text{CF} = \text{Ker}$ 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 $f \in \text{FP}$ by hypothesis. As in the proof of Proposition 4.10, 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 $\Sigma^{|i|+1} \rightarrow \Sigma^{|i|}$, there are at most $2^{|i|} - 1$ singleton classes in $R = \text{Ker}(h_i)$. If x lies in an equivalence class of size at least 2, then $\Pr_x[f(x) \neq x \mid \#[x]_R \geq 2] \geq \frac{1}{2}$. Thus $\Pr_x[f(x) \neq x] = \Pr_x[f(x) \neq x \mid \#[x]_R \geq 2] \Pr_x[\#[x]_R \geq 2] \geq \frac{1}{2} \left(\frac{1}{2} + \frac{1}{2^{|i|+1}} \right) > \frac{1}{4}$. \square

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

Fortunately or unfortunately, the subgroup equivalence problem for permutation groups on $\{1, \dots, n\}$ has a polynomial-time canonical form, via a simple modification [Bab08] of the classic techniques of Sims [Sim70, Sim71], whose analysis was completed by Furst, Hopcroft, and Luks [FHL80] and Knuth [Knu91].

4.2.4. *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 [Luk99] 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 $\text{PEq} \setminus \text{Ker}$.

4.2.5. *Complete problems?* Equivalence problems that are P-complete under NC or L reductions may lie in $\text{PEq} \setminus \text{Ker}$ 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 4.12. An equivalence relation R *kernel-reduces* to an equivalence relation S , denoted $R \leq_{ker}^P S$, if there is a function $f \in \text{FP}$ such that

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

Note that $R \in \text{Ker}$ iff R kernel-reduces to the relation of equality. Also note that if $R \leq_{ker}^P S$ via f , then $R \leq_m^P S$ via $(x, y) \mapsto (f(x), f(y))$, leading to the question:

Open Question 4.13. 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 $R \leq_m^P S$ but $R \not\leq_{ker}^P S$?

An equivalence relation $R \in \text{PEq}$ is *PEq-complete* if every $S \in \text{PEq}$ kernel-reduces to R . For any PEq -complete R , $R \in \text{Ker}$ iff $\text{Ker} = \text{PEq}$ iff the relation of equality is PEq -complete. Unlike NP-completeness, however, the notion of PEq -completeness does not become trivial if $\text{Ker} = \text{PEq}$: the relation of equality does not kernel-reduce to the trivial relation. More generally, if $R \leq_{ker}^P S$, then S cannot have fewer equivalence classes than R , even without a complexity bound on the reduction; a complexity bound further implies a relationship between the densities of the two relations.

Open Question 4.14. Are there PEq -complete equivalence problems?

5. ORACLES

We make significant use of generic oracles for various notions of genericity, i. e., forcing. Similar to Fenner, Fortnow, Kurtz, and Li [FFKL03], when we say

Let O be an X -generic oracle ...

it should be read

Let $O = A \oplus B$ where A is PSPACE-complete and B is an X -generic oracle relative to A ...

For some of these results, we will need a new notion of genericity: *transitive genericity*. A *transitive condition* σ is a Cohen condition satisfying

- (1) Length restriction: $\langle x, y \rangle \in \sigma$ only if $|x| = |y|$, and
- (2) Transitivity: $\langle x, y \rangle \in \sigma$ and $\langle y, z \rangle \in \sigma$ implies $\langle x, z \rangle \in \sigma$.

By Lemma 2.6, transitive generic oracles exist.

Theorem 5.1. *There are oracles A and B relative to which $P \neq NP$ and*

$$\begin{aligned} \text{CF}(\text{FP}^A) &\neq \text{Ker}(\text{FP}^A) \neq \text{P}^A\text{Eq}, \\ \text{CF}(\text{FP}^B)_p &= \text{Ker}(\text{FP}^B)_p \text{ and } \text{Ker}(\text{FP}^B) \neq \text{P}^B\text{Eq} \end{aligned}$$

We break most of the proof into three lemmata. The proofs of Lemmata 5.3 and 5.4 are adaptations of the proofs in [BG84a] to generic oracles. The proof of Lemma 5.5 is new.

We start by restating a useful combinatorial lemma:

Lemma 5.2 ([BG84a] 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 5.2 can be proved by a simple counting argument.

Lemma 5.3. *There is a (transitive generic) oracle relative to which $\text{Ker} \neq \text{PEq}$.*

Proof. Let τ be a transitive condition, and let $\bar{\tau}$ denote the minimal transitive oracle extending τ , that is, $(a, a) \in \bar{\tau}$ for every $a \in \Sigma^*$, but the only pairs $(x, y) \in \bar{\tau}$ are those in τ . Let M be a polynomial-time oracle transducer running in time $p(|x|)$. Let n be a length such that $p(n) < 2^{n-1}$ and τ is not defined on (a, b) for any strings a and b of length $> n$. If there are distinct strings x and y of length n such that $M^{\bar{\tau}}(x) = M^{\bar{\tau}}(y)$, then extend τ to length $p(n)$ as $\bar{\tau}$. Then $x \not\sim_{\tau} y$ but $M^{\tau}(x) = M^{\tau}(y)$.

Otherwise, $M^{\bar{\tau}}(x) \neq M^{\bar{\tau}}(y)$ for every two distinct strings x and y . Say that x *affects* y if M queries $\bar{\tau}$ about (x, y) or (y, x) in the computation of $M^{\bar{\tau}}(y)$. Let G be a digraph on the strings of length n , where there is a directed edge from x to y if x affects y . By the condition on n , the out-degree of each vertex is at most 2^{n-1} . Since there are 2^n vertices, Lemma 5.2 implies that there are two strings x and y of length n such that neither affects the other. Put (x, y) and (y, x) into τ . Thus $M^{\tau}(x) \neq M^{\tau}(y)$ but $x \sim_{\tau} y$.

Thus there is a transitive generic oracle O such that $\text{Ker} \neq \text{PEq}$ relative to O . \square

Lemma 5.4. *There is a (Cohen generic) oracle relative to which $\text{CF} \neq \text{Ker}$.*

Proof. We describe the oracle O over the alphabet $\{0, 1, 2\}$ for simplicity. Let $\text{read}^O: \Sigma^* \rightarrow \Sigma^*$ denote the oracle function

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

where k is the least value such that $O(x01^k) = 2$. Note that the bits used by read^O on input x are disjoint from those used by read^O on any input $y \neq x$. Let $R^O = \text{Ker}(\text{read}^O)$.

Let τ be a Cohen condition, and let $\bar{\tau}$ denote the oracle extending τ which has value 2 outside $\text{dom}(\tau)$. Let M be a polynomial-time oracle transducer running in time $p(|x|)$. Let n be a length such that $p(n) < 2^{n-1}$ and $\text{read}^{\bar{\tau}}(x)$ is the empty string ε for all strings of length $\geq n$. For a string x of length n , let τ_x denote the minimal extension of τ such that read^{τ_x} is the identity on all strings of length n except $\text{read}^{\tau_x}(x) = 1^{n+1}$. Note that read^{τ_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^{τ_x} must be the identity on strings of length n .

If there is an x of length n such that $M^{\overline{\tau x}}(x) \neq x$, then $M^{\overline{\tau x}}(x)$ is not the identity on strings of length n , so $M^{\overline{\tau x}}$ is not a canonical form for $R^{\overline{\tau x}}$. Extend τ to τ_x .

Otherwise, $M^{\overline{\tau x}}(x) = x$ for all x of length n . Find x and y of length n such that $M^{\overline{\tau x}}(x)$ does not query the oracle about y and $M^{\overline{\tau y}}(y)$ does not query the oracle about x . This is possible by Lemma 5.2, as in the proof of Lemma 5.3. Then update τ so that $read^\tau(x) = read^\tau(y)$. Again, M^τ cannot be a canonical form for R^τ .

Thus there is a Cohen generic oracle relative to which $CF \neq Ker$. □

Lemma 5.5. *If A is PSPACE-complete and O has at most one string of each length tower(k) and no other strings, then relative to $A \oplus O$, $CF(FP)_p = Ker(FP)_p$.*

Proof. Relativize to a base PSPACE-complete oracle. 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 $R = Ker(f^O)$, and suppose that $(x, y) \in R$ 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 $> q(p(|x|))$ in which case they cannot be queried by f on any input $y \sim x$. Following a technique used in [BF99], 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

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

$r_y = \min R_y$, 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 $f^O(z) = y$, answering 0 to any queries made by f to strings of length ℓ between $\log p(|x|) \leq \ell \leq q(p(|x|))$.
- (5) Run $f^O(z)$. If $f^O(z) = y$, then $z = b_y$, 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 5.1. ($CF \neq Ker \neq PEq$) Let $A = A_0 \oplus A_1$ where A_0 is transitive generic and A_1 is Cohen generic. The constructions in the proofs of Lemmata 5.3 and 5.4 go through *mutatis mutandis*.

($CF_p = Ker_p$ and $Ker \neq PEq$) Let B be a transitive UP-generic oracle. Then the proof of Lemmata 5.3 and 5.5 go through. □

Open Question 5.6. Does $CF = Ker$ imply $P = NP$? Or is there an oracle relative to which $CF = Ker$ but nonetheless $P \neq NP$? Further, is there an oracle relative to which $P \neq NP$ but $CF = Ker = PEq$?

Open Question 5.7. Is there an oracle relative to which $CF \neq Ker = PEq$?

6. FUTURE WORK

Here we present several directions for future work, in addition to the open problems mentioned throughout the paper. In no particular order:

- Is LEq contained in $CF(FL^{NL})$? Is it contained in $CF(FP)$? In $Ker(FP)$? We note that the straightforward binary search technique used to show $PEq \subseteq LexEqFP^{NP}$ does not work in logarithmic space. Jenner and Torán [JT97] showed that the lexicographically minimal (or maximal – in this case the same technique works) solution of any NL search problem can be computed in FL^{NL} . However, the notion of an NL search problem is based on the following characterization of NL due to Lange [Lan86]: a language A is in NL iff there is a logspace machine $M(x, \vec{y})$ that reads its second input in one direction only, indicated by “ \vec{y} ”, such that

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

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 is in $LexEqFL^{NL}$, but it is not clear that this captures all of LEq .

- Does $CF(FL) = Ker(FL)$ imply $NL = UL$? Note that $NL = UL$ iff $FL^{NL} \subseteq \#L$ [AJ93].
- Does $CF(FL) = LEq$ imply $UL \subseteq RL$? A positive answer to this question and the previous one would give very strong evidence that $CF(FL) \neq LEq$, as significant progress has been made towards showing $L = RL$ [RTV05].
- Study expected polynomial-time canonical forms. If every $R \in Ker(FP)$ has an expected polynomial-time canonical form, does PH collapse?
- Find a class of groups for which the group membership problem is in P but no efficient complete invariant is known for the subgroup equivalence problem (see Section 4.2.3).
- If $Ker = PEq$, does PH collapse?
- $LexEqFP^{\Sigma_i P} \stackrel{?}{=} CF(FP^{\Sigma_i P}) \stackrel{?}{=} Ker(FP^{\Sigma_i P}) \stackrel{?}{=} P^{\Sigma_i P}Eq$. If $Ker(FP^{\Sigma_i P}) = P^{\Sigma_i P}Eq$ does PH collapse?
- Study counting classes of equivalence relations. For an equivalence relation R , the associated counting function is $f(x) = \#[x]_R$.
- Study complexity classes of lattices, partial orders, and total orders.

ACKNOWLEDGMENTS

The authors would like to 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 equivalence of permutation groups [Bab08]. We would also like to thank Andreas Blass for pointing us to the original two papers [BG84a, BG84b].

REFERENCES

- [AJ93] Carme Álvarez and Birgit Jenner, *A very hard log-space counting class*, Theoret. Comput. Sci. **107** (1993), no. 1, 3–30.
- [AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793.

- [AT96] Manindra Agrawal and Thomas Thierauf, *The boolean isomorphism problem*, FOCS '96: 36th Annual IEEE Symposium on Foundations of Computer Science, 1996, preliminary version of [AT00], pp. 442–430.
- [AT00] Manindra Agrawal and Thomas Thierauf, *The formula isomorphism problem*, SIAM J. Comput. **30** (2000), no. 3, 990–1009, originally appeared as [AT96].
- [AV00] V. Arvind and N. V. Vinodchandran, *The counting complexity of group-definable languages*, Theoret. Comput. Sci. **242** (2000), no. 1-2, 199–218.
- [Bab08] László Babai, *Canonical generators for permutation groups*, May 2008, personal communication.
- [Bea97] Robert Beals, *Quantum computation of Fourier transforms over symmetric groups*, STOC '97: 29th Annual ACM Symposium on Theory of Computing, ACM, 1997, pp. 48–53.
- [BF99] Harry Buhrman and Lance Fortnow, *Two queries*, J. Comput. System Sci. **59** (1999), no. 2, 182–194, 13th Annual IEEE Conference on Computation Complexity (Buffalo, NY, 1998).
- [BG84a] Andreas Blass and Yuri Gurevich, *Equivalence relations, invariants, and normal forms*, SIAM J. Comput. **13** (1984), no. 4, 682–689.
- [BG84b] Andreas Blass and Yuri Gurevich, *Equivalence relations, invariants, and normal forms, II*, Logic and Machines: Decision Problems and Complexity, Lecture Notes in Computer Science, vol. 171, Springer, 1984, pp. 24–42.
- [BGM82] László Babai, D. Yu. Grigoryev, and David M. Mount, *Isomorphism of graphs with bounded eigenvalue multiplicity*, STOC '82: 14th Annual ACM Symposium on Theory of Computing, ACM, 1982, pp. 310–324.
- [BH97] Gilles Brassard and Peter Høyer, *An exact quantum polynomial-time algorithm for Simon's problem*, Proc. 5th Israeli Symp. on Theory of Computing Systems, 1997, pp. 12–23.
- [BL83] László Babai and Eugene M. Luks, *Canonical labeling of graphs*, STOC '83: 15th Annual ACM Symposium on Theory of Computing, ACM, 1983, pp. 171–183.
- [Boo57] William 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.
- [BSnP95] S. Bakhtiari, R. Safavi-naini, and J. Pieprzyk, *Cryptographic hash functions: a survey*, Tech. report, Department of Comp. Sci., University of Wollongong, 1995.
- [Dam88] Ivan Damgård, *Collision free hash functions and public key signature schemes*, EuroCrypt87, Lecture Notes in Comp. Sci., vol. 304, Springer, 1988, pp. 203–216.
- [EH99] Mark Ettinger and Peter Høyer, *A quantum observable for the graph isomorphism problem*, arXiv:quant-ph/9901029.
- [FFKL03] Stephen A. Fenner, Lance Fortnow, Stuart A. Kurtz, and Lide Li, *An oracle builder's toolkit*, Inform. and Comput. **182** (2003), no. 2, 95–136.
- [FHL80] Merrick Furst, John Hopcroft, and Eugene Luks, *Polynomial-time algorithms for permutation groups*, FOCS '80: 21st Annual IEEE Symposium on Foundations of Computer Science, IEEE, 1980, pp. 36–41.
- [FIM⁺03] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen, *Hidden translation and orbit coset in quantum computing*, STOC '03: 35th Annual ACM Symposium on Theory of Computing, ACM, 2003, pp. 1–9.
- [FSS83] Martin Fürer, Walter Schnyder, and Ernst Specker, *Normal forms for trivalent graphs and graphs of bounded valence*, STOC '83: 15th Annual ACM Symposium on Theory of Computing, ACM, 1983, pp. 161–170.
- [GSVV01] Michelangelo Grigni, Leonard J. Schulman, Monica Vazirani, and Umesh Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, STOC '01: 33rd Annual ACM Symposium on Theory of Computing, ACM, 2001, preliminary version of [GSVV04], pp. 68–74.
- [GSVV04] Michelangelo Grigni, Leonard J. Schulman, Monica Vazirani, and Umesh Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, Combinatorica **24** (2004), no. 1, 137–154, originally appeared as [GSVV01].
- [Gur97] Yuri Gurevich, *From invariants to canonization*, Bull. Euro. Assoc. Theor. Comp. Sci. (BEATCS) **63** (1997), 115–119.
- [HNOS94] Lane A. Hemaspaandra, Ashish V. Naik, Mitsunori Ogihara, and Alan L. Selman, *Computing solutions uniquely collapses the polynomial hierarchy*, Algorithms and Computation (Beijing, 1994), Lecture Notes in Comput. Sci., vol. 834, Springer, Berlin, 1994, Available as ECCC Tech. Report TR96-027. preliminary version of [HNOS96], pp. 56–64.
- [HNOS96] Lane A. Hemaspaandra, Ashish V. Naik, Mitsunori Ogihara, and Alan L. Selman, *Computing solutions uniquely collapses the polynomial hierarchy*, SIAM J. Comput. **25** (1996), no. 4, 697–708, originally appeared as [HNOS94].
- [HT72] John Hopcroft and Robert Tarjan, *Isomorphism of planar graphs*, STOC '74: 6th Annual ACM Symposium on Theory of Computing, Plenum, 1972, pp. 172–184.

- [IMS03] Gábor Ivanyos, Frédéric Magniez, and Miklos Santha, *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem*, Internat. J. Found. Comput. Sci. **14** (2003), no. 5, 723–739, Quantum computing.
- [JT97] Birgit Jenner and Jacobo Torán, *The complexity of obtaining solutions for problems in NP and NL*, Complexity theory retrospective, II, Springer, New York, 1997, pp. 155–178.
- [Kit95] Alexei Kitaev, *Quantum measurements and the abelian stabilizer problem*, arXiv:quant-ph/9511026.
- [Knu91] Donald E. Knuth, *Efficient representation of perm groups*, Combinatorica **11** (1991), no. 1, 33–43.
- [Kup05] Greg Kuperberg, *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*, SIAM J. Comput. **35** (2005), no. 1, 170–188.
- [KW95] Johannes Köbler and Osamu Watanabe, *New collapse consequences of NP having small circuits*, ICALP '95: Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (London, UK), Springer-Verlag, 1995, preliminary version of [KW99], pp. 196–207.
- [KW99] Johannes Köbler and Osamu Watanabe, *New collapse consequences of NP having small circuits*, SIAM J. Comput. **28** (1999), no. 1, 311–324, originally appeared as [KW95].
- [Lan86] Klaus-Jörn Lange, *Two characterizations of the logarithmic alternation hierarchy*, Proceedings of the 12th symposium on Mathematical foundations of computer science 1986 (New York, NY, USA), Springer-Verlag New York, Inc., 1986, pp. 518–526.
- [Lau83] Clemens Lautemann, *BPP and the polynomial hierarchy*, Inform. Process. Lett. **17** (1983), no. 4, 215–217.
- [Luk99] Eugene M. Luks, *Hypergraph isomorphism and structural equivalence of boolean functions*, STOC '99: 31st Annual ACM Symposium on Theory of Computing (New York, NY, USA), ACM, 1999, pp. 652–658.
- [Mil80] Gary Miller, *Isomorphism testing for graphs of bounded genus*, STOC '80: 12th Annual ACM Symposium on Theory of Computing (New York, NY, USA), ACM, 1980, pp. 225–235.
- [NC00] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.
- [Nov55] P. S. Novikov, *Ob algoritmičeskoj nerazrešimosti problemy toždestva slov v teorii grupp*, Trudy Mat. Inst. im. Steklov. no. 44, Izdat. Akad. Nauk SSSR, Moscow, 1955, English translation: [Nov58].
- [Nov58] P. S. Novikov, *On the algorithmic insolubility of the word problem in group theory*, American Mathematical Society Translations, Ser 2, Vol. 9, American Mathematical Society, Providence, R. I., 1958, Translation of [Nov55], pp. 1–122.
- [Rab80] Michael O. Rabin, *Probabilistic algorithm for testing primality*, J. Number Theory **12** (1980), no. 1, 128–138.
- [Reg02] Oded Regev, *Quantum computation and lattice problems*, FOCS '02: 43rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 2002, pp. 520–529.
- [RTV05] Omer Reingold, Luca Trevisan, and Salil Vadhan, *Pseudorandom walks in biregular graphs and the RL vs. L problem*, Tech. Report TR05-022, Electronic Colloquium on Computational Complexity, 2005.
- [Sel92] Alan L. Selman, *A survey of one-way functions in complexity theory*, Math. Systems Theory **25** (1992), no. 3, 203–221.
- [Sel94] Alan L. Selman, *A taxonomy of complexity classes of functions*, J. Comput. System Sci. **48** (1994), no. 2, 357–381.
- [Sho94] P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, FOCS '94: 35th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1994, preliminary version of [Sho97], pp. 124–134.
- [Sho97] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509, originally appeared as [Sho94].
- [Sim70] Charles C. Sims, *Computational methods in the study of permutation groups*, Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967), Pergamon, Oxford, 1970, pp. 169–183.
- [Sim71] Charles C. Sims, *Computation with permutation groups*, SYMSAC '71: Proceedings of the second ACM symposium on Symbolic and algebraic manipulation (New York, NY, USA), ACM, 1971, pp. 23–28.
- [Sim94] Daniel R. Simon, *On the power of quantum computation*, FOCS '94: 35th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1994, preliminary version of [Sim97], pp. 116–123.
- [Sim97] Daniel R. Simon, *On the power of quantum computation*, SIAM J. Comput. **26** (1997), no. 5, 1474–1483, originally appeared as [Sim94].
- [Sip83] Michael Sipser, *A complexity theoretic approach to randomness*, STOC '83: 15th Annual ACM Symposium on Theory of Computing (New York, NY, USA), ACM, 1983, pp. 330–335.
- [SS77] Robert Solovay and Volker Strassen, *A fast Monte-Carlo test for primality*, SIAM J. Comput. **6** (1977), no. 1, 84–85.

- [SXB83] Alan L. Selman, Mei Rui Xu, and Ronald V. Book, *Positive relativizations of complexity classes*, SIAM J. Comput. **12** (1983), no. 3, 565–579.
- [Thi00] Thomas Thierauf, *The computational complexity of equivalence and isomorphism problems*, Lecture Notes in Computer Science, no. 1852, Springer, New York, 2000.
- [Zac88] Stathis Zachos, *Probabilistic quantifiers and games*, J. Comput. System Sci. **36** (1988), no. 3, 433–451, Structure in Complexity Theory Conference (Berkeley, CA, 1986).
- [ZH86] Stathis Zachos and Hans Heller, *A decisive characterization of BPP*, Inform. and Control **69** (1986), no. 1-3, 125–135.