The Complexity of Equivalence Relations

Joshua A. Grochow

November 20, 2008

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, 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 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 using generic oracles, 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; $\mathsf{CF} \subseteq \mathsf{Ker} \subseteq \mathsf{PEq}$, and we ask whether these inclusions are proper. If $x \sim y$ implies $|y| \leq poly(|x|)$, we say that $\sim$ is polynomially bounded; we denote the corresponding classes of equivalence relation $\mathsf{CF_p}$, $\mathsf{Ker_p}$, and $\mathsf{PEq_p}$. Our main results are:

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

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.

Many of the new results in this thesis were obtained in collaboration with Lance Fortnow, and have been submitted for conference presentation.

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 — genus, orientability, and number of boundary components together form a complete homeomorphism invariant of connected surfaces that can be calculated easily from any triangulation [DH07] — and some are of great interest to complexity theorists, such as graph isomorphism (GI).

Complete invariants, as in the example of surface homeomorphism, 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 ($\overline{\mathsf{FI}}$) is in $\mathsf{AM}^{\mathsf{NP}}$. 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 if and only if it has a polynomial-time canonical form (see Section 3.1.1 for details; 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 thesis 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 (NP problems with at most one witness for each input), RP (problems solvable by a probabilistic algorithm in polynomial time with one-sided error), and BQP (bounded-error quantum polynomial time), 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 thesis.

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 $x \sim_R y$ if and only if $f(x) = f(y)$.

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. $\mathrm{Ker}(\mathsf{FP})$ consists of those equivalence relations that have a polynomial-time computable complete invariant. $\mathrm{CF}(\mathsf{FP})$ consists of those equivalence relations that have a polynomial-time canonical form. $\mathsf{LexEqFP}$ consists of those equivalence relations whose first canonical form is computable in polynomial time.

Note that PEq and $\mathsf{LexEqFP}$ are both defined in terms of the polynomial-time computability of particular functions, whereas $\mathrm{Ker}(\mathsf{FP})$ and $\mathrm{CF}(\mathsf{FP})$ are defined in terms of the existence of polynomial-time functions with a certain property. The notations are designed partially to suggest these similarities and differences.

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?

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

Proposition 4.24. † *If* $\mathsf{CF} = \mathsf{Ker}$ *then collision-free hash functions that can be evaluated in deterministic polynomial time do not exist.*

Theorem 4.16. † *If* $\mathsf{Ker} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{BQP}$*. If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{UP} \subseteq \mathsf{RP}$*.*

We also show the following two related results:

Corollary 4.14. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{NP} = \mathsf{UP}$ *and* $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$*.*

Corollary 4.17. † *If* $\mathsf{CF} = \mathsf{PEq}$ *then* $\mathsf{NP} = \mathsf{UP} = \mathsf{RP}$ *and in particular,* $\mathsf{PH} = \mathsf{BPP}$*.*

Corollary 4.14 follows from the slightly stronger Theorem 4.11, but we do not give the statement here as it requires further definitions.

It is rare for complexity classes to be defined by a *type* of algorithm, rather than an *amount* of resources, such as time, space, nondeterminism, randomness, or interaction. We believe this makes these classes and their connections to more standard complexity classes all the more interesting.

The remainder of this thesis is organized as follows. In Section 2 we give preliminary definitions and background. In Section 3 we discuss and give background on some of the motivations for studying complexity classes of equivalence relations. In particular we review the complexity-theoretic upper bounds on GI, the equivalence of complete invariants and canonical forms for GI, and Agrawal and Thierauf's result on formula isomorphism [AT00]. In Section 4.1 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 4.2 we prove new results connecting these classes with probabilistic and quantum computation. In Section 4.2.1, 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 4.16 from UP to NP, giving much stronger

†The dagger "† " indicates results or definitions we believe are significantly original.

evidence that $\mathsf{Ker} \neq \mathsf{PEq}$. 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 4.3.2 and 4.3.3 we discuss connections with integer factoring and collision-free hash functions, respectively. In Section 4.3.7 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 thesis.

The new results in this thesis were obtained in collaboration with Lance Fortnow, and have been submitted for conference presentation [FG08].

2 Preliminaries

This section serves to introduce standard concepts, and fix notation and conventions. We expect it is mostly review and make little or no attempt at proofs or excessive formality.

We assume the reader is familiar with standard models of computation. We use the multi-tape Turing machine with read-only input tape and write-only output tape as our standard model of computation, and make no further mention of the model except where it is relevant. Oracle Turing machines have a separate oracle tape and oracle query state. When the machine enters the query state, it transitions to one of two specified states depending on whether the string on the oracle tape is in the oracle. An oracle Turing machine with unspecified oracle is denoted $M^{\Box}$ for emphasis.

Alphabet and strings Throughout, Σ denotes a finite set, called the *alphabet*, and is usually taken to be $\{0, 1\}$. We often use the term "bit" rather than the more general "symbol" because of this convention. The set of strings of length exactly k over Σ is denoted Σ^k. The empty string is denoted ε. The notation $\Sigma^{\leq k}$ is used to denote $\bigcup_{n=0}^{k} \Sigma^n$, and Σ^* is used to denote the set of all finite strings. The length of a string is denoted by absolute value: thus $|x| = k$ if and only if $x \in \Sigma^k$.

Lexicographic order. When Σ is an initial segment of the natural numbers, it is equipped with the usual ordering, but even otherwise we may think of Σ as having an ordering $<_\Sigma$. The lexicographic ordering on Σ^* is given by $x <_{lex} y$ if $|x| < |y|$ or $|x| = |y|$, and if j is the leftmost position at which x and y differ, then $x[j] <_\Sigma y[j]$.

There is a bijective correspondence between Σ^* and $\mathbb{N}$, given by the lexicographic ordering on Σ^*, and we use this correspondence freely, referring to elements of Σ^* as "numbers" and speaking of the "length of the number n." Note that the length of the number n is $\lceil \log_{|\Sigma|}(n) \rceil$. We use log to denote $\log_2$.

Tuples. Ordered tuples are written with parentheses, such as $(u_0, \ldots, u_k)$. When needed, an ordered tuple is encoded into a single string by the iterated application of an easily computable and easily invertible bijective pairing function $\langle \cdot, \cdot \rangle \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such as $\langle x, y \rangle = \frac{1}{2}(x+y)(x+y+1)+y$. The iteration is performed as follows: $\langle u_0, \ldots, u_k \rangle = \langle u_0, \langle u_1, \ldots, u_k \rangle \rangle$. We find that, in writing, we never need to explicitly invoke this tupling function.

2.1 Computational Problems

A subset $L \subseteq \Sigma^*$ is called a *language*. The complement of L is denoted $\overline{L} = \Sigma^* \backslash L$. The *decision problem* for a language L is: given $x \in \Sigma^*$, decide whether or not $x \in L$. Many computational problems can be stated as decision problems, or are computationally equivalent to decision problems.

However, some problems are more naturally stated as *search problems*. A search problem is: given $x \in \Sigma^*$, find some y such that (x, y) satisfies some condition. For example, given an (encoding of) a graph G, find a Hamiltonian path in G if one exists. A solution to a search problem is a function f such that $(x, f(x))$ satisfies the desired condition, or $f(x) = \perp$ if there is no string y such that (x, y) satisfies the desired condition. Hence the computational complexity of search problems is closely related to the computational complexity of functions.

The *indicator function* of a language L is the function

$$L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$$

It is standard to abuse notation and use the same letter for both the language and its indicator function. Algorithmically solving the decision problem L is the same as computing the function L.

2.2 Reductions

A *Turing reduction* from language A to language B is an oracle Turing machine $M^\square$ such that $A(x) = M^B(x)$ for all $x \in \Sigma^*$. We write $M\colon A \leq_T B$. (The function-like notation "$M\colon A \leq_T B$ is not standard, but is a fairly natural combination of standard function notation $f\colon X \to Y$ and the standard reduction notation $A \leq_T B$.)

A *truth-table reduction* from A to B is a *nonadaptive* Turing reduction: the queries made by $M^\square$ on input x are determined solely by x, and not by the oracle answers to previous queries. We write $M\colon A \leq_{tt} B$.

A *many-one reduction* or m-reduction from A to B is a (computable) function $f\colon \Sigma^* \to \Sigma^*$ such that $x \in A \iff f(x) \in B$. We write $f\colon A \leq_m B$.

A *one-one reduction* or 1-reduction is an injective many-one reduction, denoted $f\colon A \leq_1 B$.

A *majority reduction* from A to B is a function f such that, if $f(x) = (y_1, \ldots, y_k)$ then

$$x \in A \iff y_i \in B \text{ for more than } k/2 \text{ values of } i.$$

We write $A \leq_{maj} B$.

For any notion of reduction r, $A \equiv_r B$ denotes that $A \leq_r B$ and $B \leq_r A$. If $\mathcal{C}$ is a complexity class, then $\leq_r^{\mathcal{C}}$ denotes that the reducing machine lies in $\mathcal{C}$. In particular, the polynomial-time-bounded versions of the above reductions are denoted $\leq_T^P$, $\leq_{tt}^P$, $\leq_m^P$, $\leq_1^P$, and $\leq_{maj}^P$, respectively.

Polynomial-time Turing reductions are known as *Cook reductions* and polynomial-time many-one reductions are known as *Karp reductions*, since these were the types of reductions originally used by their respective namesakes to define NP-completeness [Coo71, Kar72].

A class (collection) of languages $\mathcal{C}$ is said to be *closed under r reductions* if $B \in \mathcal{C}$ and $A \leq_r B$ implies $A \in \mathcal{C}$.

2.3 Complexity Classes

Polynomial time. The class of languages decidable in deterministic polynomial time is denoted P.

The class of languages decidable in *nondeterministic* polynomial time is denoted NP. Equivalently, $A \in \mathsf{NP}$ if there is a set $B \in \mathsf{P}$ such that

$$x \in A \iff (\exists^p w)[(x, w) \in B]$$

where the right hand side is taken to mean "there exists a polynomial q such that $|w| \leq q(|x|)$ and $(x, w) \in B$." Such a string w is said to *witness* that $x \in A$, and is called a witness for x.

If $\mathcal{C}$ is a class of languages, then $\mathsf{co}\mathcal{C} = \{L : \overline{L} \in \mathcal{C}\}$. For example, $A \in \mathsf{coNP}$ if and only if there is a set $B \in \mathsf{P}$ such that

$$x \in A \iff (\forall^p w)[(x, w) \in B]$$

where $\forall^p$ has the obvious meaning. Note that we can use $(x, w) \in B$ or $(x, w) \notin B$ in the above characterization, since P is closed under complementation, i. e., $\mathsf{P} = \mathsf{coP}$.

The following basic questions (and many more) are open: $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, $\mathsf{NP} \stackrel{?}{=} \mathsf{coNP}$, $\mathsf{P} \stackrel{?}{=} \mathsf{NP} \cap \mathsf{coNP}$.

Hardness and completeness. If $\mathcal{C}$ is a class of languages and r is a notion of reduction, a language L is said to be *hard for* $\mathcal{C}$ *under* r *reductions* if $X \leq_r L$ for every $X \in \mathcal{C}$. If, furthermore $L \in \mathcal{C}$, then L is said to be r*-complete for* $\mathcal{C}$.

In many cases, a standard notion of reduction is used. For example, a language L is said to be NP-hard if it is hard for NP under Karp ($\leq_m^P$) reductions.

Logarithmic space. The class of languages decidable in deterministic logarithmic space is denoted L. The class of languages decidable in nondeterministic logarithmic space is denoted NL. Unlike the situation for NP, it is known that $\mathsf{NL} = \mathsf{coNL}$ [Imm88, Sze88].

Polynomial space. The class of languages decidable in polynomial space is denoted PSPACE. The nondeterministic analogue, $\mathsf{NPSPACE}$ is often mentioned only up to the point of Savitch's Theorem, which says that $\mathsf{PSPACE} = \mathsf{NPSPACE}$ [Sav69]. We make no further (explicit) mention of $\mathsf{NPSPACE}$.

Relativizing complexity classes. For a language A, and a class of oracle Turing machines $\mathcal{M}$, we can define the relativized class $\mathcal{M}^A$ as the class of languages that are Turing-reducible to A by some machine in $\mathcal{M}$. For a class of machines $\mathcal{M}$ and a class of languages $\mathcal{D}$, we define $\mathcal{M}^{\mathcal{D}} = \bigcup_{X \in \mathcal{D}} \mathcal{M}^X$.

It is standard to abuse this terminology and use classes of languages instead of classes of machines for the base of the oracle, but the meaning is as expected. For example, P^A is the set of all languages that are polynomial-time Turing-reducible to A.

The polynomial hierarchy, lowness, and highness. Relativizing to a language L is essentially the same as relativizing to its complement $\overline{L}$. Hence, for example $\mathsf{NP}^{\mathsf{NP}}$ contains both NP and coNP. Based on this observation, we may define the *polynomial hierarchy*, originally introduced by Meyer and Stockmeyer [MS72] in analogy with the arithmetic hierarchy from computability theory:

$$\begin{aligned} \Sigma_0\mathsf{P} &= \mathsf{P} \\ \Sigma_1\mathsf{P} &= \mathsf{NP} \\ \Sigma_{k+1}\mathsf{P} &= \mathsf{NP}^{\Sigma_k\mathsf{P}} \\ \Delta_{k+1}\mathsf{P} &= \mathsf{P}^{\Sigma_k\mathsf{P}}. \end{aligned}$$

From these, we define $\mathsf{\Pi_k P} = \mathsf{co\Sigma_k P}$; for example, $\mathsf{\Pi_1 P} = \mathsf{coNP}$. Thus $\mathsf{\Sigma_0 P} = \mathsf{\Pi_0 P} = \mathsf{\Delta_0 P} = \mathsf{\Delta_1 P} = \mathsf{P}$. Note that $\mathsf{\Sigma_{k+1} P} = \mathsf{\Sigma_k P^{NP}}$.

It is clear that $\mathsf{\Sigma_k P} \cup \mathsf{\Pi_k P} \subseteq \mathsf{\Delta_{k+1} P} \subseteq \mathsf{\Sigma_{k+1} P} \cap \mathsf{\Pi_{k+1} P}$. The polynomial hierarchy is the union $\mathsf{PH} = \bigcup_{k=0}^{\infty} \mathsf{\Sigma_k P} = \bigcup_{k=0}^{\infty} \mathsf{\Pi_k P} = \bigcup_{k=0}^{\infty} \mathsf{\Delta_k P}$.

The following are equivalent: (1) $\mathsf{\Sigma_k P} = \mathsf{\Pi_k P}$, (2) $\mathsf{\Sigma_j P} = \mathsf{\Sigma_k P}$ for some $j \geq k$, and (3) $\mathsf{PH} = \mathsf{\Sigma_k P}$. If any (and hence all) of these conditions holds, we say the hierarchy *collapses to the k-th level.* If this does not hold for any level k, we say that PH is infinite. It is widely believed that PH is infinite.

For a language $L \in \mathsf{NP}$, $\mathsf{\Sigma_k P} \subseteq \mathsf{\Sigma_k P}^L \subseteq \mathsf{\Sigma_{k+1} P}$. Hence, L is said to be low$_k$ if $\mathsf{\Sigma_k P}^L = \mathsf{\Sigma_k P}$ and high$_k$ if $\mathsf{\Sigma_k P}^L = \mathsf{\Sigma_{k+1} P}$. The classes $\mathsf{L_k P}$ and $\mathsf{H_k P}$ consist of the low$_k$, respectively high$_k$, languages in NP [Sch83]. The following basic results are easy to show:

- $\mathsf{L_k P} \subseteq \mathsf{L_{k+1} P}$ for all k, and similarly $\mathsf{H_k P} \subseteq \mathsf{H_{k+1} P}$,
- $\mathsf{L_0 P} = \mathsf{P}$,
- $\mathsf{L_1 P} = \mathsf{NP} \cap \mathsf{coNP}$,
- $\mathsf{H_0 P} = \{L : L \text{ is } \leq_T^P\text{-complete for } \mathsf{NP}\}$,
- if $\mathsf{H_k P} \cap \mathsf{L_k P}$ is nonempty, then $\mathsf{PH} = \mathsf{\Sigma_k P}$.

The low hierarchy and the high hierarchy are thus thought to stratify NP. However, it is also believed that there are problems in NP that are neither low nor high. Indeed, Ladner's Theorem [Lad75] can be used to show that if $\mathsf{L_k P} \neq \mathsf{H_k P}$, or equivalently if $\mathsf{PH} \neq \mathsf{\Sigma_k P}$, then there are problems in $\mathsf{NP} \backslash (\mathsf{L_k P} \cup \mathsf{H_k P})$.

Complexity class operators. We now define the operators $\forall\cdot$ and $\exists\cdot$ on complexity classes. If $\mathcal{C}$ is a complexity class, then $\forall \cdot \mathcal{C}$ consists of those languages L for which there is a language $L' \in \mathcal{C}$ such that

$$x \in L \iff (\forall^p y)[(x, y) \in L'].$$

The $\exists\cdot$ operator is defined similarly. It is clear from our definitions that $\mathsf{NP} = \exists \cdot \mathsf{P}$ and $\mathsf{coNP} = \forall \cdot \mathsf{P}$. Indeed, it holds generally that $\mathsf{co}\exists \cdot \mathcal{C} = \forall \cdot \mathsf{co}\mathcal{C}$.

It is a standard exercise to show that

$$\forall \cdot \mathsf{\Sigma_k P} = \mathsf{\Pi_{k+1} P} \text{ and } \exists \cdot \mathsf{\Pi_k P} = \mathsf{\Sigma_{k+1} P}.$$

Hence we may consider $\mathsf{\Sigma_k}$ as the operator $\exists \cdot \forall \cdot \cdots Q_k\cdot$ where there are k operators total and Q_k is $\forall$ or $\exists$ depending on whether k is even or odd, respectively. Similarly, we may consider $\mathsf{\Pi_k}$ to be an operator $\forall \cdot \exists \cdot \cdots \cdot Q_k'\cdot$.

Randomness. Several complexity classes have been defined to capture various notions of randomized computation. Bounded-error probabilistic polynomial time, denoted BPP, consists of those languages L for which there is a language $L' \in \mathsf{P}$ and a polynomial p such that, for all x of length n:

$$\Pr_{r \in \Sigma^{p(n)}}[L'(x, r) = L(x)] \geq 2/3$$

Here, $2/3$ can be replaced by any function of n that is bounded below by $1/2 + \varepsilon$ for some constant $\varepsilon > 0$. By running an algorithm for L' several times with independent random bits r and taking the majority vote, the probability of correctness can be increased to $1 - 2^{q(n)}$ for any polynomial

q. Note that BPP allows *two-sided error*: L' can err on strings $x \in L$ and on strings $x \notin L$. BPP algorithms are sometimes referred to as *polynomial-time Monte Carlo* algorithms.

The classes RP and coRP are the one-sided error version of BPP. The class RP consists of those languages L for which there is a language $L' \in \mathsf{P}$ and a polynomial p such that

$$\begin{aligned} x \in L &\implies \Pr_{r \in \Sigma^{p(|x|)}}[L'(x,r) = 1] > 1/2 \\ x \notin L &\implies \Pr_{r \in \Sigma^{p(|x|)}}[L'(x,r) = 1] = 0 \end{aligned}$$

Probabilistic classes can also be defined in terms of nondeterministic Turing machines. A probabilistic Turing machine is a nondeterministic Turing machine where each binary nondeterministic choice, referred to as a "coin flip," is assigned a probability of 1/2. The probability of any given branch of the computation is the product of the probabilities of the coin flips that occur on that branch. In this model, it is clear that $\mathsf{RP} \subseteq \mathsf{NP}$.

The class ZPP, or zero-error probabilistic polynomial time, consists of those languages for which there is a randomized algorithm that never errs, and runs in *expected* polynomial time, the expectation being taken over the random coin flips. It is an easy exercise to show that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$. ZPP algorithms are sometimes referred to as *polynomial-time Las Vegas* algorithms [Bab79].

The relationship between BPP and NP is unknown. Today it is an easy exercise to show that if $\mathsf{NP} \subseteq \mathsf{BPP}$ then $\mathsf{NP} = \mathsf{RP}$, though this was originally proved by Ko [Ko82]. Sipser [Sip83], with help from Gács, and Lautemann [Lau83] showed that $\mathsf{BPP} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$.

Similar to the $\forall\cdot$ and $\exists\cdot$ operators, we can define the $\mathsf{BP}\cdot$ operator. The class $\mathsf{BP}\cdot\mathcal{C}$ consists of those languages L for which there is a language $L' \in \mathcal{C}$ and a polynomial p such that

$$\Pr_{r \in \Sigma^{p(|x|)}}[L'(x,r) = L(x)] \geq 2/3.$$

It is clear that $\mathsf{BP}\cdot\mathsf{P} = \mathsf{BPP}$. Schöning [Sch89] noticed that Lautemann's proof in fact shows if a class $\mathcal{C}$ is closed under majority reducibility, then $\mathsf{BP}\cdot\mathcal{C} \subseteq \forall\cdot\exists\cdot\mathcal{C} \cap \exists\cdot\forall\cdot\mathcal{C}$.

Mixing randomness and nondeterminisim: interactive proofs and Arthur-Merlin games. Arthur-Merlin games, and in particular the complexity class AM, were introduced by Babai [Bab85]. The basic idea is that the mere mortal King Arthur (with access to random coins) wishes the all-powerful wizard Merlin to prove a fact to him. For our purposes, it is simplest to define the class of Arthur-Merlin games as:

$$\mathsf{AM} = \mathsf{BP}\cdot\mathsf{NP}.$$

Babai showed [Bab85] that for any fixed number of alternations of the operators BP and $\exists\cdot$, $\mathsf{BP}\cdot\exists\cdot\mathsf{BP}\cdot\exists\cdot\cdots\cdot\mathsf{P} = \mathsf{AM}$. Often such extensions are denoted, for example, $\mathsf{MAM} = \exists\cdot\mathsf{BP}\cdot\exists\cdot\mathsf{P}$.

Note that in this definition, Arthur's coins are public. At the same time, Goldwasser, Micali, and Rackoff [GMR89] defined a similar class in which the coin tosses are all private — they need not be revealed to the verifier. Both of these models are known as *interactive proofs*. Subsequently, Goldwasser and Sipser [GS86] showed that "public coins are as good as private coins," that is, the class of languages with constant-round interactive proofs is exactly AM.

Subsequently it was shown that $\mathsf{GI} \in \mathsf{coAM}$ [GMW86, GS86]; see Section 3.1 for more details.

Although it will not be relevant, we feel we should mention one of the crowning achievements of complexity theory in the 1990s. The class IP consists of those languages that have interactive proofs with a polynomially bounded number of rounds, that is, the number of rounds can grow as a polynomial of the size of the input. The one non-relativizing proof technique currently known — arithmetization — was developed in the course of proving that $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN90, Sha90] (see also [BFL90, BF90, AW08] for related work).

Quantum complexity. The class BQP consists of those languages that can be decided on a quantum computer in polynomial time with error strictly bounded away from $1/2$, as in the definition of BPP. For more details on quantum computing, we recommend the book by Nielson and Chuang [NC00].

2.4 Function Classes

Complexity-bounded function classes are defined in terms of *Turing transducers*: Turing machines with an additional write-only output tape. 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. Whenever we say "partial" or "multivalued," we mean "potentially partial" and "potentially multivalued." For such a function f, we write

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

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 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.

Logarithmic-space functions. The class FL is the class of all (single-valued, total) functions computable by a logspace transducer. Note that neither the input tape nor the output tape is counted in the space usage.

Nondeterministic functions. 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 $\textit{set-}g(x)$.

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 $\mathsf{P} = \mathsf{NP}$ [Sel92] if and only if $\mathsf{NPSV} \subseteq \mathsf{PF}$ [SXB83]. Selman [Sel94] is one of the classic works in this area, and gives many more results regarding these function classes.

The following theorem is our main formal evidence for believing that $\mathsf{NPMV} \not\subseteq_c \mathsf{NPSV}$:

Theorem 2.1 ([HNOS94])**.** *The following conditions are requivalent:*

1. *There is a function $f \in \mathsf{NPSV}$ such that, for any formula φ, $f(\varphi)$ is a satisfying assignment of φ, if one exists, or $\perp$ otherwise;*

2. $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$*;*

3. $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV}$ *[Sel94].*

If any, and hence all, of the above conditions hold, then $\mathsf{PH} = \Sigma_2\mathsf{P}$*.*

In fact, they showed that the conditions of the above theorem imply $\mathsf{SAT} \in (\mathsf{NP} \cap \mathsf{coNP})/poly$ [HNOS94]. At the time, this was only known to imply $\mathsf{PH} = \Sigma_2\mathsf{P}$, but shortly thereafter the collapse was improved to $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$ [KW95].

We note that $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$ obviously implies $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV}$, and hence that the conditions of the above theorem hold, but that the converse, namely the implication $\mathsf{NPMV} \subseteq_c \mathsf{NPSV} \implies \mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$, is not known to hold.

It is not difficult to show that $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$ implies $\mathsf{NP} = \mathsf{UP}$; we review a proof of this fact in the proof of Corollary 4.14. Again, the converse is not known to hold. We also note that it is still an open question as to whether $\mathsf{NP} = \mathsf{UP}$ implies any collapse of PH at all.

The following diagram summarizes these implications:

$$
\begin{tikzcd}
\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \mathsf{NPMV_g} \subseteq_c \mathsf{NPSV} \arrow[r, Rightarrow, "\text{[HNOS94]}"] \arrow[d, Leftrightarrow, "\text{[Sel94]}"] & \mathsf{SAT} \in (\mathsf{NP} \cap \mathsf{coNP})/poly \arrow[d, Rightarrow] \\
\mathsf{NP} = \mathsf{UP} & \mathsf{NPMV} \subseteq_c \mathsf{NPSV} & \begin{array}{c} \mathsf{PH} = \Sigma_2\mathsf{P} \text{ [HNOS94]} \\ \mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}} \text{ [KW95]} \end{array}
\end{tikzcd}
$$

Any implication not present in the above diagram is not known to hold, nor are there oracles known to settle these non-implications either way.

2.5 Equivalence Relations

A binary relation R is a subset of $\Sigma^* \times \Sigma^*$. If R is an equivalence relation and $(x, y) \in R$, we write $x \sim_R y$. An *equivalence relation* is

- reflexive: $x \sim_R x$ for all x;
- symmetric $x \sim_R y \iff y \sim_R x$ for all x, y;
- transitive: if $x \sim_R y$ and $y \sim_R z$ then $x \sim_R z$.

If f is any function, then the *kernel* of f is

$$\mathrm{Ker}(f) = \{(x, y) : f(x) = f(y)\}.$$

It is clear that $\mathrm{Ker}(f)$ is an equivalence relation. If $R = \mathrm{Ker}(f)$ then f is said to be a *complete invariant* for R. A *canonical form* for an equivalence relation R is a function g such that $x \sim_R g(x)$ for all x, and $x \sim_R y$ if and only if $g(x) \sim_R g(y)$. Note that if g is a canonical form for R, then $\mathrm{Ker}(g) = R$ and g is idempotent, that is, $g \circ g = g$. Indeed, g is a canonical form for R if and only if g is an idempotent complete invariant for R.

If R is an equivalence relation, then the *equivalence class* of the string x is:

$$[x]_R = \{y : x \sim_R y\}.$$

The equivalence classes of R partition Σ^*.

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. The *discrete relation* is the relation of equality, that is, each string is equivalent only to itself under the discrete relation. Equivalently, the discrete relation satisfies $[x] = \{x\}$ for every x.

3 Motivation

In this section, we help motivate the study of complexity classes of equivalence relations. In Section 3.1 we review some naturally occurring isomorphism and equivalence problems, and some of the algorithmic and complexity-theoretic results that make them interesting, aside from their intrinsic mathematical interest. In Section 3.2 we give an indication of how studying complexity classes of equivalence relations can shed new light on function classes. In Section 3.3 we discuss chain conditions on languages, inspired by chain conditions in algebra and topology, and how they initially led us to study complexity classes of equivalence relations.

3.1 Naturally Occurring Isomorphism and Equivalence Problems

Many naturally occurring isomorphism and equivalence problems are of great algorithmic and complexity-theoretic interest. Their study has led to novel algorithmic techniques, and they are some of the few remaining candidates for naturally occurring problems of intermediate complexity.

In the original paper in which Karp defines NP-completeness [Kar72], he noted that $\mathsf{GraphIso}$, $\mathsf{Composites}$, and $\mathsf{LinearProgramming}$ were not known to be NP-complete nor known to be in P. At the time, it was unknown whether this situation was due to a lack of proof or whether there were any problems at all in that were not NP-complete but not in P. Such problems are known as problems of *intermediate complexity*.

Shortly thereafter, Richard Ladner [Lad75] proved:

Theorem 3.1 (Ladner's Theorem [Lad75])**.** *If $\mathsf{P} \neq \mathsf{NP}$ then there is a family of sets $\{A_q : q \in \mathbb{Q}\} \subseteq \mathsf{NP}$ such that $A_0 = \emptyset$, A_1 is m-complete for NP, $p \leq q$ implies $A_p \leq_m^P A_q$, and $p > q$ implies $A_p \not\leq_T^P A_q$.*

This showed that intermediate problems exist in abundance, but did not resolve the questions about the naturally occurring potentially intermediate problems mentioned above. Indeed, the sets constructed in Ladner's Theorem have a somewhat artificial flavor. For example, $A_{1/2}$ looks like A_1 for certain stretches of input and like A_0 for other stretches; it is essentially by choosing these stretches appropriately that the proof works.

Since then, both $\mathsf{LinearProgramming}$ [Hač79] and $\mathsf{Composites}$ [AKS04] have been shown to be in P. Graph isomorphism and related problems — such as graph automorphism (decide if a graph has any nontrivial automorphism) and ring iso- and automorphism — are some of the few remainining naturally occurring problems potentially of intermediate complexity.

Our main evidence that $\mathsf{GI} \notin \mathsf{P}$ is anecdotal: many smart people have tried to prove that $\mathsf{GI} \in \mathsf{P}$ and failed. Usually these attempts are not total failures: the isomorphism problems for many restricted classes of graphs have been shown to be in P. For example, the following classes of graphs have polynomial-time isomorphism algorithms: trees [Kel57] (now an easy exercise), planar graphs [HT72, HW74], graphs of bounded genus [Mil80], graphs of bounded eigenvalue multiplicity [BGM82], and graphs of bounded degree [Luk82].

However, we have more technical evidence that GI is not NP-complete:

Theorem 3.2 ([Bab77, Mat79]). *Counting the number of isomorphisms between two graphs is Cook equivalent to deciding whether two graphs are isomorphic. Briefly:* $\sharp\mathsf{GI} \equiv_T^P \mathsf{GI}$.

The above result contrasts GI from many NP-complete problems. For example, $\sharp\mathsf{SAT}$ is $\sharp\mathsf{P}$-complete; if L is an NP-complete language for which there is a parsimonious reduction from SAT to L, that is, a reduction that preserves the number of witnesses, then $\sharp L$ is also $\sharp\mathsf{P}$-complete. Such parsimonious reductions are known for many NP-complete problems, and hence the counting versions of many NP-complete problems are $\sharp\mathsf{P}$-complete. Since $\mathsf{P}^{\sharp\mathsf{P}}$ contains the whole polynomial hierarchy [Tod89], $\sharp\mathsf{P}$-hard problems are thought to be much harder than NP problems.

The following gives further technical evidence that GI is not NP-complete:

Theorem 3.3. *If GI is NP-complete, then* $\mathsf{PH} = \Sigma_2\mathsf{P}$.

Outline of proof. First, $\mathsf{GI} \in \mathsf{coAM}$; there is a two-round interactive proof for $\overline{\mathsf{GI}}$ using private coins [GMW86], and any such interactive proof can be converted to one using public coins [GS86]. Babai [Bab85] observed that Lautemann's proof [Lau83] that $\mathsf{BPP} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$ directly extends to give $\mathsf{AM} \subseteq \Pi_2\mathsf{P}$. Finally, Boppana, Håstad, and Zachos [BHZ87] showed that if $\mathsf{NP} \subseteq \mathsf{coAM}$ then $\mathsf{PH} = \Sigma_2\mathsf{P} = \mathsf{AM}$. This last result also follows directly from the fact that $\mathsf{MAM} = \mathsf{AM}$ [Bab85] (see [BM88] Section 1.9). Babai and Moran [BM88] gave an alternative direct proof of this theorem. $\square$

Schöning extended this result by a slightly different argument as follows:

Theorem 3.4 ([Sch88]). *The graph isomorphism problem is low for $\Sigma_2\mathsf{P}$. Briefly: $\mathsf{GI} \in \mathsf{L}_2\mathsf{P}$.*

Corollary 3.5. *The graph isomorphism problem is not in $\mathsf{H}_\mathsf{k}\mathsf{P}$ unless $\mathsf{PH} = \Sigma_{\max(\mathsf{k},2)}\mathsf{P}$. In particular, GI is not $\leq_T^P$-complete for NP unless $\mathsf{PH} = \Sigma_2\mathsf{P}$.*

Outline of proof of Theorem 3.4. As with the previous result, this result relies on the fact that $\mathsf{GI} \in \mathsf{NP} \cap \mathsf{coAM}$ [GMW86, GS86]. It is relatively easy to show that $\mathsf{AM} \cap \mathsf{coAM}$ is low for AM, and thus GI is low for AM. Finally, $\forall \cdot \mathsf{AM} = \Pi_2\mathsf{P}$, and this result relativizes. Thus GI is low for $\Pi_2\mathsf{P}$, and the result follows by complementation. $\square$

The book by Köbler, Schöning, and Torán [KST93] gives a nice, relatively self-contained overview of GI and the various complexity-theoretic results surrounding it.

Boolean formula isomorphism, abbreviated FI, is an exemplar potentially of intermediate complexity one level higher in the polynomial hierarchy. Recall that two Boolean formulas F and G are *equivalent* if $F(x) = G(x)$ for all assignments x. Two Boolean formulas F and G are *isomorphic* if there is some permutation π of the inputs of F such that $F \circ \pi$ is equivalent to G. Note that Boolean formula equivalence is coNP-complete and Boolean formula isomorphism lies in $\Sigma_2\mathsf{P}$. There are also complexity-theoretic upper bounds on FI, analogous to the above bounds on GI:

Theorem 3.6 ([AT00])**.** *The formula isomorphism problem cannot be* $\Sigma_2\mathsf{P}$*-complete unless* $\mathsf{PH} = \Sigma_3\mathsf{P}$*.*

This result relies on a *randomized canonical form* for the formula equivalence problem.

Definition 3.7 ([Thi00])**.** A *randomized canonical form* for an equivalence relation R is a (potentially partial) function $f(x, r)$ such that

1. $\Pr_r[x \sim_R f(x, r)] \geq 3/4$, and $f(x, r)$ makes no output otherwise;
2. If $x \sim_R y$, then $f(x, r) = f(y, r)$ for all r.

The reason a canonical form for formula equivalence is needed is that

$$\begin{aligned} F_0 &= x_1 \wedge (\overline{x_1} \vee \overline{x_2}) \\ F_1 &= \overline{x_1} \wedge x_2 \end{aligned}$$

are isomorphic by interchanging x_1 and x_2, but this permutation does not make them synactically identical. The analogous result that was needed to show that $\mathsf{GI} \in \mathsf{coAM}$ is: if $G \cong H$ then there is a permutation π of the vertices such that $\pi(G) = H$, where here equality is understood as syntactic equality.

The randomized canonical form for formula equivalence is based on a circuit learning algorithm of Bshouty *et al.*[BCKT96]:

Theorem 3.8 ([BCKT96])**.** *There is a randomized canonical form for Boolean circuit equivalence in* $\mathsf{FP}^{\mathsf{NP}}$*.*

Agrawal and Thierauf noted that this randomized canonical form works just as well for Boolean formula equivalence. In fact, all of the following results on Boolean formula equivalence also hold for Boolean circuit equivalence, since they all rely on the above theorem. Using essentially the same two-round interactive proof for graph non-isomorphism, we get:

Corollary 3.9 ([AT00])**.** *The formula non-isomorphism problem is in* $\mathsf{AM}^{\mathsf{NP}} = \mathsf{BP} \cdot \Sigma_2\mathsf{P}$*.*

Using this result, the reasoning of Theorem 3.3 extends to give:

Corollary 3.10 ([AT00])**.** *The formula isomorphism problem cannot be* $\Sigma_2\mathsf{P}$*-complete unless* $\mathsf{PH} = \Sigma_3\mathsf{P}$*.*

Much as the initial result about the non-NP-completeness of GI was extended to show that $\mathsf{GI} \in \mathsf{L}_2\mathsf{P}$, we would like to extend Agrawal and Thierauf's results about FI to show that FI is in some sense "low." It is easy to see that FI is coNP-hard, so FI cannot be in $\mathsf{L}_\mathsf{k}\mathsf{P}$ unless PH collapses.

Furthermore, Agrawal and Thierauf [AT00] showed that $\mathsf{UOClique} \leq_m^P \mathsf{FI}$. Since $\mathsf{UOClique}$ is not known to be in the Boolean closure of NP, this is taken as evidence that FI is strictly above coNP. If this is the case, then the usual notion of lowness is not even applicable to FI. However, the reasoning used in Theorem 3.4 [Sch88] extends directly to show that an FI oracle cannot move classes up PH by two levels, unless PH collapses:

Proposition 3.11. † $\Sigma_2\mathsf{P}^{\mathsf{FI}} = \Sigma_3\mathsf{P}$.

Corollary 3.12. † *If* $\Sigma_\mathsf{k}\mathsf{P}^{\mathsf{FI}} = \Sigma_{\mathsf{k}+2}\mathsf{P}$*, then* $\mathsf{PH} = \Sigma_{\max(\mathsf{k},3)}\mathsf{P}$*.*

3.1.1 From invariants to canonical forms

Gary Miller [Mil80], at the end of his Section II, pointed out that for graph isomorphism, "Using standard reducibility techniques it is easy to see that succinct codes [polynomial complete invariants] and canonical forms are polynomial time equivalent" (see also [Gur97]). Since the proof is fairly general, and hence may be applicable to other equivalence relations, we review it here.

Proposition 3.13. *The canonical form problem for GI uniformly Cook-reduces to the complete invariant problem. That is, there is a polynomial-time Turing reduction $R^\square$ such that if f is a complete invariant for GI then R^f is a canonical form for GI. In particular, $\mathsf{GI} \in \mathrm{Ker}(\mathsf{FP}) \iff \mathsf{GI} \in \mathrm{CF}(\mathsf{FP})$.*

Proof. Suppose f is a complete invariant for GI, and let G be a graph. If $v \in V(G)$, let G_v denote G with v individualized (colored, say, red, and then converted from a colored graph to an undirected graph via the standard many-one equivalence between colored graph isomorphism and GI). For each vertex of G, run $f(G_v)$. Let v_0 be the vertex of G minimizing $f(G_v)$. Give v_0 the label zero. Repeat the process inductively, fixing v_0, and using a different "color" for individualization at each stage. $\square$

In fact, we see that an analogous result also holds for any NP-complete equivalence problem. If R is such a problem, then any complete invariant for R provides a solution for R, and thus any complete invariant for R is NP-hard. Since the first canonical form for R can be computed with an NP oracle, the first canonical form problem for R Cook-reduces to the complete invariant problem. Note that unlike GI, the reduction here is not uniform in the complete invariant.

Since GI also shares this property, we ask the following question:

Open Question 3.14. Is it the case that for every equivalence problem $R \in \mathsf{NP}\backslash\mathsf{P}$, the canonical form problem Cook-reduces to the complete invariant problem? That is, is it the case that for every $R \in \mathsf{NP}\backslash\mathsf{P}$ and every complete invariant f for R, there is a polynomial-time reduction $M^\square$ such that M^f is a canonical form for R? Note that there is no restriction on the complexity of f or the resulting canonical form M^f.

We believe an answer to this question either way would provide interesting information regarding the structure of NP.

We should mention that an *expected* linear-time canonical form for GI was discovered by Babai and Kučera [BK79], the expectation being taken over the uniform distribution on inputs of a given size (note that the input size of graphs on n vertices is $O(n^2)$). Questions about expected polynomial-time canonical forms may prove interesting, but we do not consider them here.

3.2 Understanding Function Classes

Studying function classes can shed light on complexity classes of equivalence problems, and vice versa.

If f is a complete invariant for R and there is a function g such that $fgf = f$, then gf is a canonical form for R. Conversely, if c is a canonical form for R, then $c^2 = c$, so $g = \mathrm{id}$ is as above. Indeed, a canonical form is nothing more than an idempotent complete invariant. Thus we have shown:

Proposition 3.15. *A relation R has a canonical form if and only if it has a complete invariant f and there is a function g such that $fgf = f$.*

Although the g in the proposition is slightly weaker than a right inverse for f, any right inverse for f obviously satisfies the property of g. Hence answers to questions about function inversion imply results about complexity classes of equivalence problems. In particular, if $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ then every honest FP function has an inverse in NPSV, so $\mathrm{Ker}(\text{honest } \mathsf{FP}) \subseteq \mathrm{CF}(\mathsf{NPSV})$. Blass and Gurevich [BG84b] showed a partial converse to this result, restated here as Theorem 4.7.

In contrast to this, it is possible that there are functions that cannot be easily inverted, yet their kernels have canonical forms. For example, if one-one one-way functions exist, the kernel of any such function is the relation of equality, which has a trivial canonical form.

Although many function classes behave much like their language class counterparts, a notable exception concerns variations of the class P^{NP} in which the oracle access is restricted in various ways. $\mathsf{P}^{\mathsf{NP}}_{tt}$ is the class of all languages that are $\leq^P_{tt}$-reducible to an NP oracle. $\mathsf{P}^{\mathsf{NP}[\log]}$ denotes the class of languages that are Cook-reducible to a language in NP by a Cook reduction that makes at most $O(\log n)$ queries on inputs of length n. Both of the following statements can be proved using the same elementary argument:

$$\mathsf{P}^{\mathsf{NP}[\log]} \subseteq \mathsf{P}^{\mathsf{NP}}_{tt} \subseteq \mathsf{P}^{\mathsf{NP}} \quad \text{and} \quad \mathsf{FP}^{\mathsf{NP}[\log]} \subseteq \mathsf{FP}^{\mathsf{NP}}_{tt} \subseteq \mathsf{FP}^{\mathsf{NP}}.$$

Selman [Sel94] showed that $\mathsf{FP}^{\mathsf{NP}}_{tt} = \mathsf{FP}^{\mathsf{NP}}$ if and only if $\mathsf{P}^{\mathsf{NP}}_{tt} = \mathsf{P}^{\mathsf{NP}}$, so the larger two function classes are related in the same manner as the larger two language classes. However, in the case of the smaller two language classes, $\mathsf{P}^{\mathsf{NP}[\log]} = \mathsf{P}^{\mathsf{NP}}_{tt}$ [Wag87, Hem87, BH91], yet for the function classes, $\mathsf{FP}^{\mathsf{NP}[\log]} = \mathsf{FP}^{\mathsf{NP}}_{tt}$ implies $\mathsf{NP} = \mathsf{RP}$ and $\mathsf{P} = \mathsf{UP}$ [Sel94][1]. Given the comments above, it is possible that studying $\mathrm{CF}(\mathsf{FP}^{\mathsf{NP}[\log]})$, $\mathrm{Ker}(\mathsf{FP}^{\mathsf{NP}[\log]})$, $\mathrm{CF}(\mathsf{FP}^{\mathsf{NP}}_{\mathsf{tt}})$, and $\mathrm{Ker}(\mathsf{FP}^{\mathsf{NP}}_{\mathsf{tt}})$ could shed further light on these classes.

3.3 Chain Conditions on Languages

The following discussion was the genesis of this thesis. A *chain condition* is a condition on ascending or descending chains $A_1 \subsetneq A_2 \subsetneq A_3 \subsetneq \cdots$ of sub-objects of some mathematical object. Chain conditions have been quite successful in mathematics, particularly in algebra. For example, the ascending chain condition on a ring R is that there is no infinite strictly ascending chain of ideals of R; rings satisfying this condition are called Noetherian, and rings satisfying the analogous descending chain condition are called Artinian. In group theory, the definitions of nilpotent

[1]Selman [Sel94] actually showed that $\mathsf{FP}^{\mathsf{NP}[\log]} = \mathsf{FP}^{\mathsf{NP}}_{tt}$ implies $\mathsf{P} = \mathsf{FewP}$, where FewP is the class of NP problems with at most $poly(n)$ witnesses for inputs of length n. We mention this only for completeness; we will not mention FewP hereafter.

and solvable groups are defined by requiring that particular chains of subgroups must end in the trivial subgroup. What would a chain condition look like on a language, from the point of view of complexity theory?

The first question is which notion of sub-object to use. If we consider all subsets, most chain conditions become trivial. One possibility is to consider subsets in a certain complexity class. For example:does any NP-complete problem have an infinite, strictly ascending chain of subsets each of which is in P. If we allow finite differences, then the answer is trivially "yes." So a strictly ascending chain might be a set of languages $L_1 \subsetneq^* L_2 \subsetneq^* \cdots$ where each $L_i \in \mathsf{P}$ and $\subsetneq^*$ denotes that $L_i \subset L_{i+1}$ and $L_{i+1} \backslash L_i$ is infinite. However, it is again easy to construct such chains, so this notion did not seem particularly fruitful.

A natural way to extend the notion of an ideal in a ring or a normal subgroup in a group is to consider sub-objects that are the kernels of morphisms. Although the kernel of a function is in general an equivalence relation, in algebra, this equivalence relation is often entirely determined by a sub-object, which is also called the kernel of the function. For example, if $\varphi \colon G \to H$ is a group homomorphism, then $\{g \in G : \varphi(g) = 1\}$ is a normal subgroup of G, and completely defines the equivalence relation $\mathrm{Ker}(\varphi)$ (indeed, this subgroup is denoted $\mathrm{Ker}(\varphi)$).

In the case of complexity theory, a natural notion of morphism is a polynomial-time computable function. In thinking about chains of such equivalence relations, we were naturally led to the initial question of this thesis: is every polynomial-time decidable equivalence relation the kernel of some polynomial-time function?

4 Complexity Classes of Equivalence Relations

In this section we present new results on complexity classes of equivalence relations, except for new oracle separations, which we present in Section 5. In Section 4.1 we review the previously known results, including oracle separations, and results that follow from combinations of previous results but that have not been announced before. In Section 4.2 we present new consequences of the collapses of various classes of equivalence relations. In Section 4.3 we present several problems that are believed to be hard, but if the classes of equivalence relation collapse, become easy. We also define the notion of *kernel reduction*; any kernel-complete problem for PEq lies in Ker if and only if $\mathsf{Ker} = \mathsf{PEq}$, so any kernel-complete problem is a natural candidate for problems in $\mathsf{PEq} \backslash \mathsf{Ker}$.

4.1 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 $R \in \mathsf{PEq}$, 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 4.1 ([BG84a] Theorem 1)**.** *There is an equivalence relation $R \in \mathsf{CF}$ whose first canonical form is in $\mathsf{FP}^{\mathsf{NP}} = \mathsf{F}\Delta_2\mathsf{P}$ and is $\Delta_2\mathsf{P}$-hard, that is, it is essentially $\Delta_2\mathsf{P}$-complete.* □

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 4.2 ([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 $R \notin \mathrm{Ker}(\mathsf{FP}^{\mathsf{R}})$*, i. e.,* $\mathrm{Ker}(\mathsf{FP}^{\mathsf{R}}) \neq \mathsf{P}^{\mathsf{R}}\mathsf{Eq}$*.*

b. There is a function $f \in \mathsf{FP}$ *such that* $\mathrm{Ker}(f) \notin \mathrm{CF}(\mathsf{FP}^{\mathsf{f}})$*, i. e.,* $\mathrm{CF}(\mathsf{FP}^{\mathsf{f}}) \neq \mathrm{Ker}(\mathsf{FP}^{\mathsf{f}})$*.*

c. There is an idempotent function $f \in \mathsf{FP}$ *such that* $\mathrm{Ker}(f) \notin \mathsf{LexEqFP}^{\mathsf{f}}$*, i. e.,* $\mathsf{LexEqFP}^{\mathsf{f}} \neq \mathrm{CF}(\mathsf{FP}^{\mathsf{f}})$*.* □

The above theorem is proved by diagonalization. The proof of Theorem 4.2a in [BG84a] constructs an R with at most one nontrivial equivalence class at each length. The following extension of this result, giving an upper bound on R, is achieved by ensuring that R has *exactly* one nontrivial equivalence class of each length:

Theorem 4.3 ([BG84a] Theorem 3)**.** *There is an equivalence relation* $R \in \mathsf{LexEqNPSV}_{\mathsf{t}}^{\mathsf{R}}$ *that is not in* $\mathrm{Ker}(\mathsf{FP}^{\mathsf{R}})$*. In other words,* $\mathsf{P}^{\mathsf{R}}\mathsf{Eq} \cap \mathsf{LexEqNPSV}_{\mathsf{t}}^{\mathsf{R}} \not\subseteq \mathrm{Ker}(\mathsf{FP}^{\mathsf{R}})$*.*

Note that the relationship between $\mathrm{Ker}(\mathsf{FP})$ and $\mathsf{LexEqNPSV}_{\mathsf{t}}$ is unclear in either direction; Corollary 4.8, below, suggests that neither class is contained in the other. However, $\mathrm{Ker}(\mathsf{FP}) \subseteq \mathrm{Ker}(\mathsf{NPSV}_{\mathsf{t}})$, and one consequence of the above theorem is a relativized world in which $\mathrm{Ker}(\mathsf{FP}) \neq \mathsf{PEq} \cap \mathrm{Ker}(\mathsf{NPSV}_{\mathsf{t}})$.

The final result we mention in this direction is from Blass and Gurevich's second paper:

Theorem 4.4 ([BG84b] Theorem 5)**.** *There is an equivalence relation* $R \notin \mathrm{Ker}(\mathsf{NPSV}_{\mathsf{t}}^{\mathsf{R}})$*, i. e.,* $\mathsf{P}^{\mathsf{R}}\mathsf{Eq} \not\subseteq \mathrm{Ker}(\mathsf{NPSV}_{\mathsf{t}}^{\mathsf{R}})$*.*

In the following results, Blass and Gurevich [BG84b] showed that collapses of various classes of equivalence problems are equivalent to more standard complexity-theoretic hypotheses.

Theorem 4.5 ([BG84b] Theorem 1)**.** *The following statements are equivalent:*

1. $\mathsf{NPEq} \subseteq \mathsf{coNPEq}$
2. $\mathsf{coNPEq} \subseteq \mathsf{NPEq}$
3. $\mathrm{CF}(\mathsf{FP}) \subseteq \mathsf{LexEqNPSV}_{\mathsf{t}}$
4. $\mathsf{NP} = \mathsf{coNP}$

Proof [BG84b]. To show that (1), (2), and (4) are equivalent, consider a language $A \in \mathsf{NP}$ (resp., $A \in \mathsf{coNP}$). The equivalence follows easily by considering the equivalence relation generated by setting, for all x,

$$x \in A \iff 1x \sim 0x.$$

Next we show that if $\mathsf{NP} = \mathsf{coNP}$, then $\mathsf{PEq} \subseteq \mathsf{LexEqNPSV}_{\mathsf{t}}$, a stronger statement than (3). Let $R \in \mathsf{PEq}$. Then the language $R' = \{x : (\exists y)[y <_{lex} x \text{ and } (x, y) \in R]\}$ is in NP. Note that $x \in R'$ if and only if x is not the first member of its equivalence class. Since $\mathsf{NP} = \mathsf{coNP}$ by assumption, the language of strings that *are* the first member of their equivalence class is also in NP. Hence the first canonical form function is in $\mathsf{NPSV_t}$.

Finally we show that (3) implies (4). Let R be the equivalence relation constructed in Theorem 4.1. The first canonical form for R is $\Delta_2\mathsf{P}$-hard, but by assumption lies in $\mathsf{NPSV_t}$. Hence languages in $\mathsf{coNP} \subseteq \Delta_2\mathsf{P}$ can be recognized by an $\mathsf{NPSV_t}$ function, and thus $\mathsf{NP} = \mathsf{coNP}$. Blass and Gurevich [BG84b] also gave a more direct proof of this implication.

□

Theorem 4.6. *The following statements are equivalent:*

1. $\mathsf{LexEqNPSV_t} \subseteq \mathsf{PEq}$
2. $\mathsf{NPSV_t} \subseteq \mathsf{FP}$
3. $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$

Proof. The equivalence of (1) and (3) is exactly the statement of Theorem 2 from Blass and Gurevich [BG84b]. The equivalence of (2) and (3) follows from the fact that $\mathsf{NPSV_t} = \mathsf{FP}^{\mathsf{NP}\cap\mathsf{coNP}}$, which was essentially shown in [Sel94, HNOS94]. □

Some definitions are required before stating the next theorem. The *shrinking property* for NP is the statement that, for any two sets $A, B \in \mathsf{NP}$, there are subsets $A' \subseteq A$ and $B' \subseteq B$ such that A' and B' are disjoint and $A \cup B = A' \cup B'$. The *uniformization principle* for NP is the statement that $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$. Blass and Gurevich [BG84b] introduced both of these principles, based on analogous principles of the same name in computability theory and descriptive complexity theory. The original definition of the uniformization principle was stated somewhat differently, however, since function classes such as NPSV had not yet become standard, even though they were introduced by Book, Long, and Selman [BLS84] one issue before [BG84b].

Theorem 4.7 ([BG84b] Theorem 3)**.** *The following statements are equivalent:*

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

Finally, here we collect the results on the relationship between $\mathrm{Ker}(\mathsf{FP})$ and $\mathsf{LexEqNPSV_t}$:

Corollary 4.8. *If* $\mathrm{Ker}(\mathsf{FP}) \subseteq \mathsf{LexEqNPSV_t}$ *then* $\mathsf{NP} = \mathsf{coNP}$ *and* $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$*. If* $\mathsf{LexEqNPSV_t} \subseteq \mathrm{Ker}(\mathsf{FP})$ *then* $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$*.*

Proof. The first statement follows from Theorems 4.5 and 4.7. The second statement follows from Theorem 4.6. □

Corollary 4.9. $\mathrm{Ker}(\mathsf{FP}) = \mathsf{LexEqNPSV_t}$ *if and only if* $\mathsf{P} = \mathsf{NP}$ *if and only if* $\mathsf{NPSV} = \mathsf{PF}$*.*

Proof. The first equivalence follows from the previous corollary. The second equivalence was first announced in [Sel94]. □

Hemaspaandra, Naik, Ogihara, and Selman [HNOS94] showed that if $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ then $\mathsf{SAT} \in (\mathsf{NP} \cap \mathsf{coNP})/poly$. At the time, they showed that this implied $\mathsf{PH} = \Sigma_2\mathsf{P}$; shortly therefater Köbler and Watanabe [KW95] improved the collapse to $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$. Combined with Theorem 4.7, this immediately implies a result that has not been announced previously:

Corollary 4.10. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$.

4.2 New Collapses

Blass and Gurevich's [BG84b] proof that $\mathrm{Ker}(\mathsf{FP})_= \subseteq \mathrm{CF}(\mathsf{NPSV_t}) \implies \mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ essentially shows the following slightly stronger result. We reproduce the core of the proof here:

Theorem 4.11. *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 graph(f). If $\mathsf{CF} = \mathsf{Ker}$, then the equivalence relation

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

has a canonical form $c \in \mathsf{FP}$. Then the following machine 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 $f \in_c \mathsf{NPSV_g}$. □

Remark 4.12. Similar to the original result [BG84b], we can weaken the assumption of this theorem to $\mathrm{Ker}(\mathsf{FP})_{\mathsf{p}} \subseteq \mathrm{CF}(\mathsf{NPSV_t})$, without modifying the proof. ◁

Remark 4.13. Although this result follows from Blass and Gurevich's *proof* [BG84b], this result does not follow directly from their *result*, as $\mathsf{NPMV} \subseteq_c \mathsf{NPSV}$ is not known to imply $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$. ◁

Corollary 4.14. *If* $\mathsf{CF} = \mathsf{Ker}$ *then* $\mathsf{NP} = \mathsf{UP}$ *and* $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$.

Proof. Here we reproduce the proof that $\mathsf{NPMV_g} \subseteq_c \mathsf{NPSV_g}$ implies $\mathsf{NP} = \mathsf{UP}$, originally due to Selman [Sel94, GS88]. Let $L \in \mathsf{NP}$ and let V be a polynomial-time verifier for L, that is, $x \in L \iff (\exists^p y)[V(x,y)]$. Let f be the partial multi-valued function defined by

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

Then graph$(f) = V \in \mathsf{P}$, so $f \in \mathsf{NPMV_g}$. By assumption, then, f has a refinement $f' \in \mathsf{NPSV_g}$. Let V' be a polynomial-time decider for graph(f'). Then L is the projection of V' onto the first coordinate, and V' allows at most one witness for each $x \in L$. Thus $L \in \mathsf{UP}$.

The second claim — that $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$ — is exactly Corollary 4.10. □

Remark 4.15. Note that Corollary 4.10 does not imply $\mathsf{NP} = \mathsf{UP}$, as neither of the statements $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$ 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 4.16. † *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 and let V be a nondeterminstic polynomial-time machine with at most one accepting path for each input, such that $x \in L \iff (\exists y)[|y| \leq p(|x|)$ 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 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 fact 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 $\mathsf{CF} = \mathsf{PEq}$. 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. □

Corollary 4.17. † *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 4.11 and 4.16 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 $\mathsf{PH} \subseteq \mathsf{BPP}$ [Zac88], and since $\mathsf{BPP} \subseteq \mathsf{PH}$ [Lau83, Sip83], the two are equal. □

The collapse inferred here is stronger than that of Corollary 4.10, since $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ [Sip83] [2]. However, this result is incomparable to Corollary 4.10 since it also makes the stronger assumption $\mathsf{CF} = \mathsf{PEq}$, rather than only assuming $\mathsf{CF} = \mathsf{Ker}$.

4.2.1 Groupy witnesses for NP problems

We would like to extend the first half of Theorem 4.16 from UP to NP to give stronger evidence that $\mathsf{Ker} \neq \mathsf{PEq}$, 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 4.16 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

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

[2] In fact, what Sipser [Sip83] shows, as a corollary to a theorem of Gács, is that $\mathsf{BPP} \subseteq \mathsf{RP}^{\mathsf{NP}} \cap \mathsf{coRP}^{\mathsf{NP}}$. The proof that $\mathsf{RP} \cap \mathsf{coRP} = \mathsf{ZPP}$ relativizes, so $\mathsf{RP}^{\mathsf{NP}} \cap \mathsf{coRP}^{\mathsf{NP}} = \mathsf{ZPP}^{\mathsf{NP}}$. The result $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ has subsequently been re-proven by several different techniques.

Despite the Sipser-Gács result, Zachos and Heller's paper [ZH86] is often cited with the first proof that $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$.

Goldreich and Zuckerman [GZ97] gave another proof that $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ by showing that $\mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$.

More recently, Cai [Cai07] shows that $\mathsf{S}_2^{\mathsf{P}} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. Combined with the result by Canetti [Can96] and Russell and Sundaram [RS95] that $\mathsf{BPP} \subseteq \mathsf{S}_2^{\mathsf{P}}$, this gives yet another proof that $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$.

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 4.16 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 cosets of some subgroup $H \leq G$, find a generating set for H [Kit95]. Hidden subgroup problems have played a central role in 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 shortest vector problem in a lattice reduces to the dihedral HSP, which is solvable in subexponential quantum time [Kup05]. The graph isomorphism reduces to the HSP for the symmetric group, 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 [IMS03, FIM+03, GSVV04].

The proof of Theorem 4.16 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 [Sim94] 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 [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 4.18. † Let $L \in \mathsf{NP}$. For each a let $W(a)$ denote 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 mul, gen, 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 two results are corollaries to the proof, rather than the result, of Theorem 4.16.

Corollary 4.19. † *If* $\mathsf{Ker} = \mathsf{PEq}$ *and a language* $L \in \mathsf{NP}$ *has groupy witnesses in a family of groups* $\mathcal{G}$*, 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 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 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 $W(a)$. Therefore L reduces to the hidden subgroup problem. $\square$

Corollary 4.20. † *If* $\mathsf{Ker} = \mathsf{PEq}$ *and the language* L *has Abelian groupy witnesses, then* $L \in \mathsf{BQP}$.

Corollary 4.21. † *Every language in* UP *has Abelian groupy witnesses.*

Open Question 4.22. Are there NP-complete problems with Abelian groupy witnesses? Assuming $\mathsf{P} \neq \mathsf{NP}$, are there any problems in $\mathsf{NP} \setminus \mathsf{UP}$ with Abelian groupy witnesses?

4.3 New Hard Problems

4.3.1 Hard problems from NP-complete problems

By the technique in the proof of Theorem 4.11, any NP-complete problem L can be transformed into an equivalence relation $R \in \mathsf{Ker}$ such that $R \notin \mathsf{CF}$ unless $\mathsf{NP} = \mathsf{UP}$.

4.3.2 Factoring integers

Proposition 4.23. † *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 [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 $f \in \mathsf{FP}$.

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. $\square$

4.3.3 Collision-free hash functions

Collision-free hash functions are a useful cryptographic primitive (see, e. g., [BSnP95]). Proposition 4.23 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 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, $E \in \mathsf{FP}$.

Proposition 4.24. † *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 4.23, 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 $\Pr_x[f(x) \neq x] = \Pr_x[f(x) \neq x | \#[x]_R \geq 2] \Pr_x[\#[x]_R \geq 2] \geq \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2^{|i|+1}}\right) > \frac{1}{4}$. □

4.3.4 Cospectral mates

Determining whether two graphs have the same spectrum is simple linear algebra, hence the relation of graph cospectrality, Cospec, is in Ker. Finding non-isomorphic cospectral graphs, called *cospectral mates*, is an active area of research (see §13.2 of Brouwer and Haemers [BH], and references therein). However, no polynomial-time canonical form, nor even expected polynomial-time canonical form (see the end of Section 3.1.1) is known for Cospec. Graph cospectrality is thus a natural equivalence problem that may lie in $\mathsf{Ker} \backslash \mathsf{CF}$.

4.3.5 Subgroup equivalence

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 $x \in \langle g_1, \ldots, g_t \rangle$ (here $\langle \cdots \rangle$ denotes group generation, not tuple encoding). 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 equivalence problem. Thus subgroup equivalence problems are a potential source of candidates for problems in $\mathsf{PEq} \backslash \mathsf{Ker}$.

Fortunately or unfortunately, the subgroup equivalence problem for permutation groups on $\{1, \ldots, n\}$ has a polynomial-time canonical form, via a simple modification [Bab08] of the classic techniques of Sims [Sim70, Sim71]. The analysis showing that these techniques yield polynomial-time algorithms was not initially obvious, but was eventually performed by Furst, Hopcroft, and Luks [FHL80] and Knuth [Knu91]. Knuth [Knu91] gave further historical remarks at the end of his paper.

4.3.6 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 the formula isomorphism problem, discussed here in Section 3.1. If f and g are given by their complete 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 is a candidate problem in $\mathsf{PEq}\backslash\mathsf{Ker}$.

4.3.7 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 4.25. † An equivalence relation R *kernel-reduces* to an equivalence relation S, denoted $R \leq_{ker}^{P} S$, 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 $R \leq_{ker}^{P} S$ via f, then $R \leq_{m}^{P} S$ via $(x, y) \mapsto (f(x), f(y))$, leading to the question:

Open Question 4.26. Are kernel reduction are 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 $R \in \mathsf{PEq}$ 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. This shows that if $\mathsf{P} = \mathsf{NP}$ then kernel reduction and Karp reduction are distinct on PEq. More generally, if $R \leq_{ker} S$, 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 4.27. Are there PEq-complete equivalence problems?

5 Oracles

We make extensive use of generic oracles for various notions of genericity, i. e., forcing. For an overview of these techniques and their use in complexity theory, see [FFKL03]. Similar to [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 $\langle x, z\rangle \in \sigma$.

It follows from the general results of [FFKL03] that transitive generic oracles exist.

Theorem 5.1. † *There are oracles A, B, and C relative to which* $\mathsf{P} \neq \mathsf{NP}$ *and*

$$\mathrm{CF}(\mathsf{FP^A}) \neq \mathrm{Ker}(\mathsf{FP^A}) \neq \mathsf{P^AEq},$$
$$\mathrm{CF}(\mathsf{FP^B})_\mathsf{p} = \mathrm{Ker}(\mathsf{FP^B})_\mathsf{p} \textit{ and } \mathrm{Ker}(\mathsf{FP^B}) \neq \mathsf{P^BEq}$$

We break most of the proof into three lemmata. The proofs of Lemmata 5.3 and 5.4 are straightforward 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* $\mathsf{Ker} \neq \mathsf{PEq}$*.*

Proof. Let τ be a transitive condition, and let $\overline{\tau}$ denote the minimal transitive oracle extending τ, that is, $(a, a) \in \overline{\tau}$ for every $a \in \Sigma^*$, but the only pairs $(x, y) \in \overline{\tau}$ 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 $M^{\overline{\tau}}(x) = M^{\overline{\tau}}(x)$, then extend τ to length $p(n)$ as $\overline{\tau}$. Then $x \not\sim_\tau y$ but $M^\tau(x) = M^\tau(y)$.

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 (x, y) or (y, x) in the computation of $M^{\overline{\tau}}(y)$. 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 $\mathsf{Ker} \neq \mathsf{PEq}$ relative to O. □

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

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

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

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 $R^O = \mathrm{Ker}(read^O)$.

Let τ be a Cohen condition, and let $\overline{\tau}$ denote the oracle extending τ which has value 2 outside $\mathrm{dom}(\tau)$. 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 $read^{\overline{\tau}}(x)$ 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 $read^{\overline{\tau_x}}$ is the identity on all strings of length n except $read^{\overline{\tau_x}}(x) = 1^{n+1}$. 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 $M^{\overline{\tau_x}}(x) \neq x$, 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 $\mathsf{CF} \neq \mathsf{Ker}$. □

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 $A \oplus O$,* $\mathrm{CF}(\mathsf{FP})_{\mathsf{p}} = \mathrm{Ker}(\mathsf{FP})_{\mathsf{p}}$.

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 $R = \mathrm{Ker}(f^O)$, and suppose that R is polynomially bounded by q. That is, if $(x, y) \in R$ then $|x| \leq q(|y|)$. 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

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

$r_y = \min R_y$, 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 $\mathsf{FP^O}$. 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 $\log p(|x|) \leq \ell \leq q(p(|x|))$.

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.

$\square$

Proof of Theorem 5.1. ($\mathsf{CF} \neq \mathsf{Ker} \neq \mathsf{PEq}$) 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*.

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

Open Question 5.6. 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 5.7. Is there an oracle relative to which $\mathsf{CF} \neq \mathsf{Ker} = \mathsf{PEq}$?

6 Future Work

In this thesis, we developed new connections between complexity classes of equivalence relations and probabilistic and quantum computation. We extended previous collapse results, gave new oracles for these classes, and provided several natural problems that are candidate witnesses for the distinctness of these complexity classes of equivalence relations.

Here we present several directions for future work. We collect the open problems listed throughout this thesis for convenience, and present several other possible directions.

In textual order:

(3.14) Is it the case that for every equivalence problem $R \in \mathsf{NP} \backslash \mathsf{P}$, the canonical form problem Cook-reduces to the complete invariant problem?

(4.22) 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?

(4.26) Are kernel-reduction are 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 $R \not\leq_{ker}^P S$?

(4.27) Are there PEq-complete equivalence problems?

(5.6) 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 $\mathsf{CF} = \mathsf{Ker} = \mathsf{PEq}$?

(5.7) Is there an oracle relative to which $\mathsf{CF} \neq \mathsf{Ker} = \mathsf{PEq}$?

Here are some further directions for future researh, in no particular order:

- Is LEq contained in $\mathrm{CF}(\mathsf{FL}^{\mathsf{NL}})$? Is it contained in $\mathrm{CF}(\mathsf{FP})$? In $\mathrm{Ker}(\mathsf{FP})$? 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 [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 $\mathsf{FL}^{\mathsf{NL}}$. 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 if and only if there is a logspace machine $M(x, \vec{y})$ that reads is 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 $\mathsf{LexEqFL}^{\mathsf{NL}}$, but it is not clear that this captures all of LEq.

- Does $\mathrm{CF}(\mathsf{FL}) = \mathrm{Ker}(\mathsf{FL})$ imply $\mathsf{NL} = \mathsf{UL}$? Note that $\mathsf{NL} = \mathsf{UL}$ if and only if $\mathsf{FL}^{\mathsf{NL}} \subseteq \sharp\mathsf{L}$ [AJ93].

- Does $\mathrm{CF}(\mathsf{FL}) = \mathsf{LEq}$ imply $\mathsf{UL} \subseteq \mathsf{RL}$? A positive answer to this question and the previous one would give very strong evidence that $\mathrm{CF}(\mathsf{FL}) \neq \mathsf{LEq}$, as significant progress has been made towards showing $\mathsf{L} = \mathsf{RL}$ [RTV05].

- Study expected polynomial-time canonical forms (see the end of Section 3.1.1). If every $R \in \mathrm{Ker}(\mathsf{FP})$ has an expected polynomial-time canonical form, does PH collapse?

- What is the exact relationship between $\mathrm{CF}(\mathsf{FP}^{\mathsf{NP}[\log]})$, $\mathrm{Ker}(\mathsf{FP}^{\mathsf{NP}[\log]})$, $\mathrm{CF}(\mathsf{FP}^{\mathsf{NP}}_{\mathsf{tt}})$, and $\mathrm{Ker}(\mathsf{FP}^{\mathsf{NP}}_{\mathsf{tt}})$? In particular, is $\mathrm{Ker}(\mathsf{FP}^{\mathsf{NP}[\log]}) \subseteq \mathrm{CF}(\mathsf{FP}^{\mathsf{NP}}_{\mathsf{tt}})$ (see Section 3.2)?

- 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 4.3.5).

- If $\mathsf{Ker} = \mathsf{PEq}$, does PH collapse?

- $\mathsf{LexEqFP}^{\Sigma_i \mathsf{P}} \stackrel{?}{=} \mathrm{CF}(\mathsf{FP}^{\Sigma_i \mathsf{P}}) \stackrel{?}{=} \mathrm{Ker}(\mathsf{FP}^{\Sigma_i \mathsf{P}}) \stackrel{?}{=} \mathsf{P}^{\Sigma_i \mathsf{P}}\mathsf{Eq}$. If $\mathrm{Ker}(\mathsf{FP}^{\Sigma_i \mathsf{P}}) = \mathsf{P}^{\Sigma_i \mathsf{P}}\mathsf{Eq}$ does PH collapse?

- Study counting classes of equivalence relations. For an equivalence relation R, the associated counting function is $f(x) = \#[x]_R$.

- Study complexity classes of lattices, partial orders, and total orders.

Acknowledgments

I would like to thank Lance Fortnow for his guidance and effort, as well as for his joint work on these problems. I would like to thank Lance Fortnow and Laci Babai for carefully editing portions of this thesis. I 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 4.16, and Laci pointed out the canonical form for subgroup equivalence of permutation groups [Bab08]. I would also like to thank Andreas Blass for pointing me to the original two papers [BG84a, BG84b]. Last but by no means least, I would like to thank my family and friends, without whose guidance and love none of this would have been possible. Friendship, romance, love, and parternship are some of life's greatest joys; I would like to thank Nikki Pfarr for all of these and more.

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].

[AW08] Scott Aaronson and Avi Wigderson, *Algebrization: a new barrier in complexity theory*, STOC '08: 40th Annual ACM Symposium on Theory of Computing, ACM, 2008, pp. 731–740.

[Bab77] László Babai, *On the isomorphism problem*, September 1977, manuscript, distributed at the 1977 International Symposium on the Fundamentals of Computation Theory.

[Bab79] László Babai, *Monte Carlo algorithms in graph isomorphism testing*, Tech. Report DMS 79-10, Université de Montréal, 1979.

[Bab85] László Babai, *Trading group theory for randomness*, STOC '85: 17th Annual ACM Symposium on Theory of Computing, ACM, 1985, pp. 421–429.

[Bab08] László Babai, *Canonical generators for permutation groups*, May 2008, personal communication.

[BCKT96] Nader H. Bshouty, Richard Cleve, Sampath Kannan, and Christino Tamon, *Oracles and queries that are sufficient for exact learning*, J. Comput. System Sci. **52** (1996), 421–433.

[BF90] László Babai and Lance Fortnow, *A characterization of* ♯P *by arithmetic straight line programs*, FOCS '90: 31st Annual IEEE Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press, 1990, preliminary version of [BF91], pp. 26–34.

[BF91] László Babai and Lance Fortnow, *Arithmetization: a new method in structural complexity theory*, Comput. Complexity **1** (1991), no. 1, 41–66, originally appeared as [BF90].

[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).

[BFL90] László Babai, Lance Fortnow, and Carsten Lund, *Nondeterministic exponential time has two-prover interactive protocols*, FOCS '90: 31st Annual IEEE Symposium on Foundations of Computer Science, vol. 1, IEEE Computer Society, 1990, preliminary version of [BFL91], pp. 16–25.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund, *Nondeterministic exponential time has two-prover interactive protocols*, Comput. Complexity **1** (1991), no. 1, 3–40, originally appeared as [BFL90].

[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.

[BH] Andries Brouwer and Willem Haemers, *Spectra of graphs*, lecture notes, available from `http://www.cwi.nl/~aeb/math/ipm.pdf`.

[BH88] Samuel R. Buss and Louise Hay, *On truth-table reducibility to SAT and the difference hierarchy over NP*, Proceedings of the Structure in Complexity Conference, IEEE Computer Society Press, 1988, preliminary version of [BH91], pp. 224–233.

[BH91] Samuel R. Buss and Louise Hay, *On truth-table reducibility to SAT*, Inform. and Comput. **91** (1991), no. 1, 86–102, originally appeared as [BH88].

[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.

[BHZ87] Ravi Boppana, Johan Håstad, and Stathis Zachos, *Does co-NP have short interactive proofs?*, Inform. Process. Lett. **25** (1987), 27–32.

[BK79] László Babai and Ludik Kučera, *Canonical labelling of graphs in linear average time*, FOCS '79: 20th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 1979, pp. 39–46.

[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.

[BLS84] Ronald Book, Timothy Long, and Alan L. Selman, *Quantitative relativizations of complexity classes*, SIAM J. Comput. **13** (1984), no. 3, 461–847.

[BM88] László Babai and Shlomo Moran, *Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity class*, J. Comput. System Sci. **36** (1988), no. 2, 254–276.

[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.

[Cai07] Jin-Yi Cai, $\mathrm{S}_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$, J. Comput. System Sci. **73** (2007), no. 1, 25–35.

[Can96] Ran Canetti, *More on BPP and the polynomial-time hierarchy*, Inform. Process. Lett. **57** (1996), no. 5, 237–241.

[Coo71] Stephen Cook, *The complexity of theorem-proving procedures*, STOC '71: 3rd Annual ACM Symposium on Theory of Computing, ACM Press, 1971, pp. 151–158.

[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.

[DH07] Max Dehn and Poul Heegaard, *Analysis situs*, Enzyklopädie der Mathematischen Wissenschaften, vol. 3, Leipzig, 1907.

[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.

[FG08] Lance Fortnow and Joshua A. Grochow, *Complexity classes of equivalence problems revisited*, 2008, submitted for conference publication.

[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.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput. **18** (1989), no. 1, 186–208.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that yield nothing but their validity, and a methodology of cryptographic protocol design*, FOCS '86: 27th Annual IEEE Symposium on Foundations of Computer Science, 1986, preliminary version of [GMW91], pp. 174–187.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that yield nothing but their validity, or All languages in NP have zero-knowledge proof systems*, J. Assoc. Comput. Mach. **38** (1991), no. 3, 691–729, originally appeared as [GMW86].

[GS86] Shafi Goldwasser and Michael Sipser, *Private coins versus public coins in interactive proof systems*, STOC '86: 18th Annual ACM Symposium on Theory of Computing, ACM, 1986, pp. 59–68.

[GS88] Joachim Grollman and Alan L. Selman, *Complexity measures for public-key cryptosystems*, SIAM J. Comput. **17** (1988), no. 2, 309–335.

[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.

[GZ97] Oded Goldreich and David Zuckerman, *Another proof that BPP subseteq PH (and more)*, Tech. Report TR97-045, Electronic Colloquium on Computational Complexity, 1997.

[Hač79] L. G. Hačijan, *A polynomial algorithm in linear programming*, Dokl. Akad. Nauk SSSR **244** (1979), no. 5, 1093–1096, English translation: [Kha79].

[Hem87] Lane A. Hemachandra, *Counting in structural complexity theory*, Ph.D. thesis, Cornell University, 1987, note: L. A. Hemachandra is the pre-marriage name of L. A. Hemaspaandra.

[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, pp. 56–64.

[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.

[HW74] John Hopcroft and J. K. Wong, *Linear time algorithm for isomorphism of planar graphs (preliminary report)*, STOC '74: 6th Annual ACM Symposium on Theory of Computing, ACM, 1974, pp. 172–184.

[Imm88] Neil Immerman, *Nondeterministic space is closed under complementation*, SIAM J. Comput. **17** (1988), no. 5, 935–938.

[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.

[Kar72] Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), Plenum, New York, 1972, pp. 85–103.

[Kel57] Paul J. Kelly, *A congruence theorem for trees*, Pacific J. Math. **7** (1957), 961–968.

[Kha79] L. G. Khachiyan, *A polynomial algorithm in linear programming*, Soviet Math. Dokl. **20** (1979), no. 1, 191–194, English translation of [Hač79].

[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.

[Ko82] Ker-I Ko, *Some observations on the probabilistic algorithms and NP-hard problems*, Inform. Process. Lett. **14** (1982), no. 1, 39–43.

[KST93] Johannes Köbler, Uwe Schöning, and Jacobo Torán, *The graph isomorphism problem: its structural complexity*, Progress in Theoretical Computer Science, Birkhäuser Boston Inc., Boston, MA, 1993.

[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].

[Lad75] Richard E. Ladner, *On the structure of polynomial time reducibility*, J. Assoc. Comput. Mach. **22** (1975), 155–171.

[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.

[LFKN90] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan, *Algebraic methods for interactive proof systems*, FOCS '90: 31st Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 1990, pp. 2–10 vol.1.

[Luk82] Eugene M. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, J. Comput. System Sci. **25** (1982), no. 1, 42–65.

[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.

[Mat79] Rudolf Mathon, *A note on the graph isomorphism counting problem*, Inform. Process. Lett. **8** (1979), no. 3, 131–132.

[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.

[MS72] Albert R. Meyer and Larry J. Stockmeyer, *The equivalence problem for regular expressions with squaring requires exponential space*, SWAT '72: 13th Annual Symposium on Switching and Automata Theory (Washington, DC, USA), IEEE Computer Society, 1972, pp. 125–129.

[NC00] Michael A. Nielson and Isaac L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[Nov55] 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: [Nov58].

[Nov58] P. S. Novikov, *On the algorithmic insolvability 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.

[RS95] Alexander Russell and Ravi Sundaram, *Symmetric alternation captures BPP*, Tech. Report 7, 1995.

[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.

[Sav69] Walter J. Savitch, *Deterministic simulation of non-deterministic turing machines (detailed abstract)*, STOC '69: 1st Annual ACM Symposium on Theory of Computing (New York, NY, USA), ACM, 1969, pp. 247–248.

[Sch83] Uwe Schöning, *A low and a high hierarchy within NP*, J. Comput. System Sci. **27** (1983), no. 1, 14–28.

[Sch88] Uwe Schöning, *Graph isomorphism is in the low hierarchy*, J. Comp. System. Sci. **37** (1988), 312–323.

[Sch89] Uwe Schöning, *Probabilistic complexity classes and lowness*, J. Comp. System. Sci. **39** (1989), 84–100.

[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.

[Sha90] Adi Shamir, *IP = PSPACE*, FOCS '90: 31st Annual IEEE Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press, 1990, preliminary version of [Sha92], pp. 11–15.

[Sha92] Adi Shamir, *IP = PSPACE*, J. ACM **39** (1992), no. 4, 869–877, originally appeared as [Sha90].

[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.

[Sze88] Róbert Szelepcsényi, *The method of forced enumeration for nondeterministic automata*, Acta Inf. **26** (1988), no. 3, 279–284.

[Thi00] Thomas Thierauf, *The computational complexity of equivalence and isomorphism problems*, Lecture Notes in Computer Science, no. 1852, Springer, New York, 2000.

[Tod89] Seinosuke Toda, *On the computational power of PP and #P*, FOCS '89: 30th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 1989, pp. 514–519.

[Wag87] Klaus W. Wagner, *Log query classes*, Tech. Report TR-145, Universtät Augsburg, Institut für Mathematik, 1987.

[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.