On the computational complexity of equivalence relations under kernel reductions

Jeffrey Finkelstein

August 17, 2010

Abstract

In this paper we analyze the notion of kernel reductions among equivalence problems, as defined in Fortnow and Grochow. We first examine what kernel reductions look like, practically, among feasible equivalence problems. We next provide some evidence that the restriction of a polynomial time kernel reduction is not on the time complexity of the reduction, but on the number of equivalence classes in the equivalence relations being reduced. We then examine the graph isomorphism problem and problems equivalent to it under polynomial time many-one reductions, and find that of these problems the ones which are equivalence problems are in fact equivalent to the graph isomorphism problem under polynomial time kernel reductions.

1 Introduction

Problems of determining equivalence of objects abound in theoretical computer science and have widespread applications in a more practical setting. Indeed, one of the most well-studied open problems in computer science is determining the computational complexity of the graph isomorphism problem (GI). If $\mathsf{P} \neq \mathsf{NP}$, then GI is a candidate for $\mathsf{P} \backslash \mathsf{NP}$.

A recent paper by Fortnow and Grochow [FG09] defines a new kind of reduction among equivalence relations, a *kernel reduction*, which may be more natural than the usual many-one reduction for problems of determining equivalence. They suggest considering polynomial time kernel reductions for equivalence problems in P, but in section 3 we will provide some evidence that allowing a polynomial time kernel reduction among equivalence relations for which membership can be decided in polynomial time may be too powerful. The main result to this effect is Theorem 3.22, in which we show that the problem of computing a polynomial time kernel reduction among feasible equivalence problems can, under certain conditions, be reduced to the problem of computing representatives of equivalence classes for each respective equivalence relation.

In section 4 we attempt to determine whether there exist any equivalence problems which are complete under polynomial time kernel reductions, and what they look like. We also analyze the class of equivalence problems which are equivalent under polynomial time many-one reductions to the graph isomorphism problem. Every known reduction that we have found among equivalence problems in NP and the graph isomorphism problem are in fact kernel reductions.

We show in Lemma 2.13 that a kernel reduction implies a many-one reduction. One of our goals in this paper is to gain some intuition as to whether the two kinds of reductions are different. However, in section 5 we describe our difficulties in determining whether a many-one reduction implies a kernel reduction or whether they are different.

2 Preliminaries

Definition 2.1. Let $\Sigma = \{0, 1\}$. Then Σ^* is the set of all strings consisting of elements of Σ. In particular, Σ^* is the set of all binary strings.

Definition 2.2. A subset of Σ^* is called a *language*.

Since we wish to consider only sets whose members are binary strings, in order to define sets of pairs of binary strings we need to make use of a *pairing function*, $\langle \cdot, \cdot \rangle \colon \Sigma^* \times \Sigma^* \to \Sigma^*$. For example, consider the pairing function defined as follows: $\langle x, y \rangle$ duplicates each bit in x and y to produce $\hat{x}$ and $\hat{y}$, respectively, then outputs $\hat{x}01\hat{y}$. For example, the pair $(0110, 1100)$ can be encoded as $\langle 0110, 1100 \rangle = 001111000111110000$.

For tuples of arbitrary fixed length we define the encoding inductively using the pairing function. For example, the triple (x, y, z) can be encoded as $\langle x, y, z \rangle = \langle \langle x, y \rangle, z \rangle$, and so on for tuples with a greater number of elements.

For a more thorough treatment of languages, alphabets, encodings, pairing functions, etc., see [BDG95].

Definition 2.3. P is the class of languages for which membership can be decided by a Turing machine running in deterministic polynomial time.

NP is the class of languages for which membership can be decided by a Turing machine running in non-deterministic polynomial time.

NP can alternately be defined as the class of languages for which membership can be verified by a Turing machine running in deterministic polynomial time.

The equivalence of these two definitions of the complexity class NP can be found, for example, in [Sip06].

Definition 2.4. Let S, T be sets, let $S' \subseteq S$, and let $f : S' \to T$ be a function. Then f is called a *partial function from S to T*. If $S' = S$ then f is called a *total function from S to T*.

Definition 2.5. FP is the class of total functions computable by a Turing machine running in polynomial time. Specifically, $f \colon \mathbb{N} \to \mathbb{N}$ is in FP if there exists a Turing machine which, on input n, halts with $f(n)$ on its tape.

Definition 2.6. Let A, B be languages. We say A *polynomial time, many-one reduces to* B if $\exists f \in \mathsf{FP} : \forall w \in \Sigma^*, w \in A \iff f(w) \in B$. We denote this by $A \leq_m^p B$.

We say A *is polynomial time, many-one equivalent to* B if $A \leq_m^p B$ and $B \leq_m^p A$. We denote this by $A \equiv_m^p B$.

Definition 2.7. If a language A is in NP and $\forall B \in \mathsf{NP}$, $B \leq_m^p A$, then we say A is NP*-complete*, A is *complete under $\leq_m^p$ reductions in* NP, or A is $\leq_m^p$*-complete*.

Definition 2.8. Let U be a set. Let $R \subseteq U \times U$. Then R is an *equivalence relation* on U if the following conditions hold:

i. (reflexivity) $\forall x \in U$, $(x, x) \in R$

ii. (symmetry) $\forall x, y \in U$, if $(x, y) \in R$ then $(y, x) \in R$

iii. (transitivity) $\forall x, y, z \in U$, if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$

If $(x, y) \in R$, we say x *relates to* y or x *is equivalent to* y *under* R. We denote this by $x \sim_R y$ or $x \sim y$.

Note that an equivalence relation is a generalization of the idea of equality; whereas every object is equal to itself and only itself, many objects can be equivalent with respect to some equivalence relation.

Definition 2.9. Let U be a set, let R be an equivalence relation on U, let $x \in U$. The *equivalence class* of x is $[x]_R = \{y \in U | (x, y) \in R\}$. When the context is clear, we sometimes omit the subscript, and just write $[x]$.

For all equivalence relations on a set U, each element of U is in exactly one equivalence class and $U = \bigcup_{x \in U} [x]$, so the equivalence classes of an equivalence relation provide a partition of U.

In [FG09], Fortnow and Grochow give a formal definition of a new class of languages: the class of equivalence problems which can be decided in polynomial time. They also provide a natural reduction for languages in this complexity class. We provide the definition, along with its generalization in NP, here.

Definition 2.10. PEq is the class of equivalence relations for which membership can be decided by a Turing machine running in deterministic polynomial time.

NPEq is the class of equivalence relations for which membership can be decided by a Turing machine running in non-deterministic polynomial time.

NPEq can alternately be defined as the class of equivalence relations for which membership can be verified by a Turing machine running in deterministic polynomial time.

Technically, an equivalence relation on Σ^* is a set of pairs of binary strings, but PEq and NPEq are classes of *languages*, which are sets of binary strings (not pairs). We use the pairing function discussed above to encode pairs of binary strings in an equivalence relation (that is, the binary strings which relate) as single binary strings. For example an equivalence relation $R \subseteq \Sigma^* \times \Sigma^*$ can be encoded as a language $L_R = \{\langle x, y \rangle | (x, y) \in R\}$. Then we can discuss membership of L_R in complexity classes such as PEq or NPEq, etc. Throughout this paper, we will abuse these formalities and use the convention that an equivalence relation is defined as its corresponding encoding described here, so that we may simply consider equivalence relations as members of complexity classes, and consider reductions among them.

We explicitly show here the containments among P, NP, PEq, and NPEq.

Lemma 2.11.

i. $\mathsf{P} \subseteq \mathsf{NP}$

ii. $\mathsf{PEq} \subseteq \mathsf{NPEq}$

iii. $\mathsf{PEq} \subsetneq \mathsf{P}$

iv. $\mathsf{NPEq} \subsetneq \mathsf{NP}$

Proof. Any language which can be decided in deterministic polynomial time can be verified in deterministic polynomial time, by deciding whether the input is in the language, thus $\mathsf{P} \subseteq \mathsf{NP}$. The same argument applies to equivalence problems, so $\mathsf{PEq} \subseteq \mathsf{NPEq}$. $\mathsf{PEq} \subseteq \mathsf{P}$ and $\mathsf{NPEq} \subseteq \mathsf{NP}$ follows immediately from the definitions. A language in P which is not an equivalence relation is $MAJORITY = \{w|$ the number of ones in w is greater than $\frac{|w|}{2}\}$, so $\mathsf{PEq} \subsetneq \mathsf{P}$. In fact, if $MAJORITY \in \mathcal{C}$, where $\mathcal{C}$ is any complexity class, then $\mathcal{C}\mathsf{Eq} \neq \mathcal{C}$. $MAJORITY \in \mathsf{P} \subseteq \mathsf{NP}$, therefore $\mathsf{NPEq} \subsetneq \mathsf{NP}$. □

Definition 2.12. Let R and S be equivalence relations on Σ^*. We say R *kernel reduces to* S if $\exists f \colon \Sigma^* \to \Sigma^* : \forall x, y \in \Sigma^*, \langle x, y \rangle \in R \iff \langle f(x), f(y) \rangle \in S$. We denote this by $R \leq_{ker} S$.

We say R *is kernel equivalent to* S if $R \leq_{ker} S$ and $S \leq_{ker} R$. We denote this by $R \equiv_{ker} S$.

If $f \in \mathsf{FP}$, then we say R *polynomial time kernel reduces to* S, and use the notation $R \leq^p_{ker} S$ and $R \equiv^p_{ker} S$.

Note the difference between a kernel reduction and a many-one reduction: a kernel reduction maps $\langle x, y \rangle \in R$ to $\langle f(x), f(y) \rangle \in S$, whereas a many-one reduction maps $\langle x, y \rangle \in R$ to $f(\langle x, y \rangle) \in S$, for some polynomial time computable function f. Informally, a function which computes a many-one reduction has access to both x and y, but a function which computes a kernel reduction has access to only one of x and y at a time. Because the many-one reduction seems to be at least as powerful as the kernel reduction, we can prove the following lemma, which is essential for proving Theorem 4.22.

Lemma 2.13. *Let R, S be equivalence relations on Σ^*. If $R \leq^p_{ker} S$ then $R \leq^p_m S$.*

Proof. Since $R \leq^p_{ker} S$, $\exists f \in \mathsf{FP} : \forall x, y \in \Sigma^*, \langle x, y\rangle \in R \iff \langle f(x), f(y)\rangle \in S$. Define $g \in \mathsf{FP}$ by $g(\langle x, y\rangle) = \langle f(x), f(y)\rangle$. Therefore $R \leq^p_m S$ by g. □

As an analog to polynomial time many-one completeness in NP, we define a similar notion of polynomial time completeness under kernel reductions in NPEq.

Definition 2.14. An equivalence relation R is PEq-*complete* if $R \in \mathsf{PEq}$ and $\forall S \in \mathsf{PEq}$, $S \leq^p_{ker} R$.

An equivalence relation R is NPEq-*complete* if $R \in \mathsf{NPEq}$ and $\forall S \in \mathsf{NPEq}$, $S \leq^p_{ker} R$.

3 Kernel reductions among feasible equivalence problems

With these definitions in place, we can now define some examples of equivalence relations in PEq, feasible equivalence problems.

Definition 3.1. The *parity function*, $\pi\colon \Sigma^* \to \{0, 1\}$, is defined by $\pi(w) = 0$ if the number of ones in w is even, and $\pi(w) = 1$ if the number of ones in w is odd, for all $w \in \Sigma^*$.

Definition 3.2. $R_{par} = \{\langle x, y\rangle | \pi(x) = \pi(y)\}$

We use the subscript "*par*" to denote that pairs of binary strings in this equivalence relation have the same parity.

Definition 3.3. The *bitcount function*, $\beta\colon \Sigma^* \to \mathbb{N}$, is defined by $\beta(w)$ equals the number of ones in w, for all $w \in \Sigma^*$.

Definition 3.4. $R_{bc} = \{\langle x, y\rangle | \beta(x) = \beta(y)\}$

We use the subscript "*bc*" to denote that pairs of binary strings in this equivalence relation have the same bitcount.

Definition 3.5. $R_{eq} = \{\langle x, y\rangle | x = y\}$

R_{eq} is called the *equality relation*.

Definition 3.6. The *bitwise complement function*, $\bar{\cdot}\colon \Sigma^* \to \Sigma^*$, is defined for all $w \in \Sigma^*$, where $w = w_1 w_2 \cdots w_{|w|}$, by $\overline{w_i} = 0$ if $w_i = 1$ and $\overline{w_i} = 1$ if $w_i = 0$, for all $i \leq |w|$.

Definition 3.7. $R_{eqc} = \{\langle x, y\rangle | x = y \text{ or } x = \bar{y}\}$

We use the subscript "*eqc*" to denote that pairs of binary strings in this equivalence relation are either equal or bitwise complements.

Definition 3.8. The *exclusive or operation*, $\oplus\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ is defined for all $x, y \in \Sigma^*$, where $|x| = |y| = n$, $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$, by $x_i \oplus y_i = 0$ if $x_i = y_i$ and $x_i \oplus y_i = 1$ if $x_i \neq y_i$, for all $i \leq n$.

Definition 3.9. Let $a \in \Sigma^*$. $R_a = \{\langle x, y\rangle | x = y \text{ or } x \oplus y = a\}$

Note that R_a is a family of equivalence relations. R_a is a different relation for each fixed $a \in \Sigma^*$.

Notice the following equivalent definitions, where $n = |x| = |y| = |a|$: $R_{eq} = \{\langle x, y\rangle | x \oplus y = 0^n\}$, $R_{eqc} = \{\langle x, y\rangle | x \oplus y = 0^n \text{ or } x \oplus y = 1^n\}$, and $R_a = \{\langle x, y\rangle | x \oplus y = 0^n \text{ or } x \oplus y = a\}$. From these we can see that if $a = 1^n$, then $R_a = R_{eqc}$, and if $a = 0^n$, then $R_a = R_{eq}$.

3.1 Containments and equivalence classes

Theorem 3.10. $R_{eq} \subsetneq R_{bc} \subsetneq R_{par}$

Proof. Let $\langle x, y, \in \rangle R_{eq}$, so $x = y$. Then x has exactly the same number of ones as y (and the same number of zeros, and in the same order), so $\langle x, y \rangle \in R_{bc}$. Therefore, $R_{eq} \subset R_{bc}$.

Consider $x = 1100$ and $y = 0101$. Then $\langle x, y \rangle \in R_{bc}$ but $\langle x, y \rangle \notin R_{eq}$. Therefore $R_{eq} \neq R_{bc}$.

Let $\langle x, y \rangle \in R_{bc}$, so x and y have the same number of ones. Let k be the number of ones in x, and l be the number of ones in y. Then $l = k$, which implies $l \equiv k \pmod 2$. Therefore x and y have the same parity, so $\langle x, y \rangle \in R_{par}$. Therefore $R_{bc} \subset R_{par}$.

Consider $x = 1000$ and $y = 1011$. Then $\langle x, y \rangle \in R_{par}$ but $\langle x, y \rangle \notin R_{bc}$. Therefore $R_{bc} \neq R_{par}$. □

Theorem 3.11. $R_{eq} \subsetneq R_{eqc}$

Proof. Let $\langle x, y \rangle \in R_{eq}$, so $x = y$. Then $\langle x, y \rangle$ satisfies the property specified in the definition of R_{eqc}, specifically that $x = y$, so $\langle x, y \rangle \in R_{eqc}$. Therefore $R_{eq} \subset R_{eqc}$.

Consider $x = 1000$ and $y = 0111$. Then $\langle x, y \rangle \in R_{eqc}$ but $\langle x, y \rangle \notin R_{eq}$. Therefore $R_{eq} \neq R_{eqc}$. □

Since membership in R_{par}, R_{bc}, R_{eq}, R_{eqc}, and R_a can be decided in polynomial time, they are all members of PEq. Now we can examine kernel reductions between these members of PEq. Notice that in each of the reductions in the next section, the more general equivalence relation reduces to the more restrictive equivalence relation.

We will here point out that R_{par} is the most general of the above equivalence relations, and it has only two equivalence classes: [0] and [1]. The only equivalence relation which is more general is the trivial equivalence relation in which all elements of Σ^* relate to all other elements; Σ^* itself is the only equivalence class of this equivalence relation. The most restrictive equivalence relation then is the equality relation, R_{eq}, in which each binary string is the sole element in its equivalence class. The other equivalence relations defined here also have an infinite number of equivalence classes.

3.2 Reductions

We wish to provide some intuition that problems in PEq under polynomial time kernel reductions are restricted *not* by the reductions themselves, but by the number of equivalence classes in the languages. This point becomes somewhat clearer if we consider a kernel reduction from an equivalence relation with exactly two equivalence classes (like R_{par}) to an equivalence relation with at least two equivalence classes (like R_{eq}). Since we are allowed polynomial time for the kernel reduction, we can simply *decide* whether a given word is in one equivalence class or the other, then map it to a corresponding representative of an equivalence class in the language to which we are reducing, depending in which of the two equivalence classes the given word is a member.

Theorem 3.12. *Let $R, S \in$ PEq. Suppose R has exactly two equivalence classes, $[a]$ and $[b]$, and S has at least two equivalence classes, including $[j]$ and $[k]$. Given a, b, j, and k, $R \leq^p_{ker} S$.*

Proof. Since $R \in$ PEq, there exists a Turing machine, M_R, running in deterministic polynomial time such that $\forall x, y \in \Sigma^*, \langle x, y \rangle \in R \iff M_R(\langle x, y \rangle)$ accepts.

Construct machine $M \in$ FP on input $w \in \Sigma^*$:

```
1 if M_R(⟨w, a⟩) accepts then
2     return j
3 else
4     return k
```

Suppose $\langle x, y \rangle \in R$, and without loss of generality, $x, y \in [a]$. Then $M(x) = j$ and $M(y) = j$, so $\langle M(x), M(y) \rangle = \langle j, j \rangle \in S$.

Suppose $\langle x, y\rangle \notin R$, and without loss of generality, $x \in [a]$ and $y \in [b]$. Then $M(x) = j$ and $M(y) = k$. Since equivalence classes form a partition, their intersection is empty, so $(j, k) \notin S$. Thus $\langle M(x), M(y)\rangle \notin S$.

Therefore $\langle x, y\rangle \in R \iff \langle M(x), M(y)\rangle \in S$, and hence $R \leq^p_{ker} S$. □

Corollary 3.13. *Let $S \in \mathsf{PEq}$ be an equivalence relation with at least two equivalence classes, including $[j]$ and $[k]$. Given j and k, $R_{par} \leq^p_{ker} S$.*

Proof. The two equivalence classes of R_{par} are [0] and [1]. Use Theorem 3.12 to construct a kernel reduction from R_{par} to S. □

Corollary 3.14. $R_{par} \leq^p_{ker} R_{eq}$

Proof. Two equivalence classes of R_{eq} are [0] and [1]. Use Corollary 3.13 to construct a kernel reduction from R_{par} to R_{eq}. □

The kernel reductions we provide below from R_{par} to R_{bc} and then to R_{eq} somewhat obfuscate this "decide then map" process, but that is essentially what is happening when these reductions are composed.

Theorem 3.15. $R_{par} \leq^p_{ker} R_{bc}$

Proof. Construct $M \in \mathsf{FP}$ on input $w \in \Sigma^*$:

```
1 for i = 1 to |w| − 1 do
2     if w_i = 1 then
3         for j = i + 1 to |w| do
4             if w_j = 1 then
5                 write 0 to both w_i and w_j
6                 break
```

Notice that this is the machine which finds pairs of ones and writes zeros in their place, one pair at a time.

Suppose $\langle x, y\rangle \in R_{par}$, so either x and y both have even parity or x and y both have odd parity.

If x and y both have even parity, x contains $2k$ ones and y contains $2l$ ones, for some $k, l \in \mathbb{N}$. $M(x)$ and $M(y)$ both output the string $0^{|x|}$, and since both $M(x)$ and $M(y)$ have a bitcount of zero, $\langle M(x), M(y)\rangle \in R_{bc}$.

If x and y both have odd parity, x contains $2k + 1$ ones and y contains $2l + 1$ ones, for some $k, l \in \mathbb{N}$. $M(x)$ and $M(y)$ both output a string containing a single one, so both $M(x)$ and $M(y)$ have a bitcount of one, $\langle M(x), M(y)\rangle \in R_{bc}$.

Suppose $\langle x, y\rangle \notin R_{par}$, so without loss of generality, x has even parity and y has odd parity. Then x contains $2k$ ones and y contains $2l + 1$ ones, for some $k, l \in \mathbb{N}$. Thus $M(x)$ outputs the string $0^{|x|}$ and $M(y)$ outputs the string containing a single one. Since the bitcount of $M(x)$ is zero and the bitcount of $M(y)$ is one, $\langle M(x), M(y)\rangle \notin R_{bc}$.

Therefore $\langle x, y\rangle \in R_{par} \iff \langle M(x), M(y)\rangle \in R_{bc}$, so $R_{par} \leq^p_{ker} R_{bc}$. □

Theorem 3.16. $R_{bc} \leq^p_{ker} R_{eq}$

Proof. Construct $M \in \mathsf{FP}$ on input $w \in \Sigma^*$, where $w = w_1 w_2 \cdots w_{|w|}$:

```
1 Sort the bits of w
2 i ← 1
3 while w_i = 0 do
4     Write a blank at index i
5     i ← i + 1
```

Notice that if w contains k ones, this machine outputs the string 1^k.

Suppose $\langle x, y\rangle \in R_{bc}$, so x and y have the same number of ones, say k. Thus $M(x) = M(y) = 1^k$, so $\langle M(x), M(y)\rangle \in R_{eq}$.

Suppose $\langle x, y\rangle \notin R_{bc}$, so x and y have a different number of ones. Suppose x has k ones and y has l ones, for some $k, l \in \mathbb{N}$, with $k \neq l$. Then $M(x) = 1^k$ and $M(y) = 1^l$, so $M(x) \neq M(y)$. Thus $\langle M(x), M(y)\rangle \notin R_{eq}$.

Therefore $\langle x, y\rangle \in R_{bc} \iff \langle M(x), M(y)\rangle \in R_{eq}$, so $R_{bc} \leq^p_{ker} R_{eq}$. □

The equivalence relations R_{eqc}, which is a generalization of R_{eq}, and R_a, which is a generalization of R_{eqc}, also reduce to equality.

Theorem 3.17. *$R_{eqc} \leq^p_{ker} R_{eq}$*

Proof. Construct machine $M \in \mathsf{FP}$ on input $w \in \Sigma^*$, where $w = w_1 w_2 \cdots w_{|w|}$:

```
1 if w_1 = 0 then
2     return w
3 else
4     return w̄
```

Suppose $\langle x, y\rangle \in R_{eqc}$, so either $x = y$ or $x = \bar{y}$. In the case that $x = y$, $M(x)$ and $M(y)$ produce the same output. In the case that $x = \bar{y}$, then either $x_1 = 1$ and $y_1 = 0$ or $x_1 = 0$ and $y_1 = 1$. Consider without loss of generality the case that $x_1 = 1$ and $y_1 = 0$. Then $M(x)$ outputs $\bar{x}$ and $M(y)$ outputs y. Now $\bar{x} = \bar{\bar{y}} = y$, so $M(x) = M(y)$. Therefore $\langle M(x), M(y)\rangle \in R_{eq}$.

Suppose $\langle x, y\rangle \notin R_{eqc}$, so $x \neq y$ and $x \neq \bar{y}$.

In the case that $x_1 = 0$ and $y_1 = 0$, then $M(x) = x$ and $M(y) = y$. Since $x \neq y$, then $M(x) \neq M(y)$, so $\langle M(x), M(y)\rangle \notin R_{eq}$.

In the case that $x_1 = 0$ and $y_1 = 1$, then $M(x) = x$ and $M(y) = \bar{y}$. Since $x \neq \bar{y}$, then $M(x) \neq M(y)$, so $\langle M(x), M(y)\rangle \notin R_{eq}$.

In the case that $x_1 = 1$ and $y_1 = 0$, then $M(x) = \bar{x}$ and $M(y) = y$. Since $x \neq \bar{y}$, then $\bar{x} \neq \bar{\bar{y}} = y$, so $M(x) \neq M(y)$, and $\langle M(x), M(y)\rangle \notin R_{eq}$.

In the case that $x_1 = 1$ and $y_1 = 1$, then $M(x) = \bar{x}$ and $M(y) = \bar{y}$. Since $x \neq y$, then $\bar{x} \neq \bar{y}$, so $M(x) \neq M(y)$, and $\langle M(x), M(y)\rangle \notin R_{eq}$.

Therefore $\langle x, y\rangle \in R_{eqc} \iff \langle M(x), M(y)\rangle \in R_{eq}$, so $R_{eqc} \leq^p_{ker} R_{eq}$. □

Theorem 3.18. *Let $a \in \Sigma^*$. Then $R_a \leq^p_{ker} R_{eq}$.*

Proof. Construct machine $M \in \mathsf{FP}$ on input $w \in \Sigma^*$, where $|w| = |a| = n$, $w = w_1 w_2 \cdots w_n$ and $a = a_1 a_2 \cdots a_n$:

```
1 s ← ((a_1, w_1), (a_2, w_2), ..., (a_n, w_n))
2 Perform a stable sort on s with each a_i as the key, to obtain two subsequences, r and r′, where
  r = (a_i, w_i)|_{a_i=0} and r′ = (a_j, w_j)|_{a_j=0}
3 Run the algorithm in Theorem 3.17 on the concatenation of each w_j in r′ to obtain a new string c
4 return the concatenation of each w_i, in the order of r, with the string c
```

Notice that informally this machine splits up the problem into two easier problems whose solutions are known: the first problem is the problem of determining equality, the second problem is the problem of determining either equality or bitwise complement.

Suppose $\langle x, y\rangle \in R_a$, so either $x = y$ or $x \oplus y = a$. In the case that $x = y$, then $M(x)$ and $M(y)$ output the same string, so $M(x) = M(y)$, and hence $\langle M(x), M(y)\rangle \in R_{eq}$. Now consider the case in which $x \oplus y = a$, which implies $x \oplus a = y$ and $y \oplus a = x$. Then $M(x)$ and $M(y)$ will first rearrange the bits of x and y, respectively, by stable sorting the bits of a. For all bits of a which are zero, the corresponding bits of x and y are equal, by hypothesis. For all bits of a which are one, the corresponding bits of x and y are complements of one another. Let a_i be the leftmost bit of a which is one. In the case that $x_i = 1$ and $y_i = 0$, then the algorithm from Theorem 3.17 will flip all the bits of x corresponding to bits of a which are one. Since y and x were complements at these bits only, now $M(x) = M(y)$, so $\langle M(x), M(y)\rangle \in R_{eq}$. The argument for the case that $x_i = 0$ and $y_i = 1$ is symmetric.

Suppose $\langle x, y\rangle \notin R_a$, so $x \neq y$ and $x \oplus y \neq a$. So $\exists i, j \in \{1, 2, \ldots, |x|\} : x_i \neq y_i$ and $x_j \oplus y_j \neq a_j$. Now $M(x)$ and $M(y)$ will first rearrange the bits of x and y, respectively, by stable sorting the bits of a in step 2. Call these intermediate strings x', y' and a', and the corresponding permutation of indices i and j to be i' and j', respectively. Note that the value of $a'_{j'}$ completely determines whether $\langle M(x), M(y)\rangle \in R_{eq}$.

In the case that $a'_{j'} = 0$ then $x'_{j'} \oplus y'_{j'} \neq a'_{j'} \implies x'_{j'} \oplus y'_{j'} \neq 0 \implies x'_{j'} \neq y'_{j'}$ and hence $\langle M(x), M(y)\rangle \notin R_{eq}$.

Consider the case in which $a'_{j'} = 1$. Let k be the index of the leftmost one bit in a, and the index of its image after the stable sorting in step 2 k'. In the case that $x'_{k'} = 1$ then $y'_{k'} = x'_{k'} \oplus a'_{k'} = 1 \oplus 1 = 0$. Since $x'_{k'} = 1$, step 3 in the above algorithm will invert each of the bits of x' from index k' to the end of that string. Since $y'_{k'} = 0$, step 3 will leave y' unchanged. Since $x'_{j'} \oplus y'_{j'} \neq a'_{j'}$ by assumption, then $x'_{j'} \oplus y'_{j'} \neq 1 \implies x'_{j'} \oplus y'_{j'} = 0 \implies x'_{j'} = y'_{j'} \implies \overline{x'_{j'}} \neq y'_{j'}$. Hence $M(x) \neq M(y)$, so $\langle M(x), M(y)\rangle \notin R_{eq}$. The argument for the case that $x'_{k'} = 0$ is symmetric.

Therefore $\langle x, y\rangle \in R_a \iff \langle M(x), M(y)\rangle \in R_{eq}$, so $R_a \leq^p_{ker} R_{eq}$. □

It appears that all of these "bitwise" equivalence problems reduce under polynomial time kernel reductions to the equality relation, R_{eq}, the most restrictive equivalence relation in PEq. However, there is evidence[FG09] that the equality relation is not PEq-complete. We restate this evidence in the next section.

We now present theorems formalizing the intuition we have gained about reductions among equivalence relations with different numbers of equivalence classes.

Theorem 3.19. *Let R and S be equivalence relations on Σ^*. Suppose R has n equivalence classes and S has m equivalence classes. If $n > m$ then $R \not\leq_{ker} S$ (that is, R does not kernel reduce to S, regardless of any time bound on the function computing the reduction).*

Proof. Assume with the intention of producing a contradiction that $R \leq_{ker} S$. Then $\exists f : \Sigma^* \to \Sigma^* : \forall x, y \in \Sigma^*, \langle x, y\rangle \in R \iff \langle f(x), f(y)\rangle \in S$.

Since R has n equivalence classes, each equivalence class is non-empty, and the equivalence classes partition R, then $\exists r_1, \ldots, r_n \in \Sigma^* : R = [r_1]_R \cup \cdots \cup [r_n]_R$. Since each element of R is in exactly one equivalence class, $\forall i, j \leq n$, $i = j \iff \langle r_i, r_j\rangle \in R \iff \langle f(r_i), f(r_j)\rangle \in S$. Therefore the image of each r_i is in some equivalence class in S. Also, $\forall i, j \leq n$, $i \neq j \iff \langle r_i, r_j\rangle \notin R \iff \langle f(r_i), f(r_j)\rangle \notin S$. Therefore, the image of each r_i does not relate to the image of any other r_j, for $i \neq j$, and $i, j \leq n$. Therefore each of the equivalence classes $[f(r_1)]_S, \ldots, [f(r_n)]_S$ is disjoint, so S has at least n equivalence classes. But $n > m$. This is a contradiction with the hypothesis that S has m equivalence classes.

Therefore $R \not\leq^p_{ker} S$. □

Corollary 3.20. $R_{eq} \not\leq_{ker} R_{par}$

Proof. R_{eq} has a (countably) infinite number of equivalence classes (one for each binary string). R_{par} has two equivalence classes. Therefore these equivalence classes meet the conditions of Theorem 3.19. □

Our intuition now is that the problem of determining whether one equivalence relation kernel reduces to another reduces to the problem of determining representatives of equivalence classes in both equivalence relations.

Theorem 3.21. *Let R, S be equivalence relations. Suppose $R \in$ PEq. Suppose R has a finite number of equivalence classes, n, and the number of equivalence classes of S is greater than or equal to n (possibly countably infinite). Let $REP(R)$ be a set of representatives of equivalence classes in R, and $REP(S)$ be a set of representatives of equivalence classes in S. If $\exists f : REP(R) \to REP(S)$ such that $f \in$ FP and f is injective, then $R \leq^p_{ker} S$.*

Proof. Since $R \in$ PEq, $\exists M_R$, a deterministic Turing machine running in polynomial time, such that $\langle x, y\rangle \in R \iff M_R(\langle x, y\rangle)$ accepts. Construct $M \in$ FP on input $w \in \Sigma^*$:

```
1 for r_i ∈ REP(R) do
2     if M_R(⟨w, r_i⟩) accepts then
3         return f(r_i)
```

Notice that each $w \in \Sigma^*$ is in exactly one equivalence class in R, because the equivalence classes partition Σ^*, so $M_R(\langle w, r_i\rangle)$ accepts exactly once when $\langle w, r_i\rangle \in R$.

Suppose $\langle x, y\rangle \in R$, so $\langle x, r_i\rangle \in R$ and $\langle y, r_i\rangle \in R$ for some $r_i \in REP(R)$. Then $M(x)$ outputs $f(r_i)$ and $M(y)$ outputs $f(r_i)$. Since $\langle f(r_i), f(r_i)\rangle \in S$, then $\langle M(x), M(y)\rangle \in S$.

Suppose $\langle x, y\rangle \notin R$. So $\exists r_i, r_j \in REP(R)$, with $r_i \neq r_j$, such that $\langle x, r_i\rangle \in R$ and $\langle y, r_j\rangle \in R$. Then $M(x)$ outputs $f(r_i)$ and $M(y)$ outputs $f(r_j)$. Since f is injective, $r_i \neq r_j \implies f(r_i) \neq f(r_j)$. Since $f(r_i)$ and $f(r_j)$ are distinct elements in $REP(S)$, they are representatives of two distinct equivalence classes. Since every element of S is in exactly one equivalence class, and since $[f(r_i)] \neq [f(r_j)]$, it follows that $\langle f(r_i), f(r_j)\rangle = \langle M(x), M(y)\rangle \notin S$.

Therefore $\langle x, y\rangle \in R \iff \langle M(x), M(y)\rangle \in S$, so $R \leq^p_{ker} S$. □

Theorem 3.22. *Let R, S be equivalence relations. Suppose $R \in \mathsf{PEq}$. Suppose R has a finite number of equivalence classes, n, and the number of equivalence classes of S is greater than or equal to n (possibly countably infinite). If $\exists E_R, E_S \in \mathsf{FP}$ and $\exists i, j \in \Sigma^*$ such that E_R on input i outputs the encoding of $REP(R)$ and E_S on input j outputs the encoding of at least n elements of $REP(S)$, then $R \leq^p_{ker} S$.*

Proof. We compute the kernel reduction using the following procedure.

Run E_R on input i to produce the encoding of $REP(R)$. The output of this machine is $\langle r_1, r_2, \ldots, r_n\rangle$, where $r_1, \ldots, r_n$ are representatives of the equivalence classes in R.

Run E_S on input j to produce the encoding of at least n elements of $REP(S)$. The length of the output of E_S on input j is bounded by some polynomial in the length of j, say $p(|j|)$. Since $p(|j|)$ is finite, then the number of elements of $REP(S)$ which E_S outputs when run on input j must be finite. Call this number m. By hypothesis $m \geq n$. So the output of this function is $\langle s_1, s_2, \ldots, s_m\rangle$, where $s_1, \ldots, s_m$ are representatives of m equivalence classes in S.

Define $f \in \mathsf{FP}$ by $f(r_\ell) = s_\ell$, for all $\ell \leq n$. We now wish to show that f is injective. Suppose $r_\ell, r_q \in R$ and $r_\ell \neq r_q$. Then $f(r_\ell) = s_\ell$ and $f(r_q) = s_q$. Since each element in the encoding of m elements of $REP(S)$ which is output by E_S on input j is unique (as each representative is in its own equivalence class), $s_\ell \neq s_q$, and hence $f(r_\ell) \neq f(r_q)$. Therefore f is injective.

The function f now satisfies the conditions in the hypothesis of Theorem 3.21, so the result follows. □

3.3 Complete invariants

Definition 3.23. Let R be an equivalence relation on A, and let $f\colon A \to A$. Then f is a *complete invariant* for R if $(x, y) \in R$ if and only if $f(x) = f(y)$.

Definition 3.24. $\mathsf{Ker}(\mathsf{FP}) = \{L | L \text{ is an equivalence relation on } \Sigma^* \text{ with a complete invariant in } \mathsf{FP}\}$

Lemma 3.25. $\mathsf{Ker}(\mathsf{FP}) \subseteq \mathsf{PEq}$

Proof. Let $R \in \mathsf{Ker}(\mathsf{FP})$. Then by definition $\exists f \in \mathsf{FP} : \langle x, y\rangle \in R \iff f(x) = f(y)$. To decide membership of $\langle x, y\rangle \in \mathsf{R}$, use f to decide whether $f(x) = f(y)$. This is true if and only if $\langle x, y\rangle \in R$. Therefore membership in R can be decided in polynomial time, hence $\mathsf{Ker}(\mathsf{FP}) \subseteq \mathsf{PEq}$. □

For the sake of clarity, we provide a proof of the following claim made by Fortnow and Grochow:

Theorem 3.26. $\mathsf{Ker}(\mathsf{FP}) = \mathsf{PEq}$ *if and only if R_{eq} is* PEq*-complete.*

Proof. For the forward direction, suppose $\mathsf{Ker}(\mathsf{FP}) = \mathsf{PEq}$. Let $S \in \mathsf{PEq} = \mathsf{Ker}(\mathsf{FP})$, so $\exists f \in \mathsf{FP} : \langle x, y\rangle \in S \iff f(x) = f(y) \iff \langle f(x), f(y)\rangle \in R_{eq} \iff S \leq^p_{ker} R_{eq}$. Thus R_{eq} is PEq-complete.

Conversely, suppose R_{eq} is PEq-complete. Then $\forall S \in \mathsf{PEq}$, $\exists f \in \mathsf{FP} : \langle x, y\rangle \in S \iff \langle f(x), f(y)\rangle \in R_{eq} \iff f(x) = f(y)$. Thus f is a complete invariant for S computable in polynomial time. Thus $S \in \mathsf{Ker}(\mathsf{FP})$, and hence $\mathsf{PEq} \subseteq \mathsf{Ker}(\mathsf{FP})$. By Lemma 3.25, $\mathsf{Ker}(\mathsf{FP}) \subseteq \mathsf{PEq}$, therefore $\mathsf{PEq} = \mathsf{Ker}(\mathsf{FP})$. □

Fortnow and Grochow provide evidence that R_{eq} is *not* PEq-complete, by showing that if Ker(FP) = PEq then UP $\subseteq$ BQP [FG09]. To separate PEq from Ker(FP), we need to exhibit an equivalence problem for which membership can be decided in polynomial time, but no polynomial time algorithm exists for computing a complete invariant, or equivalently, no reduction to the equality relation exists. Two such candidates identified by Fortnow and Grochow are the Boolean function congruence problem and the subgroup equivalence problem.

3.4 Completeness

It would appear that proving some language PEq-complete should be easier than proving, for example, P-completeness directly, because the kernel reduction allows us a polynomial time computable function, even in the set of equivalence problems *decidable* in polynomial time, whereas proofs for P-completeness under many-one reductions allow only logarithmic space. This means we are allowed to examine the entire computation history of the machine which decides any language in PEq (for example, as the tableau of the deterministic polynomial time machine). And from our examination of reductions among feasible equivalence problems, we can see that some reductions seem to solve the problem to be reduced first. This provides some intuition that polynomial time kernel reductions for problems in PEq may be too powerful. Again, the real restriction here is on the number of equivalence classes in each language.

4 Kernel reductions among intractable problems

4.1 The graph isomorphism problem

An equivalence problem of particular importance is the graph isomorphism problem. Although it is in NP, it is not known to be NP-complete and it is not known to be in P. It is therefore a candidate for NP\P (if P $\neq$ NP). Since it is an equivalence problem in NP, it is a member of NPEq, so it may be of particular value to study kernel reductions to and from the graph isomorphism problem.

Definition 4.1. Let V be a finite set, let E be a collection of subsets of V of size exactly 2. Then $G = (V, E)$ is called an *undirected graph*. Elements of V are called *vertices*, and elements of E are called *edges*.

If instead $E \subseteq V \times V$, then $G = (V, E)$ is a *directed graph*.

Definition 4.2. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be undirected graphs. Then G_1 is *isomorphic to* G_2 if $\exists \phi \colon V_1 \to V_2 : \{u, v\} \in E_1 \iff \{\phi(u), \phi(v)\} \in E_2$ and ϕ is a bijection.

If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are directed graphs, then G_1 is *isomorphic to* G_2 if $\exists \phi \colon V_1 \to V_2 : (u, v) \in E_1 \iff (\phi(u), \phi(v)) \in E_2$ and ϕ is a bijection.

Intuitively, the bijection ϕ is a relabeling of vertices of G_1 which preserves adjacency in G_2.

Definition 4.3. Let $GI = \{\langle G_1, G_2 \rangle | G_1$ and G_2 are undirected graphs and G_1 is isomorphic to $G_2\}$. Deciding membership in GI is the *graph isomorphism problem*.

Let $DirGI = \{\langle G_1, G_2 \rangle | G_1$ and G_2 are directed graphs and G_1 is isomorphic to $G_2\}$. Deciding membership in $DirGI$ is the *directed graph isomorphism problem*.

4.2 Kernel reductions to graph isomorphism

We begin by examining kernel reductions from the PEq equivalence relations defined above to the graph isomorphism problem. The reductions should intuitively be easy because the graph isomorphism problem is (seemingly) of greater complexity. We examine these upward reductions in order to gain some intuition that a kernel reduction makes sense, practically.

Theorem 4.4. $R_{par} \leq^p_{ker} GI$

Proof. Construct $M \in \mathsf{FP}$ on input $w \in \Sigma^*$:

```
1 V_w ← {v_p, v_o}
2 E_w ← {}
3 for i = 1 to |w| do
4     if w_i = 1 then
5         if {v_o, v_p} ∈ E_w then
6             add {v_o, v_p} to E_w
7         else
8             remove {v_o, v_p} from E_w
9 return G_w = (V_w, E_w)
```

Suppose $\langle x, y\rangle \in R_{par}$, so either x and y both have even parity or x and y both have odd parity.

If x and y both have even parity, x contains $2k$ ones and y contains $2l$ ones, for some $k, l \in \mathbb{N}$. Since $2k$ is even, machine M on input x adds then removes the edge $\{v_o, v_p\}$ to and from E_x an equal number of times. Similarly for M on input y. Therefore $M(x)$ outputs $G_x = (V_x, E_x)$, where $V_x = \{v_o, v_p\}$ and $E_x = \{\}$, and $M(y)$ outputs $G_y = (V_y, E_y)$, where $V_y = \{v_o, v_p\}$ and $E_y = \{\}$. Then G_x is isomorphic to G_y by the identity function, $I \colon V_x \to V_y$, defined by $I(v) = v, \forall v \in V_x$.

If x and y both have odd parity, x contains $2k + 1$ ones and y contains $2l + 1$ ones, for some $k, l \in \mathbb{N}$. Since $2k + 1$ is odd, machine M on input x adds edge $\{v_o, v_p\}$ to E_x one more time than it removes the edge. Similarly for M on input y. Therefore $M(x)$ outputs $G_x = (V_x, E_x)$, where $V_x = \{v_o, v_p\}$ and $E_x = \{\{v_o, v_p\}\}$, and $M(y)$ outputs $G_y = (V_y, E_y)$, where $V_y = \{v_o, v_p\}$ and $E_y = \{\{v_o, v_p\}\}$. Then G_x is isomorphic to G_y by the identity function, $I \colon V_x \to V_y$, defined by $I(v) = v, \forall v \in V_x$.

Suppose $\langle x, y\rangle \notin R_{par}$, so without loss of generality, x has even parity and y has odd parity. Then x contains $2k$ ones and y contains $2l + 1$ ones, for some $k, l \in \mathbb{N}$. Since $2k$ is even, machine M on input x adds then removes the edge $\{v_o, v_p\}$ to and from E_x an equal number of times. Since $2l + 1$ is odd, machine M on input y adds edge $\{v_o, v_p\}$ to E_y one more time than it removes the edge. Therefore $M(x)$ outputs $G_x = (V_x, E_x)$, where $V_x = \{v_o, v_p\}$ and $E_x = \{\}$, and $M(y)$ outputs $G_y = (V_y, E_y)$, where $V_y = \{v_o, v_p\}$ and $E_y = \{\{v_o, v_p\}\}$. Since $\{v_o, v_p\} \in E_y$ but $\{v_o, v_p\} \notin E_x$, so no bijection exists between V_x and V_y which preserves edges. Therefore, G_x is not isomorphic to G_y so $\langle M(x), M(y)\rangle \notin GI$.

Therefore $\langle x, y\rangle \in R_{par} \iff \langle M(x), M(y)\rangle \in GI$, so $R_{par} \leq^p_{ker} GI$. □

Theorem 4.5. $R_{bc} \leq^p_{ker} GI$

Proof. Construct $M \in \mathsf{FP}$ on input $w \in \Sigma^*$:

```
1 V_w ← {v_1, v_2, ..., v_|w|, v_zero, v_one,0, v_one,1, v_one,2}
2 E_w ← {{v_one,0, v_one,1}, {v_one,1, v_one,2}, {v_one,2, v_one,0}}
3 for i = 1 to |w| do
4     if w_i = 1 then
5         add {v_i, v_one,0} to E_w
6     else
7         add {v_i, v_zero} to E_w
8 return G_w = (V_w, E_w)
```

Suppose $\langle x, y\rangle \in R_{bc}$, so x and y have the same number of ones, say $k \in \mathbb{N}$. Assume $|x| = |y| = n$, so both x and y have $n - k$ zeros. Define $E_{w,1} = \{\{v_i, v_{one,0}\} | i \in \{1, 2, \ldots, n\}, w_i = 1\}$ and $E_{w,0} = \{\{v_i, v_{zero}\} | i \in \{1, 2, \ldots, n\}, w_i = 0\}$, so $E_x = E_{x,1} \cup E_{x,0}$ and $E_y = E_{y,1} \cup E_{y,0}$ by construction. Define $V_{w,b} = \{v_i | w_i = b\}$, so $V_x = V_{x,1} \cup V_{x,0}$ and $V_y = V_{y,1} \cup V_{y,0}$. Note that $|V_{x,1}| = |V_{y,1}| = k$ and $|V_{x,0}| = |V_{y,0}| = n - k$. Since $|V_{x,1}| = |V_{y,1}| = k$, there exists a bijection between them, call it $\phi_1 \colon V_{x,1} \to V_{y,1}$. Similarly, since

$|V_{x,0}| = |V_{y,0}| = k$, there exists a bijection between them, call it $\phi_0 \colon V_{x,0} \to V_{y,0}$. Define $\phi \colon V_x \to V_y$ by

$$\phi(v) = \begin{cases} \phi_0(v) & \text{if } v = v_i \text{ and } x_i = 1, \text{ for some } i \in \{1, \ldots, n\} \\ \phi_1(v) & \text{if } v = v_i \text{ and } x_i = 0, \text{ for some } i \in \{1, \ldots, n\} \\ v & \text{if } v \in \{v_{zero}, v_{one,0}, v_{one,1}, v_{one,2}\} \end{cases}$$

for all $v \in V_x$. Notice that each v_i "corresponds" to a single x_i, because each x_i can be either a one or a zero, exclusively.

Since the only edges in E_x are the edges $\{v_i, v_{one,0}\}$ when $x_i = 1$ and $\{v_i, v_{zero}\}$ when $x_i = 0$, then

$$(v_i, v_{one,0}) \in E_x \iff (\phi(v_i), \phi(v_{one,0})) = (\phi_1(v_i), v_{one,0}) \in E_y$$

and

$$(v_i, v_{zero}) \in E_x \iff (\phi(v_i), \phi(v_{zero})) = (\phi_0(v_i), v_{zero}) \in E_y$$

Therefore ϕ describes a graph isomorphism, so G_1 is isomorphic to G_2.

Suppose $\langle x, y \rangle \notin R_{bc}$, so x and y have a different number of ones. Let k be the number of ones in x, and l be the number of ones in y, with $k \neq l$. Suppose without loss of generality that $k > l$. Define $E_{w,0}$ and $E_{w,1}$ as above. Now $|E_{x,1}| = k$ and $|E_{y,1}| = l$. Since $k > l$, $E_{x,1}$ has at least one more edge adjacent to the triangle created by the vertices $\{v_{one,0}, v_{one,1}, v_{one,2}\}$ than does $E_{y,1}$. Thus no possible bijection exists between V_x and V_y which preserves all edges. Thus G_x is not isomorphic to G_y, so $\langle M(x), M(y) \rangle \notin GI$.

Therefore $\langle x, y \rangle \in R_{bc} \iff \langle M(x), M(y) \rangle \in GI$, so $R_{bc} \leq^p_{ker} GI$. □

Unlike the reductions from R_{par} and R_{bc} to GI, in the reductions from R_{eq}, R_{eqc}, and R_a to GI the order of bits in each string x and y is significant. As such, it may be easier to create kernel reductions from these equivalence relations to the directed graph isomorphism problem, which we can prove equivalent under kernel reductions to the graph isomorphism problem.

Lemma 4.6 ([KST93] and [Mil77])**.** $GI \equiv^p_{ker} DirGI$

Proof. To show $GI \leq^p_{ker} DirGI$, replace each edge in the undirected graph with a pair of complementary edges in a directed graph with same set of vertices.

To show $DirGI \leq^p_{ker} GI$, replace each edge in the directed graph with a set of vertices and edges that enforces an ordering between the original pair of vertices adjacent in the directed graph. In [Mil77], for example, when given vertices x and y with directed edge (x, y), the author adds the vertices $\{v_1, v_2, \ldots, v_7\}$ with undirected edges $\{\{x, v_1\}, \{v_1, v_4\}, \{v_4, y\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_7\}\}$. □

From Lemma 4.6, we can use a kernel reduction to the directed graph isomorphism problem to show that a language kernel reduces to the undirected graph isomorphism problem, as in the following theorems.

Theorem 4.7. $R_{eq} \leq^p_{ker} DirGI$

Proof. Construct machine $M \in \mathsf{FP}$ on input $w \in \Sigma^*$:

```
1 V_w ← {v_1, v_2, ..., v_|w|}
2 E_w ← {(v_1, v_2), (v_2, v_3), ..., (v_|w|−1, v_|w|)}        // directed edges
3 for i = 1 to |w| do
4     if w_i = 1 then
5         add vertex v'_i to V_w
6         add directed edge (v_i, v'_i) to E_w
7 return G_w = (V_w, E_w)
```

Notice that this machine constructs a "spine" of vertices, with an extra vertex v'_i and directed edge (v_i, v'_i) adjacent to the spine whenever w_i is a one, $\forall i \in \{1, 2, \ldots, |w|\}$.

Suppose $\langle x, y \rangle \in R_{eq}$, so $x = y$. Then $M(x)$ and $M(y)$ produce the same graph, so G_x is isomorphic to G_y by the identity mapping. Therefore $\langle M(x), M(y) \rangle \in DirGI$.

Suppose $\langle x, y\rangle \notin R_{eq}$, so $x \neq y$. Suppose $|x| = |y| = n$. Run M on input x to yield $G_x = (V_x, E_x)$, and run M on input y to yield $G_y = (V_y, E_y)$. Since the graphs are directed, the "spine" created by the vertices $\{v_1, v_2, \ldots, v_n\}$ and the edges $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$ must correspond in both G_x and G_y. Let i be the index of the first bit at which x and y differ. Suppose without loss of generality that $x_i = 1$ and $y_i = 0$. Then $v_i' \in V_x$ and $(v_i, v_i') \in E_x$, but $v_i' \notin V_y$ so $(v_i, v_i') \notin E_y$. Assume with the intention of producing a contradiction that a bijection exists between V_x and V_y which satisfies the conditions for a graph isomorphism. Since vertices along the "spine" of the V_x must map to vertices along the "spine" of V_y, and specifically v_i in V_x must map to v_i in V_y, $(v_i, v_i') \in E_x$ implies $(v_i, v_i') \in E_y$. But $(v_i, v_i') \notin E_y$ because $y_i = 0$. This is a contradiction. Therefore no such mapping exists, so G_x is not isomorphic to G_y, and hence $\langle M(x), M(y)\rangle \notin DirGI$.

Therefore $\langle x, y\rangle \in R_{eq} \iff \langle M(x), M(y)\rangle \in DirGI$, so $R_{eq} \leq^p_{ker} DirGI$. □

Corollary 4.8. $R_{eq} \leq^p_{ker} GI$

Proof. Follows directly from Theorem 4.7 and Lemma 4.6. □

Theorem 4.9. $R_{eqc} \leq^p_{ker} DirGI$

Proof. Construct machine $M \in \mathsf{FP}$ on input $w \in \Sigma^*$:

```
1 V_w ← {v_1, v_2, ..., v_|w|, v_zero, v_one}
2 E_w ← {(v_1, v_2), (v_2, v_3), ..., (v_|w|−1, v_|w|)}        // directed edges
3 for i = 1 to |w| do
4     if w_i = 1 then
5         add directed edge (v_i, v_one) to E_w
6     else
7         add directed edge (v_i, v_zero) to E_w
8 return G_w = (V_w, E_w)
```

Notice that this machine, as in the machine in the proof of Theorem 4.7, creates a "spine" representing each bit of word w.

Suppose $\langle x, y\rangle \in R_{eqc}$, so either $x = y$ or $x = \bar{y}$.

In the case that $x = y$, $M(x)$ and $M(y)$ output exactly the same graph, so G_x is isomorphic to G_y, and hence $\langle M(x), M(y)\rangle \in DirGI$.

In the case that $x = \bar{y}$, define $\phi\colon V_x \to V_y$ by

$$\phi(v) = \begin{cases} v & \text{if } v = v_i, \text{ for some } i \in \{1, 2, \ldots, |x|\} \\ v_{zero} & \text{if } v = v_{one} \\ v_{one} & \text{if } v = v_{zero} \end{cases}$$

for all $v \in V_x$. Notice that ϕ maps each $v_i \in V_x$ to the corresponding $v_i \in V_y$, and maps $v_{zero} \in V_x$ to $v_{one} \in V_y$ and $v_{one} \in V_x$ to $v_{zero} \in V_y$. Then $\forall i \in \{1, 2, \ldots, |x|\}$, $x_i = 0 \iff y = 1$, so $(v_i, v_{zero}) \in E_x \iff (\phi(v_i), \phi(v_{zero})) = (v_i, v_{one}) \in E_y$. Similarly, $x_i = 1 \iff y = 0$, so $(v_i, v_{one}) \in E_x \iff (\phi(v_i), \phi(v_{one})) \in E_y \iff (v_i, v_{zero}) \in E_y$. The rest of the edges in E_x map directly to the corresponding edges in E_y by $(v_{i-1}, v_i) \mapsto (\phi(v_{i-1}), \phi(v_i)) = (v_{i-1}, v_i)$, $\forall i \in \{2, 3, \ldots, |x|\}$. Therefore, ϕ describes an isomorphism between G_x and G_y, so $\langle M(x), M(y)\rangle \in DirGI$.

Suppose $\langle x, y\rangle \notin R_{eqc}$, so $x \neq y$ and $x \neq \bar{y}$. Thus $\exists i, j \in \{1, 2, \ldots, |x|\}, i \neq j$, such that $x_i = y_i \wedge x_j \neq y_j$. Suppose without loss of generality that $x_i = y_i = 0$ and $0 = x_j \neq y_j = 1$. Now $x_i = y_i = 0$ implies $(v_i, v_{zero}) \in E_x$ and $(v_i, v_{zero}) \in E_y$. Also, $x_j = 0$ implies $(v_j, v_{zero}) \in E_x$ and $y_j = 1$ implies $(v_j, v_{one}) \in E_y$. Assume, with the goal of producing a contradiction, that there exists a bijection, $\phi\colon V_x \to V_y$ such that $(u, v) \in E_x \iff (\phi(u), \phi(v)) \in E_y$, $\forall u, v \in V_x$. Since ϕ must map vertices on the "spine" of G_x to corresponding vertices in G_y, and since $(v_i, v_{zero}) \in E_x$, then $(\phi(v_i), \phi(v_{zero})) = (v_i, \phi(v_{zero}))$ must be in E_y. The only edge of this form in E_y is (v_i, v_{zero}) so $\phi(v_{zero}) = v_{zero}$. Since $(v_j, v_{zero}) \in E_x$, $(\phi(v_j), \phi(v_{zero})) =$

$(v_j, v_{zero}) \in E_y$. But the only edge in E_y with source vertex v_j is, by construction, (v_j, v_{one}). This is a contradiction. Hence no such bijection exists, so G_x is not isomorphic to G_y, and $\langle M(x), M(y)\rangle \notin DirGI$.

Therefore $\langle x, y\rangle \in R_{eqc} \iff \langle M(x), M(y)\rangle \in DirGI$, so $R_{eqc} \leq^p_{ker} DirGI$. □

Corollary 4.10. $R_{eqc} \leq^p_{ker} GI$

Proof. Follows directly from Theorem 4.9 and Lemma 4.6. □

Theorem 4.11. $\forall a \in \Sigma^*$, $R_a \leq^p_{ker} DirGI$

Proof. Let $a \in \Sigma^*$. Construct machine $M \in \mathsf{FP}$ on input $w \in \Sigma^*$ (let $n = |w|$):

```
1  U_w ← {v_1, v_2, ..., v_n}
2  U'_w ← {v'_1, v'_2, ..., v'_n}
3  F_w ← {(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n)}        // directed edges
4  F'_w ← {(v'_1, v'_2), (v'_2, v'_3), ..., (v'_{n-1}, v'_n)}  // directed edges
5  for i = 1 to n do
6      if w_i = 1 then
7          add vertex u_i to U_w
8          add directed edge (v_i, u_i) to F_w
9      if w_i ⊕ a_i = 1 then
10         add vertex u'_i to U'_w
11         add directed edge (v'_i, u'_i) to F'_w
12 V_w ← U_w ∪ U'_w
13 E_w ← F_w ∪ F'_w
14 return G_w = (V_w, E_w)
```

Notice that for each input string w, this machine produces a graph which is the disjoint union of two distinct graphs, one representing the bits of w and the other representing the bits of $w \oplus a$.

Suppose $\langle x, y\rangle \in R_a$, so either $x = y$ or $x \oplus y = a$.

In the case that $x = y$, $M(x)$ and $M(y)$ output exactly the same graph, so G_x is isomorphic to G_y, and hence $\langle M(x), M(y)\rangle \in DirGI$.

Consider the case in which $x \neq y$, but $x \oplus y = a$. Define $\phi\colon V_x \to V_y$ by

$$\phi(v) = \begin{cases} v'_i & \text{if } v = v_i \text{ for some } i \in \{1, 2, \ldots, n\} \\ v_i & \text{if } v = v'_i \text{ for some } i \in \{1, 2, \ldots, n\} \\ u'_i & \text{if } v = u_i \text{ for some } i \in \{1, 2, \ldots, n\} \\ u_i & \text{if } v = u'_i \text{ for some } i \in \{1, 2, \ldots, n\} \end{cases}$$

If no u_i exists or no u'_i exists for some $i \in \{1, 2, \ldots, n\}$ then we don't define ϕ for those values not in the domain. ϕ maps each $v_i \in V_x$ to $v'_i \in V_y$, so $(v_i, v_{i+1}) \in E_x \iff (\phi(v_i), \phi(v_{i+1})) = (v'_i, v'_{i+1}) \in E_y, \forall i \in \{1, 2, \ldots, n-1\}$, and ϕ maps each $v'_i \in V_x$ to $v_i \in V_y$, so $(v'_i, v'_{i+1}) \in E_x \iff (\phi(v'_i), \phi(v'_{i+1})) = (v_i, v_{i+1}) \in E_y, \forall i \in \{1, 2, \ldots, n-1\}$.

Now for each $i \in \{1, 2, \ldots, n\}$ for which u_i exists and is in V_x, $(v_i, u_i) \in E_x$ by construction. This occurs if and only if $x_i = 1$, and since by hypothesis $x \oplus y = a \iff y = x \oplus a$, $y_i = x_i \oplus a_i$. In the case that $a_i = 0$, then $y_i = x_i \oplus a_i = 1 \oplus 0 = 1$. Since $y_i = 1$ and $y_i \oplus a_i = 1 \oplus 0 = 1$, $M(y)$ produces graph G_y containing vertex $u'_i \in V_y$ and directed edge $(v'_i, u'_i) \in E_y$. Now $\phi(u_i) = u'_i$ is well-defined, and $(v_i, u_i) \in E_x \iff (\phi(v_i), \phi(u_i)) = (v'_i, u'_i) \in E_y$. In the case that $a_i = 1$, then $y_i = x_i \oplus a_i = 1 \oplus 1 = 0$. Since $y_i = 0$ and $y_i \oplus a_i = 0 \oplus 1 = 1$, $M(y)$ produces graph G_y containing vertex $u'_i \in V_y$ and directed edge $(v'_i, u'_i) \in E_y$. Now $\phi(u_i) = u'_i$ is well-defined, and $(v_i, u_i) \in E_x \iff (\phi(v_i), \phi(u_i)) = (v'_i, u'_i) \in E_y$.

Now for each $i \in \{1, 2, \ldots, n\}$ for which u'_i exists and is in V_x, $(v'_i, u'_i) \in E_x$ by construction. This occurs if and only if $x_i \oplus a_i = 1$, and since by hypothesis $x \oplus y = a \iff y = x \oplus a$, then $y_i = x_i \oplus a_i = 1$. Since $y_i = 1$, $M(y)$ produces graph G_y containing vertex $u_i \in V_y$ and directed edge $(v_i, u_i) \in E_y$. Now $\phi(u'_i) = u_i$

is well-defined, and $(v'_i, u'_i) \in E_x \iff (\phi(v'_i), \phi(u'_i)) = (v_i, u_i) \in E_y$. Therefore ϕ describes an isomorphism between graphs G_x and G_y, so $\langle M(x), M(y)\rangle \in DirGI$.

Suppose $\langle x, y\rangle \notin R_a$, so $x \neq y$ and $x \oplus y \neq a$. Thus $\exists i, j \in \{1, 2, \ldots, n\} : x_i \neq y_i$ and $x_j \oplus y_j \neq a_j$ (with the possibility that $i = j$). Because $M(x)$ and $M(y)$ both produce graphs which are the disjoint union of two distinct subgraphs, (U_x, F_x) and (U'_x, F'_x) in G_x and (U_y, F_y) and (U'_y, F'_y) in G_y, if the graphs G_x and G_y were isomorphic, the bijection between them must either map vertices of U_x to U_y and U'_x to U'_y or map vertices of U_x to U'_y and U'_x to U_y. The only possible bijections must map either $v_i \in U_x$ to $v_i \in U_y$ and $v'_i \in U'_x$ to $v'_i \in U'_y$ or $v_i \in U_x$ to $v'_i \in U'_y$ and $v'_i \in U'_x$ and $v_i \in U_y$, $\forall i \in \{1, 2, \ldots, n\}$, because of the chain of directed edges between each vertex of adjacent index.

Assume without loss of generality that $x_i = 1$ and $y_i = 0$, which implies $(v_i, u_i) \in F_x$ but $(v_i, u_i) \notin F_y$. Now ϕ cannot map $v_i \in U_x$ to $v_i \in U_y$ and $v'_i \in U'_x$ to $v'_i \in U'_y$ because $(v_i, u_i) \in F_x$ but $(v_i, u_i) \notin F_y$.

Assume ϕ describes a graph isomorphism which maps $v_j \in U_x$ to $v'_j \in U'_y$ and $v'_j \in U'_x$ to $v_j \in U_y$. In the case that $x_j = 0$, $y_j = 0$ and $a_j = 1$, then $y_j \oplus a_j = 1$, so $(v_j, u_j) \notin F_x$ but $(v'_j, u'_j) \in F'_y$. This is a contradiction with the assumption that ϕ describes a graph isomorphism. In the case that $x_j = 1$, $y_j = 1$ and $a_j = 1$, then $y_j \oplus a_j = 0$, so $(v_j, u_j) \in F_x$ but $(v'_j, u'_j) \notin F'_y$. This is a contradiction. In the case that $x_j = 0$, $y_j = 1$, and $a_j = 0$, then $y_j \oplus a_j = 1$, so $(v_j, u_j) \notin F_x$ but $(v'_j, u'_j) \in F'_y$. This is a contradiction. In the case that $x_j = 1$, $y_j = 0$, and $a_j = 0$, then $y_j \oplus a_j = 0$, so $(v_j, u_j) \in F_x$ but $(v'_j, u'_j) \notin F'_y$. This is a contradiction. Therefore no such bijection ϕ exists. Therefore G_x is not isomorphic to G_y, thus $\langle M(x), M(y)\rangle \notin DirGI$.

Therefore $\langle x, y\rangle \in R_a \iff \langle M(x), M(y)\rangle \in DirGI$, so $R_a \leq^p_{ker} DirGI$. □

Corollary 4.12. $R_a \leq^p_{ker} GI$

Proof. Follows directly from Theorem 4.11 and Lemma 4.6. □

4.3 Graph isomorphism and NPEq-completeness

One question we wish to address is whether there exist NPEq-complete problems (indeed, whether there exist PEq-complete problems as well). While we can't yet describe what an NPEq-complete problem looks like, we know some NP-complete equivalence problems. We first define a complete graph and a clique.

Definition 4.13. Let $K_n = (V, E)$ be an undirected graph such that $|V| = n$ and $\forall u, v \in V$, with $u \neq v$, $\{u, v\} \in E$. Then K_n is called the *complete graph* with n vertices.

Definition 4.14. Let $G = (V, E)$ be an undirected graph. Then $C \subseteq V$ is a *clique* if $\forall u, v \in C$, with $u \neq v$, $\{u, v\} \in E$. In other words, the subgraph of G induced by C is a complete graph.

Definition 4.15. $R_{KC} = \{\langle\langle G_1, k_1\rangle, \langle G_2, k_2\rangle\rangle | k_1 = k_2$ and ($G_1 \cong G_2$ or (G_1 has a clique of size k_1 and G_2 has a clique of size k_2))$\}$

Theorem 4.16. *R_{KC} is an equivalence relation on Σ^*.*

Proof. To show that R_{KC} is an equivalence relation, we must show that it is reflexive, symmetric, and transitive.

Since G is isomorphic to G for all graphs G, $\langle\langle G, k\rangle, \langle G, k\rangle\rangle \in R_{KC}$, for all $k \in \mathbb{N}$, so R_{KC} is reflexive.

To show that R_{KC} is symmetric, suppose $\langle\langle G_1, k_1\rangle, \langle G_2, k_2\rangle\rangle \in R_{KC}$. In the case that G_1 is isomorphic to G_2, then G_2 is isomorphic to G_1 because the isomorphism relation is symmetric, so $\langle\langle G_2, k_2\rangle, \langle G_1, k_1\rangle\rangle \in R_{KC}$. In the case that G_1 has a clique of size k_1 and G_2 has a clique of size k_2 and $k_1 = k_2$, then $\langle\langle G_2, k_2\rangle, \langle G_1, k_1\rangle\rangle \in R_{KC}$ because the logical conjunction operation is commutative over propositions.

To show that R_{KC} is transitive, suppose $\langle\langle G_1, k_1\rangle, \langle G_2, k_2\rangle\rangle \in R_{KC}$ and $\langle\langle G_2, k_2\rangle, \langle G_3, k_3\rangle\rangle \in R_{KC}$. Since $k_1 = k_2$ and $k_2 = k_3$, then $k_1 = k_3$ by the transitivity of the equality relation. There are four possible cases for the remaining properties.

In the case that G_1 is isomorphic to G_2 and G_2 is isomorphic to G_3, then G_1 is isomorphic to G_3 so $\langle\langle G_1, k_1\rangle, \langle G_3, k_3\rangle\rangle \in R_{KC}$.

In the case that G_1 is isomorphic to G_2, G_2 has a clique of size k_2, G_3 has a clique of size k_3, then G_1 has a clique of size $k_2 = k_1$, so $\langle\langle G_1, k_1\rangle, \langle G_3, k_3\rangle\rangle \in R_{KC}$.

In the case that G_1 has a clique of size k_1, G_2 has a clique of size k_2, and G_2 is isomorphic to G_3, then G_3 has a clique of size $k_2 = k_3$, so $\langle\langle G_1, k_1\rangle, \langle G_3, k_3\rangle\rangle \in R_{KC}$.

In the case that G_1 has a clique of size k_1, G_2 has a clique of size k_2, and G_3 has a clique of size k_3, then $\langle\langle G_1, k_1\rangle, \langle G_3, k_3\rangle\rangle \in R_{KC}$.

Therefore R_{KC} is reflexive, symmetric, and transitive, hence it is an equivalence relation. □

Definition 4.17. $CLIQUE = \{\langle G, k\rangle | G \text{ has a clique of size } k\}$

Lemma 4.18. *$CLIQUE$ is* NP*-complete.*

Proof. The proof is omitted; we refer the reader to [GJ79] for the reduction. □

Lemma 4.19. $R_{KC} \in \mathsf{NP}$

Proof. Since $GI \in \mathsf{NP}$, it has a deterministic polynomial time verifier, M_1, which accepts on input $\langle G_1, G_2, c\rangle$, where c is the isomorphism from vertices of G_1 to vertices of G_2.

Since $CLIQUE \in \mathsf{NP}$, it has a deterministic polynomial time verifier, M_2, which accepts on input $\langle G, k, c\rangle$, where c is the set of vertices in G which comprise a clique of size k.

To show that $R_{KC} \in \mathsf{NP}$, we construct a deterministic polynomial time verifier M for R_{KC}. On input $\langle\langle G_1, k_1\rangle, \langle G_2, k_2\rangle, c\rangle$:

```
1 If k_1 ≠ k_2, reject
2 if c is the encoding of a mapping then
3     Run M_1 on input ⟨G_1, G_2, c⟩
4     If M_1 accepts, accept; otherwise reject
5 if c = ⟨c_1, c_2⟩ is the encoding of two cliques then
6     Run M_2 on input ⟨G_1, k_1, c_1⟩
7     Run M_2 on input ⟨G_2, k_2, c_2⟩
8     If M_2 accepts on both inputs, accept; otherwise reject
```

Machine M is a verifier for R_{KC}, so $R_{KC} \in \mathsf{NP}$. □

Corollary 4.20. $R_{KC} \in \mathsf{NPEq}$

Proof. Since $R_{KC} \in \mathsf{NP}$ by Lemma 4.19 and R_{KC} is an equivalence problem, then by Definition 2.10, $R_{KC} \in \mathsf{NPEq}$. □

Theorem 4.21. *R_{KC} is* NP*-complete.*

Proof. Since $R_{KC} \in \mathsf{NP}$ by Lemma 4.19, we need only show that R_{KC} is NP-hard. To do this, we construct a polynomial time many-one reduction from $CLIQUE$, which is NP-complete, to R_{KC}.

Construct machine $M \in \mathsf{FP}$ on input $\langle G, k\rangle$ which outputs $\langle\langle G, k\rangle, \langle K_k, k\rangle\rangle$, where K_k is the complete graph with k vertices.

Suppose $\langle G, k\rangle \in CLIQUE$, so G has a clique of size k. Then $M(\langle G, k\rangle) = \langle\langle G, k\rangle, \langle K_k, k\rangle\rangle \in R_{KC}$, because G has a clique of size k by hypothesis and K_k has a clique of size k by construction, specifically, the set of all vertices in K_k.

Suppose $\langle G, k\rangle \notin CLIQUE$, so G does not have a clique of size k and $M(\langle G, k\rangle) = \langle\langle G, k\rangle, \langle K_k, k\rangle\rangle$. If G were isomorphic to K_k, then it would have a clique of size k, specifically the set of all its vertices, but this is a contradiction with the hypothesis so no such isomorphism exists. Although K_k certainly has a clique of size k, specifically the set of all its vertices, G does not have a clique of size k by hypothesis, so $\langle\langle G, k\rangle, \langle K_k, k\rangle\rangle \notin R_{KC}$.

Therefore $\langle G, k\rangle \in CLIQUE \iff M(\langle G, k\rangle) \in R_{KC}$, so $CLIQUE \leq_m^p R_{KC}$, and hence R_{KC} is NP-complete. □

So we have now shown that $R_{KC} \in \mathsf{NPC} \cap \mathsf{NPEq}$, where NPC is the set of all NP-complete languages. With this fact we can show a relationship between completeness under kernel reductions and completeness under many-one reductions in NPEq which follows from the fact that a kernel reduction implies a many-one reduction.

Theorem 4.22. *If an equivalence relation A is NPEq-complete, then A is NP-complete.*

Proof. If A is NPEq-complete then $R_{KC} \leq^p_{ker} A$, since $R_{KC} \in \mathsf{NPEq}$ by Corollary 4.20. By Lemma 2.13, $R_{KC} \leq^p_{ker} A \implies R_{KC} \leq^p_m A$. Since R_{KC} is NP-complete by Theorem 4.21, then A is NP-complete. □

Of course, Theorem 4.22 is not all that useful if NPEq-complete problems do not exist. We wish to give some evidence that R_{KC} (or a language similar to it) may be NPEq-complete. We know that many problems in NP, specifically many in NPEq, kernel reduce to the graph isomorphism problem (in the past, the reductions have not explicitly been presented as kernel reductions, but we can retroactively identify the techniques used as such). If we can show a polynomial time kernel reduction from the graph isomorphism problem to R_{KC}, then we have found many NPEq problems which reduce to R_{KC}. Indeed, we defined R_{KC} in such a way to allow a kernel reduction from the graph isomorphism problem to be possible.

Theorem 4.23. $GI \leq^p_{ker} R_{KC}$

Proof. Construct machine $f \in \mathsf{FP}$ defined for all graphs $G = (V, E)$ by $f(G) = \langle G, |V| + 1\rangle$.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, and suppose $\langle G_1, G_2\rangle \in GI$, so G_1 is isomorphic to G_2. This implies $|V_1| = |V_2|$ and thus $|V_1| + 1 = |V_2| + 1$. Now $f(G_1) = \langle G_1, |V_1| + 1\rangle$ and $f(G_2) = \langle G_2, |V_2| + 1\rangle$. Since G_1 is isomorphic to G_2 and $|V_1| + 1 = |V_2| + 1$, then $\langle\langle G_1, |V_1| + 1\rangle, \langle G_2, |V_2| + 1\rangle\rangle = \langle f(G_1), f(G_2)\rangle \in R_{KC}$.

Suppose $\langle G_1, G_2\rangle \notin GI$, so G_1 is not isomorphic to G_2. Even in the case that $|V_1| = |V_2|$, G_1 cannot have a clique of size $|V_1| + 1$, and G_2 cannot have a clique of size $|V_2| + 1$. Thus $\langle\langle G_1, |V_1| + 1\rangle, \langle G_2, |V_2| + 1\rangle\rangle = \langle f(G_1), f(G_2)\rangle \notin R_{KC}$.

Therefore $\langle G_1, G_2\rangle \in GI \iff \langle f(G_1), f(G_2)\rangle \in R_{KC}$, so $GI \leq^p_{ker} R_{KC}$. □

We constructed R_{KC} in order to make this reduction easy, but there is in fact another well-known problem in $\mathsf{NPEq} \cap \mathsf{NPC}$. Garey and Johnson provide a single NP-complete equivalence problem: a databases problem concerning the equivalence of tableaux.[GJ79] The original proof is a reduction from 3-SAT.[ASU79]

Reductions between the graph isomorphism problem and other equivalence problems in NPEq abound, and though they have traditionally been presented as many-one reductions, they are in fact kernel reductions. Zemlyachenko, Korneenko and Tyshkevich notice this in [ZKT85], and describe the kernel reductions in the terminology of category theory as follows: Let C_1 and C_2 be concrete categories, let x, y be objects in C_1, let x', y' be objects in C_2, let $Mor_{C_1}(x, y)$ be the morphism between objects x and y in C_1, let $Mor_{C_2}(x', y')$ be the morphism between objects x' and y' in C_2, and let $f\colon C_1 \to C_2$ be a functor. Define $g\colon Mor(C_1) \to Mor(C_2)$ by $g(Mor_{C_1}(x, y)) = Mor_{C_2}(f(x), f(y))$. If g is bijective then g is called a *complete embedding*. If g is a complete embedding then we say the isomorphism problem for objects in category C_1 *reduces functorially* to the isomorphism problem for objects in category C_2.

Notice that in the above definition, if we let $R = \{\langle x, y\rangle | x$ and y are objects of C_1 and x is isomorphic to y by $Mor_{C_1}(x, y)\}$ and $S = \{\langle x, y\rangle | x$ and y are objects of C_2 and x is isomorphic to y by $Mor_{C_2}(x, y)\}$, then $R \leq^p_{ker} S$ by f, and $R \leq^p_m S$ by g, the many-one reduction induced by the kernel reduction f.

A functorial reduction as defined above is *more restrictive* than a kernel reduction, because it requires that there exist a bijection between the isomorphisms of the two categories. The definition of a kernel reduction does not demand this bijection, so a functorial reduction implies a kernel reduction.

In [ZKT85], the authors provide a survey of categories for which isomorphism problems are polynomial-time functorially equivalent. For our purposes, this is a simply a listing of equivalence problems which are polynomial time kernel equivalent to graph isomorphism. We have, in addition, provided some more recent results on isomorphism of objects in other categories.

- directed graph isomorphism [Mil79]
- labeled tree isomorphism [Bab79]

- polar graph isomorphism [ZKT85]
- two-color graph isomorphism [ZKT85]
- finite model isomorphism [Mil79]
- color graph isomorphism [ZKT85, Mil77, Pul64]
- stable graph isomorphism [WL68]
- multigraph isomorphism [ZKT85]
- hypergraph isomorphism [ZKT85]
- k-uniform hypergraph isomorphism [ZKT85, HJ70]
- finite automaton isomorphism [Boo78]
- lattice isomorphism [Fru50]
- isomorphism of associative algebras of finite rank over a fixed algebraically closed field with a square of the radical equal to zero and with a commutative factor with respect to the radical [Gri83]
- Markov decision process isomorphism [NR08]
- balanced incomplete block-scheme isomorphism [CC81]
- combinatorial isomorphism of convex polytopes represented by vertex-facet incidences [KS03]
- equivalence of Hadamard matrices [McK79]

We will consider the general class of problems equivalent to the graph isomorphism problem under polynomial time kernel reductions by defining the following complexity class.

Definition 4.24. $\mathsf{GI}_{ker} = \{A | A \equiv^p_{ker} GI\}$

All of the problems cited above fall in this class.

Lemma 4.25. $\mathsf{GI}_{ker} \subset \mathsf{NPEq}$

Proof. Let $A \in \mathsf{GI}_{ker}$. Then $A \leq^p_{ker} GI$ and hence $A \leq^p_m GI$. Since $GI \in \mathsf{NP}$ and NP is closed under polynomial time many-one reductions, $A \in \mathsf{NP}$. Since A is an equivalence problem by definition (only equivalence problems can kernel reduce), then $A \in \mathsf{NPEq}$. Therefore $\mathsf{GI}_{ker} \subseteq \mathsf{NPEq}$. □

There is evidence that shows that the graph isomorphism problem is not NP-complete, specifically that it is not NP-complete unless the polynomial time hierarchy collapses to the second level[Sch87]. If this is true, then the following theorem about the complexity of all equivalence problems equivalent under polynomial time kernel reductions to the graph isomorphism problems follows.

Theorem 4.26. *If GI is not $\leq^p_m$-complete in NP, then $\mathsf{GI}_{ker} \subsetneq \mathsf{NPEq}$.*

Proof. By Lemma 4.25, $\mathsf{GI}_{ker} \subset \mathsf{NPEq}$.

Assume with the intention of producing a contradiction that $\mathsf{GI}_{ker} = \mathsf{NPEq}$. Since every language A in GI_{ker} polynomial time kernel reduces to GI, GI is complete under polynomial time kernel reductions in GI_{ker}. Since $\mathsf{GI}_{ker} = \mathsf{NPEq}$, GI is complete under polynomial time kernel reductions in NPEq. By Theorem 4.22, GI is $\leq^p_m$-complete in NP. This is a contradiction with the hypothesis that GI is not $\leq^p_m$-complete. □

Intuitively, this means that if the graph isomorphism problem is not NP-complete, then there are equivalence problems harder than the graph isomorphism problem in NP, unless $\mathsf{P} = \mathsf{NP}$.

The graph isomorphism problem has been (and continues to be) extensively studied. For more information, see [KST93].

5 Kernel reductions vs. many-one reductions

Now we wish to address the question of whether kernel reductions are different from many-one reductions.

To show that they are in fact different, we need only exhibit two equivalence relations, R and S, such that $R \leq_m^p S$ but $R \not\leq_{ker}^p S$. Intuitively, this means that when $\langle x, y\rangle \in R$, there is some information shared between x and y which can not be separated from either. In other words, x can only be understood with knowledge of y and y can only be understood with knowledge of x. In attempting to construct such a relation, we have found that reflexivity, symmetry, and transitivity of a relation seem to provide a significant obstacle to the possibility of sharing information between related elements.

To show that the two reductions are the same, we need to prove that a many-one reduction implies a kernel reduction. Proving this seems even more difficult than proving the negation. A many-one reduction gets access to both related elements, and using a many-one reduction to construct a kernel reduction would require something like examining the machine which computes the many-one reduction and determining in which states that machine is "using" which of the elements.

Still, we have no reason in particular to believe that a many-one reduction is different from a kernel reduction.

6 Future work

We hope to extend the result in Theorem 3.22 to equivalence problems with an infinite number of equivalence classes. In other words, we wish to answer the question of whether we can *always* compute a kernel reduction among equivalence problems in PEq, if the reduction is allowed polynomial time.

We also hope to continue working on showing that a many-one reduction does not imply a kernel reduction for equivalence problems, by finding two equivalence problems for which there is a many-one reduction but not a kernel reduction. Perhaps Kolmogorov complexity or quantum entanglement of related members of an equivalence relation holds the key to sharing information among related binary strings.

7 Acknowledgments

Thanks to Ben Hescott, Anselm Blumer, Mary Glaser and Sam Guyer for valuable input.

References

[ASU79] A. V. Aho, Y. Sagiv, and J. D. Ullman, *Equivalences among relational expressions*, SIAM Journal on Computing **8** (1979), no. 2, 218–246. 17

[Bab79] László Babai, *Monte carlo algorithms in graph isomorphism testing*, Tech. Report DMS 79-10, Université de Montréal, 1979. 17

[BDG95] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró, *Structural complexity i*, 2 ed., Springer-Verlag, February 1995. 2

[Boo78] Kellogg S. Booth, *Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems*, SIAM Journal on Computing **7** (1978), no. 3, 273–279. 18

[CC81] Marlene J. Colbourn and Charles J. Colbourn, *Concerning the complexity of deciding isomorphism of block designs*, Discrete Applied Mathematics **3** (1981), no. 3, 155–162. 18

[FG09] Lance Fortnow and Joshua A. Grochow, *Complexity classes of equivalence problems revisited*, CoRR **abs/0907.4775** (2009). 1, 3, 8, 10

[Fru50] Robert Frucht, *Lattices with a given abstract group of automorphisms*, Canadian Journal of Mathematics **2** (1950), 417–419. 18

[GJ79] Michael R. Garey and David S. Johnson, *Computers and intractability: A guide to the theory of np-completeness*, W. H. Freeman and Company, New York, NY, 1979. 16, 17

[Gri83] D. Yu. Grigor'ev, *Complexity of "wild" matrix problems and of isomorphism of algebras and graphs*, Journal of Mathematical Sciences **22** (1983), no. 3, 1285–1289. 18

[HJ70] Pavol Hell and Nešetřil Jaroslav, *Graphs and k-societies*, Canadian Mathematical Bulletin **13** (1970), 375–381. 18

[KS03] Volker Kaibel and Alexander Schwartz, *On the complexity of polytope isomorphism problems*, Graphs and Combinatorics **19** (2003). 18

[KST93] J. Köbler, U. Schöning, and J. Torán, *The graph isomorphism problem: its structural complexity*, Springer, 1993. 12, 18

[McK79] Brendan D. McKay, *Hadamard equivalence via graph isomorphism*, Discrete Mathematics **27** (1979), no. 2, 213–214. 18

[Mil77] Gary L. Miller, *Graph isomorphism, general remarks*, Proceedings of the ninth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '77, ACM, 1977, pp. 143–150. 12, 18

[Mil79] Gary L. Miller, *Graph isomorphism, general remarks*, Journal of Computer and System Sciences **18** (1979), no. 2, 128–142. 17, 18

[NR08] Shravan Matthur Narayanamurthy and Balaraman Ravindran, *On the hardness of finding symmetries in markov decision processes*, ICML '08: Proceedings of the 25th international conference on Machine learning (New York, NY, USA), ACM, 2008, pp. 688–695. 18

[Pul64] A. Pultr, *Concerning universal categories*, Commentationes Mathematicae Universitatis Carolinae **5** (1964), no. 4, 227–239. 18

[Sch87] Uwe Schöning, *Graph isomorphism is in the low hierarchy*, Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (London, UK), STACS '87, Springer-Verlag, 1987, pp. 114–124. 18

[Sip06] Michael Sipser, *Introduction to the theory of computation*, 2 ed., Thomson Course Technology, 2006. 2

[WL68] B. Weisfeiler and A. A. Lehman, *Reduction of a graph to a canonical form and an algebra arising during this reduction*, Nauchno-Techn. Inform. **2** (1968), no. 9, 12–16. 18

[ZKT85] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich, *Graph isomorphism problem*, Journal of Mathematical Sciences **29** (1985), no. 4, 1426–1481. 17, 18