The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups ⋆

Gustav Nordh [1]

Department of Computer and Information Science
Linköpings Universitet
S-581 83 Linköping, Sweden
Fax: +46 13 28 44 99

Abstract

We study the computational complexity of the isomorphism and equivalence problems on systems of equations over a fixed finite group. We show that the equivalence problem is in P if the group is Abelian, and coNP-complete if the group is non-Abelian. We prove that if the group is non-Abelian, then the problem of deciding whether two systems of equations over the group are isomorphic is coNP-hard. If the group is Abelian, then the isomorphism problem is Graph Isomorphism-hard. Moreover, if we impose the restriction that all equations are of bounded length, then we prove that the isomorphism problem for systems of equations over finite Abelian groups is Graph Isomorphism-complete. Finally we prove that the problem of counting the number of isomorphisms of systems of equations is no harder than deciding whether there exist any isomorphisms at all.

Key words: graph isomorphism, computational complexity, systems of equations, finite groups

⋆ A preliminary version of this paper appears in *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science* (MFCS'04), LNCS 3153, Springer-Verlag, pp. 380-391, 2004.

Email address: `gusno@ida.liu.se` (Gustav Nordh).

[1] Supported by the *National Graduate School in Computer Science* (CUGS), Sweden.

1 Introduction

The computational complexity of deciding whether a system of equations over a fixed finite group is solvable has been studied in the past. Goldmann and Russel [6] proved that the problem is in P if the group is Abelian and NP-complete otherwise. This line of research continued in [8,11], where the corresponding problem for finite monoids was given a complete solution. Moreover, some very interesting results in the general case of finite semigroups have been proved by Klíma et al. in [8]. Note that even the restricted problem of determining the computational complexity of solving systems of equations over a fixed regular semigroup is still open. The problem of deciding whether systems of equations over a fixed finite group $(G, \cdot)$ are solvable is denoted by $\textsc{Eqn}^*_G$ in the literature.

The computational complexity of counting solutions to systems of equations over a fixed finite semigroup has been studied in [12], where it is proved that if the semigroup is an Abelian group, then the problem is in FP, and if the semigroup is a non-Abelian group, then the problem is #P-complete. This problem is denoted $\#\textsc{Eqn}^*_G$.

In this paper we study the computational complexity of deciding whether systems of equations over a fixed finite group are equivalent/isomorphic. More specifically, the equivalence problem is the problem of deciding whether two systems of equations have the same set of solutions and the isomorphism problem is the problem of deciding whether two systems of equations can be made equivalent by permuting the variables in one of them. We also study the problem of counting the number of isomorphisms. These fundamental problems have as far as we know eluded previous investigations from a computational complexity perspective, except for some results on the Boolean constraint equivalence and isomorphism problems, due to Böhler et al. [2,4], that are also relevant in our setting. More specifically, the equivalence problem for systems of equations over the two element group $\mathbb{Z}_2$ where each equation has a bounded number of variable occurrences is in P, and the corresponding isomorphism problem is Graph Isomorphism-complete. Note that in [2,4] Böhler et al. only study the equivalence and isomorphism problems for Boolean constraints over fixed finite constraint languages, and hence the arity of all constraints involved can be assumed to be bounded by a constant. This motivates us to additionally study the complexity of our problems under the restriction that the number of variables in each equation is bounded by a constant.

The computational complexity of several other isomorphism and equivalence problems have been intensively studied in the past, most notably the Graph Isomorphism problem [7], the formula isomorphism problem [1], and the isomorphism problem for branching programs [13]. Although there are not many results in the literature having direct implications for the equivalence and isomorphism problems for systems of equations, many of the constructions and proof techniques from [1,2,4,10] can be reused.

1.1 Definitions and Summary of Results

A system of equations over a fixed finite group $(G, \cdot)$ is a collection of equations of the form $x_1 \cdot x_2 \cdot \ldots \cdot x_k = x_{k+1} \cdot \ldots \cdot x_n$, where each x_i is either a variable or a constant in G. We will often use G as a shorthand for $(G, \cdot)$ and in the case of Abelian groups we will denote the group operation by $+$. We also assume that all equations have been simplified so that they do not contain subexpressions of the type $a \cdot b$ where a and b are group constants.

The length of an equation is defined to be the number of variable occurrences in it. A system of equations is said to be of length k if k is the length of the longest equation in the system.

Example 1 *Let S_1 be the following system of equations over $\mathbb{Z}_3$:*

$$x + x + y = 2 + z \quad (1)$$

$$x + 2z = 1 \quad (2)$$

$$2 + w + 1 + y = z + 2 \quad (3)$$

Equation (1) is of length 4 (x occurs twice, y and z occur once each), (2) is of length 3 ($2z$ amounts to two occurrences of z since $2z$ is just another way of writing $z + z$), and (3) is of length 3. Hence S_1 is of length 4.

Definition 2 *Let S be a system of equations on variables X and let π be a permutation of X. By $\pi(S)$ we denote the system of equations that results when we replace each variable x in S by $\pi(x)$.*

- $\textsc{Equiv-Eqn}^*_G$ *is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over G are equivalent, i.e., whether for every assignment of values in G to the variables in X, S_1 is satisfied if and only if S_2 is satisfied. Note that when we say that S_1 and S_2 are systems of equations on variables X we only mean that X is the union of the variables in S_1 and S_2, hence all variables in X need not to occur both in S_1 and S_2. If S_1 is equivalent to S_2 we denote this by $S_1 \equiv S_2$.*
- $\textsc{Iso-Eqn}^*_G$ *is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over G are isomorphic, i.e., whether there exists a permutation π of the variables in X such that $\pi(S_1) \equiv S_2$. If S_1 is isomorphic to S_2 we denote this by $S_1 \cong S_2$.*

- $\text{Iso-B-Eqn}^*_{G,k}$ *and* $\text{Equiv-B-Eqn}^*_{G,k}$ *are the restricted forms of* Iso-Eqn^*_G *and* Equiv-Eqn^*_G *respectively, where the systems of equations have been bounded to have length at most* k*.*
- $\#\text{Iso-Eqn}^*_G$ *is the counting version of* Iso-Eqn^*_G*, i.e., the problem of counting the number of permutations* π *of the variables in* X *such that* $\pi(S_1) \equiv S_2$*.*

The complexity of Equiv-Eqn^*_G (Iso-Eqn^*_G, $\#\text{Iso-Eqn}^*_G$) is measured in the size of the systems of equations (the size of G is fixed and does not matter). Note that in all of the problems described above, each group gives rise to a distinct problem. Hence we are trying to classify the complexity of infinite classes of problems. Moreover, the additional restriction that the systems of equations must have length at most k gives rise to infinite classes of problems for each fixed group. This framework makes it possible to give a very fine grained analysis for the complexity of the equivalence and isomorphism problems for systems of equations over finite groups. Also note that all the problems above become trivial if the group G is the (trivial) one element group. Hence when stating our hardness results for problems over Abelian groups we will always assume that the group involved is different from the trivial one.

The main results of the paper can be summarized as follows. We prove that if the group is non-Abelian, then the equivalence problem is coNP-complete and if the group is Abelian, then the equivalence problem is in P. As for the isomorphism problem we prove that if the group is non-Abelian, then the isomorphism problem is coNP-hard and in Σ_2^{P}. If the group is Abelian, then the isomorphism problem is Graph Isomorphism-hard and in NP. For the restriction of the problems where the length of all equations are bounded by k, we are able to give a very detailed picture for the complexity of $\text{Iso-B-Eqn}^*_{G,k}$ and $\text{Equiv-B-Eqn}^*_{G,k}$. $\text{Equiv-B-Eqn}^*_{G,k}$ is coNP-complete when G is non-Abelian and $k \geq 3$, otherwise $\text{Equiv-B-Eqn}^*_{G,k}$ is in P. $\text{Iso-B-Eqn}^*_{G,k}$ is in P when $k \leq 2$ (for all groups G). If $k \geq 3$, then $\text{Iso-B-Eqn}^*_{G,k}$ is Graph Isomorphism-complete when G is Abelian, and coNP-hard and in $\mathrm{P}_{||}^{\mathrm{NP}}$ for all non-Abelian groups G. Our results on the complexity of $\text{Equiv-B-Eqn}^*_{G,k}$ and $\text{Iso-B-Eqn}^*_{G,k}$ can be seen as a first attempt to extend the complexity classification of the equivalence and isomorphism problems for constraints over fixed finite Boolean constraint languages to larger domains.

For the problem of counting the number of isomorphisms, we give an algorithm that shows that it is no harder to count the number of isomorphisms than to decide whether any isomorphisms exist at all. As a corollary to this algorithm we obtain the result that Iso-Eqn^*_G for Abelian groups is powerless as an oracle to PP, and that Iso-Eqn^*_G for non-Abelian groups is no more powerful than an NP-oracle for PP. These results indicate that Iso-Eqn^*_G for Abelian groups is not NP-complete, and that Iso-Eqn^*_G for non-Abelian groups is not Σ_2^{P}-complete.

The paper is organized as follows. In Section 2 we prove our results concerning the equivalence problem, Section 3 deals with the isomorphism problem, Section 4 treats the problem of counting the number of isomorphisms, and finally in Section 5 we present our conclusions and some ideas for future research.

2 Equivalence

In this section we investigate the computational complexity of Equiv-Eqn$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are equivalent. We prove that Equiv-Eqn$^*_G$ is coNP-complete if G is non-Abelian, and that Equiv-Eqn$^*_G$ is in P if G is an Abelian group.

First note that it is easy to see that the equivalence problem is in coNP.

Theorem 3 Equiv-Eqn$^*_G$ *is in* coNP.

The following theorem states that if it is hard to decide whether systems of equations over a group G are solvable, then it is also hard to decide whether systems of equations over the same group are equivalent.

Theorem 4 *If* Eqn$^*_G$ *is* NP*-complete, then* Equiv-Eqn$^*_G$ *is* coNP*-complete.*

PROOF. If Eqn$^*_G$ is NP-complete, it follows that it is coNP-complete to decide whether a system of equations over G is insoluble. Since a system of equations is insoluble if and only if it is equivalent to an insoluble system of equations (for example a system of equations containing the equation $a = b$ where a and b are distinct constants in G), it follows that Equiv-Eqn$^*_G$ is coNP-complete. □

The previous theorem and the fact that Eqn$^*_G$ is NP-complete when G is a non-Abelian group [6] immediately implies the following corollary.

Corollary 5 Equiv-Eqn$^*_G$ *is* coNP*-complete when* G *is a non-Abelian group.*

Next we prove that if it is easy to count the number of solutions to systems of equations over a group G, then it is also easy to decide whether systems of equations over the same group are equivalent.

Theorem 6 Equiv-Eqn$^*_G$ *is in* P *if* #Eqn$^*_G$ *is in* FP.

PROOF. Let S_1 and S_2 be two systems of equations. Count the number of solutions to S_1 and S_2: if they have different number of solutions they are not equivalent. Thus we can assume that S_1 and S_2 have the same number of solutions. Count the number of solutions to the system of equations consisting of the union of S_1 and S_2. If the number of solutions to this system of equations equals the number of solutions to S_1, then we know that S_1 and S_2 have the same set of solutions and hence S_1 is equivalent to S_2, otherwise S_1 and S_2 have different sets of solutions and thus are inequivalent. $\square$

The following corollary follows directly from Theorem 6 and the fact that $\#\text{E}\textsc{qn}^*_G$ is in FP when G is an Abelian group [12].

Corollary 7 *$\text{E}\textsc{quiv}$-$\text{E}\textsc{qn}^*_G$ is in P when G is an Abelian group.*

It should be clear that Theorems 4 and 6 also hold when generalized to systems of equations over a fixed finite semigroup G. Hence interesting results on the complexity of $\text{E}\textsc{quiv}$-$\text{E}\textsc{qn}^*_G$, where G is a finite semigroup, can be deduced from the results for $\#\text{E}\textsc{qn}^*_G$ and $\text{E}\textsc{qn}^*_G$ proved in [8,11,12]. We collect these results in the following corollary.

Corollary 8 *$\text{E}\textsc{quiv}$-$\text{E}\textsc{qn}^*_G$ is coNP-complete when*

- *G is a monoid but is not commutative [11].*
- *G is a monoid but is not a union of Abelian groups [11].*
- *G is a regular semigroup but is not a strong normal band of Abelian groups [8].*

$\text{E}\textsc{quiv}$-$\text{E}\textsc{qn}^*_G$ is in P when

- *G is a direct product of an Abelian group and a rectangular band [12],*
- *G is a semigroup with zero, such that for all elements x, y, $x \cdot y = 0$ [12].*

2.1 Bounded Length

If we restrict the problem and require that all equations are of bounded length, then we can prove an even tighter correspondence between the complexity of the equivalence problem and the solvability problem.

Theorem 9 *If $\text{E}\textsc{qn}^*_{G,k}$ is in P, then $\text{E}\textsc{quiv}$-B-$\text{E}\textsc{qn}^*_{G,k}$ is in P, and if $\text{E}\textsc{qn}^*_{G,k}$ is NP-complete, then $\text{E}\textsc{quiv}$-B-$\text{E}\textsc{qn}^*_{G,k}$ is coNP-complete (where $\text{E}\textsc{qn}^*_{G,k}$ denotes the solvability problem restricted to systems of equations of length at most k).*

PROOF. For the tractability part we give a polynomial-time Turing reduction from Equiv-B-$\text{Eqn}^*_{G,k}$ to $\text{Eqn}^*_{G,k}$. The reduction is the same as the one in the proof of [2, Lemma 7]. Let $S \to E$ denote that the equation E can be inferred from the system of equations S, i.e., there exists no assignment to the variables in S such that S is satisfied and E is not satisfied. Let S_1 and S_2 be two systems of equations where all equations are of length at most k for some constant k. S_1 and S_2 are equivalent if and only if $S_1 \to E$ for every equation E in S_2, and $S_2 \to E'$ for every equation E' in S_1. Note that it is easy to check whether $S_1 \to E$ with at most $|G|^k$ queries to $\text{Eqn}^*_{G,k}$. For every assignment to the variables in E that does not satisfy E, we check whether this partial assignment of the variables can be extended to a satisfying assignment for S_1. Of course, $S_1 \to E$ if and only if none of these assignments can be extended to a satisfying assignment to S_1.

The coNP-completeness part follows from Theorem 4. $\square$

Note that the preceding theorem also holds when generalized to systems of equations over a fixed finite semigroup G.

It is easy to see that equations of length more than 3 can always be split into equations of length 3 in a solvability preserving manner by introducing new variables. Hence, Eqn^*_G is polynomial-time reducible to the restricted form where each equation have at most 3 variable occurrences, i.e., Eqn^*_G is polynomial-time reducible to $\text{Eqn}^*_{G,3}$. This observation together with the preceding theorem and the fact that Eqn^*_G is NP-complete for non-Abelian groups [6], gives us the following result.

Corollary 10 Equiv-B-$\text{Eqn}^*_{G,k}$ *is* coNP*-complete for all non-Abelian groups* G *when* $k \geq 3$*.*

Next we prove that the equivalence problem for systems of equations of length at most 2 is in P for all groups G.

Theorem 11 Equiv-B-$\text{Eqn}^*_{G,k}$ *is in* P *for all groups* G *when* $k \leq 2$*.*

PROOF. We prove that $\text{Eqn}^*_{G,2}$ is in P. Hence, the desired result then follows from Theorem 9. First of all, equations of the form $a = a$, where a is a constant from G, are redundant and can be removed, and if there exists an equation in the system of equations of the form $d = e$, where d and e are distinct constants from G, then clearly the system has no solution. Hence we can assume that all equations contain at least one variable.

Represent the system of equations S as a graph G with one vertex for each variable in S and an edge between any pair of vertices whose corresponding variables occur in the same equation in S. It should be clear that the connected components in G corresponds to independent subsystems of S. Obviously S has a solution if and only if each of these independent subsystems of equations have a solution. Hence each of these independent subsystems can be tested for solvability in isolation.

Let S' be such a subsystem corresponding to a connected component in G. Choose an arbitrary variable in S', say x. If we assign a value $a \in G$ to x, then each equation where x occurred will now be either in the form $b \cdot y = c$ or $d = e$ where b, c, d, and e are (not necessarily distinct) constants from G and y is a variable. Note that since we are working over a group, equations of the form $b \cdot y = c$ always have a unique solution. Hence y will be forced to take a value from G and the propagation can continue in the same manner until all variables in S' have been assigned. Now if there is an equation in the resulting system of equations of the form $d = e$, where d and e are distinct constants from G, then clearly there exists no solution of S' such that a is assigned to x. Otherwise S' has a solution (where a is assigned to x). Obviously S' has a solution if and only if for at least one element $a \in G$, there exists a solution where a is assigned to x. $\square$

3 Isomorphism

In this section we investigate the computational complexity of Iso-Eqn$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are isomorphic. We prove that Iso-Eqn$^*_G$ is coNP-hard if G is non-Abelian, and that Iso-Eqn$^*_G$ is Graph Isomorphism-hard if G is an Abelian group. If we restrict the problem and demand that all equations are of bounded length, then Iso-Eqn$^*_G$ in the Abelian case becomes Graph Isomorphism-complete, and Iso-Eqn$^*_G$ in the non-Abelian case is in $\mathrm{P}^{\mathrm{NP}}_{||}$, i.e., the class of problems solvable in polynomial time with parallel access to an NP-oracle.

We begin by giving upper bounds for the complexity of the isomorphism problem.

Theorem 12 Iso-Eqn$^*_G$ *is in* NP *when G is an Abelian group and* Iso-Eqn$^*_G$ *is in Σ^{P}_2 when G is a non-Abelian group.*

PROOF. The NP upper bound for Iso-Eqn$^*_G$ when G is an Abelian group follows from the results in the previous section on the equivalence problem. We know from Corollary 7 that $\pi(S_1) \equiv S_2$ can be decided in polynomial time when G is an Abelian group, hence Iso-Eqn$^*_G$ is in NP when G is an Abelian group.

For the Σ_2^{P} upper bound we nondeterministically choose a permutation π of the variables in X and use an NP-oracle to check whether $\pi(S_1) \equiv S_2$. Hence Iso-Eqn$^*_G$ is in $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$. $\square$

The following theorem states that if it is hard to decide whether systems of equations over a group G are solvable, then it is also hard to decide whether systems of equations over the same group are isomorphic.

Theorem 13 *If* Eqn$^*_G$ *is* NP*-complete then* Iso-Eqn$^*_G$ *is* coNP*-hard.*

PROOF. If Eqn$^*_G$ is NP-complete, then it follows that it is coNP-hard to decide whether a system of equations over G is insoluble. Since a system of equations is insoluble if and only if it is isomorphic to an insoluble system of equations, e.g., a system of equations containing the equation $a = b$ where a and b are distinct constants in G, it follows that Iso-Eqn$^*_G$ is coNP-hard. $\square$

The previous theorem and the fact that Eqn$^*_G$ is NP-complete when G is a non-Abelian group [6] immediately implies the following corollary.

Corollary 14 Iso-Eqn$^*_G$ *is* coNP*-hard when G is a non-Abelian group.*

The following theorem indicates that Iso-Eqn$^*_G$ for Abelian groups perhaps is not in P, or at least that it is hard to prove that Iso-Eqn$^*_G$ is in P for Abelian groups.

Theorem 15 *Let G be a finite Abelian group (different from the trivial one), then* Graph Isomorphism *is polynomial-time many-one reducible to* Iso-Eqn$^*_G$.

PROOF. We first show the result for the case where G is a cyclic group (that is groups of the form $\mathbb{Z}_m$, i.e., the integers modulo m under addition), then we show how to extend this result to all finite Abelian groups.

Since the group is Abelian, we denote the group operation by $+$. As usual we denote the elements of $\mathbb{Z}_m$ as $\{0, 1, 2 \ldots, m-1\}$, where 0 is the identity element and 1 is the group generator. It is known that Graph Isomorphism is polynomial-time equivalent to the Graph Isomorphism problem restricted to bipartite graphs. To reduce the general Graph Isomorphism problem to the Graph Isomorphism problem restricted to bipartite graphs; we simply split each edge by introducing a new vertex as a middle point, and attach a sufficiently large even cycle to each of these new vertices (forcing middle points to be mapped to middle points). The resulting bipartite graphs are isomorphic if and only if the original graphs are isomorphic. To simplify the proof we assume from now on that all graphs are bipartite and contain no isolated vertices.

Let T be the following transformation from a graph $H = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$, to a system of equations $T(H)$ over G:

$$T(H) = \{x_i + x_j + y_k = 1 \mid e_k = \{i, j\}\} \cup \{x_i + z_i + z_i' = 1 \mid i \in V\}.$$

The variables x_i and y_k correspond to vertex i and edge e_k respectively. The z and z' variables are used to distinguish x variables from y variables. This transformation is based on the construction in the proof of [2, Theorem 25].

Let $T(H) \rightarrow a+b+c = 1$ (where a, b, c are variables occurring in $T(H)$) denote that the equation $a + b + c = 1$ can be inferred from the system of equations $T(H)$, i.e., there exists no assignment to the variables in $T(H)$ such that $T(H)$ is satisfied and $a + b + c = 1$ is not satisfied. Note that since G is Abelian, we (for example) consider $a + b + c = 1$ as being identical to $b + a + c = 1$. A maximum set of equations is a set of equations S such that if $S \rightarrow E$ then $E \in S$.

The proof of the theorem relies on the following lemma, which shows that $T(H)$ is a maximum set of equations of the type $a + b + c = 1$, where a, b, c are distinct variables. More precisely, the lemma shows that there exists no equation E of length 3 having exactly 3 distinct variables (and no constants) on the left-hand side and only the constant 1 on the right-hand side, such that $T(H) \rightarrow E$ and $E \notin T(H)$.

The importance of the property that $T(H)$ is a maximum set of equations (of type $a+b+c = 1$), lies in the observation that two maximum sets of equations are equivalent if and only if the sets are equal. We will see later that $T(H)$ would not necessarily be a maximum set of equations of type $a+b+c = 1$ for non-bipartite graphs H.

Lemma 16 *The following holds for every triple a, b, c of distinct variables in $T(H)$: If $T(H) \rightarrow a + b + c = 1$, then $a + b + c = 1 \in T(H)$.*

PROOF. The proof is a minor modification of the proof of Lemma 27 in [3] and consists of a careful case analysis. Let $X = \{x_i \mid i \in V\}$, $Y = \{y_i \mid e_i \in E\}$, and $Z = \{z_i, z_i' \mid i \in V\}$. To give the proof more structure we assume that $a \leq b \leq c$, where $\leq$ is the following order on $X \cup Y \cup Z$:

$$x_1 < \ldots < x_n < y_1 < \ldots < y_m < z_1 < \ldots < z_n < z_1' < \ldots < z_n'.$$

With the aim of reaching a contradiction we assume that there exists a triple a, b, c of distinct variables in $T(H)$ such that $T(H) \rightarrow a+b+c=1$ and $a+b+c=1 \notin T(H)$. In each of the 10 possible cases that arise we show that there exists an assignment to the variables such that $T(H)$ is satisfied but $a+b+c=1$ is not satisfied.

(1) If a, b, and c are in X, then set all variables in X to 0, and all variables in Y to 1. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a+b+c=1$.

(2) If a, b are in X, and c is in Y, suppose that $c=y_k$ and let $e_k=\{i,j\}$. By the assumption that $a+b+c=1$ is not in $T(H)$, at least one of a and b is not in $\{x_i, x_j\}$. Without loss of generality, let $a \notin \{x_i, x_j\}$. Set a to 1 and set $X \setminus \{a\}$ to 0. Note that such an assignment will force $y_k=c$ to be set to 1. This assignment can be extended to an assignment satisfying $T(H)$ but not $a+b+c=1$.

(3) If a, b are in X, and c is in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G|-1$ (i.e., $m-1$ if the group is $\mathbb{Z}_m$) to every variable in Y. This assignment satisfies $T(H)$ but not $a+b+c=1$.

(4) If a is in X, and b, c are in Y, then set all variables in X to 0, and all variables in Y to 1. This can be extended to a satisfying assignment to $T(H)$ but not to $a+b+c=1$.

(5) If a is in X, b is in Y, and c is in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G|-1$ to every variable in Y. This assignment satisfies $T(H)$ but not $a+b+c=1$.

(6) If a is in X, and b, c are in Z, then consider the assignment that assigns 0 to a, b, and c. We know by assumption that $a+b+c=1 \notin T(H)$, hence this assignment can be extended to an assignment on $X \cup Z$ satisfying all equations in $T(H)$ of the form $x_i + z_i + z_i' = 1$. Now since all remaining equations in $T(H)$ contain a variable from Y and no variable in Y occurs in more than one equation, it is easy to see that this assignment can be extended to an assignment that satisfies $T(H)$ but not $a+b+c=1$.

(7) If a, b, and c are in Y, then consider the assignment that assigns 0 to all variables in Y. This assignment can be extended to a satisfying assignment to $T(H)$ but not to $a+b+c=1$. Note that the fact that the assignment can be extended follows from the bipartiteness of the graph. A two coloring of the graph gives an assignment that assigns 1 to exactly one of $\{x_i, x_j\}$ for each $e_k=\{i,j\}$ and 0 to all other elements of X.

(8) If a, b are in Y, and c is in Z, then consider the assignment that assigns 2 to c (0 to c if the group is $\mathbb{Z}_2$), 1 to every variable in X, and $|G|-1$ to every variable in Y. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a+b+c=1$.

(9) If a is in Y, and b, c is in Z, let $a=e_k$ where $e_k=\{i,j\}$. If $\{b,c\}=\{z_l, z_l'\}$ for some l, then set x_l to 1. In all cases, set b and c to 0, and for all $e_k=\{i,j\} \in Y$ set exactly one of $\{x_i, x_j\}$ to 1 (this could be x_l). Set all

other elements of X to 0, and all elements of Y to 0. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a+b+c=1$. Note again that the existence of such an assignment to the variables in X follows from the fact that the graph is bipartite.

(10) If a, b, and c are in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G|-1$ to every variable in Y. This assignment satisfies $T(H)$ but not $a+b+c=1$.

Hence if $T(H) \to a+b+c=1$, then $a+b+c=1 \in T(H)$. $\square$

The importance of the fact that $T(H)$ is a maximum set of equations of the form $a+b+c=1$ lies in the observation that $T(H_1) \equiv T(H_2)$ if and only if $T(H_1)=T(H_2)$. Note again that since G is Abelian we consider $a+b+c=1$ as being identical to $b+c+a=1$, and so on. Hence $T(H_1)$ is isomorphic to $T(H_2)$ if and only if there exists a permutation ρ of the variables in $T(H_1) \cup T(H_2)$ such that $\rho(T(H_1)) = T(H_2)$.

Let $H_1 = (V, E)$ and $H_2 = (V, E')$ be two bipartite graphs, with an equal number of vertices and edges, where $V = \{1, 2, \ldots, n\}$, $E = \{e_1, e_2, \ldots, e_m\}$, and $E' = \{e'_1, e'_2, \ldots, e'_m\}$. We will prove that there exists an isomorphism π from H_1 to H_2 if and only if there exists an isomorphism ρ from $T(H_1)$ to $T(H_2)$.

Given an isomorphism π from H_1 to H_2, we construct an isomorphism ρ from $T(H_1)$ to $T(H_2)$ as follows. For each i in V, let $\rho(x_i) = x_{\pi(i)}$, $\rho(z_i) = z_{\pi(i)}$, and $\rho(z'_i) = z'_{\pi(i)}$. For each $e_k = \{i, j\}$ in E, let $\rho(y_k) = y_l$ where $e'_l = \{\pi(i), \pi(j)\}$.

To prove the remaining implication, we first need to make some observations about the different types of variables. If we are given an isomorphism ρ from $T(H_1)$ to $T(H_2)$, then as we have already observed $\rho(T(H_1)) = T(H_2)$. Now consider the following distinguishing properties of the variables.

- Variables from X are the only variables that occur at least twice. So, ρ will be a bijection on X.
- Variables from Z are those variables that occur exactly once and that occur together with another variable that occurs exactly once. Hence, ρ will be a bijection also on Z.
- Since ρ is a bijection on both X and Z, it will of course have to be a bijection also on the variables in Y.

So, given ρ we let $\pi(i) = j$ if and only if $\rho(x_i) = x_j$. Now we must prove that $\{i, j\} \in E$ if and only if $\{\pi(i), \pi(j)\} \in E'$. If $e_k = \{i, j\}$, then we know that $x_i + x_j + y_k = 1$ is in $T(H_1)$. Hence, $\rho(x_i) + \rho(x_j) + \rho(y_k) = 1$ is in $T(H_2)$, i.e., $x_{\pi(i)} + x_{\pi(j)} + \rho(y_k) = 1$ is in $T(H_2)$. Thus we must have $\rho(y_k) = y_l$, where $e'_l = \{\pi(i), \pi(j)\}$, so $\{\pi(i), \pi(j)\}$ is an edge in E'. On the other hand, if $\{\pi(i), \pi(j)\}$ is an edge in E', then $x_{\pi(i)} + x_{\pi(j)} + y_l = 1$ is in $T(H_2)$ (where $e'_l = \{\pi(i), \pi(j)\}$). This gives us that $x_i + x_j + \rho^{-1}(y_l) = 1$ must be in $T(H_1)$. Since $\rho^{-1}(y_l) = y_k$, where $e_k = \{i, j\}$, we conclude that $\{i, j\} \in E$. Thus π is an isomorphism from H_1 to H_2 if and only if ρ is an isomorphism from $T(H_1)$ to $T(H_2)$. This completes our proof that GRAPH ISOMORPHISM is polynomial-time many-one reducible to ISO-EQN$^*_G$ where G is a finite cyclic group.

By using the fundamental theorem of finitely generated Abelian groups it is possible to extended this result to all finite Abelian groups. Note that all finite Abelian groups are of course finitely generated.

Lemma 17 ([9]) *Every finite Abelian group G is isomorphic to a direct product of cyclic groups in the form $\mathbb{Z}_{p_1^{r_1}} \times \mathbb{Z}_{p_2^{r_2}} \times \ldots \times \mathbb{Z}_{p_n^{r_n}}$, where the p_i's are (not necessarily distinct) primes, and $|G| = p_1^{r_1} \cdot \ldots \cdot p_n^{r_n}$.*

Thus given a system of equations S_1 over a finite Abelian group G, we can use the fundamental theorem of finitely generated Abelian groups to view it as n independent systems of equations, each over a cyclic group $\mathbb{Z}_{p_i^{r_i}}$. Given a group $G \cong \mathbb{Z}_{p_1^{r_1}} \times \mathbb{Z}_{p_2^{r_2}} \times \ldots \times \mathbb{Z}_{p_n^{r_n}}$, denote the element $1_{p_1^{r_1}} \times 1_{p_2^{r_2}} \times \ldots \times 1_{p_n^{r_n}}$ in G by 1_G, where $1_{p_i^{r_i}}$ denotes the element 1 in $\mathbb{Z}_{p_i^{r_i}}$. Let $T(H_1)'$ and $T(H_2)'$ denote the systems of equations over G obtained from $T(H_1)$ and $T(H_2)$ by replacing all occurrences of 1 by 1_G.

We have proved above that there exists an isomorphism π from H_1 to H_2 if and only if there exists an isomorphism ρ from $T(H_1)$ to $T(H_2)$ regardless of which finite cyclic group the systems of equations are defined over. Moreover, if ρ is an isomorphism from $T(H_1)$ to $T(H_2)$ then $\rho(T(H_1)) = T(H_2)$, and of course $\rho(T(H_1)') = T(H_2)'$ (i.e., $T(H_1)' \cong T(H_2)'$). Conversely, if there exists an isomorphism ρ' from $T(H_1)'$ to $T(H_2)'$ (i.e., $\rho'(T(H_1)') \equiv T(H_2)'$), then $\rho'(T(H_1)) \equiv T(H_2)$ (for all $\mathbb{Z}_{p_i^{r_i}} \in \{\mathbb{Z}_{p_1^{r_1}}, \mathbb{Z}_{p_2^{r_2}}, \ldots, \mathbb{Z}_{p_n^{r_n}}\}$, and hence all finite cyclic groups).

Hence given a finite Abelian group G and two graphs H_1 and H_2, then H_1 is isomorphic to H_2 if and only if the two systems of equations $T(H_1)'$ and $T(H_2)'$ over G are isomorphic. $\square$

Despite intensive research GRAPH ISOMORPHISM is not known to be in coNP. Hence, in light of the previous theorem, it seems hard to prove that ISO-EQN$^*_G$ is in coNP when G is an Abelian group.

If we restrict the isomorphism problem and require that all equations are of bounded length, then we can prove an even tighter correspondence between the complexity of Iso-Eqn$^*_{G,k}$ and Graph Isomorphism.

Theorem 18 Iso-B-Eqn$^*_{G,k}$ *is* Graph Isomorphism*-complete under polynomial-time many-one reductions when G is an Abelian group (different from the trivial one) and $k \geq 3$.*

PROOF. It follows from the proof of Theorem 15 that Graph Isomorphism is polynomial-time many-one reducible to Iso-B-Eqn$^*_{G,k}$ for Abelian groups G and $k \geq 3$. Hence what remains is to give a polynomial-time many-one reduction from Iso-B-Eqn$^*_{G,k}$ to Graph Isomorphism. We will actually give a reduction from Iso-B-Eqn$^*_{G,k}$ to the Graph Isomorphism-complete problem vertex colored graph isomorphism (VCGI) [7]. Our reduction is a minor modification of the reduction in the proof of Claim 22 from Böhler et al. [2] and our proof closely follows theirs.

Definition 19 *VCGI is the problem of, given two vertex-colored graphs $H_1 = (V_1, E_1, \chi_1)$, $H_2 = (V_2, E_2, \chi_2)$, with $\chi_1, \chi_2 \colon V \to \mathbb{N}$, to determine whether there exists an isomorphism from H_1 to H_2 that preserves colors, i.e., whether there exists a bijection $\pi : V_1 \to V_2$ such that $\{v, v'\} \in E_1$ if and only if $\{\pi(v), \pi(v')\} \in E_2$, and $\chi_1(v) = \chi_2(\pi(v))$ for all $v \in V_1$.*

Let S_1 and S_2 be two systems of equations of bounded length over an Abelian group G on variables X. The case where at least one of S_1 and S_2 are not solvable is trivially reducible to Graph Isomorphism. Hence we can assume that both S_1 and S_2 are solvable. We will first bring S_1 and S_2 into normal form. We choose an approach similar to that in Theorem 15 and reduce S_1 and S_2 to the maximum set of equations of length at most k that can be inferred from S_1 and S_2, respectively. Since both S_1 and S_2 are solvable we can assume that all equations only containing constants have been removed. Moreover, we assume that all equations have been reduced in such a way that they contain no constants from G except for in the right-most position on the right-hand side (if an equation should lack a constant we add 0 to it, where 0 denotes the identity in G). Since the group is Abelian this reduction is trivial and the resulting system of equations is of course equivalent to the original one. We call equations of this particular form reduced equations.

Let $\langle S_1 \rangle$ denote the set of all reduced equations E of length at most k such that all of E's variables occur in X and $S_1 \to E$. It should be clear that S_1 is equivalent to $\langle S_1 \rangle$, since $S_1 \subseteq \langle S_1 \rangle$ and $S_1 \to \langle S_1 \rangle$. We define $\langle S_2 \rangle$ analogously. Note that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time (in $|S_1| + |S_2|$), since there exist at most $O(|X|^k)$ reduced equations E of length at most k such that all of E's variables occur in X, and since G is Abelian we can use the polynomial-time algorithm for $\textsc{Eqn}^*_G$ to decide whether $S \to E$. Note that $S \to E$ if and only if for every assignment to the variables in E such that E is not satisfied, this assignment applied to S makes S insoluble. There is at most $O(|G|^k)$ possible assignments of the variables in E. Testing whether or not such an assignment makes S insoluble can be done in polynomial time (by the result in [6]). So, it follows that $S \to E$ can be decided in polynomial time. It should be clear that if $\langle S_1 \rangle$ is equivalent to $\langle S_2 \rangle$ then $\langle S_1 \rangle = \langle S_2 \rangle$, hence if π is a permutation of the variables in X such that $\pi(\langle S_1 \rangle) \equiv \langle S_2 \rangle$, then $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$.

Now we proceed and reduce $\langle S_1 \rangle$ and $\langle S_2 \rangle$ to vertex-colored graphs H_1 and H_2 such that $\langle S_1 \rangle \cong \langle S_2 \rangle$ if and only if $(H_1, H_2) \in VCGI$.

Let $\langle S_1 \rangle$ consist of the equations $A_1, \ldots, A_m$, i.e.,

$$\begin{array}{rcll} x_{11} + x_{12} + \ldots + x_{1j_1} & = & x_{1j_1+1} + \ldots + x_{1k_1} + c_1 & (A_1) \\ x_{21} + x_{22} + \ldots + x_{2j_2} & = & x_{2j_2+1} + \ldots + x_{2k_2} + c_2 & (A_2) \\ & \vdots & & \\ x_{m1} + x_{m2} + \ldots + x_{mj_m} & = & x_{mj_m+1} + \ldots + x_{mk_m} + c_m & (A_m) \end{array}$$

Let $T(\langle S_1 \rangle) = H_1 = (V, E, \chi)$ be the following vertex-colored graph:

- $V = \{g_0, g_1, \ldots, g_{|G|-1}\} \cup \{x \mid x \in X\} \cup \{a_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq k_i + 1\} \cup \{A_i \mid 1 \leq i \leq m\}$. Hence, the set of vertices corresponds to the elements in G, the variables in X, the elements in the equations in $\langle S_1 \rangle$, and the equations in $\langle S_1 \rangle$.
- $E = \{\{x, a_{ij}\} \mid x = a_{ij}\} \cup \{\{c, a_{ij}\} \mid c = a_{ij}\} \cup \{\{a_{ij}, A_i\} \mid 1 \leq i \leq m, 1 \leq j \leq k_i + 1\}$. Thus, the edges indicate which variables and constants that occur in a given equation.
- The vertex coloring χ is used to distinguish the different categories of vertices. Of course, we must allow any permutation of the variables, so all vertices corresponding to variables in X are assigned the same color. We also need to allow the permutation of vertices corresponding to equations, and because the group is Abelian we allow the permutation of vertices corresponding to elements within the same side of each equation.
  - $\chi(g_i) = i$, i.e., if $c_t = g_i$ then $\chi(c_t) = i$,
  - $\chi(x) = |G|$ for all $x \in X$,
  - $\chi(A_r) = |G| + 1$,
  - $\chi(a_{ij}) = |G| + 2$ if $j \leq j_i$,
  - $\chi(a_{ij}) = |G| + 3$ if $k_i \geq j > j_i$,
  - $\chi(a_{ij}) = |G| + 4$ if $j = k_i + 1$.

Define $T(\langle S_2 \rangle) = H_2$ in the analogous way.

Given a permutation π of X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$, then it is easily realized that $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$. For the converse, given a permutation π on $T(\langle S_1 \rangle)$ witnessing the fact that $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$, then we have $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. Just note that any permutation of vertices corresponding to equations forces a permutation of the vertices corresponding to elements in the equations. Thus if $\pi(A_i) = A_j$ then we must have $\pi(a_{it}) = a_{js}$, for all $1 \leq t \leq k_i + 1$ and some $1 \leq s \leq k_j + 1$. Also note that because of the coloring, π is the identity mapping on vertices corresponding to constants. Hence, the only thing left is the permutation of vertices corresponding to variables in X. Thus we have $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. $\square$

Using the same idea as in the proof of the preceding theorem we can prove a tighter upper bound for $\textsc{Iso-B-Eqn}^*_{G,k}$, compared to the trivial Σ_2^{P} upper bound for non-Abelian groups from Theorem 12. More specifically we prove that $\textsc{Iso-B-Eqn}^*_{G,k}$ for non-Abelian groups is in $\mathrm{P}_{||}^{\mathrm{NP}}$, the class of problems solvable in polynomial time with parallel (i.e., truth-table) access to an NP-oracle. This result is analogous to Corollary 23 in [2] and the proof is the same.

Theorem 20 $\textsc{Iso-B-Eqn}^*_{G,k}$ *is in* $\mathrm{P}_{||}^{\mathrm{NP}}$.

PROOF. Let S_1 and S_2 be two systems of equations on variables X over a non-Abelian group G. We use the same normal form as in the proof of the preceding theorem, i.e., we compute $\langle S_1 \rangle$ and $\langle S_2 \rangle$. Following the proof of the preceding theorem it is easy to verify that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time with parallel access to an NP-oracle. Now the existence of a permutation π of the variables in X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$ can be determined with one query to an NP-oracle. Hence, it can be decided in polynomial time with two rounds of parallel queries to NP whether S_1 and S_2 are isomorphic, and thus by the results in [5] it follows that $\textsc{Iso-B-Eqn}^*_{G,k}$ is in $\mathrm{P}_{||}^{\mathrm{NP}}$. $\square$

Böhler et al. [4] have proved that the isomorphism problem for systems of equations having length at most 2 over the group $\mathbb{Z}_2$ can be solved in polynomial time. We prove that the isomorphism problem for systems of equations of length at most 2 is in P for all groups.

Theorem 21 $\textsc{Iso-Eqn}^*_{G,k}$ *is in* P *for all groups* G *when* $k \leq 2$.

PROOF. We know from the proof of Theorem 11 that we can check in polynomial time whether systems of equations of length at most 2 are solvable. If neither S_1 nor S_2 are solvable, then they are of course isomorphic, and if just one of S_1 and S_2 are solvable, then they are not isomorphic. Hence we can assume that both S_1 and S_2 are solvable.

Remember from the proof of Theorem 11 that we can represent the system of equations S_i, $i \in \{1, 2\}$, as a graph G_i with one vertex for each variable in S_i and an edge between any pair of vertices whose corresponding variables occur in the same equation in S_i. It should be clear that the connected components in G_i correspond to independent subsystems of S_i. Note that since S_i has a solution, so do each of these independent subsystems of S_i. In fact we know, from the proof of Theorem 11, how to compute in polynomial time all the solutions of each of these subsystems and that they have at most $|G|$ solutions each. Moreover, for any two different solutions α and β of such a subsystem, $\alpha(x) \neq \beta(x)$ hold for all variables x in the subsystem. As we will see, this is the key of the proof.

We begin by treating subsystems having a unique solution. Let X_1 and X_2 be the set of all variables occurring in subsystems, of S_1 and S_2 respectively, having a unique solution. Let α_1 and α_2 be the partial solutions, to S_1 and S_2 respectively, that are induced by the subsystems having a unique solution. It is easy to check whether there exists a bijective mapping π_{FIXED} from X_1 to X_2 such that for all variables $x \in X_1$, if $\pi_{FIXED}(x) = x'$ then $\alpha_1(x) = \alpha_2(x')$. If no such mapping π_{FIXED} exists, then clearly $S_1 \ncong S_2$. On the other hand, if $S_1 \cong S_2$, then there exists an isomorphism π from S_1 to S_2 such that π and π_{FIXED} agree on the variables in X_1. Hence, we can assume that each independent subsystem of S_1 and S_2 have at least 2 solutions.

Next we show how to check in polynomial time whether two independent subsystems, IS_1 and IS_2 corresponding to connected components of G_1 and G_2 respectively, are isomorphic. Let Y_1 and Y_2 be the set of variables that occur in IS_1 and IS_2 respectively. If IS_1 and IS_2 have a different number of solutions or $|Y_1| \neq |Y_2|$, then $IS_1 \ncong IS_2$. So we assume that IS_1 and IS_2 have the same number of variables and solutions. Represent the m ($m \leq |G|$) solutions of IS_1 as a $m \times |Y_1|$ matrix M_1 where the rows of M_1 are just the solutions of IS_1. Define the matrix M_2, corresponding to the solutions of IS_2, in the analogous way. Now, if there exists a permutation of the rows and columns of M_1 (yielding a matrix M_1') such that $M_1' = M_2$, then clearly $IS_1 \cong IS_2$. If no such permutation exists, then it is easy to see that $IS_1 \ncong IS_2$. The existence of such a permutation can be determined in polynomial time since the number of permutations of the rows in M_1 is bounded by the constant $|G|!$. After each such permutation it can be checked whether the resulting matrix M_1'' admits a permutation of the columns, yielding a matrix M_1' such that $M_1' = M_2$ (e.g., by sorting the columns in M_1'' and M_2 in lexicographical order and checking for identity).

Now, we prove that if there exists an independent subsystem IS_1 of S_1 (corresponding to a connected component of G_1) such that IS_1 is not isomorphic to any of the independent subsystems of S_2 (corresponding to connected components of G_2), then $S_1 \not\cong S_2$. Assume the contrary, i.e., that IS_1 is not isomorphic to any of the independent subsystems of S_2 but there exists a permutation π such that $\pi(S_1) \equiv S_2$. Two cases emerge, either the variables in IS_1 are mapped, by π, to the variables occurring in a single independent subsystem of S_2, or they are mapped to variables from more than one independent subsystem in S_2.

If the variables in IS_1 are mapped, by π, to the variables in a single subsystem, IS_2, of S_2, then there must be a variable x in IS_2 that does not occur in $\pi(IS_1)$ (otherwise we get a contradiction with the assumption that $IS_1 \not\cong IS_2$ and $\pi(S_1) \equiv S_2$). Now, if we assign a value a to a variable y in $\pi(IS_1)$, then in IS_2, x will be forced to take a value b. By the assumption that $\pi(S_1) \equiv S_2$, we have that x must occur in a subsystem of S_1 (having at least two solutions) that is independent of IS_1. So $\pi(S_1)$ has a solution where a is assigned to y and c is assigned to x, where $c \neq b$, again contradicting $\pi(S_1) \equiv S_2$.

In the second case $\pi(IS_1)$ contains variables from two different independent subsystems of S_1. Any solution of $\pi(IS_1)$ forces a solution to these independent subsystems of S_2. But again since these subsystems of S_2 have more than one solution each, there exists a solution to S_2 not satisfying $\pi(S_1)$ (contradicting $\pi(S_1) \equiv S_2$).

The outline of the polynomial-time algorithm for solving $\textsc{Iso-Eqn}^*_{G,2}$ should now be clear. For each independent subsystem, IS_1 of S_1 corresponding to a connected component in G_1, check whether there exists an independent subsystem, IS_2 of S_2 corresponding to a connected component in G_2, such that $IS_1 \cong IS_2$. If no such subsystem IS_2 exists, then we conclude that $S_1 \not\cong S_2$. Otherwise remove IS_1 and IS_2 from S_1 and S_2 respectively, and continue with the next independent subsystem of S_1. When all independent subsystems of S_1 corresponding to connected components of G_1 have been checked, without reaching the conclusion that $S_1 \not\cong S_2$, then we know that $S_1 \cong S_2$. □

4 Counting Isomorphisms

Mathon [10] showed that the counting version of Graph Isomorphism is polynomial time Turing reducible to the decision version. Thus, Graph Isomorphism behaves differently than the known NP-complete problems, because their counting versions seems much harder than their decision versions.

This was historically the first hint that Graph Isomorphism might not be NP-complete. We prove analogous results for #Iso-Eqn$^*_G$.

Given a system of equations S on variables X over a finite group G, we are interested in the set of permutations π of the variables in X such that $\pi(S) \equiv S$. It is easy to see that this set of permutations forms a group, denoted $aut(S)$, the automorphism group of S. #Aut-Eqn$^*_G$ is the problem of counting the number of automorphisms of a system of equations S over G. That is, computing $|aut(S)|$.

Lemma 22 *#*Aut-Eqn$^*_G$ *is Turing reducible to* Iso-Eqn$^*_G$*.*

PROOF. First note that we can determine whether a system of equations S is solvable or not by making a single query to Iso-Eqn$^*_G$. If S is insoluble, then of course $|aut(S)| = |X|!$, where X is the set of variables occurring in S.

We call variables x_i and x_j equivalent with respect to S if, for any assignment α that satisfies S, we have $\alpha(x_i) = \alpha(x_j)$. By $E_S(x_i)$ we denote the set of variables in S that are equivalent to x_i. Consider any automorphism $\rho \in aut(S)$, if ρ maps x_i to x_k, then clearly $\rho(E_S(x_i)) = E_S(x_k)$; moreover, each bijection from $E_S(x_i)$ to $E_S(x_k)$ can be extended to an automorphism of S. Note that when S is a system of equations over an Abelian group, then the sets $E_S(x_i)$ for all x_i in X, can be computed in polynomial time: just check whether or not $S \equiv S \cup \{x_i = x_j\}$ for all x_j in X (remember that by Corollary 7 we know that equivalence of systems of equations over Abelian groups is in P). In the case where S is a system of equations over a non-Abelian group we can use an NP-oracle to check whether or not $S \equiv S \cup \{x_i = x_j\}$ for all x_j in X. Since Iso-Eqn$^*_G$ for non-Abelian groups is coNP-hard, we can in particular use this method when we have Iso-Eqn$^*_G$ as an oracle.

The notion of equivalence of variables is an equivalence relation on X. For each equivalence class choose a variable in that class, say x_i, and replace all occurrences of the variables from $E_S(x_i)$ by x_i in S. The resulting system of equations S' is defined on the set of variables I corresponding to the equivalence classes of X. The number of permutations ρ of X such that $\rho(S) \equiv S$ is equal to

$$|aut(S')| \prod_{x_i \in I} |E_S(x_i)|!.$$

Hence what remains to be done is to compute $|aut(S')|$, i.e., the number of permutations π of I such that $\pi(S') \equiv S'$.

We need a construction that forces a variable to be mapped to itself under any permutation ρ of I, such that $\rho \in aut(S')$. We use the same construction as the one used in the proof of [1, Lemma 5.1] to achieve this goal.

Note that ρ always maps x_i to a variable x_k such that $|E_{S'}(x_i)| = |E_{S'}(x_k)|$. The idea is to make $|E_{S'}(x_i)|$ unique by introducing new equations of the form $x_i = z_{ij}$, where z_{ij} is a new variable called a labelling variable. Recall that S' was constructed from S by eliminating all equivalent variables. Hence, for each x_i in $I = \{x_1, \dots, x_n\}$ we have $|E_{S'}(x_i)| = 1$. Let $L(x_i)$ denote $\bigcup_{j=1}^{i}\{x_i = z_{ij}\}$, and $S'_{[I]}$ the system of equations that is constructed as follows,

$$S'_{[I]} = S' \cup \bigcup_{x_i \in I} L(x_i).$$

Now with the technicalities of the labelling out of the way, we can proceed to compute $|aut(S')|$. Let $S'_{[x_1,\dots,x_n]}$ be the system of equations (corresponding to S') where all variables have been labelled in the way described above. Let $aut_{id}(S'_{[I]})$ denote the set (group) of automorphisms of $S'_{[I]}$ that acts as the identity mapping on all labelling variables. Hence, $|aut_{id}(S'_{[x_1,\dots,x_n]})| = 1$ and $aut_{id}(S') = aut(S')$. The key for computing $|aut(S')|$ lies in the fact that $|aut_{id}(S'_{[x_1,\dots,x_{i-1}]})| = d_i|aut_{id}(S'_{[x_1,\dots,x_{i-1},x_i]})|$, where d_i is the size of the orbit of x_i under the action of $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$.

Recall that the orbit of x_i under the action of $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$ is the set $\{\rho(x_i) \mid \rho \in aut_{id}(S'_{[x_1,\dots,x_{i-1}]})\}$. Let d_i be the size of the orbit o_i of x_i under the action of $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$, and let ϕ_k $(1 \leq k \leq d_i)$ be an automorphism in $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$ which maps x_i to the k-th variable in o_i. Every $\rho \in aut_{id}(S'_{[x_1,\dots,x_{i-1}]})\}$ can be decomposed as a product of a unique $\phi \in \{\phi_1, \dots, \phi_{d_i}\}$ and a unique $\tau \in aut_{id}(S'_{[x_1,\dots,x_{i-1},x_i]})$. Hence, $|aut_{id}(S'_{[x_1,\dots,x_{i-1}]})| = d_i|aut_{id}(S'_{[x_1,\dots,x_{i-1},x_i]})|$. It should be clear that $|aut(S')| = d_1 d_2 \dots d_n$. The fact that the order of a permutation group can be computed from the size of the orbits in the way described above is a standard result from group theory, cf. [7] for details.

The orbit of x_i under $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$ can be found by making $n - i$ queries to Iso-Eqn$^*_G$. Ask the query $S'_{[x_1,\dots,x_{i-1}]}[x_i] \cong S'_{[x_1,\dots,x_{i-1}]}[x_j]$ for each variable x_j, $j \geq i$, where $S'_{[x_1,\dots,x_{i-1}]}[x_i]$ and $S'_{[x_1,\dots,x_{i-1}]}[x_j]$ denotes systems of equations where x_i and x_j have been assigned the same (unique) label (using the method described above). If the answer is yes, we know that x_j is in the orbit of x_i under $aut_{id}(S'_{[x_1,\dots,x_{i-1}]})$. Hence $|aut(S')| = d_1 d_2 \dots d_n$ can be computed by making $O(n^2)$ queries to Iso-Eqn$^*_G$. This completes the proof of the fact that #Aut-Eqn$^*_G$ is Turing reducible to Iso-Eqn$^*_G$. □

The following theorem states that it is no harder to count the number of isomorphisms between two systems of equations over a fixed finite group than to decide whether an isomorphism exists at all. This indicates that Iso-Eqn$^*_G$ is not NP-complete for Abelian groups, and that Iso-Eqn$^*_G$ is not Σ_2^{P}-complete for non-Abelian groups.

Theorem 23 *$\#$*Iso-Eqn$^*_G$ *is Turing equivalent to* Iso-Eqn$^*_G$.

PROOF. The proof is based on the same principle as the analogous proof for Graph Isomorphism due to Mathon [10]. For Mathon's arguments to work, we need to make sure that all variables in X occur in both S_1 and S_2. This can be easily achieved by padding S_1 (S_2) with (dummy) equations of the form $x = x$, for all variables x that are in X but do not occur in S_1 (S_2). Denote the padded systems by S_1' and S_2' respectively. It should be clear that the number of permutations π of X such that $\pi(S_1) \equiv S_2$ equals the number of permutations π of X such that $\pi(S_1') \equiv S_2'$. Hence we will assume from now on that all variables in X occur both in S_1 and S_2.

Of course Iso-Eqn$^*_G$ is trivially polynomial-time reducible to #Iso-Eqn$^*_G$. The rest of the proof will be spent on establishing a polynomial-time Turing reduction from #Iso-Eqn$^*_G$ to Iso-Eqn$^*_G$. Let N_1 be the number of permutations π of X such that $\pi(S_1) \equiv S_2$. If S_1 is not isomorphic to S_2, i.e., there exists no permutation π of X such that $\pi(S_1) \equiv S_2$, then $N_1 = 0$. Otherwise, $N_1 = |aut(S_1)| = |aut(S_2)|$. This can be realized by noting that if π is a permutation such that $\pi(S_1) \equiv S_2$, and $\rho \in aut(S_1)$, then $\rho \circ \pi$ is also a permutation such that $\rho \circ \pi(S_1) \equiv S_2$ (note that $\rho \circ \pi$ denotes the composition where ρ is applied prior to π, hence $\rho \circ \pi(S_1) = \pi(\rho(S_1))$). In addition, any permutation π' such that $\pi'(S_1) \equiv S_2$ can be uniquely expressed as $\pi' = \rho' \circ \pi$, where $\rho' \in aut(S_1)$. Thus, the set of isomorphisms from S_1 to S_2 is the right coset, $aut(S_1) \circ \pi$, where π is an isomorphism from S_1 to S_2. Hence, it follows from Lemma 22 that #Iso-Eqn$^*_G$ is Turing reducible to Iso-Eqn$^*_G$. □

Mathon's algorithm for computing the number of isomorphisms of two graphs, has the interesting property that at each intermediate stage the number of isomorphisms of the labelled graphs are known. This property has been exploited to prove that Graph Isomorphism (GI) is low for PP, i.e., Graph Isomorphism is powerless as an oracle to PP ($\mathrm{PP}^{\mathrm{GI}} = \mathrm{PP}$) [7]. This is generally interpreted as further evidence for the hypothesis that Graph Isomorphism is not NP-complete. For the details and significance of lowness results, again consult [7]. The properties of our algorithm (that is based on Mathon's algorithm) for computing #Iso-Eqn$^*_G$ together with the same argument as that preceding Theorem 5.3 in [1] implies the following lowness results.

Corollary 24 $\mathrm{PP}^{\mathrm{IAb}} = \mathrm{PP}$, *and* $\mathrm{PP}^{\mathrm{IG}} = \mathrm{PP}^{\mathrm{NP}}$, *where* IAb *denotes* Iso-Eqn$^*_G$ *for Abelian groups and* IG *denotes* Iso-Eqn$^*_G$ *for non-Abelian groups.*

Hence Iso-Eqn$^*_G$ for Abelian groups is powerless as an oracle to PP, and Iso-Eqn$^*_G$ for non-Abelian groups is no more powerful than an NP-oracle for PP.

5 Conclusions

We give dichotomies (under the assumption that coNP≠P) for the complexity of EQUIV-EQN$^*_G$ and EQUIV-EQN$^*_{G,k}$ for all finite groups G and constants k. A natural direction for future research would be to prove similar dichotomies for EQUIV-EQN$^*_G$ and EQUIV-EQN$^*_{G,k}$ for all finite semigroups G and constants k. But, in light of the recent results in [8], we must say that these problems seem very challenging.

As for the complexity of ISO-EQN$^*_G$, the situation is not as clear. We prove that the problem is in Σ_2^{P} and coNP-hard in the non-Abelian case, and that it is in NP and GRAPH ISOMORPHISM-hard in the Abelian case. But the results in Theorem 23 and Corollary 24 give strong indications that these upper bounds (Σ_2^{P} and NP respectively) are not tight. For the isomorphism problem for systems of equations of bounded length, we prove that ISO-B-EQN$^*_{G,k}$ is in P when $k \leq 2$ (for all groups G). If $k \geq 3$, then ISO-B-EQN$^*_{G,k}$ is GRAPH ISOMORPHISM-complete when G is Abelian (and different from the trivial group), and coNP-hard and in $\mathrm{P}^{\mathrm{NP}}_{||}$ for all non-Abelian groups G.

Our results for the complexity of the equivalence and isomorphism problems for systems of equations over finite groups complement the complexity results for deciding solvability and counting solutions to systems of equations over finite groups presented in [6,12]. The equivalence and isomorphism problems for systems of equations over finite groups are natural special cases of the equivalence and isomorphism problems for constraints over finite domains. Hence, our results can also be seen as a first attempt of generalizing the results due to Böhler et al. [2,4] on the complexity of the equivalence and isomorphism problems for Boolean constraints.

It is interesting to observe that so many of the constructions and proof techniques previously used in the literature in relation to other isomorphism problems (e.g., [1,2,4,10]) can be reused. This seems to suggest that a more unified treatment of these and similar isomorphism problems is possible.

Another question left open by the present paper is that of the relative complexity of the isomorphism problems, e.g., it is far from clear whether or not there exists a polynomial-time reduction from ISO-EQN$^*_G$ to ISO-B-EQN$^*_{G,k}$ for some constant k.

Moreover, it has been proved by the use of interactive proofs, that GRAPH ISOMORPHISM and formula isomorphism are not NP-complete and Σ_2^{P}-complete respectively, unless the polynomial hierarchy collapses [1,7]. It would be interesting to investigate whether similar techniques can be used to prove analogous results for ISO-EQN$^*_G$.

Acknowledgements

The author thanks the anonymous referees for many valuable comments.

References

[1] M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM J. Comput.*, 30(3):990–1009, 2000.

[2] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *Proc. Ann. Conf. of the European Association for Computer Science Logic (CSL)*, pages 412–426, 2002.

[3] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. Technical report, arXiv:cs.CC/0202036, 2002.

[4] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. The complexity of boolean constraint isomorphism. In *Proc. 21st Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 164–175, 2004.

[5] S. Buss and L. Hay. On truth-table reducibility to SAT. *Inf. Comput.*, 90(2):86–102, 1991.

[6] M. Goldmann and A. Russel. The complexity of solving equations over finite groups. *Inf. Comput.*, 178(1):253–262, 2002.

[7] H. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhäuser, Boston, 1993.

[8] O. Klíma, P. Tesson, and D. Thérien. Dichotomies in the complexity of solving systems of equations over finite semigroups. *Electr. Colloq. on Computational Complexity (ECCC)*, 11(091), 2004.

[9] S. Lang. *Algebra.* Addison-Wesley, Reading MA, 1993.

[10] R. Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979.

[11] C. Moore, P. Tesson, and D. Thérien. Satisfiability of systems of equations over finite monoids. In *Proc. 26th Int. Symp. on Mathematical Foundations of Computer Science (MFCS)*, pages 537–547, 2001.

[12] G. Nordh and P. Jonsson. The complexity of counting solutions to systems of equations over finite semigroups. In *Proc. 10th Int. Computing and Combinatorics Conference (COCOON)*, pages 370–379, 2004.

[13] T. Thierauf. The isomorphism problem for read-once branching programs and arithmetic circuits. *Chicago J. Theor. Comput. Sci.*, 1998.