Syntactic Isomorphism of CNF Boolean Formulas is Graph Isomorphism Complete*

Giorgio Ausiello[1], Francesco Cristiano[1], Paolo Fantozzi[1] and Luigi Laura[2]

[1] Dip. di Informatica e Sistemistica Università di Roma "La Sapienza", Rome Italy.
[2] International Telematic University Uninettuno, Rome, Italy
E-mail: ausiello@dis.uniroma1.it, fra.cristiano@gmail.com,
paolo.fantozzi@uniroma1.it, luigi.laura@uninettunouniversity.net

Abstract. We investigate the complexity of the syntactic isomorphism problem of two Boolean Formulas in Conjunctive Normal Form (CNF): given two CNF Boolean formulas $\varphi(a_1,\ldots,a_n)$ and $\varphi(b_1,\ldots,b_n)$ decide whether there exists a permutation of clauses, a permutation of literals and a bijection between their variables such that $\varphi(a_1,\ldots,a_n)$ and $\varphi(b_1,\ldots,b_n)$ become syntactically identical. We first show that the CNF Syntactic Formulas Isomorphism (CSFI) problem is polynomial time reducible to the graph isomorphism problem (GI) and then we show that GI is polynomial time reducible to a special case of the CSFI problem (MCSFI) that is CSFI-complete and also **GI**-complete, thus concluding that the syntactic isomorphism problem for CNF Boolean formulas is **GI**-complete. Finally we observe that the same results hold when considering DNF Boolean formulas (DSFI).

Keywords: Boolean isomorphism · Complexity theory · Graph isomorphism · Boolean formulas · Semantic isomorphism · Syntactic isomorphism.

1 Introduction

A Boolean function of arity n $f = f(x_1,\ldots,x_n)$ is a function $f : \{0,1\}^n \to \{0,1\}$. The truth table of a Boolean function fully specifies the function but it does not provide any information about its syntactic representation; on the opposite side, a Boolean formula is a syntactic representations of a Boolean function but such representation is not unique since each Boolean function may be represented by a set of Boolean formulas. Calling *equivalent* the Boolean formulas that represent the same function, we have that a Boolean function is defined by a Boolean formula modulo logical equivalences.

Given two Boolean formulas F and G, representing respectively the Boolean functions f and g, the *Formula Equivalence* (FE) problem is to decide whether

* Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). This work was partially supported by MIUR National Interest Project "AlgoDEEP: Algorithmic Challenges for Data-intensive Processing on Emerging Computing Platforms"

they are semantically (or logically) equivalent (i.e., if and only if each model of F is a model of G and vice versa, or, in other terms, $f = g$ [14]), that is $F \equiv G$ [20].

Given two Boolean functions, represented by the Boolean formulas F and G defined over the variables set $\{x_1, ..., x_n\}$, a semantic isomorphism λ is a permutation of the variables of G such that G becomes logical equivalent to F i.e. such that $F \equiv G \circ \lambda = G(\lambda(x_1), ..., \lambda(x_n))$. For example the Boolean formulas $x \wedge \neg y$ and $\neg x \wedge y$ are semantically isomorphic, since we can swap x and y in the first formula to get the second, but they are not semantically equivalent (it suffices to build the truth table in order to see this). The problem of deciding whether two Boolean formulas are semantically isomorphic is called the *Formula Isomorphism* (FI) problem [1, 2]. From the definitions of Formula Equivalence and Formula Isomorphism problems it follows that two semantically equivalent Boolean formulas are also semantically isomorphic since the semantic equivalence relationship preserves the semantic isomorphism.

The Formula Isomorphism problem has been widely studied by Agrawal and Thierauf showing that, though FI is in $\Sigma_2\mathbf{P}$, i.e. the second level of the polynomial hierarchy, it cannot be $\Sigma_2\mathbf{P}$-complete unless the polynomial time hierarchy collapses [1, 2]. In Thierauf's work [20], where the author considers several problems related to equivalence and isomorphism, it is shown that Graph Isomorphism (GI) is polynomial time reducible to FI.

A different notion from the semantic isomorphism is the syntactic isomorphism, i.e. a permutation λ of the variables $\{x_1, ..., x_n\}$ of G such that it becomes syntactically identical to F, i.e., $F = G \circ \lambda = G(\lambda(x_1), ..., \lambda(x_n))$. It is straightforward to verify that each syntactic isomorphism is also a semantic isomorphism, since it is a permutation of variables that leads two Boolean formulas to be equivalent, but not the converse. By definition, we have that literals are variables and negated variables, terms are conjunction of literals and clauses are disjunction of literals. Recalling that each Boolean function can be represented as a disjunction of terms, called Disjunctive Normal Form (DNF), or as a conjunction of clauses, called Conjunctive Normal Form (CNF), we say that two Boolean formulas represented in CNF (or DNF) are syntactically isomorphic if and only if they can be written in identical way under a suitable permutation of variables, literals and clauses (terms); e.g., the Boolean formulas, $x \wedge \neg y$ and $\neg x \wedge y$ are syntactical isomorphic since we can swap x and y in the first formula, and we can permute the two literals, getting $\neg x \wedge y$.

In this paper we focus on the Syntactic Isomorphism of CNF Boolean Formulas (CSFI): given two CNF Boolean formulas $\varphi(a_1, \ldots, a_n)$ and $\varphi(b_1, \ldots, b_n)$, decide whether there exist a permutation of clauses, a permutation of literals, and a bijection between their variables such that $\varphi(a_1, \ldots, a_n)$ and $\varphi(b_1, \ldots, b_n)$ become syntactically identical. We prove that this problem is **GI**-complete: in particular, we i) show that the CSFI problem is polynomial time reducible to the graph isomorphism problem (GI, that is solvable in quasi-polynomial time as in [5]) and ii) show that a special case of the CSFI problem, limited to monotone formulas Monotone CNF Syntactic Formula Isomorphism (MCSFI), is both

CSFI-complete and **GI**-complete; combining these results it holds that the syntactic isomorphism problem of CNF Boolean formulas is **GI**-complete.

The end results of our findings are twofold. First, the GI-completeness of CSFI supports the existence of a "graph theoretic independent" isomorphism class, as already observed by Fortin [11] about the Term Equality Problem (TEP) [6]; furthermore, it is interesting to observe that CSFI represents a special (and easier) case of TEP and, as a consequence of our results, TEP reduces to CSFI. The CSFI problem is peculiar amongst the graph isomorphism-complete problems in that i) it regards a subclass of Boolean formulas and so it is not defined in terms of a graph theoretic problem, ii) it is a special case of a more general isomorphism problem that is FI; therefore all **GI**-complete problems (either of a graph theoretic nature or not) can be reduced to a subset of FI.

Second, our result increases, in some sense, the similarities between the Formula Isomorphism and the Graph Isomorphism problems:

1. a) If GI is NP-complete, then the polynomial hierarchy collapses to its second level [19].
   b) If FI is $\mathrm{NP}^{\mathrm{NP}}$-complete, then the polynomial hierarchy collapses to its third level [1].
2. a) The counting version of GI can be reduced to its decision version [15].
   b) The counting version of FI can be reduced to its decision version [1].
3. a) GI $\equiv_p$ CSFI, that is poly-time reducible to its monotone case MCSFI [this paper].
   b) FI is poly-time reducible to its monotone case [13].

Thus, in addition to the similarities regarding noncompleteness and counting versions of each problem [1, 2], thanks to the results proved in this paper, we can deduce that both FI and CSFI are polynomial time reducible to their monotone special case.

A preliminary version of this paper [4] appeared in the Electronic Colloquium on Computational Complexity[3] (ECCC). This preliminary version has been cited in [18] as related result for the graph isomorphisms and in [3] as related result about isomorphism testing of computable functions. It has been further cited in [17] as base to build a solution for the incremental analysis of source code and in [16] in the related works section as equivalent to the max-cut problem.

This paper is organized as follows: in the next section we provide the necessary background, whilst our main result is discussed in Section 3.

2 Preliminaries and problems definitions

In this section we provide the necessary background and definitions of the problems considered. We begin, from [12], with the already mentioned

Definition 1 (Graph Isomorphism (GI)). *Given two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, are G_1 and G_2 isomorphic, i.e. is there a bijection $f : V_1 \to V_2$ such that $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$?*

[3] ECCC papers have the status of technical reports.

A well known example of a **GI**-complete problem is the following [7]:

Definition 2 (Bipartite Graph Isomorphism (BGI)). *Given two undirected bipartite graphs $G_1 = (V_1, W_1, E_1)$ and $G_2 = (V_2, W_2, E_2)$, are G_1 and G_2 isomorphic, i.e. are there two bijections $f : V_1 \to V_2$ and $g : W_1 \to W_2$ such that $(u, v) \in E_1$ if and only if $(f(u), g(v)) \in E_2$?*

We also consider, from [10], the following:

Definition 3 (Matrix Isomorphism (MI)). *Given two $n \times m$ matrices A and B with entries respectively $a_{(i,j)}$ and $b_{(i,j)}$ ($1 \leq i \leq n$, $1 \leq j \leq m$) defined over an integers set Σ, the Matrix Isomorphism (MI) problem is to determine whether there exists an isomorphism between the matrices, i.e. a permutation of the rows of A σ_r and a permutation of the columns of A σ_c such that $a_{(\sigma_r(i),\sigma_c(j))} = b_{(i,j)}$.*

We remind that other relevant **GI**-complete problems concern the isomorphisms of finite automata, hypergraphs, and context-free grammars [21]. We can now formally introduce

Definition 4 (CNF Syntactic Formula Isomorphism (CSFI)). *Given two Boolean formulas in CNF:*

$$\varphi(a_1, ..., a_n) = \bigwedge_{c=1}^{m} \left(\bigvee_{l=1}^{2n} \alpha_{(c,l)} \right) = \left(\alpha_{(1,1)} \vee, ..., \vee \alpha_{(1,2n)}\right) \wedge, ..., \wedge \left(\alpha_{(m,1)} \vee, ..., \vee \alpha_{(m,2n)}\right)$$

$$\varphi(b_1, ..., b_n) = \bigwedge_{c=1}^{m} \left(\bigvee_{l=1}^{2n} \beta_{(c,l)} \right) = \left(\beta_{(1,1)} \vee, ..., \vee \beta_{(1,2n)}\right) \wedge, ..., \wedge \left(\beta_{(m,1)} \vee, ..., \vee \beta_{(m,2n)}\right)$$

the syntactic isomorphism problem of CNF Boolean formulas (CSFI) is to decide whether there exists a permutation of the clauses σ_c and a permutation of the literals σ_l in $\varphi(a_1, ..., a_n)$ and a bijection f between their variables such that $\varphi(a_1, ..., a_n)$ and $\varphi(b_1, ..., b_n)$ may be written in the same way i.e.:

$$\bigwedge_{c=1}^{m} \left(\bigvee_{l=1}^{2n} f\left(\alpha_{(\sigma_c(c),\sigma_l(l))}\right) \right) = \bigwedge_{c=1}^{m} \left(\bigvee_{l=1}^{2n} \beta_{(c,l)} \right)$$

As an example, considering the following formulas defined over the variables set $\{x, y, z\} \cup \{a, b, c\}$:

$$\varphi(z, y, x) = (\neg z \vee z) \wedge (y \vee x) \wedge (z \vee \neg y \vee \neg x)$$

$$\varphi(a, c, b) = (a \vee b) \wedge (\neg a \vee c \vee \neg b) \wedge (\neg c \vee c)$$

a possible solution consists in the following bijection $f = \{\langle y, a\rangle, \langle x, b\rangle, \langle z, c\rangle\}$ and a suitable permutations of σ_c and σ_l.

Recalling that a Boolean formula is said to be *monotone* if it does not contain any negation, the following problem can be seen as a special case of CSFI, in which there are no negated variables.

Definition 5 (Monotone CNF Syntactic Formula Isomorphism (MCSFI)). *Considering two Boolean monotone formulas in CNF, the syntactic isomorphism problem for CNF monotone Boolean formulas (MCSFI) is the CSFI problem restricted to CNF monotone formulas.*

3 CSFI is GI-complete

Before presenting our main result, we need to prove few lemmas that describe some relationship between the problems defined in the previous section. In particular, the combination of these lemmas allows us to provide the following results:

$$CSFI \leq_p MI_{\{-1,0,1,2\}} \leq_p GI \leq_p BGI \equiv_p MI_{\{0,1\}} \equiv_p MCSFI \leq_p CSFI$$

where $\leq_p$ and $\equiv_p$ represents, respectevely, polynomial-time reduction and polynomial-time equivalency, from which we can derive that CSFI is **GI**-complete.

Lemma 1. *The matrix isomorphism problem between matrices with entries defined over the integers set $\Sigma = \{0,1\}$ is* ***GI****-complete.*

Proof. We prove the above lemma by showing that it holds both $BGI \leq_p MI_{\{0,1\}}$ and $MI_{\{0,1\}} \leq_p BGI$:

1. $BGI \leq_p MI_{\{0,1\}}$: by definition two graphs G_1 and G_2 whose adjacency matrices are respectively A_1 and A_2, are isomorphic if and only if there exists a permutation matrix P such that $A_2 = PA_1P^{-1}$ [9].
2. $MI_{\{0,1\}} \leq_p BGI$: each binary matrix M_1 corresponds to a bipartite graph G_1 whose adjacency matrix A_1 is:

$$A_1 = \begin{pmatrix} 0 & M_1 \\ M_1^T & 0 \end{pmatrix}$$

Where M_1^T is the transpose of M_1. Is simple to verify that two binary matrices M_1 and M_2 are isomorphic if and only if the respective bipartite graph, whose adjacency matrices are build upon the showed reduction, are isomorphic. So, it holds $BGI \equiv_p MI_{\{0,1\}}$. □

Theorem 1. *CSFI is deterministic polynomial time reducible to MI (i.e., CSFI $\leq_p$ MI).*

We first provide the reduction, whilst its correctness hinges on Lemma 2 and Lemma 3.

Reduction. Each CNF formula can be represented by a matrix whose n rows represent variables, belonging or not to the literals of each clause, and whose m columns represent clauses, and entries are defined over the set of integers $\Sigma = \{-1,0,1,2\}$ such that the generic entry $a_{(i,j)}$ at the i-th row and the j-th column may be:

- 0 if the i-th variable does not belong to the literals of the j-th clause, neither positive nor negated.
- 1 if the i-th variable belongs to the literals of the j-th clause.
- -1 if the i-th negated variable belongs to the literals of the j-th clause.
- 2 if both the i-th negated and the i-th non negated variables belong to the literals of the j-th clause.

For the previous example, we have:

$$\varphi(z,y,x) = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & -1 \end{pmatrix} \qquad \varphi(a,c,b) = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 2 \\ 1 & -1 & 0 \end{pmatrix}$$

Such a matrix can be built in deterministic polynomial time from φ since there are exactly $n \times m$ entries to place in the matrix where n is the cardinality of the variables set and m is the cardinality of the clauses set.

In order to prove the theorem we divide it in two distinct lemmas.

Lemma 2. *If two CNF formulas $\varphi(x_1, ..., x_n)$ and $\varphi(y_1, ..., y_n)$ are syntactically isomorphic then the respective matrices $M[\varphi(x_1, ..., x_n)]$ and $M[\varphi(y_1, ..., y_n)]$, built upon the mentioned reduction, are isomorphic:*

$$(\varphi(x_1, ..., x_n), \varphi(y_1, ..., y_n)) \in CSFI \Rightarrow (M[\varphi(x_1, ..., x_n)], M[\varphi(y_1, ..., y_n)]) \in MI$$

Proof. We can see that each permutation of clauses in $\varphi(x_1, ..., x_n)$ corresponds to a column permutation in $M[\varphi(x_1, ..., x_n)]$ and each permutation of literals, present or not in the clauses of $\varphi(x_1, ..., x_n)$, corresponds to a row permutation of $M[\varphi(x_1, ..., x_n)]$.Moreover, when $\varphi(x_1, ..., x_n)$ is syntactically isomorphic to $\varphi(y_1, ..., y_n)$ then the matrix of this isomorphism, by construction, matches with $M[\varphi(y_1, ..., y_n)]$. So deciding if two CNF formulas $\varphi(x_1, ..., x_n)$ and $\varphi(y_1, ..., y_n)$ are syntactically isomorphic corresponds to decide if, permutating rows and columns of $M[\varphi(x_1, ..., x_n)]$, we can switch from $M[\varphi(x_1, ..., x_n)]$ to $M[\varphi(y_1, ..., y_n)]$. But deciding if there exist row and column permutations between two matrices, with entries defined over an integers set, such that the two matrices coincide, is a special case of the MI problem (case with entries defined over the set $\Sigma = \{-1, 0, 1, 2\}$). □

Lemma 3. *If two matrices $M[\varphi(x_1, ..., x_n)]$ and $M[\varphi(y_1, ..., y_n)]$, built respectively from the CNF formulas $\varphi(x_1, ..., x_n)$ and $\varphi(y_1, ..., y_n)$ upon the showed reduction, are isomorphic then the CNF formulas $\varphi(x_1, ..., x_n)$ and $\varphi(y_1, ..., y_n)$ are syntactically isomorphic:*

$$(\varphi(x_1, ..., x_n), \varphi(y_1, ..., y_n)) \in CSFI \Leftarrow (M[\varphi(x_1, ..., x_n)], M[\varphi(y_1, ..., y_n)]) \in MI$$

Proof. Suppose that the two matrices $M[\varphi(x_1, ..., x_n)]$ and $M[\varphi(y_1, ..., y_n)]$ are isomorphic, then, by definition, it exists a row and column permutation in $M[\varphi(x_1, ..., x_n)]$ such that it coincides with $M[\varphi(y_1, ..., y_n)]$. Since, by the showed reduction, each matrix with entries over the integers set $\Sigma = \{-1, 0, 1, 2\}$ corresponds to a CNF formula and vice versa, we have that, each column permutation in $M[\varphi(x_1, ..., x_n)]$ corresponds to a clause permutation in $\varphi(x_1, ..., x_n)$ and each row permutation in $M[\varphi(x_1, ..., x_n)]$ corresponds to a permutation of literals, present or not in the clauses of $\varphi(x_1, ..., x_n)$, and when the two matrices coincide then, by construction, the respective CNF formulas are syntactically isomorphic (they are written in identical way except a bijection between their variables). So, deciding if there exist row and column permutations in $M[\varphi(x_1, ..., x_n)]$ such that it coincides with $M[\varphi(y_1, ..., y_n)]$ is equivalent to decide if there exists a permutation of clauses and literals and a bijection of variables in $\varphi(x_1, ..., x_n)$ such that it is syntactically isomorphic to $\varphi(y_1, ..., y_n)$. $\square$

Theorem 2. *The MI problem, between two matrices M_A and M_B whose entries are defined over the integers set $\Sigma = \{-1, 0, 1, 2\}$, is deterministic polynomial time reducible to GI (i.e., $MI_{\{-1,0,1,2\}} \leq_p GI$).*

Reduction. Each $n \times m$ matrix of a MI instance, between two matrices M_A and M_B whose entries are defined over the integers set $\Sigma = \{-1, 0, 1, 2\}$, can be represented with a graph $G = (V, A)$ built as follow (see Figure 1 for an example of the construction):

- Set of vertices $V = \{R \cup C \cup E \cup S \cup X\}$ where:
  - $R = \{r_1, ..., r_n\}$ is the set of rows.
  - $C = \{c_1, ..., c_m\}$ is the set of columns.
  - X is the set of 3 vertices used to build 2 cliques of degrees 1 and 2 each of which identifies, respectively, the set of columns and the set of rows.
  - $E = \{(1, 1), (1, 2), ..., (m, n)\}$ is the set of ordered pairs (i, j) that represents the coordinates of each generic entry $a_{(i,j)}$ positioned at the i-th row and j-th columns.
  - S is the set of 18 vertices used to build 4 cliques of degrees 3,4,5, and 6 each of which codifies, respectively, the integers numbers: -1,0,1, and 2.
- Set of edges A obtained linking the following vertices:
  - every vertex r_i is linked to the pairs (i, j) and is linked to the clique of degree 2 in order to represent the membership of r_i to the set of rows.
  - every vertex c_j is linked to the pairs (i, j) and is linked to the clique of degree 1 in order to represent the membership of c_j to the set of columns.
  - every vertex (i, j) is linked to the clique of degree k if and only if the entry $a_{(i,j)}$ is codified by the clique of degree k.

It is simple to verify that each vertex (i, j) is linked to the clique corresponding to the codified number as specified in the matrix and so the built graph matches the whole relational structure of the matrix. Two examples of reductions from isomorphic matrices are shown in Figure 1; vertices are labelled for clarity. We note that the corresponding graph can be built in polynomial time from the matrix M_A.

In order to prove the theorem we divide it in two distinct lemmas.

Lemma 4. *If two matrices M_A and M_B, with entries defined over the integers set $\Sigma = \{-1, 0, 1, 2\}$, are isomorphic then the corresponding graphs $G[M_A]$ and $G[M_B]$, built following the shown reduction, are isomorphic:*

$$(M_A, M_B) \in MI_{\{-1,0,1,2\}} \Rightarrow (G[M_A], G[M_B]) \in GI$$

Proof. If M_A and M_B are isomorphic then they are the same matrix modulo row and column permutations. We can see that each permutation of rows and each permutation of columns in the matrix M_A corresponds, respectively, to a

$$A = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 2 & 0 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

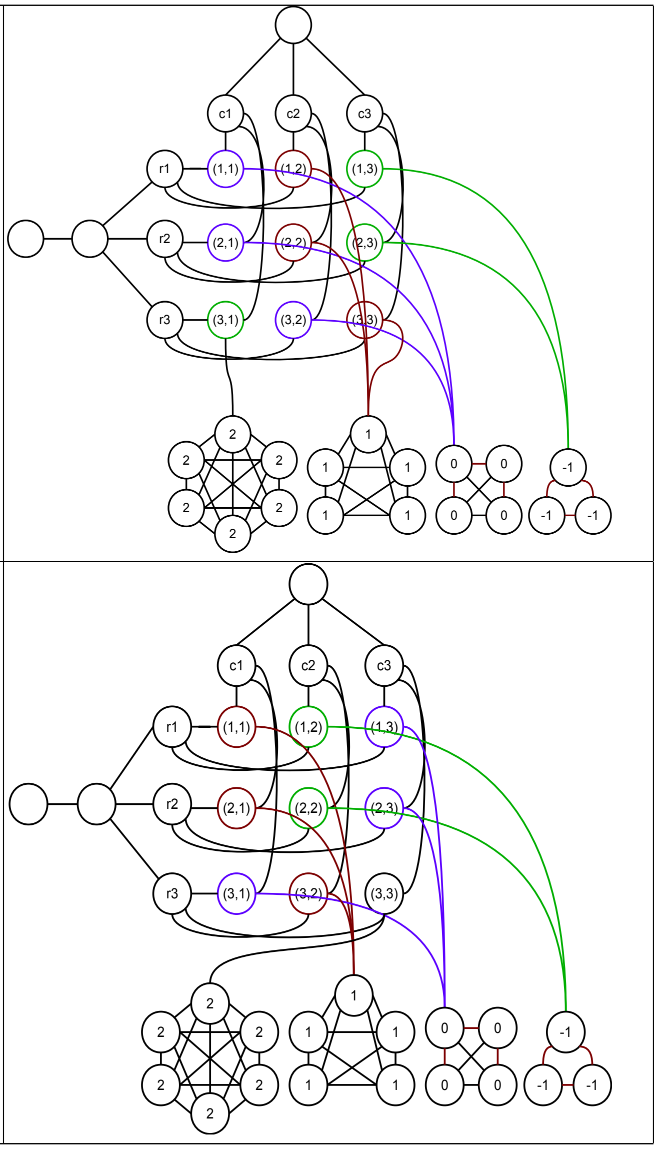

Fig. 1. An example of the reduction used in the proof of Theorem 2

permutation of vertices aligned with the rows and to a permutation of vertices aligned with the columns in $G[M_A]$ and when an isomorphism between M_A and M_B exists, then the graph of the matrix of this isomorphism, by construction, is isomorphic to $G[M_B]$. But permuting vertices aligned with rows and/or columns in $G[M_A]$ generate graphs isomorphic to $G[M_A]$. So deciding if two matrices M_A

and M_B, whose entries are defined over the integers set $\Sigma = \{-1, 0, 1, 2\}$, are isomorphic corresponds to deciding if the two graphs $G[M_A]$ and $G[M_B]$, built upon the shown reduction, are isomorphic. □

Lemma 5. *If two graphs $G[M_A]$ and $G[M_B]$, built respectively from the matrices M_A and M_B upon the shown reduction, are isomorphic then the matrices M_A and M_B are isomorphic:*

$$(M_A, M_B) \in MI_{\{-1,0,1,2\}} \Leftarrow (G[M_A], G[M_B]) \in GI$$

Proof. It is straightforward to see that, when the graphs $G[M_A]$ and $G[M_B]$ are isomorphic, then, since by construction each graph represents the whole relational structure of a matrix, the two matrices M_A and M_B from which they were built, have to be isomorphic. □

Theorem 3. *$MI_{\{0,1\}}$ between two binary matrices M_A and M_B is deterministic polynomial time reducible to MCSFI (i.e., $MI_{\{0,1\}} \leq_p$ MCSFI).*

Reduction. As seen in Theorem 1, each CNF formula can be represented by a matrix whose entries are defined over an integers set and vice versa. More formally, each CNF monotone formula can be represented as a matrix in which rows represents the variables and columns represents the clauses and matrix entries are defined over the integers set $\Sigma = \{0, 1\}$ such that the generic entry $a_{(i,j)}$ positioned at the i-th row and j-th column is:

- 0 if the i-th literal is not presents in the j-th clause
- 1 if the i-th literal is present in the j-th clause

Example: the binary matrix $A_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ corresponds to the CNF monotone formula defined over the variables set $\{z, y, x\}$:

$$\varphi(z, y, x) = (z \vee x) \wedge (y \vee x) \wedge (z \vee y)$$

Note that such a CNF formula can be built in polynomial time from a $n \times m$ matrix since there are at most $n \times m$ literals to place in each formula.

As we did for the previous theorems, we divide the proof in two distinct lemmas.

Lemma 6. *If two binary matrices M_A and M_B are isomorphic then the respective CNF monotone formulas $\varphi(a_1, ..., a_n)$ and $\varphi(b_1, ..., b_n)$, built upon the showed reduction, are syntactically isomorphic:*

$$(M_A, M_B) \in MI \Rightarrow (\varphi(a_1, ..., a_n), \varphi(b_1, ..., b_n)) \in MCSFI$$

Proof. As seen, each rows permutation in M_A corresponds to literals permutation in $\varphi(a_1, ..., a_n)$ and each columns permutation in M_A corresponds to clauses permutation in $\varphi(a_1, ..., a_n)$ and moreover, when M_A is isomorphic to M_B then the CNF monotone formula of this isomorphism will matches, by construction, with $\varphi(b_1, ..., b_n)$ unless for a variables bijection. So, deciding if two binary matrices M_A and M_B are isomorphic is equivalent to decide if permutating literals and clauses of $\varphi(a_1, ..., a_n)$, and using a variables bijection, we can switch from $\varphi(a_1, ..., a_n)$ to $\varphi(b_1, ..., b_n)$ but this problem is, by definition, the syntactic isomorphism problem between CNF monotone formulas. □

Lemma 7. *If two CNF monotone formulas $\varphi(a_1, ..., a_n)$ and $\varphi(b_1, ..., b_n)$, built respectively from the matrices M_A and M_B following the above reduction, are syntactically isomorphic then the two binary matrices M_A and M_B are isomorphic:*

$$(M_A, M_B) \in MI \Leftarrow (\varphi(a_1, ..., a_n), \varphi(b_1, ..., b_n)) \in MCSFI$$

Proof. Suppose that the two CNF monotone formulas $\varphi(a_1, ..., a_n)$ and $\varphi(b_1, ..., b_n)$ are isomorphic, then, by definition, there exist literal and clause permutations in $\varphi(a_1, ..., a_n)$ and a variables bijection such that it is syntactically identical to $\varphi(b_1, ..., b_n)$. Since, by the shown reduction, each matrix corresponds to a CNF monotone formula and vice versa, we have that, by construction, each clauses permutation in $\varphi(a_1, ..., a_n)$ corresponds to columns permutation in M_A and each literals permutation in $\varphi(a_1, ..., a_n)$ corresponds to rows permutation in M_A, and when the two CNF formulas are syntactically isomorphic then the respective binary matrices coincides. For the above reasons, deciding if, unless a variables bijection, there exists a clauses and literals permutation in $\varphi(a_1, ..., a_n)$ such that it is syntactically isomorphic to $\varphi(b_1, ..., b_n)$ is equivalent to deciding if there exists a rows and columns permutation in M_A so that it is isomorphic to M_B. □

As seen, each binary matrix can be regarded as a CNF monotone formula and vice versa, hence using the same argument as in Theorem 1 it is simple to prove the following:

Corollary 1. *MCSFI between two monotone formulas $\varphi(a_1, ..., a_n)$ and $\varphi(b_1, ..., b_n)$ is deterministic polynomial time reducible to $MI_{\{0,1\}}$ (i.e., MCSFI $\leq_p$ $MI_{\{0,1\}}$).*

Therefore we have:

Corollary 2. *$MI_{\{0,1\}} \equiv_p$ MCSFI.*

Since $MI_{\{0,1\}}$ is **GI**-complete we have:

Corollary 3. *MCSFI is **GI**-complete.*

We can now present our main result.

Theorem 4. *CSFI is **GI**-complete.*

Proof. Combining the results from Lemma 1 through Corollary 2, we can state the following:

$$CSFI \leq_p MI_{\{-1,0,1,2\}} \leq_p GI \leq_p BGI \equiv_p MI_{\{0,1\}} \equiv_p MCSFI \leq_p CSFI$$

Hence, CSFI is **GI**-complete. □

4 Conclusion

In this paper we have shown that the CSFI problem is graph isomorphism complete. This result is interesting first because CSFI is one of the few GI-complete problems that are not of a graph theoretic nature. Second, it allows to extend the similarities between the Formula Isomorphism and the Graph Isomorphism problems, already observed [1, 2]. In particular, we have shown that GI is equivalent to the problem of testing syntactic isomorphism for monotone CNF Boolean Formulas (MCSFI), as FI is equivalent to the problem of testing semantic isomorphism of monotone Boolean Formulas [13]. An interesting aspect for future researches is the extension of this result for weighted graphs.

Finally, let us observe that our result easily extends to the syntactic isomorphism of DNF Boolean Formulas. In fact, it is simple to verify that, for the commutativity of the operators AND and OR, identical theorems can be proved when the Boolean formula is in disjunctive normal form (DNF), since analogously to Theorem 1, we can replace each CNF formula with a DNF formula, that can be associated to a matrix whose columns represent terms and rows represent variables.

References

1. Agrawal, M., Thierauf, T.: The boolean isomorphism problem. Foundations of Computer Science, Annual IEEE Symposium on **0**, 422 (1996)
2. Agrawal, M., Thierauf, T.: The formula isomorphism problem. SIAM Journal on Computing **30**(3), 990–1009 (2000)
3. Arvind, V., Vasudev, Y.: Isomorphism testing of boolean functions computable by constant-depth circuits. Information and Computation **239**, 3–12 (2014)
4. Ausiello, G., Cristiano, F., Laura, L.: Syntactic isomorphism of CNF boolean formulas is graph isomorphism complete. Electronic Colloquium on Computational Complexity (ECCC) **19**, 122 (2012), http://eccc.hpi-web.de/report/2012/122
5. Babai, L.: Graph isomorphism in quasipolynomial time. CoRR **abs/1512.03547** (2015), http://arxiv.org/abs/1512.03547
6. Basin, D.A.: A term equality problem equivalent to graph isomorphism. Inf. Process. Lett. **51**, 61–66 (July 1994)
7. Booth, K.S., Colburn, C.J.: Problems polynomially equivalent to graph isomorphism. Tech. Rep. CS-77-04, Computer Science Department, University of Waterloo (1979)

8. Borchert, B., Ranjan, D., Stephan, F.: On the computational complexity of some classical equivalence relations on boolean functions. Theory of Computing Systems **31**(6), 679–693 (1998)
9. Eikenberry, J.: Approaches to solving the graph isomorphism problem. Tech. Rep. GIT-CC-04-09, Georgia Institute of Technology (2004)
10. Ettinger, M.: The complexity of comparing reaction systems. Bioinformatics **18**(3), 465–469 (2002)
11. Fortin, S.: The graph isomorphism problem. Tech. Rep. 96-20, Dept. of Computing Science, University of Alberta (1996)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
13. Goldsmith, J., Hagen, M., Mundhenk, M.: Complexity of dnf minimization and isomorphism testing for monotone formulas. Inf. Comput. **206**, 760–775 (June 2008)
14. Huth, M., Ryan, M.: Logic in Computer Science: modelling and reasoning about systems (second edition). Cambridge University Press (2004)
15. Mathon, R.: A note on the graph isomorphism counting problem. Information Processing Letters **8**, 131–132 (1979)
16. Miao, D., Cai, Z.: On the hardness of reachability reduction. In: International Computing and Combinatorics Conference. pp. 445–455. Springer (2019)
17. Mudduluru, R., Ramanathan, M.K.: Efficient incremental static analysis using path abstraction. In: International Conference on Fundamental Approaches to Software Engineering. pp. 125–139. Springer (2014)
18. Schmidt-Schauß, M., Rau, C., Sabel, D.: Algorithms for extended alpha-equivalence and complexity. In: 24th International Conference on Rewriting Techniques and Applications (RTA 2013). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2013)
19. Schöning, U.: Graph isomorphism is in the low hierarchy. J. Comput. Syst. Sci. **37**(3), 312–323 (1988)
20. Thierauf, T.: The computational complexity of equivalence and isomorphism problems. Springer-Verlag, Berlin, Heidelberg (2000)
21. Zemlyachenko, V.N., Korneenko, N.M., Tyshkevich, R.I.: Graph isomorphism problem. Journal of Mathematical Sciences **29**, 1426–1481 (1985)