

# The Complexity of Boolean Constraint Isomorphism<sup>\*</sup>

Elmar Böhler<sup>1</sup>, Edith Hemaspaandra<sup>2</sup>, Steffen Reith<sup>3</sup>, and Heribert Vollmer<sup>4</sup>

<sup>1</sup> Theoretische Informatik, Universität Würzburg, Am Hubland, D-97074 Würzburg, Germany, e-mail: boehler@informatik.uni-wuerzburg.de

<sup>2</sup> Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, U.S.A., e-mail: eh@cs.rit.edu

<sup>3</sup> LengfelderStr. 35b, D-97078 Würzburg, Germany, e-mail: streit@streit.cc

<sup>4</sup> Universität Hannover, Appelstr. 4, D-30167 Germany, email: vollmer@informatik.uni-hannover.de

**Abstract.** In 1978, Schaefer proved his famous dichotomy theorem for generalized satisfiability problems. He defined an infinite number of propositional satisfiability problems (nowadays usually called Boolean constraint satisfaction problems) and showed that all these satisfiability problems are either in P or NP-complete. In recent years, similar results have been obtained for quite a few other problems for Boolean constraints. Almost all of these problems are variations of the satisfiability problem.

In this paper, we address a problem that is not a variation of satisfiability, namely, the isomorphism problem for Boolean constraints. Previous work by Böhler et al. showed that the isomorphism problem is either coNP-hard or reducible to the graph isomorphism problem (a problem that is in NP, but not known to be NP-hard), thus distinguishing a hard case and an easier case. However, they did not classify which cases are truly easy, i.e., in P. This paper accomplishes exactly that. It shows that Boolean constraint isomorphism is coNP-hard (and GI-hard), or equivalent to graph isomorphism, or in P, and it gives simple criteria to determine which case holds.

**Keywords:** computational complexity, propositional logic, constraints, logic in computer science, graph isomorphism

## 1 Introduction

In 1978, Schaefer proved his famous dichotomy theorem for generalized satisfiability problems [Sch78]. He defined an infinite number of propositional satisfiability problems (nowadays usually called Boolean constraint satisfaction problems), showed that all these satisfiability problems are either in P or NP-complete, and gave a simple criterion to determine which of the two cases holds. This result is surprising in light of Ladner's Theorem which states that there are an infinite

---

<sup>\*</sup> Research supported in part by grants NSF-INT-9815095/DAAD-315-PPP-gü-ab, NSF-CCR-0311021, and by an RIT FEAD grant.

number of complexity classes between P and NP-complete (assuming that P is not equal to NP).

To make the discussion more concrete, we will quickly define what a constraint is and what a constraint problem is. In this paper, we will be looking at *Boolean* constraints. See for example Feder and Vardi [FV98] for a discussion about general constraint satisfaction problems. A *constraint* is a Boolean operator of fixed arity, specified as a Boolean function. For  $C$  a constraint of arity  $k$ , and  $x_1, \dots, x_k$  propositional variables (or constants),  $C(x_1, \dots, x_k)$  is a *constraint application* of  $C$ . For example,  $\lambda xy.(x \vee y)$  is a constraint, and  $x_1 \vee x_2$  is a constraint application of this constraint. Each finite set of constraints  $\mathcal{C}$  gives rise to a satisfiability problem  $\text{CSP}(\mathcal{C})$ :  $\text{CSP}(\mathcal{C})$  is the problem of, given a set of constraint applications of  $\mathcal{C}$ , determining whether this set has a satisfying assignment. We can view a set of constraint applications as a CNF formula. For example, 2-CNF-SAT corresponds to  $\text{CSP}(\{\lambda xy.(x \vee y), \lambda xy.(x \vee \bar{y}), \lambda xy.(\bar{x} \vee \bar{y})\})$ . Note that not all classes of formulas correspond to sets of constraint applications, since not every class of formulas can be viewed as conjunction of “clauses” of bounded length. For example, there is no constraint analogue of the class of 3DNF formulas, nor of the class of CNF formulas.

Using constraint terminology, Schaefer’s dichotomy theorem [Sch78] can now be formulated as follows: For any finite set of constraints  $\mathcal{C}$ , either  $\text{CSP}(\mathcal{C})$  is in P, or  $\text{CSP}(\mathcal{C})$  is NP-complete.

In recent years, dichotomy theorems<sup>1</sup> have been obtained for quite a few other problems about Boolean constraints. Almost all of these problems are variations of the satisfiability problem. For example, dichotomy theorems have been obtained for the problem of determining whether a set of constraint applications has exactly one satisfying assignment [Jub99], the problem of finding a satisfying assignment that satisfies a maximum number of constraint applications [Cre95], the problem of computing the number of satisfying assignments [CH96], the problem of finding the minimal satisfying assignment [KK01b], the inverse satisfiability problem [KS98], and the equivalence problem [BHRV02]. Khanna, Sudan, Trevisan, and Williamson examined the approximability of some of these problems [KSTW01]. Consult the excellent monograph [CKS01] for an almost completely up-to-date overview of dichotomy theorems for Boolean constraint satisfaction problems.

It should be noted that there exist some dichotomy results for problems in propositional logic that are not variants of satisfiability problems, for example, the work of Kirousis and Kolaitis on propositional circumscription [KK01a].

In this paper, we address a problem that is not a variation of satisfiability, namely, the isomorphism problem for Boolean constraints. Isomorphism is a more complicated problem than satisfiability. The exact complexity of the isomorphism problem for Boolean formulas is still unknown: It is trivially coNP-hard and in  $\Sigma_2^P$  and Agrawal and Thierauf showed that it is not  $\Sigma_2^P$ -hard unless the polynomial hierarchy collapses.

---

<sup>1</sup> We will also use the term “dichotomy theorem” for classification results that split into a finite number of cases greater than 2.

The isomorphism problem for Boolean constraints was first studied by Böhler et al. [BHRV02]. They showed that this problem is either coNP-hard or reducible to the graph isomorphism problem (a problem that is in NP, but not known to be NP-hard), thus distinguishing a hard case and an easier case. However, they did not classify which cases are truly easy, i.e., in P. This paper accomplishes exactly that. It shows that Boolean constraint isomorphism is coNP-hard (and GI-hard), or equivalent to graph isomorphism, or in P, and it gives simple criteria to determine which case holds.

## 2 Preliminaries

We start by formally introducing constraint problems. The following section is essentially from [BHRV02], following the standard notation developed in [CKS01].

- Definition 1.** 1. A *constraint*  $C$  (of arity  $k$ ) is a Boolean function from  $\{0, 1\}^k$  to  $\{0, 1\}$ .
2. If  $C$  is a constraint of arity  $k$ , and  $x_1, x_2, \dots, x_k$  are (not necessarily distinct) variables, then  $C(x_1, x_2, \dots, x_k)$  is a *constraint application* of  $C$ . In this paper, we view a constraint application as a Boolean function on a specific set of variables. Thus, for example,  $x_1 \vee x_2 = x_2 \vee x_1$
  3. If  $C$  is a constraint of arity  $k$ , and for  $1 \leq i \leq k$ ,  $x_i$  is a variable or a constant (0 or 1), then  $C(x_1, x_2, \dots, x_k)$  is a constraint application of  $C$  *with constants*.
  4. If  $A$  is a constraint application [with constants], and  $X$  a set of variables that includes all variables that occur in  $A$ , we say that  $A$  is a constraint application [with constants] *over variables*  $X$ . Note that we do not require that every element of  $X$  occurs in  $A$ .

The complexity of Boolean constraint problems depends on those properties of constraints that we define next.

**Definition 2.** Let  $C$  be a constraint.

- $C$  is *0-valid* if  $C(\mathbf{0}) = 1$ . Similarly,  $C$  is *1-valid* if  $C(\mathbf{1}) = 1$ .
- $C$  is *Horn* (or *weakly negative*) if  $C$  is equivalent to a CNF formula where each clause has at most one positive literal.
- $C$  is *anti-Horn* (or *weakly positive*) if  $C$  is equivalent to a CNF formula where each clause has at most one negative literal.
- $C$  is *bijunctive* if  $C$  is equivalent to a 2CNF formula.
- $C$  is *affine* if  $C$  is equivalent to an XOR-CNF formula.
- $C$  is *2-affine* (or, affine with width 2) if  $C$  is equivalent to a XOR-CNF formula such that every clause contains at most two literals.

Let  $\mathcal{C}$  be a finite set of constraints. We say  $\mathcal{C}$  is 0-valid, 1-valid, Horn, anti-Horn, bijunctive, or affine, if *every* constraint  $C \in \mathcal{C}$  is 0-valid, 1-valid, Horn, anti-Horn, bijunctive, or affine, respectively. Finally, we say that  $\mathcal{C}$  is *Schaefer* if  $\mathcal{C}$  is Horn or anti-Horn or affine or bijunctive.

Like all dichotomy results for Boolean constraints, our proofs heavily use Schaefer's characterization of Boolean functions.

**Lemma 3** ([Sch78, Hor51]). Let  $f$  be a Boolean function of arity  $k$ .

1.  $f$  is Horn if and only if for all assignments  $s, t \in \{0, 1\}^k$  that satisfy  $f$ ,  $s \cap t$  satisfies  $f$ .
2.  $f$  is anti-Horn if and only if for all assignments  $s, t \in \{0, 1\}^k$  that satisfy  $f$ ,  $s \cup t$  satisfies  $f$ .
3.  $f$  is bijunctive if and only if for all assignments  $s, t, u \in \{0, 1\}^k$  that satisfy  $f$ ,  $\text{majority}(s, t, u)$  satisfies  $f$ .
4.  $f$  is affine if and only if for all assignments  $s, t, u \in \{0, 1\}^k$  that satisfy  $f$ ,  $s \oplus t \oplus u$  satisfies  $f$ .

The operations on assignments mentioned above are bitwise operations, i.e.,  $(s \cap t)_i = s_i \wedge t_i$ ,  $(s \cup t)_i = s_i \vee t_i$ ,  $(s \oplus t)_i = s_i \oplus t_i$ , and  $\text{majority}(s, t, u) = (s \cap t) \cup (s \cap u) \cup (t \cap u)$ .

The question studied in this paper is that of whether a set of constraint applications can be made equivalent to a second set of constraint applications using a suitable renaming of its variables. We need some definitions.

- Definition 4.**
1. Let  $S$  be a set of constraint applications with constants over variables  $X$  and let  $\pi$  be a permutation of  $X$ . By  $\pi(S)$  we denote the set of constraint applications that results when we replace simultaneously all variables  $x$  in  $S$  by  $\pi(x)$ .
  2. Let  $S$  be a set of constraint applications over variables  $X$ . The number of satisfying assignments of  $S$ ,  $\#_1(S)$ , is defined as  $||\{I \mid I \text{ is an assignment to all variables in } X \text{ that satisfies every constraint application in } S\}||$ .

The isomorphism problem for Boolean constraints, first defined and examined in [BHRV02] is formally defined as follows.

- Definition 5.**
1.  $\text{ISO}(\mathcal{C})$  is the problem of, given two sets  $S$  and  $U$  of constraint applications of  $\mathcal{C}$  over variables  $X$ , to decide whether  $S$  and  $U$  are isomorphic, i.e., whether there exists a permutation  $\pi$  of  $X$  such that  $\pi(S)$  is equivalent to  $U$ .
  2.  $\text{ISO}_c(\mathcal{C})$  is the problem of, given two sets  $S$  and  $U$  of constraint applications of  $\mathcal{C}$  with constants over variables  $X$ , to decide whether  $S$  and  $U$  are isomorphic.

Böhler et al. obtained results about the complexity of the just-defined problem that, interestingly, pointed out relations to another isomorphism problem: the graph isomorphism problem (GI).

**Definition 6.** GI is the problem of, given two graphs  $G$  and  $H$ , to determine whether  $G$  and  $H$  are isomorphic, i.e., whether there exists a bijection  $\pi: V(G) \rightarrow V(H)$  such that for all  $v, w \in V(G)$ ,  $\{v, w\} \in E(G)$  iff  $\{\pi(v), \pi(w)\} \in E(H)$ . Our graphs are undirected, and do not contain self-loops. We also assume a standard enumeration of the edges, and will write  $E(G) = \{e_1, \dots, e_m\}$ .

GI is a problem that is in NP, not known to be in P, and not NP-complete unless the polynomial hierarchy collapses. For details, see, for example, [KST93]. Recently, Torán showed that GI is hard for NL, PL,  $\text{Mod}_k\text{L}$ , and DET under logspace many-one reductions [Tor00]. Arvind and Kurur showed that GI is in the class SPP [AK02], and thus, for example in  $\oplus P$ .

The main result from [BHRV02] can now be stated as follows.

**Theorem 7.** *Let  $\mathcal{C}$  be a set of constraints. If  $\mathcal{C}$  is Schaefer, then  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are polynomial-time many-one reducible to GI, otherwise,  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are coNP-hard.*

Note that if  $\mathcal{C}$  is Schaefer, then the isomorphism problems  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  cannot be coNP-hard, unless  $\text{NP} = \text{coNP}$ . (This follows from Theorem 7 and the fact that GI is in NP.) Under the (reasonable) assumption that  $\text{NP} \neq \text{coNP}$ , and that GI is neither in P, nor NP-complete, Theorem 7 thus distinguishes a hard case (coNP-hard) and an easier case (many-one reducible to GI).

Böhler et al. also pointed out that there are some bijunctive, Horn, or affine constraint sets  $\mathcal{C}$  for which actually  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are equivalent to graph isomorphism. On the other hand, certainly there are  $\mathcal{C}$  for which  $\text{ISO}(\mathcal{C}), \text{ISO}_c(\mathcal{C}) \in \text{P}$ . In the upcoming section we will completely classify the complexity of  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$ , obtaining for which  $\mathcal{C}$  exactly we are equivalent to GI and for which  $\mathcal{C}$  we are in P.

### 3 A Trichotomy Theorem

The main result of this paper is a complete complexity-theoretic classification of the isomorphism problem for Boolean constraints.

**Theorem 8.** *Let  $\mathcal{C}$  be a finite set of constraints.*

1. *If  $\mathcal{C}$  is not Schaefer, then  $\text{ISO}(\mathcal{C})$  is coNP-hard and  $\text{ISO}_c(\mathcal{C})$  is coNP-hard and GI-hard.*
2. *If  $\mathcal{C}$  is Schaefer and not 2-affine, then  $\text{ISO}_c(\mathcal{C})$  is polynomial-time many-one equivalent to GI.*
3. *Otherwise,  $\mathcal{C}$  is 2-affine and  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are in P.*

The rest of the paper is organized as follows. The coNP lower-bound part from Theorem 8 follows from Theorem 7. In Section 4 we will prove the polynomial-time upper bound if  $\mathcal{C}$  is 2-affine (Theorem 11). The GI upper bound if  $\mathcal{C}$  is Schaefer again is part of Theorem 7. Finally, in Section 5 we will show that  $\text{ISO}_c(\mathcal{C})$  is GI-hard if  $\mathcal{C}$  is not 2-affine (Theorems 18 and 22).

### 4 Upper Bounds

A central step in our way of obtaining upper bounds is to bring sets of constraint applications into a unique normal form. This approach is also followed in the

proof of the  $\text{coIP}[2]^{\text{NP}}$  upper bound<sup>2</sup> for the isomorphism problem for Boolean formulas [AT00] and the GI upper bound from Theorem 7 [BHRV02].

**Definition 9.** Let  $\mathcal{C}$  be a set of constraints.  $nf$  is a *normal form function* for  $\mathcal{C}$  if and only if for all sets  $S$  and  $U$  of constraint applications of  $\mathcal{C}$  with constants over variables  $X$ , and for all permutations  $\pi$  of  $X$ ,

1.  $nf(S, X)$  is a set of Boolean functions over variables  $X$ ,
2.  $S \equiv U$  (here we view  $S$  as a set of Boolean functions, and define equivalence for such sets as logical equivalence of corresponding propositional formulas),
3.  $nf(\pi(S), X) = \pi(nf(S, X))$ , and
4. if  $S \equiv U$ , then  $nf(S, X) = nf(U, X)$  (here, “ $=$ ” is equality between sets of Boolean functions).

It is important to note that  $nf(S, X)$  is not necessarily a set of constraint applications of  $\mathcal{C}$  with constants.

An easy property of the definition is that  $S \equiv U$  iff  $nf(S, X) = nf(U, X)$ . Also, it is not too hard to observe that using normal forms removes the need to check whether two sets of constraint applications with constants are equivalent, more precisely:  $S$  is isomorphic to  $U$  iff there exists a permutation  $\pi$  of  $X$  such that  $\pi(nf(S)) = nf(U)$ .

There are different possibilities for normal forms. The one used by [BHRV02] is the maximal equivalent set of constraint applications with constants, defined by  $nf(S, X)$  to be the set of all constraint applications  $A$  of  $\mathcal{C}$  with constants over variables  $X$  such that  $S \rightarrow A$ .

For the P upper bound for 2-affine constraints, we use a normal form described in the following lemma. Note that this normal form is not necessarily a set of 2-affine constraint applications with constants.

**Lemma 10.** *Let  $\mathcal{C}$  be a set of 2-affine constraints. There exists a polynomial-time computable normal form function  $nf$  for  $\mathcal{C}$  such that for all sets  $S$  of constraint applications of  $\mathcal{C}$  with constants over variables  $X$ , the following hold:*

1. if  $S \equiv 0$ , then  $nf(S, X) = \{0\}$ ,
2. if  $S \not\equiv 0$ , then  $nf(S, X) = \{\overline{Z}, 0\} \cup \bigcup_{i=1}^{\ell} \{(X_i \wedge \overline{Y_i}) \vee (\overline{X_i} \wedge Y_i)\}$ , where  $Z, O, X_1, Y_1, \dots, X_{\ell}, Y_{\ell}$  are pairwise disjoint subsets of  $X$  such that  $X_i \cup Y_i \neq \emptyset$  for all  $1 \leq i \leq \ell$ , and for  $W$  a set of variables,  $W$  in a formula denotes  $\bigwedge W$ , and  $\overline{W}$  denotes  $\neg \bigvee W$ .

**Proof.** Let  $S$  be a set of constraint applications of  $\mathcal{C}$  with constants over variables  $X$ . Since  $\mathcal{C}$  is 2-affine, we can in polynomial time check whether  $S \equiv 0$ . If so,  $nf(S, X) = \{0\}$ . Now suppose that  $S$  is not equivalent to 0. Let  $\mathcal{D}$  be the set of all unary and binary 2-affine constraints, i.e.,  $\mathcal{D} = \{\lambda a.a, \lambda a.\overline{a}, \lambda ab.a \oplus b, \lambda ab.\neg(a \oplus b)\}$ .

<sup>2</sup> Here  $\text{IP}[2]$  means an interactive proof system where there are two messages exchanged between the verifier and the prover.

Let  $S''$  be the set of all  $\mathcal{D}$  constraint applications  $A$  with constants over variables  $X$  such that  $S \rightarrow A$ . In other words, we are using the maximal equivalent set normal form used in [BHRV02], that we described in the paragraph preceding Lemma 10. It follows from [BHRV02] that  $S''$  is computable in polynomial time. Certainly,  $S \equiv S''$ , since every constraint application in  $S$  can be written as the conjunction of  $\mathcal{D}$  constraint applications. We will now show how to compute  $nf(S, X)$ .

Let  $Z$  be the set of those variables  $x \in X$  that, when set to 1, make  $S''$  equivalent to 0. Let  $O$  be the set of those variables that, when set to 0, make  $S''$  equivalent to 0. Note that  $S'' \rightarrow \overline{Z} \wedge O$ . Let  $S'$  be the set of constraint applications that is obtained from  $S''$  by setting all elements of  $Z$  to 0 and all elements of  $O$  to 1. Note that  $S'' \equiv \overline{Z} \wedge O \wedge S'$ .

Let  $\hat{S}$  be the set of all constraint applications from  $S'$  that do not contain constants. We claim that  $\hat{S} \equiv S'$ . For suppose that  $\alpha$  is an assignment that satisfies  $\hat{S}$ , but  $\alpha$  does not satisfy  $S'$ . Then there is a constraint application with constants  $A \in S' \setminus \hat{S}$  such that  $\alpha$  does not satisfy  $A$ . Note that  $A$  must contain a constant, since  $A \notin \hat{S}$ .  $A$  must contain a variable, since  $A$  is satisfiable (since  $S$  is satisfiable). But then  $A$  contains exactly one occurrence of exactly one variable, and thus  $A$  is equivalent to  $x$  or  $\neg x$  for some variable  $x$ . But then  $x$  would have been put in  $Z$  or  $O$ , and  $x$  would not occur in  $S'$ .

So,  $S \equiv \overline{Z} \wedge O \wedge \hat{S}$ , and every element of  $\hat{S}$  is of the form  $x \oplus y$  where  $x, y \in X$  and  $x \neq y$  or of the form  $\neg(x \oplus y)$ , where  $x, y \in X$ ,  $x \neq y$ . Note that it is possible that  $\hat{S} = \emptyset$ . Also note that, since  $S''$  contains all of its implicates that are of the right form, for every three distinct variables  $x, y$ , and  $z$ ,

- if  $(x \oplus y) \in \hat{S}$  and  $(y \oplus z) \in \hat{S}$ , then  $\neg(x \oplus z) \in \hat{S}$ ,
- if  $\neg(x \oplus y) \in \hat{S}$  and  $(y \oplus z) \in \hat{S}$ , then  $(x \oplus z) \in \hat{S}$ , and
- if  $\neg(x \oplus y) \in \hat{S}$  and  $\neg(y \oplus z) \in \hat{S}$ , then  $\neg(x \oplus z) \in \hat{S}$ .

So,  $\hat{S}$  is closed under a form of transitivity.

Partition  $\hat{S}$  into  $S_1, \dots, S_\ell$ , where  $S_1, \dots, S_\ell$  are minimal sets that are pairwise disjoint with respect to occurring variables. Since the  $S_i$ s are minimal, and not equivalent to 0, it follows from the observation above about the closure of  $\hat{S}$ , that for every pair of distinct variables  $x, y$  in  $S_i$ , exactly one of  $(x \oplus y)$  and  $\neg(x \oplus y)$  is in  $S_i$ . For every  $i$ ,  $1 \leq i \leq \ell$ , let  $x_i$  be an arbitrary variable that occurs in  $S_i$ . Let  $X_i = \{y \mid \neg(x_i \oplus y) \in S_i\}$  and let  $Y_i = \{y \mid x_i \oplus y \in S_i\}$ . Then  $X_i \cap Y_i = \emptyset$  and  $X_i \cup Y_i$  = the variables that occur in  $S_i$ .

It is easy to see that  $S_i \equiv \{(X_i \wedge \overline{Y_i}) \vee (\overline{X_i} \wedge Y_i)\}$ .

We claim that

$$nf(S, X) = \{\overline{Z}, O\} \cup \bigcup_{i=1}^{\ell} \{(X_i \wedge \overline{Y_i}) \vee (\overline{X_i} \wedge Y_i)\}$$

fulfills the criteria of Lemma 10.

First of all, it is clear that  $nf$  is computable in polynomial time. From the observations above, it follows that  $Z, O, X_1, Y_1, \dots, X_\ell, Y_\ell$  are pairwise disjoint subsets of  $X$  such that

1.  $X = Z \cup O \cup \bigcup_{i=1}^{\ell} (X_i \cup Y_i)$ , and
2.  $X_i \cup Y_i \neq \emptyset$  for all  $1 \leq i \leq \ell$ ,

that  $nf(S, X) \equiv S$ , and that  $nf(\pi(S), X) = \pi(nf(S), X)$ , for all permutations  $\pi$  of  $X$ .

It remains to show that if  $U$  is a set of constraint applications of  $\mathcal{C}$  with constants, and  $S \equiv U$ , then  $nf(S, X) = nf(U, X)$ .

Let  $Z', O', X'_1, Y'_1, \dots, X'_k, Y'_k$  be subsets of  $X$  such that:

1.  $Z', O', X'_1, Y'_1, \dots, X'_k, Y'_k$  are pairwise disjoint,
2.  $X = Z' \cup O' \cup \bigcup_{i=1}^k (X'_i \cup Y'_i)$ ,
3.  $X'_i \cup Y'_i \neq \emptyset$  for all  $1 \leq i \leq k$ ,
4.  $nf(U, X) = \{\overline{Z'}, O'\} \cup \bigcup_{i=1}^k \{(X'_i \wedge \overline{Y'_i}) \vee (\overline{X'_i} \wedge Y'_i)\}$ .

Since  $S \equiv nf(S, X)$ ,  $U \equiv nf(U, X)$ , and  $S \equiv U$ , it follows that  $nf(S, X) \equiv nf(U, X)$ . From this, it is immediate that  $Z = Z'$  and that  $O = O'$ . In addition, for any two variables  $x, y \in X$ :

- $(\{x, y\} \in X_i \text{ or } \{x, y\} \in Y_i \text{ for some } i) \text{ iff } (S \text{ is satisfiable iff } x \equiv y) \text{ iff } (\{x, y\} \in X'_j \text{ or } \{x, y\} \in Y'_j \text{ for some } j), \text{ and}$
- $(\{x, y\} \cap X_i \neq \emptyset \text{ and } \{x, y\} \cap Y_i \neq \emptyset \text{ for some } i) \text{ iff } (S \text{ is satisfiable iff } x \not\equiv y) \text{ iff } (\{x, y\} \cap X'_j \neq \emptyset \text{ and } \{x, y\} \cap Y'_j \neq \emptyset \text{ for some } j).$

This implies that  $nf(U, X) = nf(S, X)$ , which completes the proof.  $\square$

Making use of the normal form, it is not too hard to prove our claimed upper bound.

**Theorem 11.** *Let  $\mathcal{C}$  be a set of constraints. If  $\mathcal{C}$  is 2-affine, then  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are in P.*

**Proof.** Let  $S$  and  $U$  be two sets of constraint applications of  $\mathcal{C}$  and let  $X$  be the set of variables that occur in  $S \cup U$ . Use Lemma 10 to bring  $S$  and  $U$  into normal form. Using the first point in that lemma, it is easy to check whether  $S$  or  $U$  are equivalent to 0. For the remainder of the proof, we now suppose that neither  $S$  nor  $U$  is equivalent to 0. Let  $Z, O, X_1, Y_1, \dots, X_\ell, Y_\ell$  and  $Z', O', X'_1, Y'_1, \dots, X'_k, Y'_k$  be subsets of  $X$  such that:

1.  $Z, O, X_1, Y_1, \dots, X_\ell, Y_\ell$  are pairwise disjoint and  $Z', O', X'_1, Y'_1, \dots, X'_k, Y'_k$  are pairwise disjoint,
2.  $X_i \cup Y_i \neq \emptyset$  for all  $1 \leq i \leq \ell$  and  $X'_i \cup Y'_i \neq \emptyset$  for all  $1 \leq i \leq k$ ,
3.  $nf(S, X) = \{\overline{Z}, O\} \cup \bigcup_{i=1}^{\ell} \{(X_i \wedge \overline{Y_i}) \vee (\overline{X_i} \wedge Y_i)\}$ , and  $nf(U, X) = \{\overline{Z'}, O'\} \cup \bigcup_{i=1}^k \{(X'_i \wedge \overline{Y'_i}) \vee (\overline{X'_i} \wedge Y'_i)\}$ .

We need to determine whether  $S$  is isomorphic to  $U$ . Since  $nf$  is a normal form function for  $\mathcal{C}$ , it suffices to check if there exists a permutation  $\pi$  on  $X$  such that  $\pi(nf(S, X)) = nf(U, X)$ . Note that

$$\pi(nf(S, X)) = \{\overline{\pi(Z)}, \pi(O)\} \cup \bigcup_{i=1}^{\ell} \{(\pi(X_i) \wedge \overline{\pi(Y_i)}) \vee (\overline{\pi(X_i)} \wedge \pi(Y_i))\}.$$

It is immediate that  $\pi(nf(S, X)) = nf(U, X)$  if and only if



- $\ell = k$ ,  $\pi(Z) = Z'$ ,  $\pi(O) = O'$ , and
- $\{\{\pi(X_1), \pi(Y_1)\}, \dots, \{\pi(X_\ell), \pi(Y_\ell)\}\} = \{\{X'_1, Y'_1\}, \dots, \{X'_\ell, Y'_\ell\}\}$ .

Since  $Z, O, X_1, Y_1, \dots, X_\ell, Y_\ell$  are pairwise disjoint subsets of  $X$ , and since  $Z', O', X'_1, Y'_1, \dots, X'_k, Y'_k$  are pairwise disjoint subsets of  $X$ , it is easy to see that there exists a permutation  $\pi$  on  $X$  such that  $nf(\pi(S), X) = nf(U, X)$  if and only if

- $\ell = k$ ,  $\|Z\| = \|Z'\|$ ,  $\|O\| = \|O'\|$ , and
- $\{\{\|X_1\|, \|Y_1\|\}, \dots, \{\|X_k\|, \|Y_k\|\}\} = \{\{\|X'_1\|, \|Y'_1\|\}, \dots, \{\|X'_k\|, \|Y'_k\|\}\}$ .

It is easy to see that the above conditions can be verified in polynomial time. It follows that  $\text{ISO}(\mathcal{C})$  and  $\text{ISO}_c(\mathcal{C})$  are in P.  $\square$

## 5 GI-hardness

In this section, we will prove that if  $\mathcal{C}$  is not 2-affine, then GI is polynomial-time many-one reducible to  $\text{ISO}_c(\mathcal{C})$ . As in the upper bound proofs of the previous section, we will often look at certain normal forms. In this section, it is often convenient to avoid constraint applications that allow duplicates.

**Definition 12.** Let  $\mathcal{C}$  be a set of constraints.

1.  $A$  is a constraint application of  $\mathcal{C}$  *without duplicates* if there exists a constraint  $C \in \mathcal{C}$  of arity  $k$  such that  $A = C(x_1, \dots, x_k)$ , where  $x_i \neq x_j$  for all  $i \neq j$ .
2. Let  $S$  be a set of constraint applications of  $\mathcal{C}$  [without duplicates] over variables  $X$ . We say that  $S$  is a maximal set of constraint applications of  $\mathcal{C}$  [without duplicates] over variables  $X$  if for all constraint applications  $A$  of  $\mathcal{C}$  [without duplicates] over variables  $X$ , if  $S \rightarrow A$ , then  $A \in S$ .

If  $X$  is the set of variables occurring in  $S$ , we will say that  $S$  is a maximal set of constraint applications of  $\mathcal{C}$  [without duplicates].

The following lemma is easy to see.

**Lemma 13.** *Let  $\mathcal{C}$  be a set of constraints. Let  $S$  and  $U$  be maximal sets of constraint applications of  $\mathcal{C}$  over variables  $X$  [without duplicates]. Then  $S$  is isomorphic to  $U$  iff there exists a permutation  $\pi$  of  $X$  such that  $\pi(S) = U$ .*

When reducing from GI, it is often useful to assume that the graphs have certain properties. The following lemma shows that the complexity of GI for certain restricted classes of graphs does not decrease.

**Lemma 14.** *GI is polynomial-time many-one reducible to the graph isomorphism problem for pairs of graphs  $G$  and  $H$  such that for some  $n$ ,  $G$  and  $H$  have the same set of vertices  $\{1, \dots, n\}$ ,  $G$  and  $H$  have the same number of edges, every vertex in  $G$  and  $H$  has degree at least two (i.e., is incident with at least two edges), and  $G$  and  $H$  do not contain triangles.*

**Proof.** Let  $G$  and  $H$  be graphs. If  $G$  and  $H$  do not contain the same number of vertices, or if  $G$  and  $H$  do not contain the same number of edges, or if  $G$  and  $H$  do not contain the same number of isolated vertices, then  $G$  is not isomorphic to  $H$ .

So suppose that  $G$  and  $H$  have the same number of vertices, the same number of edges, and the same number of isolated vertices. Let  $G_1$  be the graph that results if we remove all isolated vertices from  $G$ . Let  $H_1$  be the graph that results if we remove all isolated vertices from  $H$ . Then  $G_1$  and  $H_1$  have the same number of vertices and the same number of edges, all vertices in  $G_1$  and  $H_1$  have degree at least one, and  $G$  is isomorphic to  $H$  iff  $G_1$  is isomorphic to  $H_1$ . Without loss of generality, assume that  $G_1$  has at least 3 vertices.

We will now ensure that no vertex has degree less than two. Let  $v_0$  be a new vertex. Define  $G_2$  as follows:  $V(G_2) = V(G_1) \cup \{v_0\}$ ,  $E(G_2) = E(G_1) \cup \{\{v_0, v\} \mid v \in V(G_1)\}$ . Define  $H_2$  in the same way, i.e.,  $V(H_2) = V(H_1) \cup \{v_0\}$ ,  $E(H_2) = E(H_1) \cup \{\{v_0, v\} \mid v \in V(H_1)\}$ . Note that  $G_2$  and  $H_2$  have the same number of vertices and the same number of edges, and that all vertices in  $G_2$  and  $H_2$  have degree at least two. In addition, it is easy to see that  $G_1$  is isomorphic to  $H_1$  iff  $G_2$  is isomorphic to  $H_2$ : If  $\pi$  is an isomorphism from  $G_2$  to  $H_2$ , then we can define an isomorphism  $\rho$  from  $G_1$  to  $H_1$  as follows: For all  $v \in V(G_1)$ ,  $\rho(v) = \pi(v)$  if  $\pi(v) \neq v_0$ , and  $\rho(v) = \pi(v_0)$  if  $\pi(v) = v_0$ .

Next, we will remove triangles. Define  $G_3$  as follows:  $V(G_3) = V(G_2) \cup E(G_2)$ ,  $E(G_3) = \{\{v, w\} \mid v \in V(G_2), w \in E(G_2), v \in w\}$ . Define  $H_3$  in the same way, i.e.,  $V(H_3) = V(H_2) \cup E(H_2)$ ,  $E(H_3) = \{\{v, w\} \mid v \in V(H_2), w \in E(H_2), v \in w\}$ . Note that  $G_3$  and  $H_3$  are triangle-free graphs with the same number of vertices and the same number of edges, and that all vertices in  $G_3$  and  $H_3$  have degree at least two. We claim that  $G_2$  is isomorphic to  $H_2$  iff  $G_3$  is isomorphic to  $H_3$ . The left-to-right direction is immediate. The right-to-left direction follows since an isomorphism from  $G_3$  to  $H_3$  maps  $V(G_2)$  to  $V(H_2)$ , since these are exactly the vertices at even distance from a vertex of degree greater than 2. (Here we use the fact that the degree of  $v_0$  is greater than 2.)

Let  $n$  be the number of vertices of  $G_3$  and  $H_3$ . Rename the vertices in  $G_3$  and  $H_3$  to  $\{1, 2, \dots, n\}$ . This proves Lemma 14.  $\square$

Note that if  $\mathcal{C}$  is not 2-affine, then  $\mathcal{C}$  is not affine, or  $\mathcal{C}$  is affine and not bijective. We will first look at some very simple non-affine constraints.

**Definition 15** ([CKS01, p. 20]).

1.  $\text{OR}_0$  is the constraint  $\lambda xy.x \vee y$ .
2.  $\text{OR}_1$  is the constraint  $\lambda xy.\bar{x} \vee y$ .
3.  $\text{OR}_2$  is the constraint  $\lambda xy.\bar{x} \vee \bar{y}$ .
4.  $\text{OneInThree}$  is the constraint  $\lambda xyz.(x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge \bar{z}) \vee (\bar{x} \wedge \bar{y} \wedge z)$ .

As a first step in the general GI-hardness proof, we show that GI reduces to some particular constraints.

**Lemma 16.** 1. GI is polynomial-time many-one reducible to  $\text{ISO}(\{\text{OR}_0\})$ ,  $\text{ISO}(\{\text{OR}_1\})$ , and  $\text{ISO}(\{\text{OR}_2\})$ .

2. Let  $h$  be the 4-ary constraint  $h(x, y, x', y') = (x \vee y) \wedge (x \oplus x') \wedge (y \oplus y')$ . GI is polynomial-time many-one reducible to  $\text{ISO}(\{h\})$ .
3. Let  $h$  be a 6-ary constraint  $h(x, y, z, x', y', z') = \text{OneInThree}(x, y, z) \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$ . Then GI is polynomial-time many-one reducible to  $\text{ISO}(\{h\})$ .

**Proof.** 1. We will first show that GI is polynomial-time many-one reducible to  $\text{ISO}(\{\text{OR}_0\})$ . (We remark that this result already appears in [BRS98].) Let  $\widehat{G}$  be a graph and let  $V(\widehat{G}) = \{1, 2, \dots, n\}$ . We encode  $\widehat{G}$  in the obvious way as a set of constraint applications  $S(\widehat{G}) = \{x_i \vee x_j \mid \{i, j\} \in E(\widehat{G})\}$ . It is easy to see that  $S(\widehat{G})$  is a maximal set of constraint applications of  $\text{OR}_0$ . If  $G$  and  $H$  are two graphs without isolated vertices and with vertex set  $\{1, 2, \dots, n\}$ , then  $G$  is isomorphic to  $H$  if and only if there exists a permutation  $\pi$  of  $\{x_1, \dots, x_n\}$  such that  $\pi(S(G)) = S(H)$ . By Lemma 13 it follows that  $G$  is isomorphic to  $H$  if and only if  $S(G)$  is isomorphic to  $S(H)$ .

If we negate all occurring variables in  $S(\widehat{G})$ , i.e.,  $S(\widehat{G}) = \{\overline{x_i} \vee \overline{x_j} \mid \{i, j\} \in E(\widehat{G})\}$ , we obtain a reduction from GI to  $\text{ISO}(\{\text{OR}_2\})$ .

It remains to show that GI is reducible to  $\text{ISO}(\{\text{OR}_1\})$ . Note that the obvious encoding  $\{x_i \vee \overline{x_j} \mid \{i, j\} \in E(\widehat{G})\}$  does not work, since a 3-vertex graph with edges  $\{1, 2\}$  and  $\{1, 3\}$  will be indistinguishable from a 3-vertex graph with edges  $\{1, 2\}$ ,  $\{1, 3\}$ , and  $\{2, 3\}$ .

We solve this problem by using a slightly more complicated encoding. Let  $E(\widehat{G}) = \{e_1, \dots, e_m\}$ . Define  $S(\widehat{G}) = \{x_i \vee \overline{y_k}, x_j \vee \overline{y_k} \mid e_k = \{i, j\}\}$ . We claim that if  $G$  and  $H$  are two graphs without isolated vertices with vertex set  $\{1, 2, \dots, n\}$ , then  $G$  is isomorphic to  $H$  if and only if  $S(G)$  is isomorphic to  $S(H)$ .

The left-to-right direction is immediate. For the converse, note that for all  $\widehat{G}$ ,  $S(\widehat{G})$  is a maximal set of constraint applications of  $\text{OR}_1$ . Thus, by Lemma 13, if  $S(G)$  is isomorphic to  $S(H)$ , there exists a permutation  $\pi$  of the variables such that  $\pi(S(G)) = S(H)$ . Since the  $x_i$  variables are exactly those variables that occur positively in  $S(G)$  and  $S(H)$ ,  $\pi$  maps  $x$ -variables to  $x$ -variables, and thus induces an isomorphism from  $G$  to  $H$ .

2. We use a similar encoding as in the first case above. Let  $\widehat{G}$  be a graph and let  $V(\widehat{G}) = \{1, 2, \dots, n\}$ . We encode  $\widehat{G}$  as the following set of constraint applications of  $h$ :  $S(\widehat{G}) = \{h(x_i, x_j, x'_i, x'_j) \mid \{i, j\} \in E(\widehat{G})\}$ .

Let  $X = \{x_1, \dots, x_n, x'_1, \dots, x'_n\}$ . Note that for all variables  $x, y \in X$ ,  $(S(\widehat{G}) \rightarrow (x \vee y) \text{ and } S(\widehat{G}) \not\rightarrow (x \oplus y))$  iff there exists an edge  $\{i, j\} \in E$  such that  $\{x, y\} = \{x_i, x_j\}$ .

Let  $G$  and  $H$  be two graphs without isolated vertices and with vertex set  $\{1, 2, \dots, n\}$ . We claim that  $G$  is isomorphic to  $H$  if and only if  $S(G)$  is isomorphic to  $S(H)$ . The left-to-right direction is immediate, since an isomorphism  $\pi$  from  $G$  to  $H$  induces an isomorphism  $\rho$  from  $S(G)$  to  $S(H)$ , by letting  $\rho(x_i) = x_{\pi(i)}$  and  $\rho(x'_i) = x'_{\pi(i)}$ .

For the converse, suppose that  $\rho$  is a permutation of  $X$  such that  $\rho(S(G)) \equiv S(H)$ . As noted above, for all  $\{i, j\} \in E(G)$ , it holds that  $S(H) \rightarrow (\rho(x_i) \vee \rho(x_j))$  and  $S(H) \not\rightarrow (\rho(x_i) \oplus \rho(x_j))$ . Again by the observation above, there exists an edge

$\{k, \ell\} \in E(H)$  such that  $\{\rho(x_i), \rho(x_j)\} = \{x_k, x_\ell\}$ . Thus,  $\rho$  maps  $x$ -variables to  $x$ -variables. Let  $\pi(i) = j$  iff  $\rho(x_i) = x_j$ . It is easy to see that  $\pi$  is an isomorphism from  $G$  to  $H$ .

**3.** Let  $\widehat{G}$  be a graph such that  $V(\widehat{G}) = \{1, 2, \dots, n\}$  and  $E(\widehat{G}) = \{e_1, \dots, e_m\}$ . Define  $S(\widehat{G})$  as follows:  $S(\widehat{G}) = \{h(x_i, x_j, y_k, x'_i, x'_j, y'_k) \mid e_k = \{i, j\}\} \cup \{x_i \oplus x'_i \mid 1 \leq i \leq n\} \cup \{y_i \oplus y'_i \mid 1 \leq i \leq m\}$ .

Define  $U(\widehat{G})$  as follows:  $U(\widehat{G}) = \{\text{OneInThree}(x_i, x_j, y_k) \mid e_k = \{i, j\}\} \cup \{x_i \oplus x'_i \mid 1 \leq i \leq n\} \cup \{y_i \oplus y'_i \mid 1 \leq i \leq m\}$ . Clearly,  $U(\widehat{G})$  is equivalent to  $S(\widehat{G})$ . Let  $X$  be the set of all variables that occur in  $U(\widehat{G})$ .

We will use the following claim whose proof will be given after the proof of Lemma 16.

**Claim 17** *The set of all constraint applications of OneInThree without duplicates that occur in  $U(\widehat{G})$  is a maximal set of constraint applications of OneInThree over variables  $X$  without duplicates, where  $X$  is the set of all variables that occur in  $U(\widehat{G})$ .*

Let  $G$  and  $H$  be graphs without isolated vertices, with vertex set  $\{1, 2, \dots, n\}$ , and with the same number of edges. Also assume that every vertex in  $G$  and  $H$  is incident with at least two edges (see Lemma 14). Let  $E(G) = \{e_1, \dots, e_m\}$  and let  $E(H) = \{e'_1, \dots, e'_m\}$ .

We claim that  $G$  is isomorphic to  $H$  iff  $S(G)$  is isomorphic to  $S(H)$ . It suffices to show that  $G$  is isomorphic to  $H$  iff  $U(G)$  is isomorphic to  $U(H)$ .

The left-to-right direction is trivial, since an isomorphism between the graphs induces an isomorphism between sets of constraint applications as follows. If  $\pi : V \rightarrow V$  is an isomorphism from  $G$  to  $H$ , then we can define an isomorphism  $\rho$  from  $U(G)$  to  $U(H)$  as follows:

- $\rho(x_i) = x_{\pi(i)}$ ,  $\rho(x'_i) = x'_{\pi(i)}$ , for  $i \in V$ .
- For  $e_k = \{i, j\}$ ,  $\rho(y_k) = y_\ell$ , and  $\rho(y'_k) = y'_\ell$ , where  $e'_\ell = \{\pi(i), \pi(j)\}$ .

For the converse, suppose that  $\rho$  is an isomorphism from  $U(G)$  to  $U(H)$ . Let  $U'(G)$  and  $U'(H)$  be the sets of all constraint applications of OneInThree without duplicates that occur in  $U(G)$  and  $U(H)$ , respectively.

From Claim 17 and Lemma 13, it follows that  $\rho(U'(G)) = U'(H)$ . Note that the  $x$ -variables in  $U'(G)$  and  $U'(H)$  are exactly those variables that occur in at least two constraint applications of OneInThree without constants. Thus,  $\rho$  maps  $x$ -variables to  $x$ -variables. Likewise,  $\rho$  maps  $y$ -variables to  $y$ -variables. Let  $\pi$  be the bijection on  $\{1, 2, \dots, n\}$  defined by  $\pi(i) = j$  iff  $\rho(x_i) = x_j$ . We claim that  $\pi$  is an isomorphism from  $G$  to  $H$ . First let  $e_k = \{i, j\}$ . Thus,  $\text{OneInThree}(x_i, x_j, y_k) \in U'(G)$ . Then,  $\text{OneInThree}(\rho(x_i), \rho(x_j), \rho(y_k)) \in U'(H)$ . Thus,  $\text{OneInThree}(x_{\pi(i)}, x_{\pi(j)}, \rho(y_k)) \in U'(H)$ . This implies that  $\{\pi(i), \pi(j)\}$  is an edge in  $H$ . For the converse, suppose that  $e'_k = \{\pi(i), \pi(j)\}$  is an edge in  $H$ . Then  $\text{OneInThree}(x_{\pi(i)}, x_{\pi(j)}, y_k) \in U'(H)$ . Then,  $\text{OneInThree}(\rho(x_i), \rho(x_j), y_k) \in U'(H)$ . Then  $\text{OneInThree}(x_i, x_j, \rho^{-1}(y_k)) \in U'(G)$ . It follows that  $\{i, j\}$  is an edge in  $G$ .  $\square$

**Proof of Claim 17.** Suppose for a contradiction that  $a$ ,  $b$ , and  $c$  are three distinct variables in  $X$  such that  $U(\widehat{G}) \rightarrow \text{OneInThree}(a, b, c)$  and  $\text{OneInThree}(a, b, c) \notin U(\widehat{G})$ .

First note that that it cannot be the case that  $\{a, b, c\}$  contains  $\{x_i, x'_i\}$  or  $\{y_i, y'_i\}$  for some  $i$ , since that would imply that  $U(\widehat{G}) \rightarrow \neg d$  for some variable  $d \in X$ . But clearly, there exists a satisfying assignment for  $U(\widehat{G})$  such that the value of  $d$  is 1.

Secondly, note that if we set all  $x'$ -variables and all  $y$ -variables to 1, and all other variables in  $X$  to 0, we obtain a satisfying assignment for  $U(\widehat{G})$ . It follows that exactly one variable in  $\{a, b, c\}$  is an  $x'$ -variable or a  $y$ -variable. The proof consists of a careful analysis of the different cases. We will show that in each case, there exists an assignment that satisfies  $U(\widehat{G})$  but that does not satisfy  $\text{OneInThree}(a, b, c)$ , which contradicts the assumption that  $U(\widehat{G}) \rightarrow \text{OneInThree}(a, b, c)$ .

1. If  $\{a, b, c\} = \{x_i, x_j, y_k\}$ , then, since  $\text{OneInThree}(a, b, c) \notin U(\widehat{G})$ ,  $e_k \neq \{i, j\}$ . Without loss of generality, let  $j \notin e_k$ . It is easy to see that there is a satisfying assignment for  $U(\widehat{G})$  such that  $y_k$  and  $x_j$  are set to 1. (Set all other  $x$ -variables to 0 and set  $y_\ell$  to 1 iff  $j \notin e_\ell$ .) Thus, we have an assignment that satisfies  $U(\widehat{G})$  but not  $\text{OneInThree}(a, b, c)$ .
2. If  $\{y'_\ell, y_k\} \subseteq \{a, b, c\}$ , note that  $k \neq \ell$ , by the observation made above. Let  $i$  be such that  $i \in e_\ell$  and  $i \notin e_k$ . Set  $x_i$  to 1, and set all other  $x$ -variables to 0. This can be extended to a satisfying assignment for  $S(\widehat{U})$ , and in this assignment,  $y_\ell$  is 0 (and thus  $y'_\ell$  is 1), and  $y_k = 1$ .
3. If  $\{x'_i, x_j\} \subseteq \{a, b, c\}$ . Then  $i \neq j$ . Set  $x_j$  to 1 and all other  $x$ -variables to 0. It is easy to see that this can be extended to a satisfying assignment for  $U(\widehat{G})$ .
4. If  $\{a, b, c\} = \{x'_i, y'_k, y'_\ell\}$ , then, if  $i \notin e_\ell$  and  $i \notin e_k$ , then set  $x_i$  to 1, set all other  $x$ -variables to 0, set  $y_r$  to 1 iff  $i \notin e_r$ , and extend this to a satisfying assignment for  $U(\widehat{G})$ . If  $i \in e_\ell$  or  $i \in e_k$ , assume without loss of generality that  $i \in e_k$ , set  $x_i$  to 0, set  $y_k$  to 0, and extend this to a satisfying assignment for  $U(\widehat{G})$ .

□

The constraints  $\text{OR}_0$ ,  $\text{OR}_1$ , and  $\text{OR}_2$  are the simplest non-affine constraints. However, it is not enough to show that GI reduces to the isomorphism problem for these simple cases. In order to prove that GI reduces to the isomorphism problem for all sets of constraints that are not affine, we need to show that all such sets can “encode” a finite number of simple cases.

Different encodings are used in the lower bound proofs for different constraint problems. All encodings used in the literature however, allow the introduction of auxiliary variables. In [CKS01], Lemma 5.30, it is shown that if  $C$  is not affine, then  $C$  plus constants can encode  $\text{OR}_0$ ,  $\text{OR}_1$ , or  $\text{OR}_2$ . This implies that, for certain problems, lower bounds for  $\text{OR}_0$ ,  $\text{OR}_1$ , or  $\text{OR}_2$  transfer to  $C$  plus constants. However, their encoding uses auxiliary variables, which means that lower bounds for the isomorphism problem don't automatically transfer. For

sets of constraints that are not affine, we will be able to use part of the proof of [CKS01], Lemma 5.30, but we will have to handle auxiliary variables explicitly, which makes the constructions much more complicated.

**Theorem 18.** *If  $\mathcal{C}$  is not affine, then GI is polynomial-time many-one reducible to  $\text{ISO}_c(\mathcal{C})$ .*

**Proof.** First suppose that  $\mathcal{C}$  is weakly negative and weakly positive. Then  $\mathcal{C}$  is bijective [CH96]. From the proof of [CKS01], Lemma 5.30, which crucially uses Schaefer's characterization of Boolean functions (see Lemma 3), it follows that there exists a constraint application  $A(x, y, z)$  of  $\mathcal{C}$  with constants such that  $A(0, 0, 0) = A(0, 1, 1) = A(1, 0, 1) = 1$  and  $A(1, 1, 0) = 0$ . Since  $\mathcal{C}$  is weakly positive, we also have that  $A(1, 1, 1) = 1$ . Since  $\mathcal{C}$  is bijective, we have that  $A(0, 0, 1) = 1$ . The following truth-table summarizes all possibilities (this is a simplified version of [CKS01], Claim 5.31).

$xyz$	000	001	010	011	100	101	110	111
$A(x, y, z)$	1	1	$a$	1	$b$	1	0	1

Thus we obtain  $A(x, x, y) = (\bar{x} \vee y)$ . The required result follows from Lemma 16.1.

So, suppose that  $\mathcal{C}$  is not weakly negative or not weakly positive. We follow the proof of [CKS01], Lemma 5.30. From the proof of [CKS01], Lemma 5.26, it follows that there exists a constraint application  $A$  of  $\mathcal{C}$  with constants such that  $A(x, y) = \text{OR}_0(x, y)$ ,  $A(x, y) = \text{OR}_2(x, y)$ , or  $A(x, y) = x \oplus y$ . In the first two cases, the result follows from Lemma 16.1.

Consider the last case. From the proof of [CKS01], Lemma 5.30, there exist a set  $S(x, y, z, x', y', z')$  of  $\mathcal{C}$  constraint applications with constants and a ternary function  $h$  such that  $S(x, y, z, x', y', z') = h(x, y, z) \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$ ,  $h(000) = h(011) = h(101) = 1$ , and  $h(110) = 0$ .

The following truth-table summarizes all possibilities:

$xyz$	000	001	010	011	100	101	110	111
$h(x, y, z)$	1	$a$	$b$	1	$c$	1	0	$d$

We will first show that in most cases, there exists a set  $U$  of constraint applications of  $\mathcal{C}$  with constants such that  $U(x, y, x', y') = (x \vee y) \wedge (x \oplus x') \wedge (y \oplus y')$ . In all these cases, the result follows from Lemma 16.2 above.

- $b = 0, d = 1$ . In this case,  $S(x, y, x, x', y', x') = (x \vee y') \wedge (x \oplus x') \wedge (y \oplus y') = (x \vee y') \wedge (x \oplus x') \wedge (y \oplus y')$ .
- $b = 1, d = 0$ . In this case,  $S(x, y, x, x', y', x') = (x' \vee y') \wedge (x \oplus x') \wedge (y \oplus y')$ .
- $c = 0, d = 1$ . In this case,  $S(x, y, y, x', y', y') = (x' \vee y) \wedge (x \oplus x') \wedge (y \oplus y')$ .
- $c = 1, d = 0$ . In this case,  $S(x, y, y, x', y', y') = (x' \vee y') \wedge (x \oplus x') \wedge (y \oplus y')$ .
- $b = c = 1$ . In this case,  $S(x, y, 0, x', y', 1) = (x' \vee y') \wedge (x \oplus x') \wedge (y \oplus y')$ .
- $b = c = d = 0; a = 1$ . In this case,  $S(0, y, z, 1, y', z') = (y' \vee z) \wedge (y \oplus y') \wedge (z \oplus z')$ .

The previous cases are analogous to the cases from the proof of [CKS01], Claim 5.31. However, we have to explicitly add the  $\oplus$  conjuncts to simulate the negated variables used there, which makes Lemma 16.2 necessary.

The last remaining case is the case where  $a = b = c = d = 0$ . In the proof of [CKS01], Claim 5.31, it suffices to note that  $(\overline{y} \vee z) = \exists! x h(x, y, z)$ . But, since we are looking at isomorphism, we cannot ignore auxiliary variables. Our result uses a different argument and follows from Lemma 16.3 above and the observation that  $S(x, y, z, x', y', z') = \text{OneInThree}(x, y, z') \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$ .  $\square$

We now turn to the case where  $\mathcal{C}$  is affine, but not 2-affine. As before, we first show GI-hardness of a particular constraint. The proof uses similar constructions as the proof of Lemma 16.

**Lemma 19.** *Let  $h$  be the 6-ary constraint such that  $h(x, y, z, x', y', z') = (x \oplus y \oplus z) \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$ . GI is polynomial-time many-one reducible to  $\text{ISO}(\{h\})$ .*

**Proof.** Following [CKS01, p. 20], we use  $\text{XOR}_2$  to denote the constraint  $\lambda xy.x \oplus y$ , and  $\text{XOR}_3$  to denote  $\lambda xyz.x \oplus y \oplus z$ .

Let  $\widehat{G}$  be a graph such that  $V(\widehat{G}) = \{1, 2, \dots, n\}$ , and  $E(\widehat{G}) = \{e_1, e_2, \dots, e_m\}$ . We will use a similar encoding as in the proof of Lemma 16. Again, propositional variable  $x_i$  will correspond to vertex  $i$  and propositional variable  $y_i$  will correspond to edge  $e_i$ .

Define  $S(\widehat{G})$  as follows:  $S(\widehat{G}) = \{h(x_i, x_j, y_k, x'_i, x'_j, y'_k) \mid e_k = \{i, j\}\} \cup \{x_i \oplus x'_i \mid 1 \leq i \leq n\} \cup \{y_i \oplus y'_i \mid 1 \leq i \leq m\}$ .

Define  $U(\widehat{G})$  as follows:  $U(\widehat{G}) = \{(x_i \oplus x_j \oplus y_k), (x'_i \oplus x'_j \oplus y_k), (x'_i \oplus x_j \oplus y'_k), (x_i \oplus x'_j \oplus y'_k) \mid e_k = \{i, j\}\} \cup \{x_i \oplus x'_i \mid 1 \leq i \leq n\} \cup \{y_i \oplus y'_i \mid 1 \leq i \leq m\}$ . Clearly,  $U(\widehat{G})$  is equivalent to  $S(\widehat{G})$ . Let  $X$  be the set of variables occurring in  $U(\widehat{G})$ .

The proof relies on the following claim, which shows that  $U(\widehat{G})$  is a maximal set of constraint applications of  $\{\text{XOR}_2, \text{XOR}_3\}$  without duplicates. This claim will be proved after the proof of this lemma.

**Claim 20** *Let  $\widehat{G}$  be a triangle-free graph such that  $V(\widehat{G}) = \{1, 2, \dots, n\}$ ,  $E(\widehat{G}) = \{e_1, e_2, \dots, e_m\}$ , and every vertex has degree at least two. Then  $U(\widehat{G})$  is a maximal set of constraint applications of  $\{\text{XOR}_2, \text{XOR}_3\}$  without duplicates.*

Let  $G$  and  $H$  be graphs such that  $V(G) = V(H) = \{1, \dots, n\}$ ,  $E(G) = \{e_1, \dots, e_m\}$ ,  $E(H) = \{e'_1, \dots, e'_m\}$ , all vertices in  $G$  and  $H$  have degree at least two, and  $G$  and  $H$  do not contain triangles.

We will show that  $G$  is isomorphic to  $H$  if and only if  $S(G)$  is isomorphic to  $S(H)$ . It suffices to show that  $G$  is isomorphic to  $H$  if and only if  $U(G)$  is isomorphic to  $U(H)$ .

The left-to-right direction is trivial, since an isomorphism between the graphs induces an isomorphism between sets of constraint applications as follows. If  $\pi : V \rightarrow V$  is an isomorphism from  $G$  to  $H$ , then we can define an isomorphism  $\rho$  from  $U(G)$  to  $U(H)$  as follows:

- $\rho(x_i) = x_{\pi(i)}$ ,  $\rho(x'_i) = x'_{\pi(i)}$ , for  $i \in V$ .
- For  $e_k = \{i, j\}$ ,  $\rho(y_k) = y_\ell$ , and  $\rho(y'_k) = y'_\ell$ , where  $e'_\ell = \{\pi(i), \pi(j)\}$ .

For the converse, suppose that  $\rho$  is an isomorphism from  $U(G)$  to  $U(H)$ . By Lemma 13 and Claim 20,  $\rho(U(G)) = U(H)$ . Note that every  $y_i$  and  $y'_i$  variable occurs in exactly two constraint applications of  $\text{XOR}_3$  in  $U(G)$  and  $U(H)$ , while every  $x_i$  and  $x'_i$  variable occurs in at least four constraint applications of  $\text{XOR}_3$  in  $U(G)$  and  $U(H)$  (since every vertex in  $G$  and  $H$  is incident with at least two edges). From the  $\text{XOR}_2$  constraint applications, it is also immediate that for all  $a \in \{x_1, \dots, x_n, y_1, \dots, y_m\}$ , there exists a  $b \in \{x_1, \dots, x_n, y_1, \dots, y_m\}$  such that  $\{\rho(a), \rho(a')\} = \{b, b'\}$ .

Define  $\pi$  as follows:  $\pi(i) = j$  if and only if  $\{\rho(x_i), \rho(x'_i)\} = \{x_j, x'_j\}$ . By the observations above,  $\pi$  is total and 1-1. It remains to show that  $\{i, j\} \in E(G)$  iff  $\{\pi(i), \pi(j)\} \in E(H)$ .

Let  $e_k = \{i, j\}$ . Then  $x_i \oplus x_j \oplus e_k \in U(G)$ . Thus,  $\rho(x_i) \oplus \rho(x_j) \oplus \rho(y_k) \in U(H)$ . That is,  $a \oplus b \oplus \rho(y_k) \in U(H)$  for some  $a \in \{x_{\pi(i)}, x'_{\pi(i)}\}$  and  $b \in \{x_{\pi(j)}, x'_{\pi(j)}\}$ . But that implies that  $\rho(y_k) \in \{y_\ell, y'_\ell\}$  where  $e'_\ell = \{\pi(i), \pi(j)\}$ . This implies that  $\{\pi(i), \pi(j)\} \in E(H)$ . For the converse, suppose that  $\{\pi(i), \pi(j)\} \in E(H)$ . Then  $x_{\pi(i)} \oplus x_{\pi(j)} \oplus y_\ell \in U(H)$  for  $e'_\ell = \{\pi(i), \pi(j)\}$ . It follows that  $a \oplus b \oplus \rho^{-1}(y_\ell) \in U(G)$  for some  $a \in \{x_i, x'_i\}$  and  $b \in \{x_j, x'_j\}$ . By the form of  $U(G)$ , it follows that  $\{i, j\} \in E(G)$ .

□

**Proof of Claim 20.** Suppose that  $a$  and  $b$  are two distinct variables in  $X$  such that  $U(\widehat{G}) \rightarrow (a \oplus b)$  and  $a \oplus b \notin U(\widehat{G})$ . It is easy to see that we can set  $a$  and  $b$  to 0, and extend this to a satisfying assignment of  $U(\widehat{G})$ , which is a contradiction.

Next, let  $a, b, c$  be three distinct variables in  $X$  such that  $U(\widehat{G}) \rightarrow (a \oplus b \oplus c)$  and  $a \oplus b \oplus c \notin U(\widehat{G})$ . Let  $\widehat{X} = \{x_1, \dots, x_n, y_1, \dots, y_m\}$ . Any assignment that sets all variables in  $\widehat{X}$  to 1, and all variables not in  $\widehat{X}$  to 0, satisfies  $U(\widehat{G})$ . It follows that either exactly one or exactly three elements of  $\{a, b, c\}$  are in  $\widehat{X}$ .

If exactly one of  $\{a, b, c\}$  is in  $\widehat{X}$ , assume without loss of generality that  $a \in \widehat{X}$ . If  $a' \in \{b, c\}$ , then, without loss of generality, let  $a' = b$ . In this case,  $U(\widehat{G}) \rightarrow \bar{b}$ . But this is a contradiction, since it is immediate that we can set  $c$  to 1 and extend this to a satisfying assignment of  $U(\widehat{G})$ . Next, let  $d, e \in \widehat{X}$  be such that  $d' = b$  and  $e' = c$ . In that case,  $a, d$ , and  $e$  are distinct variables in  $\widehat{X}$  such that  $U(\widehat{G}) \rightarrow a \oplus d \oplus e$  and  $a \oplus d \oplus e \notin U(\widehat{G})$ . This falls under the next case.

Finally, suppose that  $a, b, c$  are three distinct variables in  $\widehat{X}$  such that  $U(\widehat{G}) \rightarrow a \oplus b \oplus c$  and  $a \oplus b \oplus c \notin U(\widehat{G})$ . Let  $\widehat{U}(\widehat{G}) = \{(x_i \oplus x_j \oplus y_k) \mid e_k = \{i, j\}\}$ , i.e.,  $\widehat{U}(\widehat{G})$  consists of all constraint applications in  $U(\widehat{G})$  whose variables are in  $\widehat{X}$ . Since any assignment on  $\widehat{X}$  that satisfies  $\widehat{U}(\widehat{G})$  can be extended to a satisfying assignment of  $U(\widehat{G})$  (by letting  $a' = \bar{a}$  for all  $a \in \widehat{X}$ ), the desired result follows immediately from the following claim. □

**Claim 21**  $\widehat{U}(\widehat{G})$  is a maximal set of constraint applications of  $\text{XOR}_3$  without duplicates over variables  $\widehat{X}$ .



**Proof.** Suppose that  $a, b, c$  are three distinct variables in  $\widehat{X}$  such that  $\widehat{U}(\widehat{G}) \rightarrow a \oplus b \oplus c$  and  $a \oplus b \oplus c \notin \widehat{U}(\widehat{G})$ . The proof consists of a careful analysis of the different cases. We will show that in each case, there exists an assignment on  $\widehat{X}$  that satisfies  $\widehat{U}(\widehat{G})$  but not  $(a \oplus b \oplus c)$ , which contradicts the assumption that  $\widehat{U}(\widehat{G}) \rightarrow a \oplus b \oplus c$ .

It is important to note that any assignment to  $\{x_1, \dots, x_n\}$  can be extended to a satisfying assignment of  $\widehat{U}(\widehat{G})$ .

1. If  $a, b$ , and  $c$  are in  $\{x_1, \dots, x_n\}$ , then set  $a, b$ , and  $c$  to 0. This assignment can be extended to an assignment on  $\widehat{X}$  that satisfies  $x_i \oplus x_j \oplus y_k$  for  $e_k = \{i, j\}$ . So, we now have an assignment that satisfies  $\widehat{U}(\widehat{G})$  but does not satisfy  $(a \oplus b \oplus c)$ .
2. If exactly two of  $\{a, b, c\}$  are in  $\{x_1, \dots, x_n\}$ , then without loss of generality, let  $c = y_k$  for  $e_k = \{i, j\}$ . By the assumption that  $a \oplus b \oplus c$  is not in  $\widehat{U}(\widehat{G})$ , 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 0 and set  $\{x_1, \dots, x_m\} \setminus \{a\}$  to 1. This assignment can be extended to a satisfying assignment for  $\widehat{U}(\widehat{G})$ . Note that such an assignment will set  $y_k$  to 1. It follows that this assignment does not satisfy  $a \oplus b \oplus c$ .
3. If exactly one of  $\{a, b, c\}$  are in  $\{x_1, \dots, x_n\}$ , without loss of generality, let  $a \in \{x_1, \dots, x_n\}$ . Set  $a$  to 0 and  $b$  and  $c$  to 1. It is easy to see that this can be extended to a satisfying assignment for  $\widehat{U}(\widehat{G})$ .
4. If  $a, b$ , and  $c$  are in  $\{y_1, \dots, y_m\}$ , let  $a = y_{k_1}, b = y_{k_2}, c = y_{k_3}$  such that  $e_{k_\ell} = \{i_\ell, j_\ell\}$  for  $\ell \in \{1, 2, 3\}$ . First suppose that for every  $\ell \in \{1, 2, 3\}$ , for every  $x \in \{x_{i_\ell}, x_{j_\ell}\}$ , there exists an  $\ell' \in \{1, 2, 3\}$  with  $\ell' \neq \ell$  and a constraint application  $A$  in  $\widehat{U}(\widehat{G})$  such that  $x$  and  $y_{k_{\ell'}}$  occur in  $A$ . This implies that every vertex in  $\{i_1, j_1, i_2, j_2, i_3, j_3\}$  is incident with at least 2 of the edges in  $e_{k_1}, e_{k_2}, e_{k_3}$ . Since these three edges are distinct, it follows that the edges  $e_{k_1}, e_{k_2}, e_{k_3}$  form a triangle in  $\widehat{G}$ , which contradicts the assumption that  $\widehat{G}$  is triangle-free. So, let  $\ell \in \{1, 2, 3\}$ ,  $x \in \{x_{i_\ell}, x_{j_\ell}\}$  be such that for all  $\ell' \in \{1, 2, 3\}$  with  $\ell' \neq \ell$ ,  $x$  and  $y_{k_{\ell'}}$  do not occur in the same constraint application in  $\widehat{U}(\widehat{G})$ . Set  $x$  to 0 and set  $\{x_1, \dots, x_n\} \setminus \{x\}$  to 1. This can be extended to a satisfying assignment of  $\widehat{U}(\widehat{G})$  and such a satisfying assignment must have the property that  $y_{k_\ell}$  is 0 and  $y_{k_{\ell'}}$  is 1 for all  $\ell' \in \{1, 2, 3\}$  such that  $\ell' \neq \ell$ .

□

**Theorem 22.** *If  $\mathcal{C}$  is affine and not bijunctive, then GI is polynomial-time many-one reducible to  $\text{ISO}_c(\mathcal{C})$ .*

**Proof.** Recall from the proof of Theorem 18 that if  $\mathcal{C}$  is not bijunctive, then  $\mathcal{C}$  is not weakly positive or not weakly negative. As in the proof of that theorem, it follows from the the proof of [CKS01], Lemma 5.26 that there exists a constraint application  $A$  of  $\mathcal{C}$  with constants such that  $A(x, y) = \text{OR}_0(x, y)$ ,  $A(x, y) = \text{OR}_2(x, y)$ , or  $A(x, y) = x \oplus y$ . Since  $A$  is affine, and  $\text{OR}_0$  and  $\text{OR}_2$  are not affine, the first two cases cannot occur.

Consider the last case. Let  $B \in \mathcal{C}$  be a constraint that is not bijective. Let  $k$  be the arity of  $B$ . Following Schaefer's characterization of bijective functions (see Lemma 3), there exist assignments  $s, t, u \in \{0, 1\}^k$  such that  $B(s) = B(t) = B(u) = 1$  and  $B(\text{majority}(s, t, u)) = 0$ . In addition, since  $\mathcal{C}$  is affine, using Schaefer's characterization of affine functions,  $B(s \oplus t \oplus u) = 1$ .

Let  $\widehat{B}(x, y, z, x', y', z') = B(x_1, \dots, x_k)$  be the constraint application of  $B$  with constants that results if for all  $1 \leq i \leq k$ ,  $x_i =$

- 1 if  $s_i = t_i = u_i = 1$ ,
- 0 if  $s_i = t_i = u_i = 0$ ,
- $x$  if  $s_i = t_i = 0$  and  $u_i = 1$ ,
- $x'$  if  $s_i = t_i = 1$  and  $u_i = 0$ ,
- $y$  if  $s_i = u_i = 0$  and  $t_i = 1$ ,
- $y'$  if  $s_i = u_i = 1$  and  $t_i = 0$ ,
- $z$  if  $s_i = 0$  and  $t_i = u_i = 1$ ,
- $z'$  if  $s_i = 1$  and  $t_i = u_i = 0$ .

Note that  $\widehat{B}(0, 0, 0, 1, 1, 1) = B(s) = 1$ ,  $\widehat{B}(0, 1, 1, 1, 0, 0) = B(t) = 1$ ,  $\widehat{B}(1, 0, 1, 0, 1, 0) = B(u) = 1$ ,  $\widehat{B}(1, 1, 0, 0, 0, 1) = B(s \oplus t \oplus u) = 1$ , and  $\widehat{B}(0, 0, 1, 1, 1, 0) = B(\text{majority}(s, t, u)) = 0$ .

Let  $S = \{\widehat{B}(x, y, z, x', y', z'), A(x, x'), A(y, y'), A(z, z')\}$ . Then  $S$  is a set of constraint applications of  $\mathcal{C}$  with constants such that there exists a ternary function  $h$  such that  $S(x, y, z, x', y', z') = h(x, y, z) \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$  and  $h(000) = h(011) = h(101) = h(110) = 1$  and  $h(001) = 0$ .

The following table summarizes the possibilities we have.

$xyz$	000	001	010	011	100	101	110	111
$h(x, y, z)$	1	0	$a$	1	$b$	1	1	$c$

We will analyze all cases.

- $a = 1$ . In this case,  $S(0, y, z, 1, y', z') = (y \vee z') \wedge (y \oplus y') \wedge (z \oplus z')$ , and the result follows from Lemma 16.2.
- $b = 1$ . In this case,  $S(x, 0, z, x', 1, z') = (x \vee z') \wedge (x \oplus x') \wedge (z \oplus z')$ , and the result follows from Lemma 16.2.
- $b = 0$  and  $c = 1$ . In this case,  $S(1, y, z, 0, y', z') = (y \vee z) \wedge (y \oplus y') \wedge (z \oplus z')$ , and the result follows from Lemma 16.2.
- $a = b = c = 0$ . In this case,  $S(x', y, z, x, y', z') = (x \oplus y \oplus z) \wedge (x \oplus x') \wedge (y \oplus y') \wedge (z \oplus z')$ , and the result follows from Lemma 19.

□

**Acknowledgements:** We would like to thank Lane Hemaspaandra for helpful conversations and suggestions, and the anonymous referees for helpful comments.

## References

- [AK02] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. Technical Report 02-37, Electronic Colloquium on Computational Complexity, 2002.

- [AT00] M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM Journal on Computing*, 30(3):990–1009, 2000.
- [BHRV02] E. Böhrer, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 412–426, Berlin Heidelberg, 2002. Springer Verlag.
- [BRS98] B. Borchert, D. Ranjan, and F. Stephan. On the computational complexity of some classical equivalence relations on Boolean functions. *Theory of Computing Systems*, 31:679–693, 1998.
- [CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.
- [CKS01] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Monographs on Discrete Applied Mathematics. SIAM, 2001.
- [Cre95] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51:511–522, 1995.
- [FV98] T. Feder and M. Vardi. Monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.
- [Hor51] A. Horn. On sentences which are true of direct unions of algebras. In *Journal of Symbolic Logic*, 16:14–21, 1951.
- [Jub99] L. Juban. Dichotomy theorem for generalized unique satisfiability problem. In *Proceedings 12<sup>th</sup> Fundamentals of Computation Theory*, volume 1684 of *Lecture Notes in Computer Science*, pages 327–337. Springer Verlag, 1999.
- [KK01a] L. Kirousis and P. Kolaitis. A dichotomy in the complexity of propositional circumscription. In *Proceedings of the 16th Symposium on Logic in Computer Science*, pages 71–80, 2001.
- [KK01b] L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability problems. In *Proceedings 18<sup>th</sup> Symposium on Theoretical Aspects of Computer Science*, volume 2010, pages 407–418. Springer Verlag, 2001.
- [KS98] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal of Computing*, 28(1):152–163, 1998.
- [KST93] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, 1993.
- [KSTW01] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863 – 1920, 2001.
- [Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10<sup>th</sup> Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.
- [Tor00] J. Torán. On the hardness of graph isomorphism. In *Proceedings 41<sup>st</sup> Foundations of Computer Science*, pages 180–186, 2000.