An Algebraic Approach to the Complexity of Generalized Conjunctive Queries*

Michael Bauland[1], Philippe Chapdelaine[2], Nadia Creignou[3], Miki Hermann[4], and Heribert Vollmer[1]

[1] Theoretische Informatik, Universität Hannover, Germany. bauland|vollmer@thi.uni-hannover.de
[2] GREYC (UMR 6072), Université de Caen, France. pchapdel@info.unicaen.fr
[3] LIF (UMR 6166), Université de la Méditerranée, France. creignou@lidil.univ-mrs.fr
[4] LIX (FRE 2653), École Polytechnique, France. hermann@lix.polytechnique.fr

Abstract. Kolaitis and Vardi pointed out that constraint satisfaction and conjunctive query containment are essentially the same problem. We study the Boolean conjunctive queries under a more detailed scope, where we investigate their counting problem by means of the algebraic approach through Galois theory, taking advantage of Post's lattice. We prove a trichotomy theorem for the generalized conjunctive query counting problem, showing this way that, contrary to the corresponding decision problems, constraint satisfaction and conjunctive-query containment differ for other computational goals. We also study the audit problem for conjunctive queries asking whether there exists a frozen variable in a given query. This problem is important in databases supporting statistical queries. We derive a dichotomy theorem for this audit problem that sheds more light on audit applicability within database systems.

1 Introduction

Constraint satisfaction is recognized as a fundamental problem in artificial intelligence, in automated deduction, in computer-aided verification, in operations research, etc. At the same time conjunctive-query containment is considered as a fundamental problem in database query evaluation and optimization. Kolaitis and Vardi pointed out in [KV00] that constraint satisfaction and conjunctive-query containment are essentially the same problem. Constraints are usually specified by means of relations. The standard constraint satisfaction problem can therefore be parameterized by restricting the set of allowed relations. In particular, given a finite set S of Boolean relations, we consider conjunctive propositional formulas consisting of clauses built over relations from S, also called S-formulas. Deciding the satisfiability of such an S-formula is known as the *generalized satisfiability problem*, denoted by $\textsc{sat}(S)$, and was first investigated by Schaefer [Sch78]. It turns out that the complexity of $\textsc{sat}(S)$ can be characterized by closure properties of S. This correspondence is obtained through a generalization of Galois theory. In order to get complexity results via this algebraic approach, conjunctive queries $\textsc{coq}(S)$ over a set of relations S turn out to be useful. Roughly speaking, a conjunctive query from $\textsc{coq}(S)$ is an S-formula with distinguished variables, where all non-distinguished variables are existentially quantified. These queries play an important role in database theory, since they represent a broad class of queries and their expressive power is equivalent to select-join-project queries in relational algebra. Thus they are also of interest in their own right and we study the complexity of some related computational problems. The algebraic approach is particularly well adapted to this study, yielding short and elegant proofs.

We focus here on the counting and the audit problems for conjunctive queries. In the former the problem is to count the number of entries in the database that match the query, i.e., the number of satisfying assignments. In the latter the problem is to audit a database to ensure protection of sensitive data, where the goal is to decide whether the conjunctive query evaluates to false or whether there is some distinguished variable that is frozen, i.e., that takes the same value in all satisfying assignments. This frozen variable would then be considered as not protected. This is a generalization of the audit problem for Boolean attributes defined in [KPR03] (see also [KJ03]), which is particularly interesting in databases supporting statistical queries. For both considered

* Supported by ÉGIDE 05835SH, DAAD D/0205776 and DFG VO 630/5-1.

problems we obtain a complete complexity classification that indicates a difference with respect to satisfiability problems of Boolean constraints.

We think that our results will have an impact on concrete database implementations and applications, since the considered formulas in our computational problems correspond better to the model of queries formulated within existing database systems than the so far mainly studied S-formulas.

2 Preliminaries

Throughout the paper we use the standard correspondence between predicates and relations. We use the same symbol for a predicate and its corresponding relation, since the meaning will always be clear from the context, and we say that the predicate *represents* the relation.

An n-ary *logical relation* R is a Boolean relation of arity n. Each element of a logical relation R is an n-ary Boolean vector $m = (m_1, \dots, m_n) \in \{0,1\}^n$. Let V be a set of variables. A *constraint* is an application of R to an n-tuple of variables from V, i.e., $R(x_1, \dots, x_n)$. An assignment $I\colon V \to \{0,1\}$ satisfies the constraint $R(x_1, \dots, x_n)$ if $(I(x_1), \dots, I(x_n)) \in R$ holds.

Example 1. Equivalence is the binary relation defined by $\mathrm{Eq} = \{(0,0),(1,1)\}$. Given the ternary relations $R_{\mathrm{nae}} = \{0,1\}^3 \smallsetminus \{(0,0,0),(1,1,1)\}$ and $R_{1/3} = \{(1,0,0),(0,1,0),(0,0,1)\}$, the constraint $R_{\mathrm{nae}}(x,y,z)$ is satisfied if not all variables are assigned the same value and the constraint $R_{1/3}(x,y,z)$ is satisfied if exactly one of the variables x, y, and z is assigned to 1.

Throughout the text we refer to different types of Boolean constraint relations following Schaefer's terminology [Sch78]. We say that a Boolean relation R is

- *1-valid* if $(1,\dots,1) \in R$ and it is *0-valid* if $(0,\dots,0) \in R$,
- *Horn* (*dual Horn*) if R can be represented by a conjunctive normal form (CNF) formula having at most one unnegated (negated) variable in each clause,
- *bijunctive* if it can be represented by a CNF formula having at most two variables in each clause,
- *affine* if it can be represented by a conjunction of linear functions, i.e., a CNF formula with $\oplus$-clauses (XOR-CNF),
- *complementive* if for each $(\alpha_1, \dots, \alpha_n) \in R$ also $(\neg\alpha_1, \dots, \neg\alpha_n) \in R$.

A set S of Boolean relations is called 0-valid (1-valid, Horn, dual Horn, affine, bijunctive, complementive) if every relation in S is 0-valid (1-valid, Horn, dual Horn, affine, bijunctive, complementive).

Let S be a non-empty finite set of Boolean relations. An *S-formula* is a finite conjunction of clauses $\varphi = c_1 \wedge \cdots \wedge c_k$, where each clause c_i is a constraint application of some logical relation $R \in S$. An assignment I satisfies the formula φ if it satisfies all clauses c_i. We denote by $\mathrm{sol}(\varphi)$ the set of satisfying assignments of a formula φ.

Schaefer in his seminal paper [Sch78] developed a complexity classification of the satisfiability problem of S-formulas, denoted by $\mathrm{SAT}(S)$. *Conjunctive queries* turn out to be useful in order to obtain this result. Given a set S of Boolean relations, we denote by $\mathrm{COQ}(S)$ the set of all formulas of the form

$$F(x_1, \dots, x_k) \quad = \quad \exists y_1 \exists y_2 \cdots \exists y_l \; \varphi(x_1, \dots, x_k, y_1, \dots, y_l),$$

where φ is an S-formula. These formulas are called *conjunctive queries over S* [KV00], with $\boldsymbol{x} = \{x_1, \dots, x_k\}$ being the *distinguished variables*.

3 Closure Properties of Constraints

There exist easy criteria to determine if a given relation is Horn, dual Horn, bijunctive, or affine. We recall these properties here briefly for completeness. An interested reader can find a more detailed description with proofs in the paper [Sch78] or in the monograph [CKS01]. The operations of

$\mathrm{Pol}(R) \supseteq \mathrm{E}_2$	$\Leftrightarrow$	R is Horn	$\mathrm{Pol}(R) \supseteq \mathrm{V}_2$	$\Leftrightarrow$	R is dual Horn
$\mathrm{Pol}(R) \supseteq \mathrm{D}_2$	$\Leftrightarrow$	R is bijunctive	$\mathrm{Pol}(R) \supseteq \mathrm{L}_2$	$\Leftrightarrow$	R is affine
$\mathrm{Pol}(R) \supseteq \mathrm{N}_2$	$\Leftrightarrow$	R is complementive	$\mathrm{Pol}(R) \supseteq \mathrm{N}$	$\Leftrightarrow$	R is complementive, 0- and 1-valid
$\mathrm{Pol}(R) \supseteq \mathrm{I}_0$	$\Leftrightarrow$	R is 0-valid	$\mathrm{Pol}(R) \supseteq \mathrm{I}_1$	$\Leftrightarrow$	R is 1-valid
$\mathrm{Pol}(R) \supseteq \mathrm{I}$	$\Leftrightarrow$	R is 0- and 1-valid	$\mathrm{Pol}(R) \supseteq \mathrm{I}_2$	$\Leftrightarrow$	R is Boolean

Fig. 1. Polymorphism correspondences

conjunction, disjunction, majority, and addition applied coordinate-wise on n-ary Boolean vectors $m, m', m'' \in \{0,1\}^n$ are defined as follows:

$$\begin{aligned} m \wedge m' &= (m[1] \wedge m'[1], \ldots, m[n] \wedge m'[n]) \\ m \vee m' &= (m[1] \vee m'[1], \ldots, m[n] \vee m'[n]) \\ \mathrm{maj}(m, m', m'') &= (m \vee m') \wedge (m' \vee m'') \wedge (m'' \vee m) \\ m \oplus m' &= (m[1] \oplus m'[1], \ldots, m[n] \oplus m'[n]) \end{aligned}$$

where $m[i]$ is the i-th coordinate of the vector m and $\oplus$ is the exclusive-or operator. Given a logical relation R, the following *closure properties* fully determine the structure of R.

- R is Horn if and only if $m, m' \in R$ implies $m \wedge m' \in R$.
- R is dual Horn if and only if $m, m' \in R$ implies $m \vee m' \in R$.
- R is bijunctive if and only if $m, m', m'' \in R$ implies $\mathrm{maj}(m, m', m'') \in R$.
- R is affine if and only if $m, m', m'' \in R$ implies $m \oplus m' \oplus m'' \in R$.

The notion of closure property of a Boolean relation has been defined more generally, see for instance [JCG97,Pip97]. Let $f \colon \{0,1\}^k \to \{0,1\}$ be a Boolean function of arity k. We say that R is *closed under* f, or that f is a *polymorphism* of R, if for any choice of k vectors $m_1, \ldots, m_k \in R$, we have that

$$\Big(f\big(m_1[1], \ldots, m_k[1]\big),\ f\big(m_1[2], \ldots, m_k[2]\big),\ \ldots,\ f\big(m_1[n], \ldots, m_k[n]\big)\Big) \in R,$$

i.e., that the new vector constructed coordinate-wise from $m_1, \ldots, m_k$ by means of f belongs to R.

We denote by $\mathrm{Pol}(R)$ the set of all polymorphisms of R and by $\mathrm{Pol}(S)$ the set of Boolean functions that are polymorphisms of every relation in S. It turns out that $\mathrm{Pol}(S)$ is a *closed set of Boolean functions* for every set of relations S. All closed classes of Boolean functions were identified by Post [Pos41]. Post also detected the inclusion structure of these classes, what is now referred to as *Post's lattice*, presented in Fig. 2 with the notation from [BCRV03]. The correspondence of the most studied classes with respect to the polymorphisms of a relation R is presented in Fig. 1. The class I_2 is the closed class of Boolean functions generated by the identity function, thus for every Boolean relation R we have $\mathrm{Pol}(R) \supseteq \mathrm{I}_2$.

An interesting Galois correspondence has been exhibited between the sets of Boolean functions $\mathrm{Pol}(S)$ and the sets of Boolean relations S. A basic introduction to this correspondence can be found in [Pip97,Pös01] and a comprehensive study in [PK79]. This theory helps us to get elegant and short proofs for results concerning the complexity of conjunctive queries. Indeed, it shows that the smaller the set of polymorphisms is, the more expressive the corresponding conjunctive queries are, what is the cornerstone for applying the algebraic method to complexity (see [BCRV04] for a survey).

Proposition 2. *Let S_1 and S_2 be two finite sets of Boolean relations. If the relation* $\mathrm{Pol}(S_2) \subseteq \mathrm{Pol}(S_1)$ *holds then* $\mathrm{COQ}(S_1 \cup \{\mathrm{Eq}\}) \subseteq \mathrm{COQ}(S_2 \cup \{\mathrm{Eq}\})$.

4 Complexity Results

The only difference between conjunctive queries and S-formulas is that the former contain some existentially quantified variables, thus distinguishing the remaining ones. While this certainly does not lead to a different complexity of the satisfiability problem, this is not any more the case for other computational goals, such as counting the number of satisfying assignments. The algebraic correspondence described above is useful to determine the complexity of the satisfiability problem, since it proves that the complexity of SAT-COQ(S) strongly depends on the set Pol(S), as shown in Proposition 2. It provides a polynomial-time reduction from the problem SAT-COQ(S_1) to SAT-COQ($S_2 \cup \{\mathrm{Eq}\}$) by locally replacing each S_1-conjunctive-query by its equivalent constraint in COQ($S_2 \cup \{\mathrm{Eq}\}$). Moreover, the equivalence relation is actually superfluous, since we can delete all equivalence constraints and identify the corresponding equivalent variables in the rest of the formula. This proves that SAT-COQ(S_1) is polynomial-time reducible to SAT-COQ(S_2). We will show in the sequel that the algebraic approach is helpful to study the complexity of the counting and the audit problems for conjunctive queries.

4.1 Introduction to Counting Problems and Their Reducibilities

A *counting problem* is typically presented using a suitable *witness* function which for every input x, returns a set of *witnesses* for x. Formally, a *witness* function is a function $w\colon \Sigma^* \longrightarrow \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are two alphabets, and $\mathcal{P}^{<\omega}(\Gamma^*)$ is the collection of all finite subsets of Γ^*. Every such witness function gives rise to the following *counting problem*: given a string $x \in \Sigma^*$, find the cardinality $|w(x)|$ of the *witness* set $w(x)$.

Let Σ, Γ be two alphabets and let $R \subseteq \Sigma^* \times \Gamma^*$ be a binary relation between strings such that, for each $x \in \Sigma^*$, the set $R(x) = \{y \in \Gamma^* \mid R(x, y)\}$ is finite. We write $\#R$ to denote the following counting problem: given a string $x \in \Sigma^*$, find the cardinality $|R(x)|$ of the witness set $R(x)$ associated with x. It is easy to see that every counting problem is of the form $\#R$ for some R.

Valiant [Val79a,Val79b] was the first to investigate the computational complexity of counting problems. To this effect, he introduced the class #P of counting functions that count the number of accepting paths of nondeterministic polynomial-time Turing machines. The prototypical problem in #P is #SAT, which is the counting version of Boolean satisfiability. Valiant [Val79a] showed that #SAT is #P-*complete* via *parsimonious* reductions, that is, every counting problem in #P can be reduced to #SAT via a polynomial-time reduction that preserves the cardinalities of the witness sets. Creignou and Hermann [CH96] proved that the counting problem #SAT(S) of S-formulas is dichotomic: #SAT(S) is in FP if S is a set of affine relations, otherwise the problem is #P-complete.

Hemaspaandra and Vollmer [HV95] have introduced higher complexity counting classes using a predicate-based framework that focuses on the complexity of membership in the witness sets. Specifically, if $\mathcal{C}$ is a complexity class of decision problems, then $\#\cdot\mathcal{C}$ is the class of all counting problems whose witness function w satisfies the following conditions:

1. There is a polynomial $p(n)$ such that for every x and every $y \in w(x)$, we have that $|y| \leq p(|x|)$, where $|x|$ is the length of x and $|y|$ is the length of y.
2. The witness recognition problem "given x and y, is $y \in w(x)$?" is in $\mathcal{C}$.

In particular, $\#\cdot$NP is the class of counting problems associated with decision problems, for which the witness size is polynomially bounded and the witness recognition problem is in NP. Following Toda [Tod91], the inclusions $\#\cdot\Sigma_k\mathrm{P} \subseteq \#\cdot\Pi_k\mathrm{P}$ and $\#\cdot\Pi_k\mathrm{P} \subseteq \#\cdot\Sigma_{k+1}\mathrm{P}$ among counting classes hold for each k. In particular, we have the inclusion $\#\cdot\mathrm{NP} \subseteq \#\cdot\mathrm{coNP}$.

Following Valiant [Val79a], we say that a reduction is *parsimonious* if it is a polynomial-time many-one reduction preserving the number of solutions. However, this reduction does not allow to prove completeness of many known #P-complete problems. Valiant [Val79b] used counting reductions in his #P-completeness proofs, but the aforementioned counting classes are not closed under this reduction, following Toda and Watanabe [TW92]. More useful for counting problems are *subtractive reductions* [DHK00]. They allow us to obtain many completeness results and at the same time they leave the $\#\cdot\Pi_k\mathrm{P}$ classes closed. Nevertheless, these reductions do not seem to be well-suited for our purposes. Indeed, we need to express the operation of halving the witness set, which is quite delicate if we require the closure of the counting classes under these reductions. For this purpose, we define the *complementive reductions* which satisfy the aforementioned requirements, provided that every witness set of the target counting problem is complementive.

A finite alphabet Σ is called *even* if $|\Sigma| = 2k$ for some $k \in \mathbb{N}$. A permutation π on an even alphabet Σ is called *bipartite* if there exists a partition of Σ into two disjoint sets Σ_0 and Σ_1 such that the following conditions hold:

– $\Sigma = \Sigma_0 \cup \Sigma_1$

– $\Sigma_0 \cap \Sigma_1 = \emptyset$
– $|\Sigma_0| = |\Sigma_1|$
– for all $x \in \Sigma_i$ we have $\pi(x) \in \Sigma_{1-i}$ for each $i = 0, 1$.

We homomorphically enlarge every permutation π on Σ to the strings Σ^* by means of the identity $\pi(x_1 \cdots x_k) = \pi(x_1) \cdots \pi(x_k)$ for each string $x_1 \cdots x_k \in \Sigma^*$.

A set of strings $A \subseteq \Sigma^*$ over an even alphabet Σ is called *complementive* if there exists a bipartite permutation π_A on Σ such that $x \in A$ holds if and only if $\pi_A(x) \in A$. If we know that a set of strings A is complementive, we always assume that we are effectively given the permutation π_A.

Definition 3. *Let Σ, Γ be two alphabets, Γ being even, and let $\#A$ and $\#B$ be two counting problems determined by the binary relations A and B between the strings from Σ and Γ, where $B(y)$ is complementive for every string $y \in \Sigma^*$.*

– *We say that the counting problem $\#A$ reduces to the counting problem $\#B$ via a **strong complementive reduction**, if there exists two polynomial-time computable functions f and g such that for every string $x \in \Sigma^*$:*
  - *$B(g(x)) \subseteq B(f(x))$*
  - *$2 \cdot |A(x)| = |B(f(x))| - |B(g(x))|$*
– *A **complementive reduction** $\#A \leq_{cr} \#B$ from a counting problem $\#A$ to $\#B$ is a transitive closure of strong complementive and parsimonious reductions.*

It is clear that complementive reductions present a special case of counting reductions, the most frequently used reductions among counting problems.

Theorem 4. $\#\mathrm{P}$ *and all higher complexity classes $\#{\cdot}\Pi_k\mathrm{P}$, $k \geq 1$, are closed under complementive reductions.*

Proof. Let k be a fixed nonnegative integer. We prove that the class $\#{\cdot}\Pi_k\mathrm{P}$ is closed under strong complementive reductions. The result will follow by induction on the number of strong complementive and parsimonious reductions used to compose the final complementive reduction. Recall that Toda [Tod91] showed that $\#{\cdot}\mathrm{P}^{\Sigma_k\mathrm{P}} = \#{\cdot}\Pi_k\mathrm{P}$.

Let $\#A$ and $\#B$ be two counting problems such that $\#B \in \#\Sigma_k\mathrm{P}$, $B(y)$ is complementive for every string $y \in \Sigma^*$, and $\#A$ reduces to $\#B$ via a strong complementive reduction. We will show that $\#A$ belongs to $\#{\cdot}\Pi_k\mathrm{P}$ by constructing a predicate A' in $\mathrm{P}^{\Sigma_k\mathrm{P}}$ such that for each string x we have $2 \cdot |A'(x)| = |B(f(x))| - |B(g(x))| = 2 \cdot |A(x)|$, where f and g are the required polynomial-time computable functions.

Let $*$ be a delimiter symbol not in the alphabets Σ and Γ. Let Γ_0 and Γ_1 are the partition sets defined by the bipartite permutation π on Γ. The predicate A' consists of all pairs (x, y') of strings x and y', such that y' is of the form $f(x) * g(x) * y$ with $(f(x), y) \in B$, $(g(x), y) \notin B$, and $\mathrm{last}(y) \in \Gamma_0$, where $\mathrm{last}(y)$ denotes the last symbol of the string y. Thus, a pair (x, y') belongs to A' if and only if (x, y') is accepted by the following algorithm:

1. extract $f(x)$, $g(x)$, and y from y';
2. check that $\mathrm{last}(y)$ belongs to Γ_0;
3. check that $(f(x), y)$ belongs to B;
4. check that $(g(x), y)$ does not belong to B.

Steps 1 and 2 take polynomial time. The test in Step 3 is in $\Pi_k\mathrm{P}$, therefore also in $\mathrm{P}^{\Sigma_k\mathrm{P}}$. The test in Step 4 is in $\Sigma_k\mathrm{P}$, hence it can be done in $\mathrm{P}^{\Sigma_k\mathrm{P}}$. Therefore the predicate A' is in $\mathrm{P}^{\Sigma_k\mathrm{P}}$. It is clear from the construction that the identity $2 \cdot |A'(x)| = |B(f(x))| - |B(g(x))|$ holds, since $B(y)$ is complementive for every string $x \in \Sigma^*$, and this implies $|A'(x)| = |A(x)|$. It follows that the counting problem $\#A$ is in $\#{\cdot}\mathrm{P}^{\Sigma_k\mathrm{P}} = \#{\cdot}\Pi_k\mathrm{P}$. If we take $k = 0$ in the proof, we get also the closure for the class $\#\mathrm{P}$. □

In view of the preceding Theorem 4, it is quite natural to ask whether the classes $\#{\cdot}\Sigma_k\mathrm{P}$ are also closed under complementive reductions. The following proposition provides the evidence that *no* class $\#{\cdot}\Sigma_k\mathrm{P}$ is closed under complementive reductions.

Proposition 5. *For every $k \in \mathbb{N}$, the counting class $\#\cdot\Sigma_k\mathrm{P}$ is not closed under complementive reductions, unless $\#\cdot\Sigma_k\mathrm{P} = \#\cdot\Pi_k\mathrm{P}$.*

Proof. We must perform a case analysis, whether k is even or odd. In the even case, take a $\Pi_{2i}\mathrm{P}$-formula $\varphi(x_1, \ldots, x_n)$ and construct the formulas

$$\begin{aligned}\tau(x_0, x_1, \ldots, x_n) &= x_0 \vee x_1 \vee \cdots \vee x_n \vee \neg x_0 \vee \neg x_1 \vee \cdots \vee \neg x_n \\ \psi(x_0, x_1, \ldots, x_n) &= \big(x_0 \wedge \neg\varphi(x_1, \ldots, x_n)\big) \vee \big(\neg x_0 \wedge \neg\varphi(\neg x_1, \ldots, \neg x_n)\big)\end{aligned}$$

where $\neg\varphi$ is formed from φ by de Morgan's laws. For the odd case, take a $\Pi_{2i+1}\mathrm{P}$-formula φ, maintain the same formula τ, and construct the formula

$$\psi(x_0, x_1, \ldots, x_n) = \big(x_0 \vee \neg\varphi(x_1, \ldots, x_n)\big) \wedge \big(\neg x_0 \vee \neg\varphi(\neg x_1, \ldots, \neg x_n)\big)$$

Both τ and ψ are complementive formulas, hence $\mathrm{sol}(\tau)$ and $\mathrm{sol}(\psi)$ are complementive sets of strings with $\Gamma_0 = \{0\}$ and $\Gamma_1 = \{1\}$.

The non-quantified part of the $\Pi_{2i}\mathrm{P}$ formula φ is in CNF, therefore the formulas $\neg\varphi(x_1, \ldots, x_n)$ and $\neg\varphi(\neg x_1, \ldots, \neg x_n)$ are in disjunctive normal form (DNF). Using the distributive law, both formulas $x_0 \wedge \neg\varphi(x_1, \ldots, x_n)$ and $\neg x_0 \wedge \neg\varphi(\neg x_1, \ldots, \neg x_n)$ can be transformed into DNF in polynomial time and linear space. Hence, the formula ψ is equivalent to a DNF-formula, which can be obtained in polynomial time and linear space. Similarly for the odd case, the non-quantified part of the $\Pi_{2i+1}\mathrm{P}$ formula φ is in DNF, therefore the formulas $\neg\varphi(x_1, \ldots, x_n)$ and $\neg\varphi(\neg x_1, \ldots, \neg x_n)$ are in CNF. Using the distributive law, we can show that the final formula ψ can be transformed in polynomial time and linear space into an equivalent CNF-formula.

In both cases, it is clear that $\mathrm{sol}(\psi) \subseteq \mathrm{sol}(\tau)$, $|\mathrm{sol}(\tau)| = 2 \cdot 2^n$, and $|\mathrm{sol}(\psi)| = 2 \cdot |\mathrm{sol}(\neg\varphi)| = 2 \cdot \big(2^n - |\mathrm{sol}(\varphi)|\big)$. Thus we conclude that $2 \cdot |\mathrm{sol}(\varphi)| = |\mathrm{sol}(\tau)| - |\mathrm{sol}(\psi)|$. Hence, we have a complementive reduction from a $\#\cdot\Pi_k\mathrm{P}$-complete problem to a counting problem in $\#\cdot\Sigma_k\mathrm{P}$. □

4.2 The Counting Problem of Conjunctive Queries

The counting problem associated with the satisfiability of generalized conjunctive queries is defined as follows.

Problem: #SAT-COQ(S)
Input: A conjunctive query $F(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ from COQ(S).
Output: Number of different satisfying assignments to the distinguished variables $\boldsymbol{x}$.

Our ultimate goal is to determine the complexity of #SAT-COQ(S) for all possible sets S. Observe first that #SAT-COQ(S) is in $\#\cdot\mathrm{NP}$ for every set of Boolean relations S.

Proposition 6. *Let S_1 and S_2 be two finite sets of Boolean relations. If the inclusion* $\mathrm{Pol}(S_2) \subseteq \mathrm{Pol}(S_1)$ *holds, then there exists a parsimonious reduction from* #SAT-COQ(S_1) *to* #SAT-COQ(S_2).

This result, together with Post's lattice, allows us to prove the following trichotomy complexity classification.

We need two propositions whose predecessors can already be found in a slightly different form in [CH93].

Proposition 7. #SAT($R_{1/3}$) *is* $\#\mathrm{P}$*-complete and* #SAT-COQ($R_{1/3}$) *is* $\#\cdot\mathrm{NP}$*-complete, both via parsimonious reductions.*

Proof. From Valiant's original results [Val79a] follows that #SAT is the generic #P-complete problem via parsimonious reductions. From the same reference and also from [DHK00] follows that #SAT-COQ is the generic $\#\cdot\mathrm{NP}$-complete counting problem under parsimonious reductions (see also [KST89]). It is clear that #SAT($R_{1/3}$) is in #P and #SAT-COQ($R_{1/3}$) is in $\#\cdot\mathrm{NP}$.

There exists a parsimonious reduction from #SAT to #3SAT as from SAT to 3SAT. Similarly, the same reduction exists from #SAT-COQ to#3SAT-COQ. Now for each clause $c = l_1 \vee l_2 \vee l_3$ of a 3SAT formula we construct the formula

$$r(c) = R_{1/3}(l_1, z_1, z_2) \wedge R_{1/3}(\neg l_2, z_1, z_3) \wedge R_{1/3}(\neg l_3, z_2, z_4) \wedge R_{1/3}(z_2, z_3, z_5)$$

where $z_1, \ldots, z_5$ are new variables. The resulting formula is the conjunction of these partial formulas $r(c)$ for all clauses c. This proves the required parsimonious reductions from $\#\textsc{sat}$ to $\#\textsc{sat}(R_{1/3})$ and from $\#\textsc{sat-coq}$ to $\#\textsc{sat-coq}(R_{1/3})$. □

Remark 8. There exists an alternative and shorter proof of Proposition 7 making use of algebraic arguments. We mention the proof here, since one of our goals is to promote the algebraic approach. The drawback of the proof is that you do not explicitly see the parsimonious reduction.

Proof. Since $\mathrm{Pol}(R_{1/3}) = \mathrm{I}_2$ and $\mathrm{I}_2 \subseteq S$ for every clone S, we conclude by Proposition 6 that $\#\textsc{sat-coq}(S)$ reduces to $\#\textsc{sat-coq}(R_{1/3})$ via parsimonious reductions. □

Proposition 9. *$\#\textsc{sat}(R_{\mathrm{nae}})$ is $\#\mathrm{P}$-complete and $\#\textsc{sat-coq}(R_{\mathrm{nae}})$ is $\#{\cdot}\mathrm{NP}$-complete, both via complementive reductions.*

Proof. It is clear that $\#\textsc{sat}(R_{\mathrm{nae}})$ is in $\#\mathrm{P}$ and $\#\textsc{sat-coq}(R_{\mathrm{nae}})$ is in $\#{\cdot}\mathrm{NP}$, respectively. For each clause $c = R_{1/3}(l_1, l_2, l_3)$ of a $\textsc{sat}(R_{1/3})$ formula we construct the formula

$$q(c) \quad = \quad R_{\mathrm{nae}}(l_1, l_2, z) \wedge R_{\mathrm{nae}}(l_2, l_3, z) \wedge R_{\mathrm{nae}}(l_3, l_1, z) \wedge R_{\mathrm{nae}}(l_1, l_2, l_3)$$

where z is a new variable. In case of conjunctive queries, z will be an existentially quantified variable. The resulting formula is the conjunction of these partial formulas $q(c)$ for all clauses c. Observe that the set of satisfying assignments for the formula $q(c)$ is complementive, therefore the resulting formula will have twice as many satisfying assignments as the original formula. This proves the required complementive reductions from $\#\textsc{sat}(R_{1/3})$ to $\#\textsc{sat}(R_{\mathrm{nae}})$ and from $\#\textsc{sat-coq}(R_{1/3})$ to $\#\textsc{sat-coq}(R_{\mathrm{nae}})$. Using Proposition 7, this proves the result. □

Theorem 10. *Let S be a non-empty finite set of Boolean relations.*

- *If S is affine, then $\#\textsc{sat-coq}(S)$ is in* FP.
- *Else if S is bijunctive, or Horn, or dual Horn, then $\#\textsc{sat-coq}(S)$ is $\#\mathrm{P}$-complete under counting reductions.*
- *Otherwise $\#\textsc{sat-coq}(S)$ is $\#{\cdot}\mathrm{NP}$-complete under complementive reductions.*

Proof. If S is affine, then the Gaussian elimination algorithm given in [CH96] for $\#\textsc{sat}(S)$ can also be used to construct a corresponding polynomial-time algorithm for $\#\textsc{sat-coq}(S)$.

If S is Horn, dual Horn, or bijunctive, then $\textsc{sat}(S)$ is in P following [Sch78] and therefore $\#\textsc{sat-coq}(S)$ is in $\#\mathrm{P}$. Moreover, we know from [CH96] that in this case $\#\textsc{sat}(S)$ is $\#\mathrm{P}$-hard. Hence, the trivial (parsimonious) reduction from $\#\textsc{sat}(S)$ to $\#\textsc{sat-coq}(S)$ finally shows that $\#\textsc{sat-coq}(S)$ is $\#\mathrm{P}$-complete.

It remains to treat the case where $\mathrm{Pol}(S) = \mathrm{N}$. In fact, observe that all the other nonconsidered classes N_2, I, I_0, I_1 or I_2 are subsets of N. Therefore according to Proposition 6 and Post's lattice, it suffices to exhibit a set S of Boolean relations, such that $\mathrm{N} \subseteq \mathrm{Pol}(S)$ but $\#\textsc{sat-coq}(S)$ is $\#{\cdot}\mathrm{NP}$-complete.

According to Proposition 9 we know that $\#\textsc{sat-coq}(R_{\mathrm{nae}})$ is $\#{\cdot}\mathrm{NP}$-complete via complementive reductions. Construct now the relations

$$\begin{aligned} R''(u,v,x,y,z) \quad &= \quad (\neg u \wedge \neg v \wedge \neg x \wedge \neg y \wedge \neg z) \vee (u \wedge v \wedge x \wedge y \wedge z) \qquad \text{and} \\ R'(u,v,x,y,z) \quad &= \quad R''(u,v,x,y,z) \vee (u \wedge \neg v \wedge R_{\mathrm{nae}}(x,y,z)) \vee (\neg u \wedge v \wedge R_{\mathrm{nae}}(x,y,z)). \end{aligned}$$

Consider now the formula $F(\boldsymbol{x}) = \exists \boldsymbol{y} \bigwedge_{i=1}^{m} R_{\mathrm{nae}}(x_1^i, x_2^i, x_3^i)$ being an instance of $\#\textsc{sat-coq}(\{R_{\mathrm{nae}}\})$. Build the formulas

$$F'(\boldsymbol{x},u,v) = \exists \boldsymbol{y} \bigwedge_{i=1}^{m} R'(u,v,x_1^i,x_2^i,x_3^i) \qquad \text{and} \qquad F''(\boldsymbol{x},u,v) = \exists \boldsymbol{y} \bigwedge_{i=1}^{m} R''(u,v,x_1^i,x_2^i,x_3^i)$$

from the relations R' and R''. The satisfying assignments of F' include those of F''. If q is the number of satisfying assignments of F then those of F' is $2q + 2$ and those of F'' is 2. Hence, we have the equality $2\,|\mathrm{sol}(F)| = |\mathrm{sol}(F')| - |\mathrm{sol}(F'')|$, implying a complementive reduction from $\#\textsc{sat-coq}(R_{\mathrm{nae}})$ to $\#\textsc{sat-coq}(\{R', R''\})$, proving that $\#\textsc{sat-coq}(\{R', R''\})$ is $\#{\cdot}\mathrm{NP}$-complete. Moreover, both R' and R'' are 0-valid, 1-valid, and complementive, since R_{nae} is complementive. Hence $\mathrm{Pol}(\{R', R''\})$ contains the class N. □

4.3 The Audit Problem

Another problem of interest, defined by Kleinberg *et al.* [KPR03] and studied from a complexity standpoint by Jonsson and Krokhin [JK03,KJ03], is the *audit problem.* This problem is related to databases that support statistical queries. It can be generalized to conjunctive queries in the following way.

Problem: AUDIT-COQ(S)
Input: A conjunctive query $F(\boldsymbol{x}) = \exists \boldsymbol{y}\ \varphi(\boldsymbol{x}, \boldsymbol{y})$ from COQ(S).
Question: Is F unsatisfiable or is there some variable among $\boldsymbol{x}$ that is frozen, i.e., that takes the same value in all satisfying assignments?

Note that our AUDIT-COQ(S) problem is different from the 1-AUDIT problem studied in [JK03], since we do not include the variable candidate to be frozen as part of the input.

It is easy to see that this problem belongs to the class coNP. We prove that the algebraic approach applies to study the complexity of this problem.

Proposition 11. *Let S_1 and S_2 be two finite sets of Boolean relations. If the inclusion* $\mathrm{Pol}(S_2) \subseteq \mathrm{Pol}(S_1)$ *holds, then* AUDIT-COQ(S_1) *is polynomial-time many-one reducible to* AUDIT-COQ(S_2).

Once more, this result together with Post's lattice allows us to get a complete complexity classification.

Theorem 12. *Let S be a non-empty finite set of Boolean relations.*

– *If S is both* 0- *and* 1-*valid, or affine, or Horn, or dual Horn or bijunctive, then* AUDIT-COQ(S) *is in* P.
– *Otherwise* AUDIT-COQ(S) *is* coNP-*complete.*

Proof. If S is both 0- and 1-valid, i.e., $\mathrm{I} \subseteq \mathrm{Pol}(S)$, then the problem is trivial.

If S is affine, Horn, dual Horn, or bijunctive, then observe that given an S-formula and a variable x, we can check in polynomial time whether both $F \wedge x$ and $F \wedge \neg x$ are satisfiable. Therefore, in this case AUDIT-COQ(S) is in P.

If S is complementive, but neither 0-valid, nor included in the four previous cases, i.e., $\mathrm{Pol}(S) = \mathrm{N}_2$, then no variable can be frozen. Therefore in this case the problem AUDIT-COQ(S) is equivalent to the coNP-complete problem UNSAT(S), asking whether an S-formula is unsatisfiable.

The remaining cases are those for which $\mathrm{Pol}(S) = \mathrm{I}_0$, I_1 or I_2. According to Proposition 6 and Post's lattice, in order to conclude the proof it suffices to exhibit a Boolean relation R_0 (resp. R_1) such that $\mathrm{I}_0 \subseteq \mathrm{Pol}(R_0)$ (resp. $\mathrm{I}_1 \subseteq \mathrm{Pol}(R_1)$) and AUDIT-COQ($R_0$) (resp. AUDIT-COQ($R_1$)) is coNP-complete. Recall first that SAT($R_{1/3}$) is NP-complete, so UNSAT($R_{1/3}$) is coNP-complete. Consider an instance of UNSAT($R_{1/3}$) defined by the formula $F(\boldsymbol{x}) = \bigwedge_{i=1}^{m} R_{1/3}(x_1^i, x_2^i, x_3^i)$. Construct the 0-valid relation

$$R_0(v, x, y, z) \quad = \quad (\neg v \wedge x \wedge y \wedge z) \vee (\neg v \wedge \neg x \wedge \neg y \wedge \neg z) \vee (v \wedge R_{1/3}(x, y, z))$$

and build the formula $F'(\boldsymbol{x}, v) = \bigwedge_{i=1}^{m} R_0(v, x_1^i, x_2^i, x_3^i)$. Clearly, the inclusion $\mathrm{I}_0 \subseteq \mathrm{Pol}(\{R_0\})$ holds since the relation R_0 is 0-valid.

Observe that F' is always satisfiable, that no variable among the $\boldsymbol{x}$ is frozen, and that F is unsatisfiable if and only if the variable v is frozen to 0 in F'. So, we have a reduction from UNSAT($R_{1/3}$) to AUDIT-COQ(R_0), therefore AUDIT-COQ(R_0) is coNP-complete. The proof is similar for $\mathrm{Pol}(S) = \mathrm{I}_1$, with a 1-valid relation R_1 similar to R_0, just flip the polarity of the variable v. □

5 Conclusion

While the complexity of conjunctive-query evaluation and constraint satisfaction is the same, we determined that this is not any more the case for other computational goals. We have shown that the counting problem for conjunctive queries has a different structure than that for conjunctive formulas. The latter displays a dichotomy behavior between the affine formulas in FP and the #P-complete other cases, as it was shown in [CH96], whereas the former presents a trichotomy structure between the affine cases in FP, the Horn, dual Horn, and bijunctive #P-complete cases, and finally the general #·NP-complete case. This shows that, under the more fine grained analysis presented by counting, the conjunctive queries present three different levels of (in)tractability. As a byproduct, we developed a new kind of reductions among counting problems, called the complementive reductions, that allow to use halving functions within the counting classes under certain circumstances , i.e., when every instance of the target set is complementive. Since there are many counting problems presenting this structure, we think that the complementive reductions will have a broader impact.

We have also shown that the corresponding audit problem for conjunctive queries displays a dichotomic behavior, where the cases of Horn, dual Horn, bijunctive, or both 0 and 1-valid constraints are in P, whereas the other cases are coNP-complete. It is interesting to observe that although the audit problem considered in Section 4.3 is different from that of Jonsson and Krokhin in [JK03], it presents the same dichotomy behavior.

Acknowledgment. We thank Elmar Böhler, Matthias Galota, and Steffen Reith for helpful discussions.

References

[BCRV03] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *SIGACT News, Complexity Theory Column 42*, 34(4):38–52, 2003.

[BCRV04] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *SIGACT News, Complexity Theory Column 43*, 35(1):22–35, 2004.

[CH93] N. Creignou and M. Hermann. On #P-completeness of some counting problems. Research report 2144, Institut de Recherche en Informatique et en Automatique, December 1993. URL = `http://www.lix.polytechnique.fr/~hermann/publications/satcount.ps.gz`.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

[CKS01] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia (PA), 2001.

[DHK00] A. Durand, M. Hermann, and P. G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. In M. Nielsen and B. Rovan, editors, *Proceedings 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Bratislava (Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 323–332. Springer-Verlag, August 2000. To appear in *Theoretical Computer Science*.

[HV95] L. A. Hemaspaandra and H. Vollmer. The satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News, Complexity Theory Column 8*, 26(1):2–13, March 1995.

[JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the Association for Computing Machinery*, 44(4):527–548, 1997.

[JK03] P. Jonsson and A. Krokhin. Computational complexity of auditing finite attributes in statistical databases. In *Proceedings Structural Theory of Automata, Semigroups and Universal Algebra, Montreal (Canada)*, July 2003.

[KJ03] A. Krokhin and P. Jonsson. Recognizing frozen variables in constraint satisfaction problems. Technical Report TR03-062, Electronic Colloquium on Computational Complexity, 2003.

[KPR03] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing Boolean attributes. *Journal of Computer and System Science*, 66(1):244–253, 2003.

[KST89] J. Köbler, U. Schöning, and J. Torán. On counting and approximation. *Acta Informatica*, 26(4):363–379, 1989.

[KV00] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Science*, 61(2):302–332, 2000.

[Pip97] N. Pippenger. *Theories of Computability*. Cambridge University Press, Cambridge, 1997.

[PK79] R. Pöschel and L. A. Kalužnin. *Funktionen- und Relationenalgebren*. Deutscher Verlag der Wissenschaften, Berlin, 1979.

[Pos41] E. L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Pös01] R. Pöschel. Galois connection for operations and relations. Technical Report MATH-AL-8-2001, Technische Universität Dresden, 2001.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th Symposium on Theory of Computing (STOC'78), San Diego (California, USA)*, pages 216–226, 1978.

[Tod91] S. Toda. *Computational complexity of counting complexity classes.* PhD thesis, Tokyo Institute of Technology, Department of Computer Science, Tokyo, Japan, 1991.

[TW92] S. Toda and O. Watanabe. Polynomial-time 1-Turing reductions from #PH to #P. *Theoretical Computer Science*, 100(1):205–221, 1992.

[Val79a] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Val79b] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

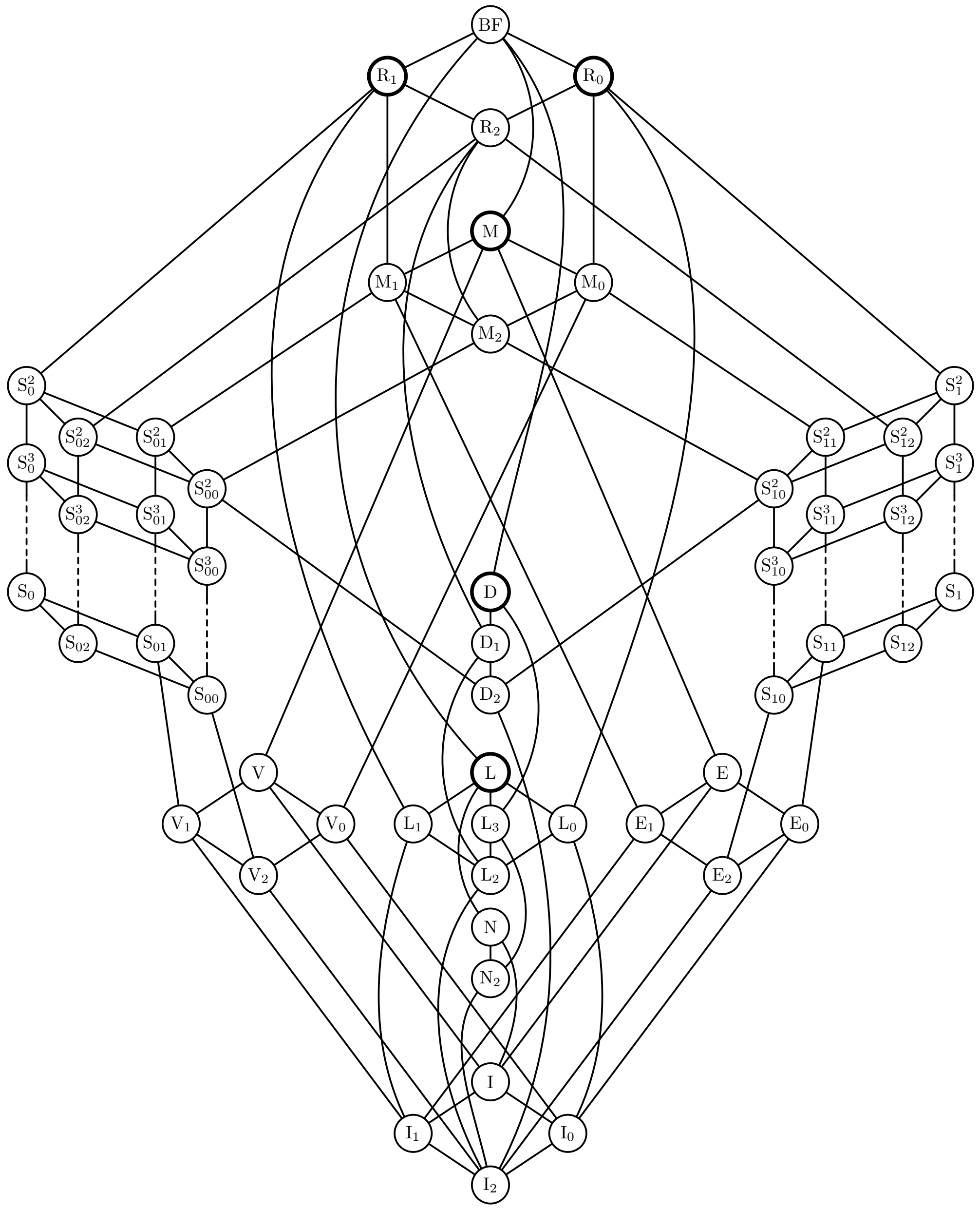

Fig. 2. Graph of all closed classes of Boolean functions