When Do Homomorphism Counts Help in Query Algorithms?

Balder ten Cate
University of Amsterdam, The Netherlands

Victor Dalmau
Universitat Pompeu Fabra, Barcelona, Spain

Phokion G. Kolaitis
University of California Santa Cruz, CA, USA
IBM Almaden Research Center, San Jose, CA, USA

Wei-Lin Wu
University of California Santa Cruz, CA, USA

Abstract

A query algorithm based on homomorphism counts is a procedure for determining whether a given instance satisfies a property by counting homomorphisms between the given instance and finitely many predetermined instances. In a left query algorithm, we count homomorphisms from the predetermined instances to the given instance, while in a right query algorithm we count homomorphisms from the given instance to the predetermined instances. Homomorphisms are usually counted over the semiring $\mathbb{N}$ of non-negative integers; it is also meaningful, however, to count homomorphisms over the Boolean semiring $\mathbb{B}$, in which case the homomorphism count indicates whether or not a homomorphism exists. We first characterize the properties that admit a left query algorithm over $\mathbb{B}$ by showing that these are precisely the properties that are both first-order definable and closed under homomorphic equivalence. After this, we turn attention to a comparison between left query algorithms over $\mathbb{B}$ and left query algorithms over $\mathbb{N}$. In general, there are properties that admit a left query algorithm over $\mathbb{N}$ but not over $\mathbb{B}$. The main result of this paper asserts that if a property is closed under homomorphic equivalence, then that property admits a left query algorithm over $\mathbb{B}$ if and only if it admits a left query algorithm over $\mathbb{N}$. In other words and rather surprisingly, homomorphism counts over $\mathbb{N}$ do *not* help as regards properties that are closed under homomorphic equivalence. Finally, we characterize the properties that admit both a left query algorithm over $\mathbb{B}$ and a right query algorithm over $\mathbb{B}$.

2012 ACM Subject Classification Theory of computation → Logic and databases

Keywords and phrases query algorithms, homomorphism, homomorphism counts, conjunctive query, constraint satisfaction

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.8

Funding *Balder ten Cate*: Supported by the European Union's Horizon 2020 research and innovation programme (MSCA-101031081).
Victor Dalmau: Supported by the MiCin under grants PID2019-109137GB-C22 and PID2022-138506NB-C22, and the Maria de Maeztu program (CEX2021-001195-M).
Phokion G. Kolaitis: Partially supported by NSF Grant IIS-1814152.

1 Introduction

Consider a scenario in which we are interested in a certain property of database instances and we wish to find out whether or not a given instance A satisfies that property by asking finitely many predetermined queries against A. Naturally, which properties can be checked in this way depends on what kind of queries we are allowed to ask. For example, if we are restricted to using Boolean conjunctive queries and evaluating them under set semantics, then only properties that are invariant under homomorphic equivalence stand a chance to be checked in this way. In particular, this means that we cannot test in this way whether a relation in a given instance contains precisely 5 tuples. In contrast, if we are also allowed to ask for the number of homomorphisms from a given conjunctive query to A (a feature that is supported by actual database management systems), then we can find out much more about the instance A, including whether it satisfies the aforementioned property.

In this paper, we embark on a systematic study of this scenario. As usual, by a *property* of instances we mean a class $\mathcal{C}$ of instances closed under isomorphisms. We write $\mathbb{B}$ for the Boolean semiring with $\vee$ and $\wedge$ as its operations, while we write $\mathbb{N}$ for the semiring of the non-negative integers with $+$ and $\times$ as its operations. Formally, we say that a class $\mathcal{C}$ of instances admits a *left query algorithm over* $\mathbb{B}$ if there exists a finite set $\mathcal{F} = \{F_1, \dots, F_k\}$ of instances such that the membership in $\mathcal{C}$ of an arbitrary instance A is completely determined by $\hom_{\mathbb{B}}(\mathcal{F}, A)$, where $\hom_{\mathbb{B}}(\mathcal{F}, A)$ is the k-tuple of Boolean values indicating for each $i \leq k$ whether or not there is a homomorphism $F_i \to A$. Similarly, we say that a class $\mathcal{C}$ of instances admits a *left query algorithm over* $\mathbb{N}$ if there exists a finite set $\mathcal{F} = \{F_1, \dots, F_k\}$ of instances such that the membership in $\mathcal{C}$ of an arbitrary instance A is completely determined by $\hom_{\mathbb{N}}(\mathcal{F}, A)$, where $\hom_{\mathbb{N}}(\mathcal{F}, A)$ is the k-tuple of non-negative integers indicating for each $i \leq k$ how many homomorphisms from F_i to A there are. *Right query algorithms* over $\mathbb{B}$ and over $\mathbb{N}$ are defined similarly, except that we now count the homomorphisms from A to each F_i instead of the homomorphisms from each F_i to A.

Assume that the class $\mathcal{C}$ of instances under consideration admits a left query algorithm over $\mathbb{B}$ or over $\mathbb{N}$ using the set $\mathcal{F} = \{F_1, \dots, F_k\}$. Let q^{F_i} be the canonical conjunctive query associated with the instance F_i, $1 \leq i \leq k$. Then $\hom_{\mathbb{B}}(\mathcal{F}, A)$ is the k-tuple of Boolean values denoting whether $q^{F_i}(A) = 1$ or $q^{F_i}(A) = 0$, i.e., whether q^{F_i} is true or false on A under set semantics. Similarly, $\hom_{\mathbb{N}}(\mathcal{F}, A)$ is the k-tuple of non-negative integers that are the answers to q^{F_i} on A under bag-set semantics. Thus, intuitively, a class $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ or over $\mathbb{N}$ if membership in $\mathcal{C}$ is answerable by evaluating finitely many Boolean conjunctive queries under set semantics or under bag-set semantics, respectively. Observe that these are data complexity notions because the queries $q^{F_1}, \dots, q^{F_k}$ are fixed while the instance A varies. Observe also that these two notions differ in expressive power: for example, if $\mathcal{C}$ is the class of instances in which a particular relation R has precisely 5 tuples, then $\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$, but not over $\mathbb{B}$.

There are two pieces of earlier work (each with different motivation and results) where the notion of a left query algorithm or variants of this notion have been explored. First, Bielecki and Van den Bussche [4] defined what it means for a query p to be *derivable* through interrogation with a query language L using a *database independent strategy*, where the interrogation consists of asking the cardinality $|q(A)|$ for finitely many queries $q \in L$. When L is the language of conjunctive queries with no existential quantifiers, such strategies correspond to left-query algorithms over $\mathbb{N}$; whereas, when L is the language of Boolean conjunctive queries, they correspond to left-query algorithms over $\mathbb{B}$. Second, when the instances are unordered graphs, the concept of a left query algorithm over $\mathbb{N}$ was studied by Chen et al. [9] under the name *non-adaptive query algorithm*; note that Chen et al. [9] were apparently unaware of the work by Bielecki and Van den Bussche [4]. The term "non-adaptive" is apt as it conveys that the instances F_i (or the associated conjunctive queries q^{F_i}), $1 \leq i \leq k$, in the set $\mathcal{F}$ depend only on the class $\mathcal{C}$ and do not change during a run of the query algorithm. It is also natural to consider *adaptive query algorithms*, where the instances F_i, $1 \leq i \leq k$, are not required to be fixed up front. In fact, such adaptive notions were explored in both [4] and [9]. In particular, Chen et al. [9] showed that adaptive left query algorithms over $\mathbb{N}$ are more powerful than non-adaptive ones over $\mathbb{N}$. It is easy to see, however, that the existence of an adaptive left query algorithm over $\mathbb{B}$ implies the existence of a non-adaptive one over $\mathbb{B}$. In this sense, adaptive left query algorithms over $\mathbb{B}$ are not more powerful than non-adaptive ones over $\mathbb{B}$. Also, since in this paper we study non-adaptive algorithms only (but over both $\mathbb{B}$ and $\mathbb{N}$), we will not use the adjective "non-adaptive" here.

Our investigation begins by focusing on left query algorithms over $\mathbb{B}$. It is easy to see that a class of instances admits a left query algorithm over $\mathbb{B}$ if and only if it is definable by a Boolean combination of conjunctive queries. Using tools developed by Rossman [27] to prove the preservation-under-homomorphisms theorem in the finite, we obtain a deeper characterization of such classes by showing that a class of instances admits a left query algorithm over $\mathbb{B}$ if and only if it is both first-order definable and closed under homomorphic equivalence. Clearly, if a class of instances is closed under homomorphic equivalence, then it is a (possibly infinite) union of homomorphic equivalence classes. We show that if $\mathcal{C}$ is a finite union of homomorphic-equivalence classes, then $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ if and only if every homomorphic-equivalence class in that union admits a left query algorithm over $\mathbb{B}$. In contrast, a similar result does not hold for arbitrary infinite unions.

After this, we turn attention to a comparison between left query algorithms over $\mathbb{B}$ and left query algorithms over $\mathbb{N}$. As discussed earlier, left query algorithms over $\mathbb{N}$ are more powerful than left query algorithms over $\mathbb{B}$. The intuitive reason is that left query algorithms over $\mathbb{B}$ do not differentiate between homomorphically equivalent instances, while those over $\mathbb{N}$ do. The main (and technically more challenging) result of this paper reveals that, in a precise sense, this is the *only* reason why left query algorithms over $\mathbb{N}$ are more powerful than left query algorithms over $\mathbb{B}$. More precisely, our main theorem asserts that if a class $\mathcal{C}$ is closed under homomorphic equivalence, then $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$. In other words and rather surprisingly, homomorphism counts over $\mathbb{N}$ do *not* help as regards properties that are closed under homomorphic equivalence. As an immediate consequence, a constraint satisfaction problem has a left query algorithm over $\mathbb{N}$ if and only if this problem is first-order definable.

Finally, we characterize the properties that admit both a left query algorithm over $\mathbb{B}$ and a right query algorithm over $\mathbb{B}$. In particular, we show that a class $\mathcal{C}$ of instances admits both a left query algorithm over $\mathbb{B}$ and a right query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ is definable by a Boolean combination of Berge-acyclic conjunctive queries. To see the point of this result, recall that if a class admits a left query algorithm over $\mathbb{B}$, then it is definable by a Boolean combination of conjunctive queries. Thus, if the class admits also a right query algorithm over $\mathbb{B}$, then these conjunctive queries can be taken to be Berge-acyclic.

Related work. A classical result by Lovász [25] characterizes graph isomorphism in terms of "left" homomorphism counts: two graphs G and H are isomorphic if and only if for every graph F, the number of homomorphisms from F to G is equal to the number of homomorphisms from F to H. In more recent years, there has been a study of relaxations of isomorphisms obtained by requiring that the number of homomorphisms from F to G is equal to the number of homomorphisms from F to H, where F ranges over a restricted class of graphs [12, 11, 6]. Furthermore, a study of relaxations of isomorphism obtained by counting the number of "right" homomorphisms to a restricted class was carried out in [3].

There has been an extensive body of research on answering queries under various types of access restrictions. Closer in spirit to the work reported here is view determinacy, which is the question of whether the answers to a query can be inferred when given access only to a certain view of the database instance [26]. We note that view determinacy is typically concerned with non-Boolean queries and non-Boolean views; in contrast, the question of whether a class admits a left query algorithm over $\mathbb{B}$ can be interpreted as the question of whether there is a finite set of Boolean conjunctive queries that determine a given Boolean query. A study of view determinacy under bag-set semantics was recently initiated in [24]. Section 7 contains additional commentary on the relationship between left query algorithms over $\mathbb{N}$ and view determinacy under bag-set semantics.

2 Basic Notions

Relational database instances. A *relational database schema* or, simply, a *schema* is a finite set $\sigma = \{R_1, \dots, R_m\}$ of relation symbols R_i, each of which has a positive integer r_i as its associated *arity*. A *relational database instance* or, simply, an *instance* is a tuple $A = (R_1^A, \dots, R_m^A)$, where each R_i^A is a relation of arity r_i. The *facts* of the instance A are the tuples in the relations R_i^A, $1 \le i \le m$. The *active domain of* A, denoted $\mathrm{adom}(A)$, is the set of all entries occurring in the facts of A. All instances A considered are assumed to be finite, i.e., $\mathrm{adom}(A)$ is finite. A *graph* is an instance A over a schema consisting of a binary relation symbol E and such that E^A is a binary relation that is symmetric and irreflexive.

The *incidence multigraph* $\mathrm{inc}(A)$ of an instance A is the bipartite multigraph whose parts are the sets $\mathrm{adom}(A)$ and $\mathrm{block}(A) = \{(R,t) : R \in \sigma \text{ and } t \in R^A\}$, and whose edges are the pairs $(a, (R,t))$ such that a is one of the entries of t. A *path of length* n in A is a sequence $a_0, a_1, \dots, a_n$ of elements in $\mathrm{adom}(A)$ for which there are elements $b_1, \dots, b_n$ in $\mathrm{block}(A)$ such that the sequence $a_0, b_1, a_1, \dots, b_n, a_n$ is a path in $\mathrm{inc}(A)$ in the standard graph-theoretic sense (disallowing traversing an edge twice in succession in opposite directions). Two elements a and a' in $\mathrm{adom}(A)$ are *connected* if $a = a'$ or there is a path $a_0, a_1, \dots, a_n$ in A with $a = a_0$ and $a' = a_n$. We say that A is *connected* if every two elements a and a' in $\mathrm{adom}(A)$ are connected. A *cycle of length* n in A is a path of length n in A with $a_n = a_0$. We say A is *acyclic* if it contains no cycles. The *girth of* A is the shortest length of a cycle in A or ∞ if A is acyclic.

An instance A is a *subinstance* of an instance B if $R^A \subseteq R^B$, for every $R \in \sigma$.

A *class* $\mathcal{C}$ of instances is a collection of instances over the same schema that is closed under isomorphism (i.e., if $A \in \mathcal{C}$ and B is isomorphic to A, then $B \in \mathcal{C}$). Every decision problem P about instances can be identified with the class of all “yes” instances of P.

Homomorphisms, conjunctive queries, and canonical instances. A *homomorphism* from an instance A to an instance B is a function $h : \mathrm{adom}(A) \to \mathrm{adom}(B)$ such that for every relation symbol $R \in \sigma$ with arity r and for all elements $a_1, \dots, a_r$ in $\mathrm{adom}(A)$ with $(a_1, \dots, a_r) \in R^A$, we have $(h(a_1), \dots, h(a_r)) \in R^B$. We write $h : A \to B$ to denote that h is a homomorphism from A to B; we also write $A \to B$ to denote that there is a homomorphism from A to B. We say that A and B are *homomorphically equivalent*, denoted $A \leftrightarrow B$, if $A \to B$ and $B \to A$. Clearly, $\leftrightarrow$ is an equivalence relation on instances. We write $[A]_\leftrightarrow$ to denote the equivalence class of A with respect to $\leftrightarrow$, i.e., $[A]_\leftrightarrow = \{B : B \leftrightarrow A\}$.

Let $\mathcal{C}$ be a class of instances. We say that $\mathcal{C}$ is *closed under homomorphic equivalence* if whenever $A \in \mathcal{C}$ and $A \leftrightarrow B$, we have that $B \in \mathcal{C}$. As an example, for every instance A, we have that the equivalence class $[A]_\leftrightarrow$ is closed under homomorphic equivalence. For a different example, the class of all 3-colorable graphs is closed under homomorphic equivalence.

We assume familiarity with the syntax and the semantics of first-order logic (FO). For a FO-sentence φ, we denote by $\mathrm{Mod}(\varphi)$ the set $\{A : A \models \varphi\}$ of instances A that satisfy φ under the active domain semantics (i.e., the quantifiers range over elements of the active domain of the instance at hand). A *Boolean conjunctive query* (Boolean CQ) is a FO-sentence of the form $\exists x_1 \dots x_n(\alpha_1 \wedge \cdots \wedge \alpha_k)$, where each α_j is an atomic formula of the form $R(y_1, \dots, y_r)$, each variable y_i is among the variables $x_1, \dots, x_n$, and each variable x_i occurs in at least one of the atomic formulas $\alpha_1, \dots, \alpha_k$.

The *canonical instance* of a conjunctive query q, denoted A^q, is the instance whose active domain consists of the variables of q, and whose facts are the conjuncts of q. Conversely, the *canonical conjunctive query* of an instance A, denoted q^A, has, for each a in $\mathrm{adom}(A)$, an existentially quantified variable x_a and, for each fact $(a_1, \dots, a_r) \in R^A$, a conjunct $R(x_{a_1}, \dots, x_{a_r})$. An immediate consequence of the semantics of FO is that, for every two instances A and D, we have that $D \models q^A$ if and only if $A \to D$.

A conjunctive query is *Berge-acyclic* if its canonical instance is acyclic, as defined earlier. The notion of Berge-acyclicity is stronger than the more standard notion of acyclicity in databse theory, which requires that the conjunctive query has a join tree (see, e.g., [1]).

Homomorphism counts, left and right profiles. Let $\mathbb{N} = (N, +, \times, 0, 1)$ be the semiring of the non-negative integers and let $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ be the Boolean semiring. If A and B are two instances, then we write $\hom_{\mathbb{N}}(A, B)$ for the number of homomorphisms from A to B. For example, if A is a graph and K_3 is the 3-clique, then $\hom_{\mathbb{N}}(A, K_3)$ is the number of 3-colorings of A. We extend this notion to $\hom_{\mathbb{B}}(A, B)$, where $\hom_{\mathbb{B}}(A, B) = 1$ if there is a homomorphism from A to B, and $\hom_{\mathbb{B}}(A, B) = 0$ otherwise. For example, $\hom_{\mathbb{B}}(A, K_3) = 1$ if A is 3-colorable, and $\hom_{\mathbb{B}}(A, K_3) = 0$ if A is not 3-colorable.

Let $\mathcal{F} = \{F_1, \dots, F_k\}$ be a finite non-empty set of instances and let A be an instance.

- The *left profile of A in $\mathcal{F}$ over* $\mathbb{N}$ is the tuple

$$\hom_{\mathbb{N}}(\mathcal{F}, A) = (\hom_{\mathbb{N}}(F_1, A), \dots, \hom_{\mathbb{N}}(F_k, A)).$$

- The *left profile of A in $\mathcal{F}$ over* $\mathbb{B}$ is the tuple

$$\hom_{\mathbb{B}}(\mathcal{F}, A) = (\hom_{\mathbb{B}}(F_1, A), \dots, \hom_{\mathbb{B}}(F_k, A)).$$

- The *right profile of A in $\mathcal{F}$ over* $\mathbb{N}$ is the tuple

$$\hom_{\mathbb{N}}(A, \mathcal{F}) = (\hom_{\mathbb{N}}(A, F_1), \dots, \hom_{\mathbb{N}}(A, F_k)).$$

- The *right profile of A in $\mathcal{F}$ over* $\mathbb{B}$ is the tuple

$$\hom_{\mathbb{B}}(A, \mathcal{F}) = (\hom_{\mathbb{B}}(A, F_1), \dots, \hom_{\mathbb{B}}(A, F_k)).$$

Let $A_1, \dots, A_n$ be instances whose active domains are pairwise disjoint.

- The *direct sum* $A_1 \oplus \cdots \oplus A_n$ of $A_1, \dots, A_n$ is the instance such that $R^{A_1 \oplus \cdots \oplus A_n} = R^{A_1} \cup \cdots \cup R^{A_n}$, for every $R \in \sigma$.
- The *direct product* $A_1 \otimes \cdots \otimes A_n$ of $A_1, \dots, A_n$ is the instance such that the relation $R^{A_1 \otimes \cdots \otimes A_n}$ consists of all tuples $(\mathbf{a}_1, \dots, \mathbf{a}_r)$ with $(\mathbf{a}_1(i), \dots, \mathbf{a}_r(i)) \in R^{A_i}$, for $1 \le i \le n$ and for every $R \in \sigma$ of arity r.

The next proposition is well known and has a straightforward proof.

▶ **Proposition 2.1.** *Let A, B_1, B_2 be instances, and let $K \in \{\mathbb{B}, \mathbb{N}\}$. Then the following statements are true.*

1. $\hom_K(A, B_1 \oplus B_2) = \hom_K(A, B_1) +_K \hom_K(A, B_2)$*, provided that A is connected;*
2. $\hom_K(A, B_1 \otimes B_2) = \hom_K(A, B_1) \cdot_K \hom_K(A, B_2)$*;*
3. $\hom_K(B_1 \oplus B_2, A) = \hom_K(B_1, A) \cdot_K \hom_K(B_2, A)$*,*

where $+_{\mathbb{N}}$ and $\cdot_{\mathbb{N}}$ stand for addition $+$ and multiplication $\times$ of non-negative integers, while $+_{\mathbb{B}}$ and $\cdot_{\mathbb{B}}$ stand for disjunction $\vee$ and conjunction $\wedge$ of the Boolean values 0 and 1.

3 Left Query Algorithms and Right Query Algorithms

In [9], Chen et al. focused on classes of graphs and introduced the notions of a left query algorithm and a right query algorithm over the semiring $\mathbb{N}$ of the non-negative integers. Here, we extend their framework in two ways: first, we consider classes of instances over some arbitrary, but fixed, schema; second, we consider left query algorithms and right query algorithms over the Boolean semiring $\mathbb{B}$.

▶ **Definition 3.1.** *Let $\mathcal{C}$ be a class of instances and let K be the semiring $\mathbb{B}$ or $\mathbb{N}$.*

- *Assume that k is a positive integer.*
  - *A* left k-query algorithm over K for $\mathcal{C}$ *is a pair $(\mathcal{F}, X)$, where $\mathcal{F} = \{F_1, \dots, F_k\}$ is a set of instances and X is a set of k-tuples over K, such that for all instances D, we have that $D \in \mathcal{C}$ if and only if* $\hom_K(\mathcal{F}, D) \in X$.
  - *A* right k-query algorithm over K for $\mathcal{C}$ *is a pair $(\mathcal{F}, X)$, where $\mathcal{F} = \{F_1, \dots, F_k\}$ is a set of instances and X is a set of k-tuples over K, such that for all instances D, we have that $D \in \mathcal{C}$ if and only if* $\hom_K(D, \mathcal{F}) \in X$.
- *We say that $\mathcal{C}$* admits a left query algorithm over K *if for some $k > 0$, there is a left k-query algorithm over K for $\mathcal{C}$. Similarly, we say that $\mathcal{C}$* admits a right query algorithm over K *if for some $k > 0$, there is a right k-query algorithm over K for $\mathcal{C}$.*

The term "query algorithm" is natural because we can think of a query algorithm as a procedure for determining if a given instance belongs to the class $\mathcal{C}$: we compute the left homomorphism-count vector (in the case of a left query algorithm) or the right homomorphism-count vector (in the case of a right query algorithm) and test whether it belongs to X. When the semiring $\mathbb{N}$ is considered, this is a somewhat abstract notion of an algorithm because it makes no requirements on the effectiveness of the set X. Not requiring X to be a decidable set makes our results regarding the non-existence of left query algorithms over $\mathbb{N}$ stronger. Moreover, in all cases where we establish the existence of a left query algorithm over $\mathbb{N}$ or a right query algorithm over $\mathbb{N}$, the set X will happen to be decidable (for the semiring $\mathbb{B}$, the set X is always finite, hence decidable).

Let K be the semiring $\mathbb{B}$ or $\mathbb{N}$. It is clear that if two classes of instances admit a left query algorithm over K, then so do their complements, their union, and their intersection. Consequently, the classes of instances that admit a left query algorithm over K are closed under Boolean combinations. Furthermore, the same holds true for right query algorithms.

By Part 3 of Proposition 2.1, we have that if K is the semiring $\mathbb{B}$ or the semiring $\mathbb{N}$, then $\hom_K$ is *multiplicative on the left*, i.e., for all instances $A_1, \dots, A_n$ and B, we have $\hom_K(A_1 \oplus \cdots \oplus A_n, B) = \hom_K(A_1, B) \cdot_K \cdots_K \hom_K(A_n, B)$. It follows that, as regards the existence of left query algorithms, we may assume that all instances in the finite set $\mathcal{F}$ of a left query algorithm over $\mathbb{B}$ or over $\mathbb{N}$ are connected. We state this observation as a proposition that will be used repeatedly in the sequel.

▶ **Proposition 3.2.** *Let $\mathcal{C}$ be a class of instances and let K be the semiring $\mathbb{B}$ or $\mathbb{N}$. Then the following statements are equivalent.*

1. *$\mathcal{C}$ admits a left query algorithm $(\mathcal{F}, X)$ over K.*
2. *$\mathcal{C}$ admits a left query algorithm $(\mathcal{F}, X)$ over K, where every instance in the set $\mathcal{F}$ is connected.*

Left profiles over $\mathbb{N}$ contain more information than left profiles over $\mathbb{B}$. Therefore, if membership in a class $\mathcal{C}$ can be determined using left profiles over $\mathbb{B}$, then it ought to be also determined using left profiles over $\mathbb{N}$. Similar considerations hold for right profiles. The next proposition makes these assertions precise.

▶ **Proposition 3.3.** *Let $\mathcal{C}$ be a class of instances and let $\mathcal{F}$ be a finite set of instances.*

If $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ of the form $(\mathcal{F}, X)$ for some set X, then $\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$ of the form $(\mathcal{F}, X')$ for some set X'. In particular, if $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$, then it also admits a left query algorithm over $\mathbb{N}$.

Furthermore, the same holds true for right query algorithms.

Proof. Assume that $\mathcal{C}$ admits a left query algorithm $(\mathcal{F}, X)$ over $\mathbb{B}$, where $\mathcal{F} = \{F_1, \dots, F_k\}$ and $X \subseteq \{0,1\}^k$, for some $k > 0$. For every $t = (t_1, \dots, t_k) \in \{0,1\}^k$, we let X_t be the set

$$X_t = \{(s_1, \cdots, s_k) \in \mathbb{N}^k : s_i = 0 \text{ if and only if } t_i = 0, \text{ for } 1 \le i \le k\}.$$

Consider the set $X' = \bigcup_{t \in X} X_t$. It is easy to verify that the pair $(\mathcal{F}, X')$ is a left k-query algorithm for $\mathcal{C}$ over $\mathbb{N}$. The argument for right query algorithms is entirely analogous. ◀

As pointed out in the Introduction, the converse of Proposition 3.3 is not true, in general. We now give several examples illustrating left and right query algorithms.

▶ **Example 3.4.** Let $\mathcal{C}$ be the class of all triangle-free graphs, i.e., the graphs G for which there is no homomorphism from K_3 to G. Clearly, $\mathcal{C}$ admits a left 1-query algorithm $(\mathcal{F}, X)$ over $\mathbb{B}$, where $\mathcal{F} = \{K_3\}$ and $X = \{0\}$ (recall that K_3 is the 3-clique).[1] Therefore, by Proposition 3.3, $\mathcal{C}$ admits a left 1-query algorithm over $\mathbb{N}$. In contrast, Chen et al. [9, Proposition 8.2] showed that (the complement of) $\mathcal{C}$ does not admit a right query algorithm over $\mathbb{N}$, hence (again by Proposition 3.3) it does not admit a right query algorithm over $\mathbb{B}$.

We now recall the definition of constraint satisfaction problems.

▶ **Definition 3.5.** *If B is an instance, then the* constraint satisfaction problem CSP(B) *is the following decision problem: given an instance A, is there a homomorphism from A to B?*

For $k \ge 2$, let K_k denote the k-clique. Then CSP(K_k) is the k-COLORABILITY problem: given a graph G, is G k-colorable? During the past three decades, there has been an extensive study of complexity of constraint satisfaction problems, motivated by the Feder-Vardi Conjecture that for every instance B, either CSP(B) is NP-complete or CSP(B) is solvable in polynomial time. This conjecture was eventually confirmed independently by Bulatov [7] and Zhuk [29].

Every constraint satisfaction problem will be identified with the class of its "yes" instances, that is, for every instance B, we have that CSP$(B) = \{A : A \to B\}$.

▶ **Example 3.6.** Let B be an instance. Clearly, CSP(B) admits a right 1-query algorithm $(\mathcal{F}, X)$ over $\mathbb{B}$, where $\mathcal{F} = \{B\}$ and $X = \{1\}$. Therefore, by Proposition 3.3, CSP(B) admits a right 1-query algorithm $(\mathcal{F}, X')$ over $\mathbb{N}$. In particular, the 3-COLORABILITY problem CSP(K_3) admits a right query algorithm over both $\mathbb{B}$ and $\mathbb{N}$. In contrast, it will follow from results in Section 4 and Section 5 that CSP(K_3) does not admit a left query algorithm over $\mathbb{B}$ or over $\mathbb{N}$ (see Remark 4.5).

▶ **Example 3.7.** Consider the homomorphic equivalence class $[K_3]_\leftrightarrow$. Note that $[K_3]_\leftrightarrow$ is the class of all graphs that are 3-colorable and also contain a triangle. From results in Section 4, it will follow that $[K_3]_\leftrightarrow$ does not admit a left query algorithm over $\mathbb{B}$ (see Remark 4.5). Furthermore, from results in Section 6, it will follow that $[K_3]_\leftrightarrow$ does not admit a right query algorithm over $\mathbb{B}$ (see Remark 6.7).

[1] This example, and several other examples in this paper, involve graphs. Here, the word "graph" may be read as "structure over a schema with one binary relation". Alternatively, it may be read as "structure over a schema with one binary relation that is symmetric and irreflexive", but, in the latter case, we only require the query algorithm to behave correctly on such graphs, and we do not require the query algorithm to distinguish such graphs from structures whose relation is not symmetric and irreflexive.

4 Left Query Algorithms over $\mathbb{B}$

In this section, we investigate which classes admit a left query algorithm over $\mathbb{B}$. It is easy to see that every class of instances that admits a left query algorithm over $\mathbb{B}$ is closed under homomorphic equivalence. In other words, closure under homomorphic equivalence is a necessary condition for the existence of a left query algorithm over $\mathbb{B}$. The next result gives an exact characterization of the classes of instances that admit a left query algorithm over $\mathbb{B}$.

▶ **Theorem 4.1.** *Let $\mathcal{C}$ be a class of instances. Then the following statements are equivalent.*

1. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$.*
2. *$\mathcal{C}$ is definable by a Boolean combination of CQs.*
3. *$\mathcal{C}$ is FO-definable and closed under homomorphic equivalence.*

Proof. We will first show the equivalence between statements (1) and (2), and then the equivalence between statements (2) and (3).
(1) $\Longrightarrow$ (2): Let a left query algorithm over $\mathbb{B}$ for $\mathcal{C}$ consist of $\mathcal{F} = \{F_1, \ldots, F_k\}$ and $X \subseteq \{0,1\}^k$, and let $q^{F_1}, \ldots, q^{F_k}$ be the canonical conjunctive queries of $F_1, \ldots, F_k$, respectively. For every tuple $t = (t_1, \ldots, t_k) \in X$, define

$$\varphi_t := \bigwedge_{t_i=1} q^{F_i} \wedge \bigwedge_{t_i=0} \neg q^{F_i}.$$

Take $\varphi := \bigvee_{t \in X} \varphi_t$. Then $\mathcal{C} = \mathrm{Mod}(\varphi)$.
(2) $\Longrightarrow$ (1): Every class $\mathcal{C}$ defined by a conjunctive query q admits a left 1-query algorithm over $\mathbb{B}$. Indeed, we can pick $\mathcal{F}$ to consist of the canonical instance A^q of q, and $X = \{1\}$. It follows by closure under Boolean combinations that every class defined by a Boolean combination of conjunctive queries also admits a left query algorithm over $\mathbb{B}$.
(2) $\Longrightarrow$ (3): This implication is immediate because CQs are first-order formulas whose truth is preserved by homomorphisms.
(3) $\Longrightarrow$ (2): We will use two results from [27] about the homomorphism preservation theorem in the finite. For instances A and B, we write $A \leftrightarrow^n B$ to mean that A and B satisfy the same existential positive FO-sentences of quantifier rank at most n, and write $A \equiv^n B$ to mean that A and B satisfy the same FO-sentences of quantifier rank at most n. The two results from [27] in our concerns are

(a) Theorem 1.9: For every n, there is some m that depends on n such that for every instances A and B with $A \leftrightarrow^m B$, there are instances A' and B' such that $A \leftrightarrow A'$, $B \leftrightarrow B'$, and $A' \equiv^n B'$.

(b) Lemma 3.9: For every m, the equivalence relation $A \leftrightarrow^m B$ has finitely many equivalence classes over the class of all instances.

Now, let $\mathcal{C}$ be a class definable by a FO-sentence φ and closed under homomorphic equivalence $\leftrightarrow$. Let n be the quantifier rank of φ, and let m be the integer in the statement of (a). Note that m depends on n only.

We claim that $\mathrm{Mod}(\varphi) = \mathcal{C}$ is closed under $\leftrightarrow^m$. Indeed, assume that A and B are two instances such that $A \models \varphi$ and $A \leftrightarrow^m B$. By (a), there are instances A' and B' such that $A \leftrightarrow A'$, $B \leftrightarrow B'$, and $A' \equiv^n B'$. It follows, successively, that

$$\begin{aligned} A' &\models \varphi && \text{(since } A \models \varphi,\ A \leftrightarrow A', \text{ and } \mathcal{C} \text{ is closed under } \leftrightarrow),\\ B' &\models \varphi && \text{(since } \varphi \text{ has quantifier rank } n \text{ and } A' \equiv^n B'),\\ B &\models \varphi && \text{(since } B \leftrightarrow B' \text{ and } \mathcal{C} \text{ is closed under } \leftrightarrow). \end{aligned}$$

By (b), the equivalence relation $\leftrightarrow^m$ has finitely many equivalence classes. Let $A_1, \ldots, A_k$ be representatives from each of these equivalence classes (one per equivalence class). For different i, j in $\{1, \ldots, k\}$, let $\psi_{i,j}$ be an existential positive FO-sentence of quantifier rank at most m such that $A_i \models \psi_{i,j}$ but not $A_j \models \psi_{i,j}$, and let $\psi(A_i)$ be the conjunction of all $\psi_{i,j}$. Each $\psi(A_i)$ is a Boolean combination of conjunctive queries and it holds for every instance B that $B \leftrightarrow^m A_i$ if and only if $B \models \psi(A_i)$. Then φ is equivalent to the disjunction $\bigvee_{A_i \models \varphi} \psi(A_i)$ since $\text{Mod}(\varphi)$ is closed under $\leftrightarrow^m$. Indeed, if $B \models \varphi$, then $B \leftrightarrow^m A_i$ for some A_i that satisfies φ, hence $B \models \psi(A_i)$. Conversely, if $B \models \bigvee_{A_i \models \varphi} \psi(A_i)$, then $B \models \psi(A_i)$ for some A_i that satisfies φ; it follows that $B \leftrightarrow^m A_i$, hence $B \models \varphi$. ◀

▶ **Corollary 4.2.** *A class $\mathcal{C}$ of instances that is closed under homomorphic equivalence admits a left query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ is FO-definable.*

Corollary 4.2, in particular, applies to classes of the form $\text{CSP}(A)$, since such classes are closed under homomorphic equivalence. It was shown in [28] that testing, for a given instance A, whether $\text{CSP}(A)$ is FO-definable, is NP-complete (and in fact, in polynomial time when A is a *core*, i.e. when there is no homomorphism from A to a proper subinstance of A). It follows that testing if a given $\text{CSP}(A)$ admits a left query algorithm over $\mathbb{B}$ is NP-complete. This extends to finite unions of CSPs:

▶ **Proposition 4.3.** *The following problem is NP-complete: given instances $A_1, \ldots, A_n$, does $\bigcup_{1 \le i \le n} \text{CSP}(A_i)$ admit a left query algorithm over $\mathbb{B}$?*

Proof. Without loss of generality, assume that the instances $A_1, \ldots, A_n$ are pairwise homomorphically incomparable (because, if $A_j \to A_k$, then A_j can be removed without affecting the class defined by $\bigcup_{1 \le i \le n} \text{CSP}(A_i)$). It is known that, in this case, $\bigcup_{1 \le i \le n} \text{CSP}(A_i)$ is FO-definable if and only if for each $1 \le i \le n$, $\text{CSP}(A_i)$ is FO-definable (this may be considered folklore, see [5, Lemma 5.13] for an explicit proof). The result follows, by Corollary 4.2. ◀

Corollary 4.2 also applies to classes of the form $[A]_{\leftrightarrow}$, and we can derive a similar complexity bound. This will be obtained using the following proposition.

▶ **Proposition 4.4.** *Let A be an instance and $K \in \{\mathbb{B}, \mathbb{N}\}$. Then the following statements are equivalent.*

1. *$[A]_{\leftrightarrow}$ has a left query algorithm over K.*
2. *$\text{CSP}(A)$ has a left query algorithm over K.*

Proof. Let $K \in \{\mathbb{B}, \mathbb{N}\}$ throughout the proof.
(2) $\Longrightarrow$ (1): Clearly, $[A]_{\leftrightarrow} = \text{CSP}(A) \cap \{B : A \to B\}$. Also, $\{B : A \to B\}$ admits an obvious left query algorithm over K. The result follows by closure under Boolean combinations.
(1) $\Longrightarrow$ (2): Assume $\text{CSP}(A)$ has no left query algorithm over K. Let $\mathcal{F}$ be an arbitrary finite set of connected instances (think: candidate left query algorithm for $[A]_{\leftrightarrow}$). Since $\text{CSP}(A)$ has no left query algorithm over K, there are instances $P \in \text{CSP}(A)$ and $Q \notin \text{CSP}(A)$ such that $\hom_K(\mathcal{F}, P) = \hom_K(\mathcal{F}, Q)$. Let $P' = P \oplus A$ and let $Q' = Q \oplus A$. Then, by construction, $P' \in [A]_{\leftrightarrow}$ and $Q' \notin [A]_{\leftrightarrow}$ but $\hom_K(\mathcal{F}, P') = \hom_K(\mathcal{F}, Q')$ (cf. Proposition 2.1). Therefore, by Proposition 3.2, $[A]_{\leftrightarrow}$ has no left query algorithm over K. ◀

Consequently, the following problem is also NP-complete: *given an instance A, does $[A]_{\leftrightarrow}$ admit a left query algorithm over $\mathbb{B}$?*

▶ Remark 4.5. In Example 3.6, we asserted that the class $\text{CSP}(K_3)$ of 3-colorable graphs admits no left query algorithm over $\mathbb{B}$. Furthermore, in Example 3.7, we asserted that $[K_3]_{\leftrightarrow}$ (i.e., the class of graphs that are 3-colorable and also contain a triangle) admits no left query algorithm over $\mathbb{B}$. Corollary 4.2 and Proposition 4.4, now account for the non-existence of a left query algorithm over $\mathbb{B}$ for these classes: the reason is these two classes are not FO-definable. In the next section (see Corollary 5.6), we will see that the same explanation accounts for the fact that these classes admit no left query algorithm over $\mathbb{N}$ either.

Proposition 4.4 extends to finite unions of homomorphic equivalence classes.

▶ **Theorem 4.6.** *For all instances $A_1, \ldots, A_n$, the following statements are equivalent.*

1. *$\bigcup_{1\leq i\leq n} [A_i]_{\leftrightarrow}$ admits a left query algorithm over $\mathbb{B}$ (equivalently, is FO-definable).*
2. *Each $[A_i]_{\leftrightarrow}$, for $i = 1, \ldots, n$, admits a left query algorithm over $\mathbb{B}$ (equivalently, is FO-definable).*

Proof. It is clear that the second statement implies the first. We will prove by induction on n that the first statement implies the second. The base case ($n = 1$) is immediate since (i) and (ii) coincide. Next, let $n > 1$ and $\mathcal{C} := \bigcup_{1\leq i\leq n} [A_i]_{\leftrightarrow}$. We proceed by contraposition: suppose that $[A_i]_{\leftrightarrow}$ does not admit a left query algorithm over $\mathbb{B}$, for some $i \leq n$. We may assume without loss of generality that $A_1, \ldots, A_n$ are pairwise not homomorphically equivalent. Note that $\rightarrow$ induces a preorder among $A_1, \ldots, A_n$ and, since n is finite, there is a maximal. Without loss of generality, assume that A_n is a maximal, that is, $A_n \not\rightarrow A_i$ for all $i < n$. We distinguish two cases.

(1) $[A_n]_{\leftrightarrow}$ admits a left query algorithm over $\mathbb{B}$. Then, for some $i \leq n-1$, $[A_i]_{\leftrightarrow}$ does not admit a left query algorithm over $\mathbb{B}$. By induction hypothesis, we have $\mathcal{C}' := \bigcup_{1\leq i\leq n-1} [A_i]_{\leftrightarrow}$ does not admit a left query algorithm over $\mathbb{B}$. Then it follows that $\mathcal{C}$ does not admit a left query algorithm over $\mathbb{B}$ either, for otherwise $\mathcal{C}' = \mathcal{C} \setminus [A_n]_{\leftrightarrow}$ would admit a left query algorithm over $\mathbb{B}$.

(2) $[A_n]_{\leftrightarrow}$ does not admit a left query algorithm over $\mathbb{B}$. By Proposition 4.4, $\mathrm{CSP}(A_n)$ does not admit a left query algorithm over $\mathbb{B}$ either. Let $\mathcal{F}$ be an arbitrary finite non-empty set of connected instances. Since $\mathrm{CSP}(A_n)$ does not admit a left query algorithm over $\mathbb{B}$, there are $P \in \mathrm{CSP}(A_n)$ and $Q \notin \mathrm{CSP}(A_n)$ such that $\hom_{\mathbb{B}}(\mathcal{F}, P) = \hom_{\mathbb{B}}(\mathcal{F}, Q)$. It follows that

- $(P \oplus A_n) \in [A_n]_{\leftrightarrow}$, because both $P, A_n \in \mathrm{CSP}(A_n)$,
- $(Q \oplus A_n) \notin [A_n]_{\leftrightarrow}$, because $Q \notin \mathrm{CSP}(A_n)$,
- for all $i < n$, $(Q \oplus A_n) \notin [A_i]_{\leftrightarrow}$, because $A_n \not\rightarrow A_i$, and
- $\hom_{\mathbb{B}}(\mathcal{F}, P \oplus A_n) = \hom_{\mathbb{B}}(\mathcal{F}, Q \oplus A_n)$, because the instances in $\mathcal{F}$ are all connected.

The first three points above give $(P \oplus A_n) \in \mathcal{C}$ and $(Q \oplus A_n) \notin \mathcal{C}$. Therefore, by Proposition 3.2, $\mathcal{C}$ has no left query algorithm over $\mathbb{B}$. ◀

Note that Theorem 4.6 only applies to finite unions of homomorphic-equivalence classes. It may fail for infinite unions. Specifically, the class of all instances trivially admits a left query algorithm over $\mathbb{B}$, and it is the union of all equivalence classes $[A]_{\leftrightarrow}$, as A varies over all instances; however, as seen earlier, $[K_3]_{\leftrightarrow}$ does not admit any left query algorithm over $\mathbb{B}$.

▶ **Corollary 4.7.** *The following problem is NP-complete: given instance $A_1, \ldots, A_n$, does $\bigcup_{1\leq i\leq n} [A_i]_{\leftrightarrow}$ admit a left query algorithm over $\mathbb{B}$?*

Each class $\mathcal{C}$ that is closed under homomorphic equivalence can trivially be represented as a possibly-infinite union of classes of the form $[A]_{\leftrightarrow}$. The algorithmic problem of testing for the existence of a left query algorithm, of course, makes sense only for finitely presented inputs. This motivates the above corollary.

As a last case study, consider Boolean Datalog programs, i.e., Datalog programs with a zero-ary goal predicate. We omit a detailed definition of the syntax and semantics of Datalog, which can be found, e.g., in [1]. Each Datalog program P naturally defines a class of instances $\mathcal{C}_P$. It is well known that *the class $\mathcal{C}_P$ is closed under homomorphic equivalence.*

Furthermore, $\mathcal{C}_P$ is FO-definable if and only if P is "bounded" (meaning that there is a fixed number n, depending only on P and not on the input instance, such that P reaches its fixed point after at most n iterations), as was first shown by [2] and also follows from [27]. The boundedness problem for Boolean Datalog programs is undecidable [14]. Therefore, we have the following result.

▶ **Corollary 4.8.** *The following problem is undecidable: given a Boolean Datalog program P, does $\mathcal{C}_P$ admit a left query algorithm over $\mathbb{B}$.*

5 Existence vs. Counting: When Does Counting Not Help?

Left query algorithms over $\mathbb{N}$ are more powerful than left query algorithms over $\mathbb{B}$. This is trivially so because query algorithms over $\mathbb{B}$ cannot distinguish homomorphically equivalent instances.

▶ **Example 5.1.** Let $\mathcal{C}$ be the isomorphism class of the instance A consisting of the fact $R(a, a)$ (in other words, the single-node reflexive digraph). Since $\mathcal{C}$ is not closed under homomorphism equivalence, it does not admit a left query algorithm (nor a right query algorithm) over $\mathbb{B}$. On the other hand, it admits a left query algorithm over $\mathbb{N}$: by counting the number of homomorphisms from A to a given input instance B, we can verify that B contains a reflexive node; and by counting the number of homomorphisms from the instance A' consisting of a single edge $R(a, b)$, we can verify that the total number of edges in the graph is equal to 1. More precisely, let $\mathcal{F} = \{A, A'\}$ and let $X = \{(1, 1)\}$. Then $(\mathcal{F}, X)$ is a left query algorithm over $\mathbb{N}$ for $\mathcal{C}$.

As it turns out, in a precise sense, this is the only reason why left query algorithms over $\mathbb{N}$ are more powerful than left query algorithms over $\mathbb{B}$: the ability to count does not give more power when it comes to classes that are closed under homomorphic equivalence. This follows from the next theorem.

▶ **Theorem 5.2.** *Let $\mathcal{C}$ be a class of instances closed under homomorphic equivalence. For every finite set $\mathcal{F}$ of connected instances, the following statements are equivalent.*

1. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$ of the form $(\mathcal{F}, X)$ for some set X.*
2. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ of the form $(\mathcal{F}, X')$ for some set X'.*

Proof. The case $\mathcal{C} = \emptyset$ is trivial, so we will assume that $\mathcal{C}$ is non-empty. The implication (2) $\Longrightarrow$ (1) is given by Proposition 3.3. Let us prove the implication (1) $\Longrightarrow$ (2).

Let $(\mathcal{F}, X)$ be a left query algorithm over $\mathbb{N}$ for $\mathcal{C}$ where $\mathcal{F} = \{F_1, \ldots, F_k\}$ of pairwise non-isomorphic instances and $X \subseteq \mathbb{N}^k$. It is enough to focus on *simple* sets X, where a set X is simple if for every $1 \leq i \leq k$ and every $\mathbf{t}, \mathbf{t}' \in X$, $\mathbf{t}(i) = 0$ iff $\mathbf{t}'(i) = 0$. Indeed, assume that (1) $\Rightarrow$ (2) holds whenever X is simple. Then, if X is not simple, partition X into maximal simple subsets $X_1, \ldots, X_r$ and, for every $1 \leq i \leq r$, let $\mathcal{C}_i$ be the class of instances that admits the left query algorithm $(\mathcal{F}, X_i)$. It is easy to verify that $\mathcal{C}_i$ is closed under homomorphic equivalence and, hence, by assumption, admits a left query algorithm over $\mathbb{B}$ of the form $(\mathcal{F}, X'_i)$ for some set X'_i. Then $(\mathcal{F}, \bigcup_{1 \leq i \leq r} X'_i)$ is a left query algorithm over $\mathbb{B}$ for $\mathcal{C}$.

Let us assume that X is simple and non-empty (otherwise $\mathcal{C} = \emptyset$, contradicting our assumption) and let $\mathbf{t} \in X$. We can assume, by reordering the instances in $\mathcal{F}$ if necessary, that there exists $s \geq 0$, such that $\mathbf{t}(i) > 0$ for every $i \leq s$ and $\mathbf{t}(i) = 0$ for every $i > s$.

Consider the FO-sentence defined as:

$$\varphi = \bigwedge_{i \leq s} q^{F_i} \wedge \bigwedge_{i > s} \neg q^{F_i}$$

We shall prove that $\mathrm{Mod}(\varphi) = \mathcal{C}$, and therefore $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ (namely, the left query algorithm $(\mathcal{F}, \{\mathbf{t}'\})$ where $\mathbf{t}'(i) = 1$ for $i \leq s$ and $\mathbf{t}'(i) = 0$ for $i > s$). Since $\mathrm{Mod}(\varphi) = \mathcal{C}$ already holds when $s = 0$, we assume $s > 0$ in the sequel.

To this end we shall need the following two technical lemmas.

▶ **Lemma 5.3.** *Let $\mathbf{t}_1, \ldots, \mathbf{t}_r \in \mathbb{N}^s$ satisfying the following conditions:*

1. *For every $1 \leq i \leq s$, there exists $1 \leq j \leq r$ such that $\mathbf{t}_j(i) \neq 0$.*
2. *For every different $i, i' \in \{1, \ldots, s\}$, there exists $1 \leq j \leq r$ such that $\mathbf{t}_j(i) \neq \mathbf{t}_j(i')$.*

Then there exists some integer $c > 0$ such that for every $\mathbf{b} \in \mathbb{Z}^s$ whose entries are integers divisible by c, there is a multivariate polynomial with integer coefficients $p(x_1, \ldots, x_r)$ of degree s satisfying $p(0, \ldots, 0) = 0$ (that is, such that the independent term is zero) and such that, for each $1 \leq i \leq s$, $p(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i)) = \mathbf{b}(i)$.

Proof. Most likely this follows from known results but, in any case, we provide a self-contained proof. Let $\mathcal{G}$ be the set of all linear combinations $a_1\mathbf{t}_1 + \cdots a_r\mathbf{t}_r$ where each a_i is a non-negative integer. It follows from condition (1) that $\mathcal{G}$ contains a tuple where all entries are positive. Moreover it follows from (2) that $\mathcal{G}$ contains a tuple where all entries are different and positive. Indeed, if $\mathbf{t}, \mathbf{u} \in \mathcal{G}$ and $\mathbf{v} = d\mathbf{t} + \mathbf{u}$ for $d \in \mathbb{N}$ large enough, then for every different $i, i' \in \{1, \ldots, s\}$, we have

$$(\mathbf{t}(i) \neq \mathbf{t}(i') \;\text{ or }\; \mathbf{u}(i) \neq \mathbf{u}(i')) \;\text{ implies }\; \mathbf{v}(i) \neq \mathbf{v}(i').$$

Now, let $\mathbf{u} = a_1\mathbf{t}_1 + \cdots + a_r\mathbf{t}_r$ be any tuple where all entries are different and positive. It is easy to see that the $(s \times s)$-matrix M whose rows are $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^s$ is non-singular. Indeed, if $\mathbf{u} = (u_1, \ldots, u_s)$, then $\det(M) = u_1 \cdots u_s \cdot \det(N)$, where N is the $(s \times s)$-matrix with rows $\mathbf{u}^0, \mathbf{u}^1, \ldots, \mathbf{u}^{s-1}$ which is Vandermonde. It is well known that Vandermonde matrices are non-singular whenever $\mathbf{u}$ has no repeated elements (see [18] for example).

To finish the proof we choose c to be $|\det(M)|$. By assumption all entries of $\mathbf{b}$ are divisible by c which implies that all entries of $M^{-1}\mathbf{b}$ are integers, that is, $\mathbf{b}$ can be expressed as $e_1\mathbf{u}^1 + \cdots + e_s\mathbf{u}^s$ for some $e_1, \ldots, e_s \in \mathbb{Z}$. Hence, the polynomial $p(x_1, \ldots, x_r) = e_1 y^1 + \cdots + e_s y^s$ where $y = a_1 x_1 + \cdots + a_r x_r$ satisfies the claim. ◀

▶ **Lemma 5.4.** *Let F, F' be instances such that there is no surjective homomorphism from F onto F'. Then there exists some subinstance H of F' such that $\hom_{\mathbb{N}}(F, H) \neq \hom_{\mathbb{N}}(F', H)$.*

Proof. This is a natural adaptation of the Lovász's proof that two instances are isomorphic if and only if they have the same left homomorphism-count vector.

For every instance G, we write $\mathrm{sur}_{\mathbb{N}}(G, F')$ for the number of surjective homomorphisms from G onto F'; moreover, for every subset $S \subseteq \mathrm{adom}(F')$, we write $\hom^S_{\mathbb{N}}(G, F')$ for the number of homomorphisms $h : G \to F'$ whose range is contained in S. Let $n := |\mathrm{adom}(F')|$. By the Inclusion-Exclusion Principle, then

$$\mathrm{sur}_{\mathbb{N}}(G, F') = \sum_{S \subseteq \mathrm{adom}(F')} (-1)^{n-|S|} \hom^S_{\mathbb{N}}(G, F').$$

Since $\mathrm{sur}_{\mathbb{N}}(F, F') = 0$ and $\mathrm{sur}_{\mathbb{N}}(F', F') > 0$, we have $\mathrm{sur}_{\mathbb{N}}(F, F') \neq \mathrm{sur}_{\mathbb{N}}(F', F')$ and it follows by the above discussion that there is a subset $S \subseteq \mathrm{adom}(F')$ such that $\hom^S_{\mathbb{N}}(F, F') \neq \hom^S_{\mathbb{N}}(F', F')$. Let H be the maximum subinstance of F' with $\mathrm{adom}(H) \subseteq S$, then we have $\hom_{\mathbb{N}}(F, H) = \hom^S_{\mathbb{N}}(F, F') \neq \hom^S_{\mathbb{N}}(F', F') = \hom_{\mathbb{N}}(F', H)$. ◀

Let us now continue with the proof that $\mathrm{Mod}(\varphi) = \mathcal{C}$. The direction $\supseteq$ is immediate. For the converse, we must prove that every $B \in \mathrm{Mod}(\varphi)$ belongs also to $\mathcal{C}$. Let $B \in \mathrm{Mod}(\varphi)$, and choose an arbitrary $A \in \mathcal{C}$ (by the assumption that $\mathcal{C} \neq \emptyset$). Let $\mathbf{t}^A = \hom_{\mathbb{N}}(\mathcal{F}, A)$ and $\mathbf{t}^B = \hom_{\mathbb{N}}(\mathcal{F}, B)$. Note that $\mathbf{t}^A(i) = \mathbf{t}^B(i) = 0$ for every $i > s$ and $\mathbf{t}^A(i)$ and $\mathbf{t}^B(i)$ are strictly positive for every $i \leq s$.

Let $\mathcal{H} = \{H_1, \ldots, H_r\}$ be the non-empty set of instances constructed as follows:

(i) For every $1 \leq i \leq s$, F_i is contained in $\mathcal{H}$.

(ii) For every $i, i' \in \{1, \ldots, s\}$ such that there is no surjective homomorphism $F_i \to F_{i'}$, $\mathcal{H}$ contains the instance H given by Lemma 5.4.

For each $1 \leq i \leq r$, let $\mathbf{t}_i = \hom_{\mathbb{N}}(\{F_1, \ldots, F_s\}, H_i)$. It can be readily verified that $\mathbf{t}_1, \ldots, \mathbf{t}_r$ satisfy the conditions of Lemma 5.3. Condition (1) is guaranteed due to step (i) in the construction of $\mathcal{H}$. For condition (2), let i, i' be any pair of different integers in $\{1, \ldots, s\}$. Since F_i and $F_{i'}$ are not isomorphic, it must be the case that there is no surjective homomorphism $F_i \to F_{i'}$ or there is no surjective homomorphism $F_{i'} \to F_i$. Hence, due to step (ii), $\mathcal{H}$ contains some instance H_j witnessing $\mathbf{t}_j(i) \neq \mathbf{t}_j(i')$.

Let $c > 0$ be given by Lemma 5.3, let $\mathbf{b} \in \mathbb{Z}^s$ where $\mathbf{b}(i) = c(\mathbf{t}^B(i) - \mathbf{t}^A(i))$ for every $1 \leq i \leq s$, and let $p(x_1, \ldots, x_r)$ be the polynomial given by Lemma 5.3 for $\mathbf{b}$. We can express $p(x_1, \ldots, x_r)$ as $p_A(x_1, \ldots, x_r) - p_B(x_1, \ldots, x_r)$ where all the coefficients in p_A and p_B are positive.

For every polynomial $q(x_1, \ldots, x_r)$ where the independent term is zero, consider the instance H_q defined inductively as follows:

- If $q = x_j$, then H_q is H_j.
- If $q = u + v$, then H_q is $H_u \oplus H_v$.
- If $q = u \cdot v$, then $H_q = H_u \otimes H_v$.

▶ **Lemma 5.5.** *Let H_q be constructed as above. Then:*

1. $H_q \to F_1 \oplus \cdots \oplus F_s$.
2. *Let F be a connected instance and let $\mathbf{t} = \hom_{\mathbb{N}}(F, \mathcal{H})$. Then $\hom_{\mathbb{N}}(F, H_q) = q(\mathbf{t})$.*

This lemma directly follows from the definition of H_q. More precisely, the first item follows from the fact that $H_j \to F_1 \oplus \cdots \oplus F_s$ for every $H_j \in \mathcal{H}$ (as can be shown by a straightforward induction argument). The second item follows from the definition of H_q and Proposition 2.1.

Let $A' = A_1 \oplus \cdots \oplus A_c \oplus H_{p_A}$ where $A_1, \ldots, A_c$ are disjoint copies of A. Similarly, define $B' = B_1 \oplus \cdots \oplus B_c \oplus H_{p_B}$ where $B_1, \ldots, B_c$ are disjoint copies of B. We claim that $\mathbf{t}^{A'} = \hom_{\mathbb{N}}(\mathcal{F}, A')$ and $\mathbf{t}^{B'} = \hom_{\mathbb{N}}(\mathcal{F}, B')$ coincide.

Let $1 \leq i \leq k$ and consider first the case $i > s$. It follows from Lemma 5.5(1) that $F_i \not\to H_{p_A}$. Indeed, if $F_i \to H_{p_A}$, then $F_i \to F_{i'}$ for some $i' \leq s$ implying that $\mathcal{C} = \emptyset$, contradicting our assumption. Similarly $F_i \not\to H_{p_B}$. Consequently, $\mathbf{t}^{A'}(i) = c \cdot \mathbf{t}^A(i) = 0 = c \cdot \mathbf{t}^B(i) = \mathbf{t}^{B'}(i)$. For the other case, namely $i \leq s$, we have

$$\begin{aligned}
\mathbf{t}^{B'}(i) - \mathbf{t}^{A'}(i) &= \hom_{\mathbb{N}}(F_i, B') - \hom_{\mathbb{N}}(F_i, A') \\
&= c \cdot \hom_{\mathbb{N}}(F_i, B) - c \cdot \hom_{\mathbb{N}}(F_i, A')) + \hom_{\mathbb{N}}(F_i, H_{p_B}) - \hom_{\mathbb{N}}(F_i, H_{p_A}) \\
&= c \cdot (\mathbf{t}^B(i) - \mathbf{t}^A(i)) + \hom_{\mathbb{N}}(F_i, H_{p_B}) - \hom_{\mathbb{N}}(F_i, H_{p_A}) \\
&= c \cdot (\mathbf{t}^B(i) - \mathbf{t}^A(i)) + p_B(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i)) - p_A(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i)) \\
&= c \cdot (\mathbf{t}^B(i) - \mathbf{t}^A(i)) - p(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i)) \\
&= c \cdot (\mathbf{t}^B(i) - \mathbf{t}^A(i)) - \mathbf{b}(i) = 0,
\end{aligned}$$

where $\hom_{\mathbb{N}}(F_i, H_{p_A}) = p_A(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i))$ and $\hom_{\mathbb{N}}(F_i, H_{p_B}) = p_B(\mathbf{t}_1(i), \ldots, \mathbf{t}_r(i))$ hold by Lemma 5.5(2).

To finish the proof, note that since $\mathbf{t}^A(i) > 0$ for every $1 \leq i \leq s$ it follows by Lemma 5.5(1) that $A' \to A$, and, hence $A \leftrightarrow A'$. Similarly we have $B' \leftrightarrow B$. Since $\mathcal{C}$ is closed under homomorphic equivalence we have that $A' \in \mathcal{C}$. Since $\mathbf{t}^{A'} = \mathbf{t}^{B'}$ it follows that B' and, hence, B belong to $\mathcal{C}$ as well. ◀

▶ **Corollary 5.6.** *Let $\mathcal{C}$ be a class of instances closed under homomorphic equivalence. Then the following statements are equivalent.*

1. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$.*
2. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$.*
3. *$\mathcal{C}$ is FO-definable.*

The equivalence of statements (1) and (2) follows from Theorem 5.2 together with Proposition 3.2. The equivalence of statements (2) and (3) was already established in Corollary 4.2.

We would like to point out some interesting special cases of Theorem 5.2. The first pertains to CSPs. Let us say that a constraint satisfaction problem CSP(B) is "*determined by* $\hom_K(\mathcal{F}, \cdot)$" for some $K \in \{\mathbb{B}, \mathbb{N}\}$ and some class $\mathcal{F}$ of instances, if the following holds for all instances A and A': if $\hom_K(\mathcal{F}, A) = \hom_K(\mathcal{F}, A')$ then $A \in \mathrm{CSP}(B)$ iff $A' \in \mathrm{CSP}(B)$. Then Theorem 5.2 can be rephrased as follows: for every finite set of connected instances $\mathcal{F}$, every CSP determined by $\hom_{\mathbb{N}}(\mathcal{F}, \cdot)$ is determined by $\hom_{\mathbb{B}}(\mathcal{F}, \cdot)$. In contrast, for $\mathcal{T}$ the infinite class of all trees, the CSPs determined by $\hom_{\mathbb{B}}(\mathcal{T}, \cdot)$ form a proper subclass of those determined by $\hom_{\mathbb{N}}(\mathcal{T}, \cdot)$.[2] This follows from results in [8], because the former are precisely the CSPs that can be solved using arc-consistency, while the latter are precisely the CSPs that can be solved using basic linear programming relaxation (BLP). See [23, Example 99] for an example of a CSP that can be solved using BLP but not using arc-consistency.

The second special case pertains to homomorphic-equivalence classes. Given the importance of homomorphic equivalence as a notion of equivalence in database theory, it is natural to ask when a database instance A can be uniquely identified up to homomorphic equivalence by means of a left query algorithm. As we saw earlier, in Section 4, for left query algorithm over $\mathbb{B}$, this is the case if and only if CSP(A) is FO-definable (a condition that can be tested effectively, and, in fact, is NP-complete to test). It follows from Corollary 5.6 that the same criterion determines whether A can be uniquely identified up to homomorphic equivalence by means of a left query algorithm over $\mathbb{N}$. This extends naturally to finite unions of homomorphic equivalence classes.

Finally, let us consider again classes $\mathcal{C}$ defined by a Boolean Datalog program P. It follows from the results we mentioned in Section 4 that such a class of instances $\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$ if and only if P is bounded, and that testing for the existence of a left query algorithm over $\mathbb{N}$ is undecidable.

6 Right Query Algorithms

Just as for left query algorithms, we have that every class of instances that admits a right query algorithm over $\mathbb{B}$ is closed under homomorphic equivalence. However, *unlike* left query algorithms, a class of instances that admits a right query algorithm over $\mathbb{B}$ is not necessarily FO-definable. Concretely, as we saw in Example 3.6, the class of 3-colorable graphs admits a right query algorithm over $\mathbb{B}$, but is not FO-definable. In fact, *every* constraint satisfaction problem CSP(A) admits a right query algorithm over $\mathbb{B}$, and the FO-definable ones are precisely those that admit a left query algorithm over $\mathbb{B}$. The next result is straightforward.

[2] Note that the definitions of $\hom_{\mathbb{N}}(\mathcal{F}, \cdot)$ and $\hom_{\mathbb{B}}(\mathcal{F}, \cdot)$ extend naturally to infinite classes $\mathcal{F}$.

▶ **Proposition 6.1.** *Let $\mathcal{C}$ be a class of instances. Then $\mathcal{C}$ admits a right query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ is definable by a Boolean combination of CSPs.*

We also saw in Section 3 that not every homomorphic-equivalence closed FO-definable class admits a right query algorithm over $\mathbb{B}$. For example, the class of triangle-free graphs does not admit a right query algorithm over $\mathbb{B}$ (or even over $\mathbb{N}$). This raises the question which homomorphic-equivalence closed FO-definable classes admit a right query algorithm over $\mathbb{B}$. Equivalently, *which classes $\mathcal{C}$ that admit a left query algorithm over $\mathbb{B}$, admit a right query algorithm over* $\mathbb{B}$? We will address this question next by making use of two known results.

▶ **Theorem 6.2** (Sparse Incomparability Lemma, [22]). *Let $m, n \geq 0$. For every instance B there is an instance B^* of girth at least m such that, for all instances D with $|\mathrm{adom}(D)| \leq n$, we have $B \in \mathrm{CSP}(D)$ if and only if $B^* \in \mathrm{CSP}(D)$.*

A *finite homomorphism duality* is a pair of finite sets $(\mathcal{F}, \mathcal{D})$ such that, for every instance A, the following are equivalent: (i) $F \rightarrow A$ for some $F \in \mathcal{F}$; (ii) $A \not\rightarrow D$ for all $D \in \mathcal{D}$. We make use of the following known characterization of this notion.

▶ **Theorem 6.3** ([13]). *Let A be an instance. Then the following statements are equivalent.*
1. *A is homomorphically equivalent to an acyclic instance.*
2. *There is a finite homomorphism duality $(\mathcal{F}, \mathcal{D})$ with $\mathcal{F} = \{A\}$.*
3. *There is a finite homomorphism duality $(\mathcal{F}, \mathcal{D})$ with $\mathcal{F} = \{A\}$ and where $\mathcal{D}$ consists of instances D for which $\mathrm{CSP}(D)$ is FO-definable.*

To see the implication (2) $\Longrightarrow$ (3), note that, if $(\mathcal{F}, \mathcal{D})$ is a finite homomorphism duality, then $\bigcup_{D \in \mathcal{D}} \mathrm{CSP}(D)$ is a FO-definable class (because it is defined by the negation of the UCQ $\bigvee_{F \in \mathcal{F}} q^F$). It follows, by the same reasoning as in the proof of Proposition 4.3, that exists $\mathcal{D}' \subseteq \mathcal{D}$ such that $(\mathcal{F}, \mathcal{D}')$ is a finite homomorphism duality and such that, for each $D \in \mathcal{D}'$, we have that $\mathrm{CSP}(D)$ is FO-definable.

Our next result characterizes the classes that admit both a left query algorithm and a right query algorithm over $\mathbb{B}$.

▶ **Theorem 6.4.** *Let $\mathcal{C}$ be a class of instances. Then the following statements are equivalent.*
1. *$\mathcal{C}$ admits both a left query algorithm and a right query algorithm over $\mathbb{B}$.*
2. *$\mathcal{C}$ admits a left query algorithm over $\mathbb{N}$ and a right query algorithm over $\mathbb{B}$.*
3. *$\mathcal{C}$ is definable by a Boolean combination of Berge-acyclic CQs.*
4. *$\mathcal{C}$ is definable by a Boolean combination of FO-definable CSPs.*

Proof. The equivalence of statements (1) and (2) follows from Corollary 5.6. The proof of the remaining equivalences is as follows.
(1) $\Longrightarrow$ (3) By Theorem 4.1 and Proposition 6.1, $\mathcal{C}$ is definable by a Boolean combination φ of CQs, as well as by a Boolean combination ψ of CSPs. Let φ' be obtained from φ by replacing each conjunctive query q by the disjunction of all homomorphic images of q that are Berge-acyclic. We claim that φ' defines $\mathcal{C}$.

By Theorem 6.3, we know that φ' is also equivalent to a Boolean combination of CSPs, which we may call ψ'. Let n be the maximum size of a CSP occurring in ψ and ψ'. Also, let m be greater than the maximum size of the CQs in φ. Let B be any instance, and let B^* now be the instance given by Theorem 6.2 (for m, n as chosen above). By construction, B and B^* agree with each other on their membership in CSPs of size at most n, which include all CSPs occurring in ψ as well as ψ', and therefore, B and B^* agree on ψ and ψ'. Since ψ is equivalent to φ and ψ' is equivalent to φ', this means that B and B^* agree on φ and φ'. Furthermore, by construction, φ' is equivalent to φ on instances of girth at least m (because

every homomorphic image of a CQ of size less than m in such an instance must be acyclic). In particular, It follows that $B^* \models \varphi$ iff $B^* \models \varphi'$. Putting everything together, we have that $B \models \varphi$ iff $B^* \models \varphi$ iff $B^* \models \varphi'$ iff $B \models \varphi'$.
(3) $\Longrightarrow$ (4) This follows from Theorem 6.3 (for A the canonical instance of q): we simply replace each Berge-acyclic conjunctive query q by the conjunction $\bigwedge_{D \in \mathcal{D}} \neg\mathrm{CSP}(D)$.
(4) $\Longrightarrow$ (1) This follows from Theorem 4.1 and Proposition 6.1. ◀

Let $\mathcal{C}$ be a class that admits a left query algorithm over $\mathbb{B}$ or, equivalently, let $\mathcal{C}$ be definable by a Boolean combination of CQs. It follows that $\mathcal{C}$ admits a right query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ is definable by a Boolean combination of *Berge-acyclic* CQs. Similarly, let $\mathcal{C}$ be a class that admits a right query algorithm over $\mathbb{B}$ or, equivalently, let $\mathcal{C}$ be definable by a Boolean combination of CSPs. It follows that $\mathcal{C}$ admits a left query algorithm over $\mathbb{B}$ if and only if $\mathcal{C}$ is definable by a Boolean combination of *FO-definable* CSPs.

Finally, we will consider the question when a class of the form $[A]_{\leftrightarrow}$ admits a right query algorithm over $\mathbb{B}$.

▶ **Theorem 6.5.** *Let A be an instance. Then the following statements are equivalent.*
1. *$[A]_{\leftrightarrow}$ has a right query algorithm over $\mathbb{B}$.*
2. *$\{B : A \to B\}$ has a right query algorithm over $\mathbb{B}$.*
3. *A is homomorphically equivalent to an acyclic instance.*

Moreover, testing whether this holds (for a given instance A) is NP-complete.

Proof. We will close a cycle of implications.
(1) $\Longrightarrow$ (2): For this, we use the exponentiation operation X^Y [17]. This operation has the property that $X \to Y^Z$ if and only if $X \otimes Z \to Y$. Assume $\{B : A \to B\}$ does not admit a right query algorithm over $\mathbb{B}$. Consider an arbitrary finite set of instances $\mathcal{F} = \{F_1, \dots, F_k\}$ (think: candidate right query algorithm for $[A]_{\leftrightarrow}$). Since $\{B : A \to B\}$ does not admit a right query algorithm over $\mathbb{B}$, for the set $\mathcal{F}^A := \{F_1^A, \dots, F_k^A\}$ there are instances P and Q with $A \to P$ and $A \not\to Q$ such that $\hom_{\mathbb{B}}(P, \mathcal{F}^A) = \hom_{\mathbb{B}}(Q, \mathcal{F}^A)$ or, equivalently, $\hom_{\mathbb{B}}(P \otimes A, \mathcal{F}) = \hom_{\mathbb{B}}(Q \otimes A, \mathcal{F})$. Let $P' := P \otimes A$ and let $Q' := Q \otimes A$. Then, by Proposition 2.1, $P' \in [A]_{\leftrightarrow}$ and $Q' \notin [A]_{\leftrightarrow}$ but $\hom_{\mathbb{B}}(P', \mathcal{F}) = \hom_{\mathbb{B}}(Q', \mathcal{F})$. Therefore $[A]_{\leftrightarrow}$ has no right query algorithm over $\mathbb{B}$.
(2) $\Longrightarrow$ (3): Assume that $\{B \mid A \to B\}$ admits a right query algorithm for $\mathbb{B}$. We will construct a finite homomorphism duality $(\mathcal{F}, \mathcal{D})$ with $\mathcal{F} = \{A\}$. It then follows by Theorem 6.3 that A is homomorphically equivalent to an acyclic instance. Let $B_1, \dots B_n$ be all those instances B_i used by the right query algorithm for which it holds that $A \not\to B_i$. We claim that $(\{A\}, \{B_1, \dots, B_n\})$ is a finite homomorphism duality. Let C be any instance. If $C \to B_i$ for some $i \le n$, then $A \not\to C$ (otherwise, by transitivity, we would have that $A \to B_i$). Conversely, if $A \not\to C$, then the algorithm must answer "no" on input C while it answers "yes" on input $C \oplus A$. Therefore, one of the right-queries made by the algorithm must differentiate C from $C \oplus A$. It is easy to see that the right-query in question must consist of an instance into which C maps but A does not. This instance must then be among the B_i, and $C \to B_i$.
(3) $\Longrightarrow$ (1): By Theorem 6.3, there is a finite homomorphism duality $(\{A\}, \mathcal{D})$. In particular, for all instances C, we have that $C \in [A]_{\leftrightarrow}$ if and only if $C \to A$ and $C \not\to D$ for all $D \in \mathcal{D}$.

To test if a given instance is homomorphically equivalent to an acyclic instance, it suffices to test that its core is acyclic (equivalently, that it has an acyclic retract). This can clearly be done in NP. The NP-hardness follows directly from Theorem 6 in [10]. ◀

The preceding Theorem 6.5 can be thought of as an analogue of Proposition 4.4 for right query algorithms. Again, this result extends to finite unions of homomorphic-equivalence classes.

▶ **Theorem 6.6.** *For all instances $A_1, \dots, A_n$, the following statements are equivalent.*

1. *$\bigcup_{1\le i\le n}[A_i]_\leftrightarrow$ admits a right query algorithm over $\mathbb{B}$.*
2. *Each $[A_i]_\leftrightarrow$, for $i = 1, \dots, n$, admits a right query algorithm over $\mathbb{B}$.*

In particular, testing whether this holds (for given instances $A_1, \dots, A_n$) is NP-complete.

Proof. It is clear that the second statement implies the first. We will prove by induction on n that the first statement implies the second. The base case ($n = 1$) is immediate since then statements (1) and (2) coincide. Next, let $n > 1$ and $\mathcal{C} := \bigcup_{1\le i\le n}[A_i]_\leftrightarrow$. We proceed by contraposition, assuming that $[A_i]_\leftrightarrow$ does not admit a right query algorithm over $\mathbb{B}$ for some $i \le n$. We may assume without loss of generality that $A_1, \dots, A_n$ are pairwise not homomorphically equivalent. Note that $\to$ induces a preorder among $A_1, \dots, A_n$ and, since n is finite, there is a minimal. Without loss of generality, assume that A_n is a minimal, that is, $A_i \not\to A_n$ for all $i < n$. We distinguish two cases.

(1) $[A_n]_\leftrightarrow$ admits a right query algorithm over $\mathbb{B}$. Then, for some $i \le n-1$, $[A_i]_\leftrightarrow$ does not admit any right query algorithm over $\mathbb{B}$. By induction hypothesis, we have $\mathcal{C}' := \bigcup_{1\le i\le n-1}[A_i]_\leftrightarrow$ does not admit any right query algorithm over $\mathbb{B}$. Then $\mathcal{C}$ does not admit a right query algorithm over $\mathbb{B}$ either, since $\mathcal{C}' = \mathcal{C} \setminus [A_n]_\leftrightarrow$.

(2) $[A_n]_\leftrightarrow$ does not admit any right query algorithm over $\mathbb{B}$. By Theorem 6.5, the class $\{B : A \to B\}$ does not admit any right query algorithm over $\mathbb{B}$, either. Consider an arbitrary finite non-empty set of instances $\mathcal{F} = \{F_1, \dots, F_k\}$. Since $\{B : A \to B\}$ does not admit any right query algorithm over $\mathbb{B}$, for the set $\mathcal{F}^{A_n} = \{F_1^{A_n}, \dots, F_k^{A_n}\}$ there are instances P and Q with $A_n \to P$ and $A_n \not\to Q$ such that $\hom_{\mathbb{B}}(P, \mathcal{F}^{A_n}) = \hom_{\mathbb{B}}(Q, \mathcal{F}^{A_n})$, which implies that $\hom_{\mathbb{B}}(P \otimes A_n, \mathcal{F}) = \hom_{\mathbb{B}}(Q \otimes A_n, \mathcal{F})$. It follows by Proposition 2.1 that

- $(P \otimes A_n) \in [A_n]_\leftrightarrow$ because $A_n \to P$,
- $(Q \otimes A_n) \notin [A_n]_\leftrightarrow$ because $A_n \not\to Q$,
- for all $i < n$, $(Q \otimes A_n) \notin [A_i]_\leftrightarrow$ because $A_i \not\to A_n$.

Let $P' := P \otimes A_n$ and let $Q' := Q \otimes A_n$. Then the above discussion yields that $P' \in \mathcal{C}$ and $Q' \notin \mathcal{C}$ while $\hom_{\mathbb{B}}(P', \mathcal{F}) = \hom_{\mathbb{B}}(Q', \mathcal{F})$. Therefore, $\mathcal{C}$ does not admit any right query algorithm over $\mathbb{B}$. ◀

▶ Remark 6.7. We saw in Example 3.4 that the class of triangle-free graphs, which clearly has a left query algorithm over $\mathbb{B}$, does not admit a right query algorithm over $\mathbb{B}$ or over $\mathbb{N}$. Observe that this class is defined by the negation of the "triangle" conjunctive query $\exists xyz(R(x, y) \wedge R(y, z) \wedge R(z, x))$. In light of Theorem 6.4, the lack of a right query algorithm over $\mathbb{B}$ for this class can be "explained" by the fact that this conjunctive query is not Berge-acyclic. Furthermore, in Example 3.7 we mentioned that the class $[K_3]_\leftrightarrow$, that is, the class of graphs that are 3-colorable and also contain a triangle, does not admit a right query algorithm over $\mathbb{B}$. This follows from Theorem 6.5.

We conclude this section with an open problem.

▶ **Question 6.8.** Does a suitable analogue of Theorem 5.2 hold for right query algorithms?

7 Summary and Discussion of Related Topics

Inspired by the work of Chen et al. [9], we extended their framework and studied various types of query algorithms, where a query algorithm for a class $\mathcal{C}$ of instances determines whether a given input instance belongs to $\mathcal{C}$ by making a finite number of (predetermined) queries that ask for the existence of certain homomorphisms or for the number of certain homomorphisms. Specifically, we introduced and studied *left query algorithms* and *right* *query algorithms* over $\mathbb{B}$, as well as over $\mathbb{N}$. Our results delineate the differences in expressive power between these four types of query algorithms. In particular, they pinpoint when the ability to count homomorphisms is essential for the existence of left query algorithms.

Relationship to view determinacy. Recently, Kwiecien et al. [24] studied view determinacy under bag semantics. In particular, they obtained a decidability result for determinacy with respect to Boolean views under bag-set semantics. We will briefly describe their framework and relate it to ours. At the most abstract level, a *view* is simply a function f that takes as input a database instance. Specifically, under set semantics, every Boolean CQ specifies a view that is a function from database instances to $\{0, 1\}$, while, under bag-set semantics, every Boolean CQ specifies a view that is a function from database instances to $\mathbb{N}$. We say that a finite collection of views $f_1, \ldots, f_k$ *determines* a view g, if for all database instances A and B, if $f_i(A) = f_i(B)$ for all $i \leq k$, then $g(A) = g(B)$. The aforementioned result from [24] asserts that the following problem is decidable: given views $f_1, \ldots, f_k$ and g specified by Boolean CQs under bag-set semantics, is g is determined by $f_1, \ldots, f_k$?

We now describe the relationship between the above notion of view determinacy and our framework. Let $\mathcal{C}$ be a class of instances and let $\mathcal{F} = \{F_1, \ldots, F_k\}$ be a finite set of instances. Then the following are equivalent:

1. There exists a set X such that $(\mathcal{F}, X)$ is a left query algorithm over $\mathbb{N}$ for $\mathcal{C}$,
2. $f_1, \ldots, f_k$ determine $g_{\mathcal{C}}$ where $f_i(A) = \hom_{\mathbb{N}}(F_i, A)$ and $g_{\mathcal{C}}$ is the indicator function of $\mathcal{C}$ (i.e., $g_{\mathcal{C}}(A) = 1$ if $A \in \mathcal{C}$ and $g_{\mathcal{C}}(A) = 0$ otherwise).

This tells that there are some important differences between our framework and the one in [24]: (i) when we study the existence of left query algorithms, the set $\mathcal{F}$ is not fixed, whereas, in the view determinacy problem, the views are given as part of the input; (ii) in the view determinacy problem studied in [24], the view g is specified by a CQ with bag-set semantics, whereas in our case g is a Boolean-valued function since it is the indicator function of a class of instances, (iii) we do not assume that the class $\mathcal{C}$ is specified by a Boolean CQ. Indeed, if $\mathcal{C}$ were specified by a Boolean CQ q, then a left query algorithm would trivially exist, where $\mathcal{F}$ simply consists of (the canonical instance of) q itself.

Other semirings. Query algorithm over $\mathbb{B}$ and query algorithm over $\mathbb{N}$ can be viewed as special cases of a more general setting, namely that of a query algorithm over a semiring. There is a body of research in the interface of databases and semirings, including the study of provenance of database queries using semirings [16, 19], the study of the query containment problem under semiring semantics [15, 21], and, more recently, the study of Datalog under semiring semantics [20]. In these studies, the semirings considered are *positive*, which means that they are commutative semirings with no zero divisors and with the property that the sum of any two non-zero elements is non-zero. It is perfectly meaningful to define homomorphism counts over positive semirings and then investigate query algorithms over such semirings. In particular, it may be interesting to investigate query algorithms over the *tropical semiring* $\mathbb{R} = (R \cup \{\infty\}, \min, +)$, where R is the set of real numbers, since it is well known that various *shortest-distance* problems can be naturally captured using this semiring.

Adaptive query algorithms. The query algorithms $(\mathcal{F}, X)$ studied in this paper are *non-adaptive*, in the sense that the set $\mathcal{F} = \{F_1, \ldots, F_k\}$ is fixed up-front. In contrast, an *adaptive* query algorithm may decide the set of instances $\mathcal{F}$ at run-time, that is to say, the choice of F_i may depend on the homomorphism-count vector for $F_1, \ldots, F_{i-1}$. As pointed out in the Introduction, whenever a class $\mathcal{C}$ admits an adaptive left (right) query algorithm over $\mathbb{B}$, then it also admits a non-adaptive left (right) query algorithm over $\mathbb{B}$. Note that the most "economical" (as regards the number of instances used) non-adaptive algorithm for a class $\mathcal{C}$ may use a larger set $\mathcal{F}$ of instances than the adaptive one, but the number of instances used by the non-adaptive algorithm is still finite. Thus, adaptive query algorithms over $\mathbb{B}$ do not offer higher expressive power than adaptive ones. The situation for $\mathbb{N}$ is quite different: it was shown in [9] that *every* isomorphism-closed class of instances (in particular, every homomorphic-equivalence closed class) admits an adaptive left k-query algorithm over $\mathbb{N}$ already for $k = 3$; therefore, adaptive left query algorithms over $\mathbb{N}$ have higher expressive power than adaptive left query algorithms over $\mathbb{B}$, even when it comes to homomorphic-equivalence closed classes. Switching sides, note that the class of triangle-free graphs (Example 3.4) does not admit an adaptive right query algorithm over $\mathbb{N}$, as was shown in [9, Proposition 8.2]; hence it is a meaningful question to ask: which homomorphic-equivalence closed classes admit an adaptive right query algorithm over $\mathbb{N}$?

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*, volume 8. Addison-Wesley Reading, 1995.

2 Miklos Ajtai and Yuri Gurevich. Datalog vs first-order logic. *Journal of Computer and System Sciences*, 49(3):562–588, 1994. 30th IEEE Conference on Foundations of Computer Science. doi:10.1016/S0022-0000(05)80071-6.

3 Albert Atserias, Phokion G Kolaitis, and Wei-Lin Wu. On the expressive power of homomorphism counts. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470543.

4 Michal Bielecki and Jan Van den Bussche. Database interrogation using conjunctive queries. In *Database Theory - ICDT 2003, 9th Int. Conf., 2003, Proceedings*, volume 2572 of *Lecture Notes in Computer Science*, pages 259–269. Springer, 2003. doi:10.1007/3-540-36285-1_17.

5 Meghyn Bienvenu, Balder Ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4), dec 2015. doi:10.1145/2661643.

6 Jan Böker, Yijia Chen, Martin Grohe, and Gaurav Rattan. The complexity of homomorphism indistinguishability. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019*, volume 138 of *LIPIcs*, pages 54:1–54:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.54.

7 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *58th IEEE Annual Symp. on Found. of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.

8 Silvia Butti and Víctor Dalmau. Fractional homomorphism, Weisfeiler-Leman invariance, and the Sherali-Adams hierarchy for the constraint satisfaction problem. In *International Symposium on Mathematical Foundations of Computer Science*, 2021. doi:10.4230/LIPIcs.MFCS.2021.27.

9 Yijia Chen, Jörg Flum, Mingjun Liu, and Zhiyang Xun. On algorithms based on finitely many homomorphism counts. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, 2022. doi:10.4230/LIPIcs.MFCS.2022.32.

10 Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002. doi:10.1007/3-540-46135-3_21.

11 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.40.

12 Zdenek Dvorák. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory*, 64(4):330–342, 2010. doi:10.1002/jgt.20461.

13 Jan Foniok, Jaroslav Nešetřil, and Claude Tardif. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *European Journal of Combinatorics*, 29(4):881–899, 2008. doi:10.1016/j.ejc.2007.11.017.

14 Haim Gaifman, Harry Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, jul 1993. doi:10.1145/174130.174142.

15 Todd J. Green. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, 49(2):429–459, 2011. doi:10.1007/s00224-011-9327-6.

16 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In Leonid Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2007*, pages 31–40. ACM, 2007. doi:10.1145/1265530.1265535.

17 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004.

18 Dan Kalman. The generalized Vandermonde matrix. *Mathematics Magazine*, 57(1):15–21, 1984. doi:10.2307/2690290.

19 Grigoris Karvounarakis and Todd J. Green. Semiring-annotated data: queries and provenance? *SIGMOD Rec.*, 41(3):5–14, 2012. doi:10.1145/2380776.2380778.

20 Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, Dan Suciu, and Yisu Remy Wang. Convergence of datalog over (pre-) semirings. In *PODS '22: Int. Conf. on Management of Data, Philadelphia, 2022*, pages 105–117. ACM, 2022. doi:10.1145/3517804.3524140.

21 Egor V. Kostylev, Juan L. Reutter, and András Z. Salamon. Classification of annotation semirings over containment of conjunctive queries. *ACM Trans. Database Syst.*, 39(1):1:1–1:39, 2014. doi:10.1145/2556524.

22 Gábor Kun. Constraints, MMSNP and expander relational structures. *Combinatorica*, 33(3):335–347, 2013. doi:10.1007/s00493-013-2405-4.

23 Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *European Journal of Combinatorics*, 52:338–367, 2016. Special Issue: Recent Advances in Graphs and Analysis. doi:10.1016/j.ejc.2015.07.011.

24 Jaroslaw Kwiecien, Jerzy Marcinkowski, and Piotr Ostropolski-Nalewaja. Determinacy of real conjunctive queries. the boolean case. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '22, pages 347–358, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3517804.3524168.

25 László Lovász. Operations with structures. *Acta Math. Acad. Sci. Hungar*, 18(3-4):321–328, 1967.

26 Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3), jul 2010. doi:10.1145/1806907.1806913.

27 Benjamin Rossman. Homomorphism preservation theorems. *Journal of the ACM (JACM)*, 55(3):1–53, 2008. doi:10.1145/1379759.1379763.

28 Claude Tardif, Cynthia Loten, and Benoit Larose. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007. doi:10.2168/LMCS-3(4:6)2007.

29 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.