Conjunctive Queries: Unique Characterizations and Exact Learnability

Balder ten Cate
Google, Mountain View, CA, USA

Victor Dalmau
Universitat Pompeu Fabra, Barcelona, Spain

Abstract

We answer the question of which conjunctive queries are uniquely characterized by polynomially many positive and negative examples, and how to construct such examples efficiently. As a consequence, we obtain a new efficient exact learning algorithm for a class of conjunctive queries. At the core of our contributions lie two new polynomial-time algorithms for constructing frontiers in the homomorphism lattice of finite structures. We also discuss implications for the unique characterizability and learnability of schema mappings and of description logic concepts.

2012 ACM Subject Classification Theory of computation → Machine learning theory; Theory of computation → Logic; Information systems → Query languages

Keywords and phrases Conjunctive Queries, Homomorphisms, Frontiers, Unique Characterizations, Exact Learnability, Schema Mappings, Description Logic

Digital Object Identifier 10.4230/LIPIcs.ICDT.2021.9

Related Version *Full Version*: `https://arxiv.org/pdf/2008.06824.pdf`

Funding *Victor Dalmau*: Victor Dalmau was supported by MICCIN grants TIN2016-76573-C2-1P and PID2019-109137GB-C22.

Acknowledgements This paper largely grew out of discussions at Dagstuhl Seminar 19361 ("Logic and Learning") in Sept. 2019. We thank Carsten Lutz and Phokion Kolaitis for helpful discussions.

1 Introduction

Conjunctive queries (CQs) are an extensively studied database query language and fragment of first-order logic. They correspond precisely to Datalog programs with a single non-recursive rule. In this paper, we study two problems related to CQs. The first problem is concerned with the existence and constructability of unique characterizations. *For which CQs q is it the case that q can be characterized (up to logical equivalence) by its behavior on a small set of data examples? And, when such a set of data examples exists, can it be constructed efficiently?* The second problem pertains to *exact learnability* of CQs in an interactive setting where the learner has access to a "membership oracle" that, given any database instance and a tuple of values, answers whether the tuple belongs to the answer of the goal CQ (that is, the hidden CQ that the learner is trying to learn). We can think of the membership oracle as a black-box, compiled version of the goal query, which the learner can execute on any number of examples. The task of the learner, then, is to reverse engineer the query based on the observed behavior.

Note that these two problems (unique characterizability and exact learnability) are closely related to each other: a learner can identify the goal query with certainty, only when the set of examples that it has seen so far constitutes a unique characterization of the goal query.

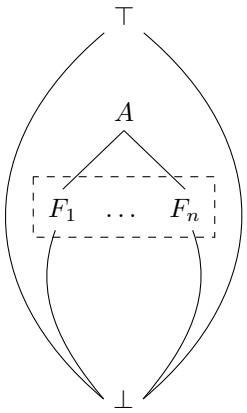

Figure 1 A frontier in the homomorphism lattice of structures.

▶ **Motivating Example 1.** *This example, although stylized and described at a high level, aims to convey an important use case that motivated the present work. The* Google Knowledge Graph *is a large database of entities and facts, gathered from a variety of sources. It is used to enhance the search engine's results for queries such as "where was Barack Obama born" with factual information in the form of* knowledge panels *[7] When a query triggers a specific knowledge panel, this may be the result of various different triggering and fulfillment mechanisms, which may involve a combination of structured queries to the knowledge graph, hard-coded business logic (in a Turing-complete language), and machine learned models. This makes it difficult to understand interactions between knowledge panels (e.g., whether the two knowledge panels are structurally equivalent or one subsumed by the other). If a declarative specification of (an approximation of) the triggering and fulfillment logic for a knowledge panel can be constructed programmatically, specified in a sufficiently restrictive formalism such as Datalog rules, this provides an avenue to the above, and other relevant static analysis tasks. The* efficient exact learnability with membership queries *that we study in this paper, can be viewed as an idealized form of such a programmatic approach, where the membership oracle is the existing, black box, implementation of the knowledge panel, and the learner must produce a CQ that exactly captures it.*

As it turns out, the above problems about CQs are intimately linked to fundamental properties of the homomorphism lattice of finite structures. In particular, the existence of a unique characterization for a CQ can be reduced to the existence of a *frontier* in the homomorphism lattice for an associated structure A, where, by a "frontier" for A, we mean a finite set of structures $F_1, \ldots, F_n$ that cover precisely the set of structures homomorphically strictly weaker than A, that is, such that $\{B \mid B \overset{\rightarrow}{\not\leftarrow} A\} = \bigcup_i \{B \mid B \to F_i\}$ (cf. Figure 1).

Known results [16, 1] imply that not every finite structure has such a frontier, and, moreover, a finite structure has a frontier if and only if the structure (modulo homomorphic equivalence) satisfies a structural property called *c-acyclicity*. These known results, however, are based on exponential constructions, and no polynomial algorithms for constructing frontiers were previously known.

▶ **Main Contribution 1** (Polynomial-time algorithms for constructing frontiers)**.** *We show that, for c-acyclic structures, a frontier can in fact be computed* in polynomial time*. More specifically, we present two polynomial-time algorithms. The first algorithm takes any c-acyclic structure and produces a frontier consisting of structures that are themselves not necessarily*

c-acyclic. Indeed, we show that this is unavoidable. The second algorithm applies to a more restricted class of acyclic structures and yields a frontier consisting entirely of structures belonging to the same class (that is, the class of structures in question is frontier-closed).

We use these to obtain new results on the existence and efficient constructability of unique characterizations for CQs:

▶ **Main Contribution 2** (Polynomial Unique Characterizations for Conjunctive Queries)**.** *We show that a CQ is uniquely characterizable by polynomially many examples, precisely if (modulo logical equivalence) it is c-acyclic. Furthermore, for c-acyclic CQs, a uniquely characterizing set of examples can be constructed in polynomial time. In the special case of unary, acyclic, connected CQs, a uniquely characterizing set of examples can be constructed consisting entirely of queries from the same class.*

Using the above results as a stepping stone, we obtain a polynomial-time exact learning algorithm for the class of c-acyclic CQs.

▶ **Main Contribution 3** (Polynomial-Time Learnability with Membership Queries)**.** *We show that c-acyclic CQs are efficiently exactly learnable in Angluin's model of exact learnability with membership queries [2].*

The above results are optimal, because, as we mentioned above, exact learnability with membership queries, requires the existence of a finite uniquely characterizing set of examples, which, in turn, requires the existence of a frontier, and frontiers exist (up to homomorphic equivalence) only for c-acyclic structures.

Although our primary interest is in conjunctive queries, we show that our results also have implications for *schema mappings* and *description logic concepts*:

▶ **Main Contribution 4** (Schema Mappings and Description Logic Concepts)**.** *As a further corollary to the above, we obtain a number of results regarding the existence of polynomial unique characterizations, as well as exact learnability, for LAV ("Local-As-View") schema mappings and for description logic concepts for the lightweight description logic $\mathcal{ELI}$.*

Outline

Section 2 reviews basic facts and definitions. In Section 3, we present our two new polynomial-time algorithms for constructing frontiers for finite structures with designated elements. We also review a result by [27], which implies the existence of (not necessarily polynomially computable) frontiers w.r.t. classes of structures of bounded expansion. In Section 4, we apply these algorithms to show that a CQ is uniquely characterizable by polynomially many examples, precisely if (modulo logical equivalence) it is c-acyclic. Furthermore, for c-acyclic CQs, a uniquely characterizing set of examples can be constructed in polynomial time. In the special case of unary, acyclic, connected CQs, a uniquely characterizing set of examples can be constructed consisting entirely of queries from the same class. In Section 5, we further build on these results, and we study the exact learnability of CQs. Section 6 presents applications to schema mappings and description logic concepts.

2 Preliminaries

Schemas, Structures, Homomorphisms, Cores. A *schema* (or, relational signature) is a finite set of relation symbols $\mathcal{S} = \{R_1, \ldots, R_n\}$, where each relation R_i as an associated arity $\text{arity}(R_i) \geq 1$. For $k \geq 0$, by a *structure over $\mathcal{S}$ with k distinguished elements* we

will mean a tuple $(A, a_1, \ldots, a_k)$, where $A = (\mathrm{dom}(A), R_1^A, \ldots, R_n^A)$ is a finite structure (in the traditional, model-theoretic sense) over the schema $\mathcal{S}$, and $a_1, \ldots, a_k$ are elements of the domain of A. Note that all structures, in this paper, are assumed to be finite, and we will drop the adjective "finite". By a *fact* of a structure A we mean an expression of the form $R(a_1, \ldots, a_n)$ where the tuple $(a_1, \ldots, a_n)$ belongs to the relation R in A. Given two structures $(A, \mathbf{a})$ and $(B, \mathbf{b})$ over the same schema, where $\mathbf{a} = a_1, \ldots a_k$ and $\mathbf{b} = b_1, \ldots, b_k$, a *homomorphism* $h : (A, \mathbf{a}) \to (B, \mathbf{b})$ is a map h from the domain of A to the domain of B, such that h preserves all facts, and such that $h(a_i) = b_i$ for $i = 1 \ldots k$. When such a homomorphism exists, we will also say that $(A, \mathbf{a})$ "homomorphically maps to" $(B, \mathbf{b})$ and we will write $(A, \mathbf{a}) \to (B, \mathbf{b})$. We say that $(A, \mathbf{a})$ and $(B, \mathbf{b})$ are *homomorphically equivalent* if $(A, \mathbf{a}) \to (B, \mathbf{b})$ and $(B, \mathbf{b}) \to (A, \mathbf{a})$.

A structure is said to be a *core* if there is no homomorphism from the structure in question to a proper substructure [20]. It is known [20] that every structure $(A, \mathbf{a})$ has a substructure to which it is homomorphically equivalent and that is a core. This substructure, moreover, is unique up to isomorphism, and it is known as *the core of* $(A, \boldsymbol{a})$.

Fact Graph, FG-Connectedness, FG-Disjoint Union. The *fact graph* of a structure $(A, \mathbf{a})$ is the undirected graph whose nodes are the facts of A, and such that there is an edge between two distinct facts if they share a non-designated element, i.e., there exists an element b of the domain of A that is distinct from the distinguished elements $\mathbf{a}$, such that b occurs in both facts. We say that $(A, \mathbf{a})$ is *fg-connected* if the fact graph is connected. A *fg-connected component* of $(A, \mathbf{a})$ is a maximal fg-connected substructure $(A', \mathbf{a})$ of $(A, \mathbf{a})$. If $(A_1, \mathbf{a})$ and $(A_2, \mathbf{a})$ are structures with the same distinguished elements, and whose domains are otherwise (except for these distinguished elements) disjoint, then the union $(A_1 \cup A_2, \mathbf{a})$ of these two structures will be called a *fg-disjoint union* and will be denoted as $(A_1, \mathbf{a}) \uplus (A_2, \mathbf{a})$. The same construction naturally extends to finite sets of structures. It is easy to see that every structure $(A, \mathbf{a})$ is equal to the fg-disjoint union of its fg-connected components. See also [15, 34], where fg-connected components are called *fact blocks*.

Direct Product, Homomorphism Lattice. Given two structures $(A, \mathbf{a})$ and $(B, \mathbf{b})$ over the same schema, where $\mathbf{a} = a_1, \ldots a_k$ and $\mathbf{b} = b_1, \ldots, b_k$, the *direct product* $(A, \mathbf{a}) \times (B, \mathbf{b})$ is defined, as usual, as $((A \times B), \langle a_1, b_1 \rangle, \ldots, \langle a_k, b_k \rangle)$, where the domain of $A \times B$ is the Cartesian product of the domains of A and B, and where the facts of $A \times B$ are all facts $R(\langle c_1, d_1 \rangle, \ldots, \langle c_n, d_n \rangle)$ for which it holds that $R(c_1, \ldots, c_n)$ is a fact of A and $R(d_1, \ldots, d_n)$ is a fact of B. The direct product of a finite collection of structures is defined analogously.

For a fixed schema $\mathcal{S}$ and $k \geq 0$, the collection of (homomorphic-equivalence classes of) structures over $\mathcal{S}$ with k distinguished elements, ordered by homomorphism, forms a lattice, where the above *direct product* operation is the meet operation. In particular, $(A, \mathbf{a}) \times (B, \mathbf{b})$ homomorphically maps to both $(A, \mathbf{a})$ and $(B, \mathbf{b})$, and a structure $(C, \mathbf{c})$ homomorphically maps to $(A, \mathbf{a}) \times (B, \mathbf{b})$ if and only if it homomorphically maps to both $(A, \mathbf{a})$ and $(B, \mathbf{b})$. The join operation of the lattice is a little more tedious to define, and we only sketch it here, as it is not used in the remainder of the paper. When two structures have the same isomorphism type of distinguished elements, their join is simply the fg-disjoint union as defined earlier. In the general case, one must first compute the smallest equivalence relation over the index set of the distinguished elements that refines the equivalence relations induced by both structures, and factor both structures through, before taking their fg-disjoint union.

For structures without designated elements, this lattice has been studied extensively (cf. for instance [21, 29]). The above exposition shows how to lift some of the basic definitions and constructions, in the appropriate way, to structures with distinguished elements.

Incidence Graph, C-Acyclicity. Given a structure $(A, \mathbf{a})$, the *incidence graph* of A is the bipartite multi-graph containing all elements of the domain of A as well as all facts of A, and an edge (a, f) whenever a is an element and f is a fact in which a occurs. Note that if an element occurs more than once in the same fact, the incidence graph contains multiple edges. $(A, \mathbf{a})$ is said to be *c-acyclic* if every cycle in its incidence graph contains at least one distinguished elements, i.e., at least one elements in $\mathbf{a}$. In particular, this means that no non-designated element occurs twice in the same fact. The concept of c-acyclicity was first introduced in [1] in the study of unique characterizability of GAV schema mappings (cf. Section 6 for more details). A straightforward dynamic-programming argument shows:

▶ **Proposition 2.1.** *For c-acyclic structures $(A, \boldsymbol{a})$ and $(B, \boldsymbol{b})$ (over the same schema and with the same number of distinguished elements), we can test in polynomial time whether $(A, \boldsymbol{a}) \to (B, \boldsymbol{b})$. The core of a c-acyclic structure can be computed in polynomial time.*

In the case without designated elements, c-acyclicity simply means that the incidence graph is acyclic, a condition better known as *Berge acyclicity* in the database theory literature [14].

C-Connectedness. We say that a structure $(A, \mathbf{a})$ is *c-connected* if every connected component of its incidence graph contains at least one designated element. Note that this condition is only meaningful for $k > 0$, and that it differs subtly from the condition of fg-connectedness we defined above. For example, the structure consisting of the facts $R(a_1, a_2)$ and $S(a_2, a_1)$ with distinguished elements a_1, a_2, is c-connected but is *not* fg-connected. For any structure $(A, \mathbf{a})$, we denote by $(A, \mathbf{a})^{\text{reach}}$ the (unique) maximal c-connected substructure, that is, the substructure containing everything reachable from the distinguished elements.

▶ **Proposition 2.2.** *If $(A, \boldsymbol{a})$ is c-connected, then $(A, \boldsymbol{a}) \to (B, \boldsymbol{b})^{reach}$ iff $(A, \boldsymbol{a}) \to (B, \boldsymbol{b})$.*

Conjunctive Queries. Let $k \geq 0$. A k-ary *conjunctive query (CQ)* q over a schema $\mathcal{S}$ is an expression of the form $q(\mathbf{x}) \text{ :- } \alpha_1 \wedge \cdots \wedge \alpha_n$ where $\mathbf{x} = x_1, \ldots, x_k$ is a sequence of variables, and where each α_i is an atomic formula using a relation from $\mathcal{S}$. Note that α_i may use variables from $\mathbf{x}$ as well as other variables. In addition, it is required that each variable in $\mathbf{x}$ occurs in at least one conjunct α_i. This requirement is referred to as the *safety* condition.

Note that, for simplicity, this definition of CQ does not allow the use of constants. Many of the results in this paper, however, can be extended in a straightforward way to CQs with a fixed finite number of constants (which can be simulated using additional free variables).

If A is a structure over the same schema as q, we denote by $q(A)$ the set of all k-tuples of values that satisfy the query q in A. We write $q \subseteq q'$ if q and q' are queries over the same schema, and of the same arity, and $q(A) \subseteq q'(A)$ holds for all structures A. We say that q and q' are *logically equivalent* if $q \subseteq q'$ and $q' \subseteq q$ both hold. We refer to any textbook on database theory for a more detailed exposition of the semantics of CQs.

There is a well-known correspondence between k-ary CQs over a schema $\mathcal{S}$ and structures over $\mathcal{S}$ with k distinguished elements. In one direction, we can associate to each k-ary CQ $q(\mathbf{x})$ over the schema $\mathcal{S}$ a corresponding structure over $\mathcal{S}$ with k distinguished elements, namely $\widehat{q} = (A_q, \mathbf{x})$, where the domain of A_q is the set of variables occurring in q, and the facts of A_q are the conjuncts of q. We will call this structure $\widehat{q}$ the *canonical structure* of q. Note that every distinguished element of $\widehat{q}$ occurs in at least one fact, as follows from the safety condition of CQs. Conversely, consider any structure $(A, \mathbf{a})$, with $\mathbf{a} = a_1, \ldots, a_k$, such that every distinguished element a_i occurs in at least one fact of A. We can associate to $(A, \mathbf{a})$ a k-ary *canonical CQ*, namely the CQ that has a variable x_a for every value a in the domain of A occurring in at least one fact, and a conjunct for every fact of A.

By the classic *Chandra-Merlin Theorem* [10], a tuple $\mathbf{a}$ belongs to $q(A)$ if and only if there is a homomorphism from $\widehat{q}$ to $(A, \mathbf{a})$; and $q \subseteq q'$ holds if and only if there is a homomorphism from $\widehat{q'}$ to $\widehat{q}$. Finally, q and q' are logically equivalent if and only if $\widehat{q}$ and $\widehat{q'}$ are homomorphically equivalent.

Exact Learning Models, Conjunctive Queries as a Concept Class. Informally, an *exact learning algorithm* is an algorithm that identifies an unknown goal concept by asking a number of queries about it. The queries are answered by an oracle that has access to the goal concept. This model of learning was introduced by Dana Angluin, cf. [2]. In this paper, we consider the two most extensively studied kinds of oracle queries: *membership queries* and *equivalence queries*. We will first review basic notions from computational learning theory, such as the notion of a *concept*, and then explain what it means for a concept class to be *efficiently exactly learnable with membership and/or equivalence queries*.

Let X be a (possibly infinite) set of *examples*. A *concept over* X is a function $c : X \to \{0, 1\}$, and a *concept class* $\mathcal{C}$ is a collection of such concepts. We say that $x \in X$ is a *positive example* for a concept c if $c(x) = 1$, and that x is a *negative example* for c if $c(x) = 0$.

Conjunctive queries (over a fixed schema $\mathcal{S}$ and with a fixed arity k) are a particular example of such a concept class, where the example space is the class of all structures over $\mathcal{S}$ with k distinct elements, and where an example $(A, \mathbf{a})$ is labeled as positive if the tuple $\mathbf{a}$ belongs to $q(A)$, and negative otherwise.

It is always assumed that concepts are specified using some representation system so that one can speak of the length of the specification of a concept. More formally, a *representation system for* $\mathcal{C}$ is a string language $\mathcal{L}$ over some finite alphabet, together with a surjective function $r : \mathcal{L} \to \mathcal{C}$. By the *size* of a concept $c \in \mathcal{C}$, we will mean the length of the smallest representation. Similarly, we assume a representation system, with a corresponding notion of length, for the examples in X. When there is no risk of confusion, we may conflate concepts (and examples) with their representations.

Specifically, for us, when it comes to *structures*, any natural choice of representation will do; we only assume that the length of the specification of a structure (for a fixed schema) is polynomial in the domain size, the number of facts and the number of distinguished elements. Likewise for *CQs*, we assume that the length of the representation of a CQ is polynomial in that of its canonical structure.

For every concept c, we denote by MEM_c the *membership oracle* for c, that is, the oracle that takes as input an example x and returns its label, $c(x)$, according to c. Similarly, for every concept $c \in \mathcal{C}$, we denote by EQ_c, the *equivalence oracle* for c, that is, the oracle that takes as input the representation of a concept h and returns "yes", if $h = c$, or returns a counterexample x otherwise (that is, an example x such that $h(x) \neq c(x)$). An *exact learning algorithm with membership and/or equivalence queries* for a concept class C is an algorithm `alg` that takes no input but has access to the membership oracle and/or equivalence oracle for an unknown *goal concept* $c \in C$.[1] The algorithm `alg` must terminate after finite amount of time and output (some representation of) the goal concept c. This notion was introduced by Angluin [2], who also introduced the notion of a *polynomial-time* exact learning algorithm. We say that an exact learning algorithm `alg` with membership and/or equivalence queries *runs in polynomial time* if there exists a two-variable polynomial $p(n, m)$ such that at any

[1] It is common in the learning theory literature to assume that the learning algorithm is given an upper bound on the size of the goal concept as input. However, it turns out that such an assumption is not needed for any of our positive results concerning learnability.

point during the run of the algorithm, the time used by `alg` up to that point (counting one step per oracle call) is bounded by $p(n, m)$, where n is the size of the goal concept and m the size of the largest counterexample returned by calls to the equivalence oracle up to that point in the run ($m = 0$ if no equivalence queries have been used). A concept class $\mathcal{C}$ is *efficiently exactly learnable with membership and/or equivalence queries* if there is an exact learning algorithm with membership and/or equivalence queries for $\mathcal{C}$ that runs in polynomial time.

There is a delicate issue about this notion of polynomial time that we now discuss. One might be tempted to relax the previous definition by requiring merely that the total running time is bounded by $p(n, m)$. However, this change in the definition would give rise to a *wrong* notion of a polynomial-time algorithms in this context by way of a loophole in the definition. Indeed, under this change, one could design a learning algorithm that, in a first stage, identifies the goal hypothesis by (expensive) exhaustive search and that, once this is achieved, forces – by asking equivalence queries with appropriate modification of the goal concept – the equivalence oracle to return large counterexamples that would make up for the time spent during the exhaustive search phase.

3 Frontiers in the homomorphism lattice of structures

In this section, we define frontiers and discuss their relationships to gaps and (restricted) homomorphism dualities. We present two polynomial-time methods for constructing frontiers. For the applications in the next sections, it is important to consider structures with designated elements. These designated elements, intuitively, correspond to the free variables of a CQ.

▶ **Definition 3.1.** *Fix a schema and $k \geq 0$, and let $\mathcal{C}$ be a class of structures with k designated elements and let $(A, \boldsymbol{a})$ be a structure with k designated elements as well. A* frontier *for $(A, \boldsymbol{a})$ w.r.t. $\mathcal{C}$, is a set of structures F such that*

1. *$(B, \boldsymbol{b}) \to (A, \boldsymbol{a})$ for all $(B, \boldsymbol{b}) \in F$.*
2. *$(A, \boldsymbol{a}) \not\to (B, \boldsymbol{b})$ for all $(B, \boldsymbol{b}) \in F$.*
3. *For all $(C, \boldsymbol{c}) \in \mathcal{C}$ with $(C, \boldsymbol{c}) \to (A, \boldsymbol{a})$ and $(A, \boldsymbol{a}) \not\to (C, \boldsymbol{c})$, we have that $(C, \boldsymbol{c}) \to (B, \boldsymbol{b})$ for some $(B, \boldsymbol{b}) \in F$.*

See Figure 1 for a graphical depiction of a frontier.

The notion of a frontier is closely related to that of a gap pair. A pair of structures (B, A) with $B \to A$ is said to be a *gap pair* if $A \not\to B$, and every structure C satisfying $B \to C$ and $C \to A$ is homomorphically equivalent to either B or A [30]. The same concept applies to structures with designated elements. It is easy to see that any frontier for a structure A must contain (modulo homomorphic equivalence) all structures B such that (B, A) is a gap pair.

▶ **Example 3.2.** The structure (A, a_1) consisting of facts Pa_1 and Qa_1 (with designated element a_1) has a frontier of size 2 (w.r.t. the class of all finite structures), namely $F = \{(B, a_1), (C, a_1)\}$ where B consists of the facts Pa_1, Pb, Qb and C consists of the facts Qa_1, Pb, Qb, respectively. Note that $((B, a_1), (A, a_1))$ and $((C, a_1), (A, a_1))$ are gap pairs. It can be shown that the structure (A, a_0) has no frontier of size 1 (as such a frontier would have to consist of a structure that contains both facts Pa_1 and Qa_1).

For another example, consider the structure (A', a_1) consisting of facts Pa_1 and Rbb. It is the right hand side of a gap pair (the left hand side being the structure (B', a_1) consisting of the facts Rbb and Pb'), but (A', a_1) has no finite frontier as follows from Theorem 3.7 below.

Frontiers are also closely related to (generalized) homomorphism dualities [16]. We say that a structure $(A, \mathbf{a})$ *has a finite duality* w.r.t. a class $\mathcal{C}$ if there is a finite set of structures

D such that for all $(C, \mathbf{c}) \in \mathcal{C}$, $(A, \mathbf{a}) \to (C, \mathbf{c})$ iff for all $(B, \mathbf{b}) \in D$, $(C, \mathbf{c}) \not\to (B, \mathbf{b})$. If $\mathcal{C}$ is the set of all structures (over the same schema as $(A, \mathbf{a})$), we simply say that $(A, \mathbf{a})$ *has a finite duality.*[2]

▶ **Example 3.3.** In the realm of digraphs, viewed as relational structures without designated elements with a single binary relation, every directed path A of, say, $k > 1$ nodes has finite duality (w.r.t. the class of digraphs). Indeed, it is not difficult to verify that for every digraph C, $A \to C$ iff $C \not\to D$ where D is the digraph with nodes $\{1, \ldots k-1\}$ and edges $\{(i, j) \mid i < j\}$.

▶ **Lemma 3.4.** *Let $\mathcal{C}$ be any class of structures.*

1. *If a structure $(A, \boldsymbol{a})$ has a finite duality w.r.t. $\mathcal{C}$ then $(A, \boldsymbol{a})$ has a finite frontier w.r.t. $\mathcal{C}$.*
2. *If a structure $(A, \boldsymbol{a}) \in \mathcal{C}$ has a finite frontier w.r.t. $\mathcal{C}$ and $\mathcal{C}$ is closed under direct products, then $(A, \boldsymbol{a})$ has a finite duality w.r.t. $\mathcal{C}$.*

Note that the construction of the frontier from the duality is polynomial, while the construction of the duality from the frontier involves an exponential blowup. The following example shows that this is unavoidable.

▶ **Example 3.5.** The path $\circ \xrightarrow{R} \circ \xrightarrow{R_1} \circ \xrightarrow{R} \circ \xrightarrow{R_2} \circ \cdots \circ \xrightarrow{R_n} \circ \xrightarrow{R} \circ$, viewed as a structure without any designated elements, has a frontier (w.r.t. the class of all finite structures) of size polynomial in n, as will follow from Theorem 3.8 below. It is known, however, that any finite duality for this structure must involve a structure whose size is exponential in n, and the example can be modified to use a fixed schema (cf. [31]).

3.1 Frontiers for classes with bounded expansion

The notion of a *class of graphs with bounded expansion* was introduced in [28]. We will not give a precise definition here, but important examples include graphs of bounded degree, graphs of bounded treewidth, planar graphs, and any class of graphs excluding a minor. The same concept of bounded expansion can be applied also to arbitrary structures: a class of structures $\mathcal{C}$ is said to have bounded expansion if the class of Gaifman graphs of structures in $\mathcal{C}$ has bounded expansion. We refer to [29] for more details. Classes of structures of bounded expansion are in many ways computationally well-behaved (cf. for example [23]).

Nešetřil and Ossona de Mendez [27, 29] show that if $\mathcal{C}$ is any class of structures with bounded expansion, then every structure has a finite duality w.r.t. $\mathcal{C}$. It follows by Lemma 3.4 that also every structure has a finite fontier w.r.t. $\mathcal{C}$. Nešetřil and Ossona de Mendez [27, 29] only consider connected structures without designated elements, but their result extends in a straightforward way to the general case of structures with designated elements. Furthermore, it yields an effective procedure for constructing frontiers, although non-elementary (i.e., not bounded by a fixed tower of exponentials).

▶ **Theorem 3.6** (from [27, 29])**.** *Let $\mathcal{C}$ be any class of structures that has bounded expansion. Then every structure $(A, \boldsymbol{a})$ has a finite frontier w.r.t. $\mathcal{C}$, which can be effectively constructed.*

[2] We note here that in the literature on Constraint Satisfaction, it is usual to consider the 'other side' of the duality, i.e, a structure A is said to have finite duality if there exists a finite set of structures F such that for every structure C, $C \to A$ iff for all $B \in F$, $B \not\to C$.

3.2 Polynomial frontiers for c-acyclic structures

Alexe et al. [1], building on Foniok et al. [16], show that a structure has a finite duality if and only if its core is c-acyclic. By Lemma 3.4, this implies that a structure has a finite frontier if and only if its core is c-acyclic.

▶ **Theorem 3.7** (from [16, 1]). *A structure has a finite frontier w.r.t. the class of all structures iff it is homomorphically equivalent to a c-acyclic structure, iff its core is c-acyclic.*

One of our main results is a new proof of the right-to-left direction, which, unlike the original, provides a polynomial-time construction of a frontier from a c-acyclic structure:

▶ **Theorem 3.8.** *Fix a schema $\mathcal{S}$ and $k \geq 0$. Given a c-acyclic structure over $\mathcal{S}$ with k distinguished elements, we can construct in polynomial time a frontier w.r.t. the class of all structures over $\mathcal{S}$ that have k distinguished elements.*

Note that the size of the smallest frontier is in general exponential in k. Indeed, consider the single-element structure $(A, \mathbf{a})$ where A consists of the single fact $P(a)$ and $\mathbf{a} = a, \ldots, a$ has length k. It is not hard to show that every frontier of this (c-acyclic) structure must contain, up to homomorphic equivalence, all structures of the form $(B, \mathbf{b})$ where B consists of two facts, $P(a_1)$ and $P(a_2)$, and $\mathbf{b} \in \{a_1, a_2\}^k$ is a sequence in which both a_1 and a_2 occur. There are exponentially many pairwise homomorphically incomparable such structures.

The proof of Theorem 3.8 is based on a construction that improves over a similar but exponential construction of gap pairs for acyclic structures given in [30, Def. 3.9]. Our results also shed new light on a question posed in the same paper: after presenting a double-exponential construction of duals (for connected structures without designated elements), involving first constructing an exponential-sized gap pair, the authors ask: "*It would be interesting to know to what extent the characterisation of duals can be simplified, and whether the indirect approach via density is optimal.*" This question appeared to have been answered in [31], where a direct method was established for constructing single-exponential size duals. Theorem 3.8 together with Lemma 3.4, however, gives another answer: single-exponential duals can be constructed by going through frontiers (i.e., "via density") as well.

Recall the definition of fg-connectedness from the preliminaries. We first prove a restricted version of Theorem 3.8 for special case of core, fg-connected, c-acyclic structures with the Unique Names Property. We subsequently lift these extra assumptions. A structure $(A, \mathbf{a})$ with $\mathbf{a} = a_1, \ldots a_k$ has the *Unique Names Property (UNP)* if $a_i \neq a_j$ for all $i \leq j$ (cf. [4]).

▶ **Proposition 3.9.** *Given a core, fg-connected, c-acyclic structure with UNP, we can construct in polynomial time (for fixed schema $\mathcal{S}$ and number of distinguished elements k) a frontier w.r.t. the class of all finite structures. Furthermore, the frontier consists of a single structure, which has the UNP.*

Proof. Let a core fg-connected c-acyclic structure $(A, \mathbf{a})$ with UNP be given. Note that each fg-connected structure either (i) consists of a single fact containing only designated elements, or (ii) consists of a number of facts that all contain at least one non-designated element. Therefore, we can distinguish two cases:

Case 1. A consists of a single fact f without non-designated elements. Let $(B, \mathbf{a})$ be the structure whose domain is $\{\mathbf{a}, b\}$, where b is a fresh value distinct from the values in $\mathbf{a}$, and which contains all facts over this domain except f. It is easy to see that $(B, \mathbf{a})$ is a homomorphism dual for $(A, \mathbf{a})$, and consequently, the direct product of the two structures constitutes a singleton frontier for $(A, \mathbf{a})$. Note that this construction is polynomial because we assume that the schema $\mathcal{S}$ and k are both fixed.

Case 2. A consists of one or more facts that each contain a non-designated element. In this case, we construct a singleton frontier $F = \{B\}$ where

- the domain of B consists of
  1. all pairs (a, f) where a is a non-designated element of A and f is a fact of A in which a occurs, and
  2. All pairs (a, id) where a is a designated element of A
- a fact $R((a_1, f_1), \ldots, (a_n, f_n))$ holds in B if and only if $R(a_1, \ldots, a_n)$ holds in A and at least one f_i is a fact that is different from the fact $R(a_1, \ldots, a_n)$ itself.

Note that, in the above construction, id is an arbitrary symbol, used only to simplify notation by ensuring that every element of B can we written as a pair. In what follows we will not distinguish between a designated element a_i and the corresponding pair (a_i, id).

We claim that $F = \{(B, \mathbf{a})\}$ is a frontier for $(A, \mathbf{a})$.

It is clear that the natural projection $h : B \to A$ is a homomorphism.

We claim that there is no homomorphism $h' : A \to B$. Assume, for the sake of a contradiction, that there was such a homomorphism. We may assume that the composition of h and h' is the identity on A (since A is a core, the composition of h and h' is an automorphism of A, that is, an isomorphism from A to itself. By composing h with the inverse of this automorphism if needed, we ensure that its composition with h' is the identity function). In particular, this means that h' maps each designated element a to (a, id) and for each non-designated element a of A, $h'(a) = (a, f)$ for some fact f. For a non-designated element a, let us denote by f_a the unique fact f for which $h'(a) = (a, f_a)$.

We will consider "walks" in A of the form

$$a_1 \xrightarrow{f_{a_1}} a_2 \xrightarrow{f_{a_2}} \ldots a_n$$

with $n \geq 1$, where

1. $a_1, \ldots, a_n$ are non-designated elements,
2. $f_{a_i} \neq f_{a_{i+1}}$, and
3. a_i and a_{i+1} co-occur in fact f_{a_i},

Since A is c-acyclic, the length of any such sequence is bounded by the diameter of A (otherwise some fact would have to be traversed twice in succession, which would violate condition 2). Furthermore, trivially, such a walk of length $n = 1$ exists: just choose as a_1 an arbitrary non-designated element of A. Furthermore, we claim that any such finite sequence can be extended to a longer one: let the fact f_{a_n} be of the form $R(b_1, \ldots, b_m)$ (where $a_n = b_i$ for some $i \leq m$). Since h is a homomorphism, it must map f_{a_n} to some fact $R((b_1, f_{b_1}), \ldots, (b_m, f_{b_m}))$ of B, where some f_{b_j} is a fact that is different from f_{a_n}. We can choose b_j as our element a_{n+1}. Thus, we reach our desired contradiction.

Finally, consider any C with $h : C \to A$ and $A \not\to C$. We construct a function $h' : C \to B$ as follows: consider any element c of C, and let $h(c) = a$. If a is a designated element, we set $h'(c) = (a, \mathsf{id})$. Otherwise, we proceed as follows: since A is c-acyclic and fg-connected, for each non-distinguished element a' of A (other than a itself) there is a unique minimal path in the incidence graph, containing only non-distinguished elements, from a' to a. We can represent this path by a sequence of the form

$$a' = a_0 \xrightarrow{(f_0, i_0, j_0)} a_1 \xrightarrow{(f_1, i_1, j_1)} a_2 \cdots \xrightarrow{(f_{n-1}, i_{n-1}, j_{n-1})} a_n = a$$

where each f_ℓ is a fact of A in which a_ℓ occurs in the i_ℓ-th position and $a_{\ell+1}$ occurs in the j_ℓ-th position. We can partition the non-distinguished elements a' of A (other than a itself) according to the last fact on this path, that is, f_{n-1}. Furthermore, it follows from

fg-connectedness that each fact of A contains a non-distinguished element. It is easy to see that if a fact contains multiple non-distinguished elements (other than a) then they must all belong to the same part of the partition as defined above. Therefore, the above partition on non-distinguished elements naturally extends to a partition on the facts of A. Note that if A contains any facts in which a is the only non-distinguished element, we will refer to these facts as "local facts" and they will be handled separately. In this way, we have essentially decomposed A into a union $A_{\text{local}} \cup \bigcup_i A_i$, where A_{local} contains all local facts and each "component" A_i is a substructure of A consisting of non-local facts, in such a way that (i) different substructures A_i do not share any facts with each other, (ii) different substructures do not share any elements with each other, except for a and distinguished elements, (iii) each A_i contains precisely one fact involving a.

Since we know that $(A, a) \not\to (C, c)$, it follows that either some local fact f of A does not map to C (when sending a to c), or some "component" A_i of A does not map to C through any homomorphism sending a to c. In the first case, we set $h'(a) = (a, f)$. In the second case, we choose such a component (if there are multiple, we choose one of minimal size) and let f be the unique fact in that component containing a (that is, f is the fact f_{n-1} that by construction connected the non-distinguished elements of the component in question to a). We set $h'(c) = (a, f)$. Intuitively, when $h'(c) = (h(c), f)$, then f is a fact of A involving $h(c)$ that "points in a direction where homomorphism from $(A, h(c))$ back to (C, c) fails".

We claim that h' is a homomorphism from C to B: let $R(c_1, \ldots, c_n)$ be a fact of C. Then $R(h(c_1), \ldots, h(c_n))$ holds in A. Let $h'(c_i) = (h(c_i), f_i)$ (where $f_i = \mathsf{id}$ if $h(c_i)$ is a designated element of A). To show that $R(h'(c_1), \ldots, h'(c_n))$ holds in B, it suffices to show that some f_i is different from the fact $R(h(c_1), \ldots, h(c_n))$ itself. If f_i is a local fact, then this follows immediately from the construction. Otherwise, let n_i be the size of the smallest "component" (as defined above) of $(A, h(c_i))$ that does not homomorphically map to (C, c_i), and choose an element c_i with minimal n_i. Then, clearly, f_i must be different from the fact $R(h(c_1), \ldots, h(c_n))$ itself. ◀

Next, we remove the assumptions of fg-connectedness and being a core.

▶ **Proposition 3.10.** *Given a c-acyclic structure with UNP, we can construct in polynomial time a frontier w.r.t. the class of all finite structures. Furthermore, the frontier consists of structures that have the UNP.*

Proof. By Proposition 2.1, we may assume that $(A, \mathbf{a})$ is a core. Note that the c-acyclicity and UNP properties are preserved under the passage from a structure to its core.

Let $(A, \mathbf{a})$ be a structure with designated elements that is UNP and that is a fg-disjoint union of homomorphically incomparable fg-connected structures $(A_1, \mathbf{a}), \ldots, (A_n, \mathbf{a})$. By Proposition 3.9, $(A_1, \mathbf{a}), \ldots, (A_n, \mathbf{a})$ have, respectively, frontiers $F_1, \ldots, F_n$, each consisting of a single structure with the UNP. We may assume without loss of generality that each F_i consists of structures that have the same designated elements $\mathbf{a}$ as A (we know that the structures in question satisfy the UNP, and therefore, modulo homomorphism, we can assume that their designated elements are precisely $\mathbf{a}$).

We claim that $F = \{\big(\biguplus_{j \neq i}(A_j, \mathbf{a})\big) \uplus (B, \mathbf{a}) \mid 1 \leq i \leq n, (B, \mathbf{a}) \in F_i\}$ is a frontier for $(A, \mathbf{a})$ w.r.t. $\mathcal{C}$.

Clearly, each structure in F maps homomorphically to A.

Suppose, for the sake of contradiction, that there were a homomorphism from $h : (A, \mathbf{a}) \to \big(\biguplus_{j \neq i}(A_j, \mathbf{a})\big) \uplus (B, \mathbf{a})$ for some $\mid 1 \leq i \leq n$ and $(B, \mathbf{a}) \in F_i$. Observe that h must send each designated element to itself, and it must send each non-designated element to a non-designated element (otherwise, the composition of h with the backward homomorphism

would be a non-injective endomorphism on $(A, \mathbf{a})$ which would contradict the fact that $(A, \mathbf{a})$ is a core). Since $(A_i, \mathbf{a})$ is fg-connected (and because h cannot send non-designated elements to designated elements), its h-image must be contained either in some $(A_j, \mathbf{a})$ $(j \neq i)$ or in B. The former would cannot happen because A_i and A_j are homomorphically incomparable. The latter cannot happen either, because B belongs to a frontier of A_i.

Finally, let $C \in \mathcal{C}$ be any structure such that there is a homomorphism $h : C \to A$ but $A \not\to C$. Let A_i be a fg-connected component of A such that $A_i \not\to C$. Since A_i is fg-connected, we can partition our structure C as $C_1 \uplus C_2$ where the h-image of C_1 is contained in A_i while the h-image of C_2 is disjoint from A_i. We know that $A_i \not\to C_1$ and therefore $C_1 \to B$ for some $B \in F_i$. Furthermore, we have that $C_2 \to \biguplus_{j \neq i} A_j$. Therefore, $C \to \left(\biguplus_{j \neq i} A_j\right) \uplus B \mid 1 \leq i \leq n, B \mid B \in F_i\}$. ◀

Finally, we can prove Theorem 3.8 itself.

Proof of Theorem 3.8. Let $(A, \mathbf{a})$ be c-acyclic. If it has the UNP, we are done. Consider the other case, where the sequence $\mathbf{a}$ contains repetitions. Let $\mathbf{a}' = a'_1, \ldots, a'_n$ consists of the same elements without repetition (in some order). We construct a frontier for it as follows:

1. Consider the structure $(A, \mathbf{a}')$, which, by construction, has the UNP. Let F be a frontier for $(A, \mathbf{a}')$ (again consisting of structures with the UNP), using Proposition 3.10. Note that, through isomorphism, we may assume that each structure in F has the same designated elements $\mathbf{a}'$. For each $(B, \mathbf{a}') \in F$, we take the structure $(B, \mathbf{a})$.
2. Let k be the length of the tuple $\mathbf{a}$. For each function $f : \{1, \ldots, k\} \to \{1, \ldots, k\}$, whose range has size strictly greater than n, consider structure $(C, \mathbf{c}^f)$ where C contains all facts over the domain $\{1, \ldots, k\}$, and $c^f_i = f(i)$. We take its direct product with $(A, \mathbf{a})$.

It is easy to see that the set of all structures constructed above, constitutes a frontier for $(A, \mathbf{a})$. Indeed, suppose a structure maps to $(A, \mathbf{a})$ but not vice versa. If the tuple of designated elements of the structure in question has the same identity type as the tuple $\mathbf{a}$ (i.e., the same equalities hold between values at different indices in the tuple) then it is easy to see that the structure in question must map to some structure $(B, \mathbf{a})$ as constructed under item 1 above. Otherwise, if the tuple of designated elements of the structure in question does *not* have the same identity type, then it is easy to see that the structure in question must map to $(C^f, \mathbf{c}^f) \times (A, \mathbf{a})$, as constructed under item 2 above, where f reflects the identity type of the designated elements of the structure in question. ◀

As a corollary of Theorem 3.8, we obtain the following interesting by-product:

▶ **Theorem 3.11.** *For a fixed schema $\mathcal{S}$ and $k \geq 0$, the following problem is solvable in NP: given a finite set of structures F and a structure A (all with k designated elements), is F a frontier for A w.r.t. the class of all structures? For some $\mathcal{S}$ and k, it is NP-complete.*

Proof. For the upper bound, we use the fact that, if A is homomorphically equivalent to a c-acyclic structure A', then the core of A is c-acyclic (cf. Theorem 3.7). The problem can therefore be solved in non-deterministic polynomial time as follows:

First we guess a substructure A' and we verify that A' is c-acyclic and homomorphically equivalent to A. Note that the existence of such A' is a necessary precondition for F to be a frontier of A. Furthermore, c-acyclicity can be checked in polynomial time using any PTIME algorithm for graph acyclicity (recall that a structure is c-acyclic if and only if its incidence graph is acyclic after removing all nodes corresponding to designated elements).

Next, we apply Theorem 3.8 to construct a frontier F' for A' (and hence for A). Finally, we verify that each $B \in F$ homomorphically maps to some $B' \in F'$ and, vice versa, every $B' \in F'$

homomorphically maps to some $B \in F$. It is not hard to see that this non-deterministic algorithm has an accepting run if and only if F is a frontier for A.

For the lower bound, we reduce from graph 3-colorability. Let A be the structure, over a 3-element domain, that consists of the facts $R(a,b)$ for all pairs a,b with $a \neq b$. In addition, each of the three elements is named by a constant symbol. Since A is c-acyclic, by Theorem 3.7, it has a frontier F. Now, given any graph G (viewed as a relational structure with binary relation R and without constant symbols), we have that G is 3-colorable if and only if F is a frontier for the disjoint union of A with G. To see that this is the case, note that if G is 3-colorable, then the disjoint union of A with G is homomorphically equivalent to A itself, whereas if G is not 3-colorable, then the disjoint union of A with G is strictly greater than A in the homomorphism order. ◀

3.3 A polynomially frontier-closed class of structures

We call a class $\mathcal{C}$ of structures *frontier-closed* if every structure $(A, \mathbf{a}) \in \mathcal{C}$ has a frontier w.r.t. $\mathcal{C}$, consisting of structures belonging to $\mathcal{C}$. If, moreover, the frontier in question can be constructed from $(A, \mathbf{a})$ in polynomial time, then we say that $\mathcal{C}$ is *polynomially frontier-closed*.

▶ **Theorem 3.12.** *The class of c-connected acyclic structures with 1 designated element is polynomially frontier-closed.* [3]

As will follow from results in Section 4 (cf. Theorems 4.7-4.9 below) the theorem fails if we drop any of the three restrictions in the statement (i.e., c-connectedness, acyclicity, and $k = 1$). However, it might quite well be the case that still holds for some other values of k. Indeed, we conjecture that Theorem 3.12 remains true for any $k > 1$.

4 Unique Characterizations for Conjunctive Queries

In this section, we study the question when a CQ is uniquely characterizable by a finite set of positive and/or negative examples.

▶ **Definition 4.1** (Data Examples, Fitting, Unique Characterizations). *Let $\mathcal{C}$ be a class of k-ary CQs over a schema $\mathcal{S}$ (for some $k \geq 0$), and let q be a k-ary query over $\mathcal{S}$.*

1. *A* data example *is a structure $(A, \boldsymbol{a})$ over schema $\mathcal{S}$ with k distinguished elements. If $\boldsymbol{a} \in q(A)$, we call $(A, \boldsymbol{a})$ a* positive example *(for q), otherwise a* negative example.
2. *Let E^+, E^- be finite sets of data examples. We say that q* fits *(E^+, E^-) if every example in E^+ is a positive example for q and every example in E^- is a negative example for q. We say that (E^+, E^-)* uniquely characterizes *q w.r.t. $\mathcal{C}$ if q fits (E^+, E^-) and every $q' \in \mathcal{C}$ that fits (E^+, E^-) is logically equivalent to q.*

It turns out that there is a precise correspondence between unique characterizations and frontiers. Recall that the canonical structure of a query q is denoted by $\widehat{q}$. Similarly, for any class of CQs $\mathcal{C}$, we will denote by $\widehat{\mathcal{C}}$ the class of structures $\{\widehat{q} \mid q \in \mathcal{C}\}$.

▶ **Proposition 4.2** (Frontiers vs Unique Characterizations). *Fix a schema $\mathcal{S}$ and $k \geq 0$. Let q be any k-ary CQ over $\mathcal{S}$ and $\mathcal{C}$ a class of k-ary CQs over $\mathcal{S}$.*

1. *If F is a frontier for $\widehat{q}$ w.r.t. $\widehat{\mathcal{C}}$, then $(E^+ = \{\widehat{q}\}, E^- = F)$ uniquely characterizes q w.r.t. $\mathcal{C}$.*

[3] Note that for structures with one distinguished element, c-connectedness is the same as connectedness.

2. *Conversely, if* (E^+, E^-) *uniquely characterizes* Q *w.r.t.* $\mathcal{C}$*, then* $F = \{\widehat{q} \times (B, \boldsymbol{b}) \mid (B, \boldsymbol{b}) \in E^-\}$ *is a frontier for* $\widehat{q}$ *w.r.t.* $\widehat{\mathcal{C}}$*.*

Proposition 4.2 allows us to take the results on frontiers from the previous section, and rephrase them in terms of unique characterizations. Incidentally, note that results in [1] imply an analogous relationship between *finite dualities* and uniquely characterizing sets of examples for *unions of conjunctive queries*. We need two more lemmas. Recall that a structure $(A, \mathbf{a})$ corresponds to a conjunctive query only if every distinguished element occurs in at least one fact. Let us call such structures *safe*. The following lemmas, essentially, allow us to ignore unsafe structures, thereby bridging the gap between structures and CQs.

▶ **Lemma 4.3.** *Let* q *be a* k*-ary CQ over schema* $\mathcal{S}$ *and* $\mathcal{C}$ *a class of* k*-ary CQs over* $\mathcal{S}$*. If* q *is uniquely characterized w.r.t.* $\mathcal{C}$ *by positive and negative examples* (E^+, E^-)*, then* q *is uniquely characterized w.r.t.* $\mathcal{C}$ *by* $(\{(A, \boldsymbol{a}) \in E^+ \mid (A, \boldsymbol{a})$ *is safe*$\}, \{(A, \boldsymbol{a}) \in E^- \mid (A, \boldsymbol{a})$ *is safe*$\})$*.*

▶ **Lemma 4.4.** *A safe structure has a finite frontier w.r.t. all structures if and only if it has a finite frontier w.r.t. the class of all safe structures.*

Putting everything together, we obtain the main result of this section. We call a CQ q c-acyclic (or acyclic, or c-connected) if the structure $\widehat{q}$ is c-acyclic (resp. acyclic, c-connected).

▶ **Theorem 4.5.** *Fix a schema and fix* $k \geq 0$*.*

1. *If* $\mathcal{C}$ *is a class of* k*-ary CQs such that* $\widehat{\mathcal{C}}$ *has bounded expansion, then every CQ* $q \in \mathcal{C}$ *is uniquely characterizable w.r.t.* $\mathcal{C}$ *by finitely many positive and negative examples (which can be effectively constructed from the query).*
2. *A* k*-ary CQ* q *is uniquely characterizable by finitely many positive and negative examples (w.r.t. the class of all* k*-ary CQs) iff* q *is logically equivalent to a c-acyclic CQ. Moreover, for a c-acyclic CQ, a uniquely characterizing set of examples can be constructed in polynomial time.*
3. *Assume* $k = 1$ *and let* $\mathcal{C}_{ca}$ *be the class of* k*-ary CQs that are c-connected and acyclic. Then every* $q \in \mathcal{C}_{ca}$ *is uniquely characterizable w.r.t.* $\mathcal{C}_{ca}$ *by finitely many positive and negative examples belonging to* $\widehat{\mathcal{C}_{ca}}$*. Moreover, the set of examples in question can be constructed in polynomial time.*

▶ Remark 4.6. For the purpose of applications discussed in Section 6, we note that Theorem 4.5 remains true if the safety condition for CQs were to be dropped. Indeed, the proof in this case is even simpler, as it does not require Lemma 4.3 and Lemma 4.4.

Theorem 4.5(3) applies to CQs with $k = 1$ that are c-connected, and acyclic. None of these restrictions can be dropped.

▶ **Theorem 4.7.** *The Boolean acyclic connected CQ* $\mathbf{T}() \coloneq Ry_1y_2 \wedge Ry_2y_3 \wedge Ry_3y_4 \wedge Ry_4y_5$ *is not characterized, w.r.t. the class of Boolean acyclic connected CQs, by finitely many acyclic positive and negative examples.*

This shows that in Theorem 4.5(3), the restriction to *unary* queries cannot be dropped. Similarly, the restriction to *c-connected* queries cannot be dropped, and acyclicity cannot be replaced by the weaker condition of c-acyclicity.

▶ **Theorem 4.8.** *The unary acyclic CQ* $\mathbf{T}'(x) \coloneq P(x) \wedge Ry_1y_2 \wedge Ry_2y_3 \wedge Ry_3y_4 \wedge Ry_4y_5$ *is not uniquely characterizable, w.r.t. the class of unary acyclic CQs, by finitely many acyclic positive and negative examples.*

▶ **Theorem 4.9.** *The unary c-acyclic c-connected CQ* $\mathbf{T}''(x) \coloneq Ry_1y_2 \wedge Ry_2y_3 \wedge Ry_3y_4 \wedge Ry_4y_5 \wedge \bigwedge_{i=1\ldots5} Rxy_i$ *is not uniquely characterizable, w.r.t. the class of unary c-acyclic c-connected CQs, by finitely many c-acyclic positive and negative examples.*

5 Exact learnability with membership queries

The unique characterization results in the previous section immediately imply (not-necessarily-efficient) exact learnability results:

▶ **Theorem 5.1.** *Fix a schema and $k \geq 0$. Let $\mathcal{C}$ be a computably enumerable class of k-ary CQs. If $\widehat{\mathcal{C}}$ has bounded expansion, then $\mathcal{C}$ is exactly learnable with membership queries.*

The learning algorithm in question simply enumerates all queries $q \in \mathcal{C}$ and uses membership queries to test if the goal query fits the uniquely characterizing set of examples of q (cf. Theorem 4.5(1)). Unfortunately, this learning algorithm does not run in polynomial time. Indeed, the number of membership queries is not bounded by any fixed tower of exponentials. For the special case of c-acyclic queries, we can do a little better by taking advantage of the fact that a uniquely characterizing set of examples can be constructed in polynomial time. Indeed, the class of c-acyclic k-ary CQs is exponential-time exactly learnable with membership queries: the learner can simply enumerates all c-acyclic queries in order of increasing size. For each query q (starting with the smallest query), it uses Theorem 4.5(2) to test, using polynomially many membership queries, whether the goal query is equivalent to q. After at most $2^{O(n)}$ many attempts (where n is the size of the goal query), the algorithm is guaranteed to find a query that is equivalent to the goal query.[4] Our main result in this section improves on this by establishing *efficient* (i.e., polynomial-time) exact learnability:

▶ **Theorem 5.2.** *For each schema and $k \geq 0$, the class of c-acyclic k-ary CQs is efficiently exactly learnable with membership queries.*

At a high level, the learning algorithm works by maintaining a c-acyclic hypothesis that is an over-approximation of the actual goal query. At each iteration, the hypothesis is strengthened by replacing it with one of the elements of its frontier, a process that is shown to terminate and yield a query that is logically equivalent to the goal query. Note, however, that the frontier of a c-acyclic structure does not, in general, consist of c-acyclic structures. At the heart of the proof of Theorem 5.2 lies a non-trivial argument showing how to turn an arbitrary hypothesis into a c-acyclic one with polynomially many membership queries.

The class of *all* k-ary queries is not exactly learnable with membership queries (even with unbounded amount of time and the ability to ask an unbounded number of oracle queries), because exact learnability with membership queries would imply that every query in the class is uniquely characterizable, which we know is not the case. On the other hand, we have:

▶ **Theorem 5.3** (from [36])**.** *For each schema $\mathcal{S}$ and $k \geq 0$, the class of all k-ary CQs over $\mathcal{S}$ is efficiently exactly learnable with membership and equivalence queries.*

In fact, it follows from results in [36] that the larger class of all *unions of conjunctive queries* is efficiently exactly learnable with membership and equivalence queries (for fixed k and fixed schema). Efficient exact learnability with membership and equivalence queries is not a monotone property of concept classes, but the result from [36] transfers to CQs as well.

▶ Remark 5.4. For the purpose of applications discussed in Section 6, we note that Theorems 5.1- 5.3 remain true if the safety condition for CQs were to be dropped.

[4] Similarly, by Theorem 4.5(3), the class of unary acyclic c-connected queries is exponential-time exactly learnable with subset queries, where a *subset query* is an oracle query asking whether a given CQ from the concept class is implied by the goal query. Subset queries correspond precisely to membership queries where the example is the canonical structure of a query from the concept class.

Related Work. There has been considerable prior work that formally studies the task of identifying some unknown goal query Q from examples. Work in this direction includes learning CQs, Xpath queries, Sparql, tree patterns, description logic concepts, ontologies, and schema mappings among others [8, 36, 18, 33]. We shall describe mostly the previous work regarding learning CQs. Some of the work in this direction ([6, 35, 12, 22, 39] for example) assumes that a background structure A is fixed and known by the algorithm. In this setting, a *example* is a k-ary tuple $(a_1, \ldots, a_k)$ of elements in A, labelled positively or negatively depending on whether it belongs or not to $Q(A)$. In the present paper (as in [36, 19]) we do not fix any background structure (i,e, examples are pairs of the form $(A, \mathbf{a})$). Our setting corresponds also to the extended instances with empty background in [13].

In both cases a number of different learning protocols has been considered. In the reverse-engineering problem (as defined in [38]) it is only required that the algorithm produces a query consistent with the examples. In a similar direction, the problem of determining whether such a query exists has been intensively studied under some variants (satisfiability, query-by-example, definability, inverse satisfiability) [6, 35, 39]. In some scenarios, it is desirable that the query produced by the learner not only explains the examples received during the training phase, but also has also predictive power. In particular, the model considered in [9] follows the paradigm of identification in the limit by Gold and requires that, additionally, there exists a finite set of examples that uniquely determines the target query Q. In a different direction, the model introduced in [18], inspired by the minimum description length principle, requires to produce a hypothesis consistent after some repairs. A third line of work (see [11, 19, 22]) studies this problem under Valiant's probably approximately correct (PAC) model. The present paper is part of a fourth direction based on the exact model of query identification by Angluin. In this model, instead of receiving labelled examples, the learner obtains information about the target query by mean of calls to an oracle. As far as we know, we are the first to study the exact learnability of CQs using a membership oracle.

6 Further Applications

While our main focus in this paper is on unique characterizability and exact learnability for CQs, in this section, we explore some implications for other application domains.

6.1 Characterizability and learnability of LAV schema mappings

A schema mapping is a high-level declarative specifications of the relationships between two database schemas [24]. Two of the most well-studied schema mapping specification languages are *LAV ("Local-as-View")* and *GAV ("Global-as-View")* schema mappings.

In [1], the authors studied the question when a schema mapping can be uniquely characterized by a finite set of data examples. Different types of data examples were introduced and studied, namely positive examples, negative examples, and "universal" examples. In particular, it was shown in [1] that a GAV schema mapping can be uniquely characterized by a finite set of positive and negative examples (or, equivalently, by a finite set of universal examples) if and only if the schema mapping in question is logically equivalent to one that is specified using c-acyclic GAV constraints.

It was shown in [1] that every LAV schema mapping is uniquely characterized by a finite set of universal examples, and that there are LAV schema mappings that are not uniquely characterized by any finite set of positive and negative examples. In this section, we will consider the question *which* LAV schema mappings are uniquely characterizable by a finite set of positive and negative examples, and how to construct such a set of examples efficiently.

We will also consider the exact learnability of LAV schema mappings with membership queries. Exact learnability of GAV schema mappings was studied in [36], where it was shown that GAV schema mappings are learnable with membership and equivalence queries (and, subsequently, also in a variant of the PAC model) but is not exactly learnable with membership queries alone or with equivalence queries alone. The exact learning algorithm for GAV schema mappings from [36] was further put to use and validated experimentally in [37]. Here, we consider exact learnability of LAV schema mappings with membership queries.

▶ **Definition 6.1.** *A* LAV ("Local-As-View") schema mapping *is a triple* $M = (S, T, \Sigma)$ *where* S *and* T *are disjoint schemas (the "source schema" and "target schema"), and* Σ *is a finite set of* LAV constraints*, that is, first-order sentences of the form* $\forall \boldsymbol{x}(\alpha(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\phi(\boldsymbol{x}, \boldsymbol{y}))$*, where* $\alpha(\boldsymbol{x})$ *is an atomic formula using a relation from* S*, and* $\phi(\boldsymbol{x}, \boldsymbol{y})$ *is a conjunction of atomic formulas using relations from* T*.*

By a *schema-mapping example* we will mean a pair (I, J) where I is a structure over schema $\mathcal{S}$ without distinguished elements, and J is a structure over schema $\mathcal{T}$ without distinguished elements. We say that (I, J) is a *positive example* for a schema mapping $M = (\mathcal{S}, \mathcal{T}, \Sigma)$ if (I, J), viewed as a single structure over the joint schema $\mathcal{S} \cup \mathcal{T}$, satisfies all constraints in Σ, and we call (I, J) a *negative example* for M otherwise. Note that schema-mapping examples were called *data examples* in [1]. Unique characterizations and learnability with membership queries are defined as before. In particular, by a *membership query*, in the context of learning LAV schema mappings, we will mean an oracle query that consists of a schema-mapping example, which the oracle then labels as positive or negative depending on whether it satisfies the constraints of the goal LAV schema mapping. It is assumed here, that the source and target schemas are fixed and known to the learner.

Given a fixed source schema $\mathcal{S}$, there are only finitely many different possible left-hand sides α for a LAV constraint, up to renaming of variables. Furthermore, if a schema mapping contains two LAV constraints with the same left-hand side, then they can be combined into a single LAV constraint by conjoining the respective right-hand sides. Since the right-hand side of a LAV constraint can be thought of as a CQ, this means that, intuitively, a LAV schema mapping can be thought of as a finite collection of CQs (one for each possible left-hand side). In the light of this observation, it is no surprise that questions about the unique characterizability and learnability of LAV schema mappings can be reduced to questions about the unique characterizability and learnability of CQs.

Let us capture this observation a little more precisely. Let k be the maximum arity of a relation in $\mathcal{S}$, and let $\text{ATOMS}_{\mathcal{S}}$ be the finite set of all atomic formulas using a relation from $\mathcal{S}$ and variables from $\{z_1, \ldots, z_k\}$. Given a LAV schema mapping $M = (\mathcal{S}, \mathcal{T}, \Sigma)$ and an $\alpha(\mathbf{z}) \in \text{ATOMS}_{\mathcal{S}}$, we denote by $q_{M,\alpha}(\mathbf{z})$ the following first-order formula over schema $\mathcal{T}$:

$$\bigwedge_{\substack{\forall \mathbf{x}(\beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\phi(\mathbf{x},\mathbf{y})) \in \Sigma \\ h : \{\mathbf{x}\} \rightarrow \{\mathbf{z}\} \text{ a function s.t. } \beta(h(\mathbf{x})) = \alpha(\mathbf{z})}} \exists \mathbf{y}\phi(h(\mathbf{x}), \mathbf{y})$$

For example, if M consists of the LAV constraints $\forall x_1, x_2, x_3.R(x_1, x_2, x_3) \rightarrow S(x_1, x_2, x_3)$ and $\forall x_1, x_2.R(x_1, x_2, x_2) \rightarrow \exists y T(x_1, y)$, and $\alpha(z_1)$ is $R(z_1, z_1, z_1)$, then $q_{M,\alpha} = S(z_1, z_1, z_1) \wedge \exists y T(z_1, y)$. Similarly, for $\alpha'(z_1, z_2, z_3) = R(z_1, z_2, z_3)$ then $q_{M,\alpha'} = S(z_1, z_2, z_3)$. Note that $q_{M,\alpha}(\mathbf{z})$ can be equivalently written as a not-necessarily-safe CQ over $\mathcal{T}$ (by pulling the existential quantifies to the front).

▶ **Lemma 6.2.** *Let* $M = (\mathcal{S}, \mathcal{T}, \Sigma)$ *be any LAV schema mapping, and let* $\alpha(\boldsymbol{z}) \in \text{ATOMS}_{\mathcal{S}}$ *have* k *many distinct variables. For every structure* $(A, \boldsymbol{a})$*, over schema* $\mathcal{T}$ *and with* k *distinguished elements, the following are equivalent:*

1. $(A, \boldsymbol{a})$ *is a positive data example for* $q_{M,\alpha}(\boldsymbol{z})$*,*
2. *The schema-mapping example* (I, J) *is a positive example for* M*, where* I *is the structure over* $\mathcal{S}$ *consisting of the single fact* $\alpha(\boldsymbol{a})$*, and* $J = A$*.*

We omit the proof, which is straightforward (note that the left-hand side of a LAV constraint can have at most one homomorphism to I, and the latter can be extended to the right-hand side of the constraint to J iff the respective conjunct of $q_{M,\alpha}$ is satisfied).

Intuitively, Lemma 6.2 shows that the behavior of $q_{M,\alpha}$ on arbitrary data examples, is fully determined by the behavior of M on arbitrary schema-mapping examples. The converse turns out to be true as well, that is, the semantics of a LAV schema mapping $M = (\mathcal{S}, \mathcal{T}, \Sigma)$ is determined (up to logical equivalence) by its associated queries $q_{M,\alpha}$ for $\alpha \in \text{ATOMS}_{\mathcal{S}}$:

▶ **Lemma 6.3.** *Two LAV schema mappings* $M_1 = (\mathcal{S}, \mathcal{T}, \Sigma_1)$*,* $M_2 = (\mathcal{S}, \mathcal{T}, \Sigma_1)$ *are logically equivalent iff, for every* $\alpha(\boldsymbol{z}) \in ATOMS_{\mathcal{S}}$*,* $q_{M_1,\alpha}(\boldsymbol{z})$ *and* $q_{M_2,\alpha}(\boldsymbol{z})$ *are logically equivalent.*

Proof. The left-to-right direction follows immediately from the preceding Lemma. For the right-to-left direction: suppose M_1 and M_2 are not logically equivalent. Then they disagree on some schema-mapping example (I, J). Without loss of generality, we may assume that (I, J) is a positive example for M_1 and a negative example for M_2. In particular, one of the LAV constraints in Σ_2 is false in (I, J). Since the left-hand side of a LAV constraint consists of a single atom, it follows that, for some fact $R(\mathbf{a})$ of I, the schema-mapping example $(\{R(\mathbf{a})\}, J)$ is a negative example for M_2. Moreover, an easy monotonicity argument shows that $(\{R(\mathbf{a})\}, J)$ is a positive example for M_1. Let α be obtained from the fact $R(\mathbf{a})$ by replacing each distinct element a_i by a corresponding variable z_i. It follows from Lemma 6.2 that $q_{M_1,\alpha}$ and $q_{M_2,\alpha}$ disagree on the structure $(J, \mathbf{a})$, and are not logically equivalent. ◀

It follows directly from the above Lemmas that the unique characterizability of a LAV schema mapping M reduces to the unique characterizability of each query $q_{M,\alpha}$:

▶ **Lemma 6.4.** *For all LAV schema mappings* $M = (\mathcal{S}, \mathcal{T}, \Sigma)$*, the following are equivalent:*
1. M *is uniquely characterizable by finitely many positive and negative schema-mapping examples (w.r.t. the class of all LAV schema mappings over* $\mathcal{S}, \mathcal{T}$*).*
2. *For each* $\alpha(z_1, \ldots, z_k) \in ATOMS_{\mathcal{S}}$*,* $q_{M,\alpha}(z_1, \ldots, z_k)$ *is uniquely characterizable by finitely many positive and negative data examples w.r.t. the class of all* k*-ary not-necessarily-safe CQs over* $\mathcal{T}$*.*

Intuitively, this shows that a LAV schema mapping is uniquely characterizable iff each of its constraints (joined together according to their left-hand side atom) are. By combining these lemmas with Theorem 4.5 (cf. Remark 4.6), we can link the unique characterizability of a LAV schema mapping to the condition of c-acyclicity. We say that a LAV schema mapping M is c-acyclic if the right-hand side of each of its LAV constraints is a c-acyclic not-necessarily-safe CQ. Note that, in this case, also $q_{M,\alpha}$ is c-acyclic, for each $\alpha \in \text{ATOMS}_{\mathcal{S}}$.

▶ **Theorem 6.5.** *Fix a source schema* $\mathcal{S}$ *and a target schema* $\mathcal{T}$*. A LAV schema mapping* $M = (\mathcal{S}, \mathcal{T}, \Sigma)$ *is uniquely characterizable by a finite set of positive and negative schema-mapping examples if and only if* M *is logically equivalent to a c-acyclic LAV schema mapping. Moreover, if* M *is c-acyclic, then a uniquely characterizing set of positive and negative schema-mapping examples can be constructed in polynomial time (for fixed* $\mathcal{S}, \mathcal{T}$*).*

Proof. The direction going from c-acyclicity to the uniquely characterizing set of schema-mapping examples, follows immediately from the above lemmas together with Theorem 4.5. For the other direction, assume that M is uniquely characterizable by finitely many positive

and negative schema-mapping examples. It follows by Lemma 6.4 that each $q_{M,\alpha}$ is uniquely characterizable by finitely many positive and negative data examples. Hence, each $q_{M,\alpha}$ is logically equivalent to a c-acyclic not-necessarily-safe conjunctive query $q'_{M,\alpha}$. Finally, let $M' = (\mathcal{S}, \mathcal{T}, \Sigma')$, where Σ' consists of all LAV constraints of the form $\forall \mathbf{z}(q_{M,\alpha}(\mathbf{z}) \to \alpha(\mathbf{z}))$ for $\alpha(\mathbf{z}) \in \text{ATOMS}_{\mathcal{S}}$. Then M' is c-acyclic and logically equivalent to M. ◀

Similarly, Lemma 6.2 and Lemma 6.3, together with Theorem 5.2, directly imply:

▶ **Theorem 6.6.** *Fix a source schema $\mathcal{S}$ and a target schema $\mathcal{T}$. The class of c-acyclic LAV schema mappings over $\mathcal{S}, \mathcal{T}$ is efficiently exactly learnable with membership queries.*

Note that the class of *all* LAV schema mappings over $\mathcal{S}, \mathcal{T}$ is *not* exactly learnable with membership queries (assuming that $\mathcal{S}$ is non-empty and $\mathcal{T}$ contains a relation of arity at least 2). This follows immediately from the existence of LAV schema mappings that are not uniquely characterizable by finitely many positive and negative schema-mapping examples.

As mentioned earlier, LAV schema mappings and GAV schema mappings are two of the most well-studied schema mapping languages. GLAV ("Global-and-Local-As-Views") schema mappings is another, which forms a common generalization. An important remaining open question in the area example-driven approaches to schema mapping design is the following [1]: *which GLAV schema mappings are uniquely characterizable by a finite set of examples*?

6.2 Learning description logic concept expressions and ABoxes

Description logics are formal specification languages used to represent domain knowledge. Example-driven and machine-learning based approaches have a long history in this area, and have received renewed interest in the last years [32]. In particular, ontologies specified in the lightweight description logic $\mathcal{ELI}$, focusing on the exact learnability of ontologies using entailment queries and equivalence queries. As we show in this section, our results on c-acyclic CQs have some implications for the exact learnability of $\mathcal{ELI}$ concept expressions.

▶ **Definition 6.7** ($\mathcal{ELI}$ Concept expressions, ABoxes, TBoxes)**.** *Let N_C, N_R, N_I be fixed, disjoint sets, whose members we will refer to as "concept names", "role names", and "individual names", respectively. N_C and N_R are assumed to be finite, while N_I is assumed to be infinite.*

A concept expression *C is an expression built up from from concept names in N_C and $\top$, using conjunction ($C_1 \sqcap C_2$) and existential restriction ($\exists r.C$ or $\exists r^-.C$, where $r \in N_R$).*

An ABox *is a finite set of* ABox axioms *of the form $P(a)$ and/or $r(a, b)$, where $P \in N_C$, $r \in N_R$, and $a, b \in N_I$.*

A TBox *is a finite set of* TBox axioms *$C \sqsubseteq D$, where C, D are concept expressions.*

The semantics of these expressions can be explained by translation to first-order logic:

▶ **Definition 6.8.** *The* correspondence schema *is the schema that contains a unary relation for every $A \in N_C$ and a binary relation for every $r \in N_R$. Through the standard translation from description logic to first-order logic (cf. Table 1), every concept expression C translates to a first-order formula $q_C(x)$ over the correspondence schema. By extension, every TBox $\mathcal{T}$ translates to a finite first-order theory $\mathcal{T}^{fo}$, where $C_1 \sqsubseteq C_2$ translates to $\forall x(q_{C_1}(x) \to q_{C_2}(x))$.*

An ABox can equivalently viewed as a finite structure (without designated elements), whose domain consists of individual names from N_I, and whose facts are the ABox assertions. Since N_I is assumed to be infinite, every finite structure over the correspondence schema can (up to isomorphism) be represented as an ABox. Therefore, in what follows we will use ABoxes and structures interchangeably.

$$
\begin{array}{rcl}
q_P(x) & = & A(x) \text{ for } P \in N_C \\
q_\top(x) & = & \top \\
q_{C_1 \sqcap C_2}(x) & = & q_{C_1}(x) \wedge q_{C_2}(x) \\
q_{\exists r.C}(x) & = & \exists y(r(x,y) \wedge q_C(y)) \\
q_{\exists r^-.C}(x) & = & \exists y(r(y,x) \wedge q_C(y))
\end{array}
$$

Table 1 Standard translation from concept expressions to first-order logic.

	Example	First-order logic translation
ABox:	$\mathcal{A} = \{P(a), r(a,b)\}$	
TBox:	$\mathcal{T} = \{P \sqsubseteq Q \sqcap \exists r.P\}$	$\mathcal{T}^{\text{fo}} = \{\forall x(P(x) \to Q(x) \wedge \exists y(r(x,y) \wedge P(y))\}$
Concept expr:	$C = \exists r.Q$	$q_C(x) = \exists y(r(x,y) \wedge Q(y))$

Table 2 Example description logic ABox, TBox and concept expression.

We can think of an ABox as a (possibly incomplete) list of facts, and a TBox as domain knowledge in the form of rules for deriving more facts. This idea underlies the next definition:

▶ **Definition 6.9.** *A* QA-example *is a pair $(\mathcal{A}, a)$ where $\mathcal{A}$ is an ABox and $a \in N_I$. We say that $(\mathcal{A}, a)$ is a* positive *QA-example for a concept expression C relative to a TBox $\mathcal{T}$ if $a \in certain(C, \mathcal{A}, \mathcal{T})$ where $certain(C, \mathcal{A}, \mathcal{T}) = \bigcap\{q_C(B) \mid \mathcal{A} \subseteq B \text{ and } B \models \mathcal{T}^{fo}\}$. If $a \notin certain(C, \mathcal{A}, \mathcal{T})$, we say that $(\mathcal{A}, a)$ is a negative QA-example for C relative to $\mathcal{T}$.*

The name *QA-example*, here, reflects the fact that the task of computing certain$(C, \mathcal{A}, \mathcal{T})$ is commonly known as *query answering*. It is one of the core inference tasks studied in the description logic literature. In general, there are two variants of the definition of certain$(C, \mathcal{A}, \mathcal{T})$: one where B ranges over finite structures, and one where B ranges over all, finite or infinite, structures. The description logic $\mathcal{ELI}$ that we consider here has been shown to be *finitely controllable* [5], meaning that both definitions are equivalent. For more expressive description logics, this is in general not the case.

▶ **Example 6.10.** Consider the ABox, TBox, and concept expression in Table 2. Every model of $\mathcal{T}^{\text{fo}}$ containing the facts in $\mathcal{A}$ must contain also $r(a,c)$ and $Q(c)$ for some $c \in N_I$. It follows that $a \in$ certain$(C, \mathcal{A}, \mathcal{T})$. In other words, $(\mathcal{A}, a)$ is a positive QA-example for C relative to $\mathcal{T}$. On the other hand, $(\mathcal{A}, b)$ is a negative QA-example for C relative to $\mathcal{T}$.

See [3] for more details on description logic syntax and semantics. We now explain how our results from Section 4 and 5 can be applied here. Although a QA-example is just a data example with one distinguished element, over the correspondence schema, the definition of *positive/negative* QA-examples diverges from the definition of positive/negative data examples, because of the TBox $\mathcal{T}$. For the special case where $\mathcal{T} = \emptyset$, the two coincide:

▶ **Lemma 6.11.** *Let $\mathcal{T} = \emptyset$. A QA-example $(\mathcal{A}, a)$ is a positive (negative) QA-example for a concept expression C relative to $\mathcal{T}$ iff $(\mathcal{A}, a)$ is a positive (negative) data example for $q_C(x)$.*

Lemma 6.11 follows from the well-known monotonicity property of CQs (i.e., whenever $A \subseteq B$, then $q(A) \subseteq q(B)$), which implies that certain$(C, \mathcal{A}, \emptyset) = q_C(\mathcal{A})$.

Concept expressions turn out to correspond precisely to unary, acyclic, c-connected CQs:

▶ **Lemma 6.12.** *The standard translation $q_C(x)$ of every $\mathcal{ELI}$ concept expression C is equivalent to a not-necessarily-safe unary CQ that is acyclic and c-connected. Conversely, every unary, acyclic, c-connected not-necessarily-safe CQ over the correspondence schema is logically equivalent to $q_C(x)$ for some $\mathcal{ELI}$ concept expression C.*

Both directions of Lemma 6.12 can be proved using a straightforward induction.

The above two lemmas, together with Theorem 4.5(2) and Theorem 5.2 (cf. Remark 4.6 and Remark 5.4) immediately yield our main result here. We say that a collection of positive and engative QA-examples *uniquely characterizes* a concept expression C relative to a TBox $\mathcal{T}$ if C fits the examples (relative to $\mathcal{T}$) and every other concept expression that does so is equivalent (relative to $\mathcal{T}$) to C. By a *QA-membership query* we mean an oracle query consisting of a QA example, where the oracle answers yes or no depending on whether the input is a positive QA example or a negative QA example for the goal concept, relative to the TBox. It is assumed that the TBox is fixed and known to the learner.

▶ **Theorem 6.13.** *Let $\mathcal{T} = \emptyset$. Every $\mathcal{ELI}$ concept expression is uniquely characterizable by a finite collection of positive and negative QA examples (relative to $\mathcal{T}$), which can be computed in polynomial time. Furthermore, the class of $\mathcal{ELI}$ concept expressions is efficiently exactly learnable with QA-membership queries.*

Moreover, by Theorem 4.5(3), the uniquely characterizing examples can be constructed so that each example $(\mathcal{A}, a)$ is the canonical QA-example of a concept expression. By the *canonical QA-example* of a concept expression C, here, we mean the QA-example that (viewed as a structure with one distinguished element) is the canonical structure of the not-necessarily-safe CQ $q_C(x)$.

Theorem 6.13 remains true when the concept language is extended with unrestricted existential quantification (of the form $\exists.C$) and a restricted form of the **I-me** self-reference construct introduced in [26], namely where the **I** operator can only occur once, and in the very front of the concept expression. Indeed, it can be shown that this extended concept language (by a straightforward extension of the standard translation) captures precisely the class of c-acyclic unary not-necessarily-safe CQs over the correspondence schema.

▶ **Open Question 6.14.** *Does Theorem 6.13 holds true for arbitrary TBoxes?*

Results in [25] imply that certain$(C, \cdot, \mathcal{T})$ can be expressed in a fragment of monadic Datalog. More precisely, for each $\mathcal{ELI}$ concept expression C and TBox $\mathcal{T}$, there is a monadic Datalog program Π that, such that, for every ABox $\mathcal{A}$, certain$(C, \mathcal{A}, \mathcal{T}) = \Pi(\mathcal{A})$. Moreover, the left-hand side of every Datalog rule of Π is an acyclic, c-connected (unary) CQ. The above question, therefore, may perhaps be approached by studying unique characterizability and exact learnability for a class of acyclic, c-connected monadic Datalog programs. [5]

References

1 Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23:1–23:48, December 2011. doi:10.1145/2043652.2043656.

2 Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, April 1988. doi:10.1023/A:1022821128753.

[5] In a recent manuscript [17], a positive answer is given to a weaker variant of the question, namely for the description logic $\mathcal{EL}$, when the learning algorithm is also allowed to ask equivalence queries.

3 Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

4 Franz Baader and Werner Nutt. *Basic Description Logics*, page 43–95. Cambridge University Press, USA, 2003.

5 Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014. doi:10.2168/LMCS-10(2:3)2014.

6 Pablo Barceló and Miguel Romero. The complexity of reverse engineering problems for conjunctive queries. In Michael Benedikt and Giorgio Orsi, editors, *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, volume 68 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

7 Google Blog. A reintroduction to our knowledge graph and knowledge panels, 2020. URL: https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/.

8 Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. Learning path queries on graph databases. In Gustavo Alonso, Floris Geerts, Lucian Popa, Pablo Barceló, Jens Teubner, Martín Ugarte, Jan Van den Bussche, and Jan Paredaens, editors, *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*, pages 109–120. OpenProceedings.org, 2015.

9 Angela Bonifati, Radu Ciucanu, and Slawek Staworko. Learning join queries from user examples. *ACM Trans. Database Syst.*, 40(4):24:1–24:38, 2016.

10 Ashok K. Chandra Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *ACM Symposium on Theory of Computing (STOC)*, pages 77–90, 1977.

11 William W. Cohen. Cryptographic limitations on learning one-clause logic programs. In Richard Fikes and Wendy G. Lehnert, editors, *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*, pages 80–85. AAAI Press / The MIT Press, 1993.

12 William W. Cohen. Pac-learning nondeterminate clauses. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1*, pages 676–681. AAAI Press / The MIT Press, 1994.

13 William W. Cohen. Pac-learning non-recursive prolog clauses. *Artif. Intell.*, 79(1):1–38, 1995.

14 Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, July 1983. doi:10.1145/2402.322390.

15 Ronald Fagin, Phokion Kolaitis, Alan Nash, and Lucian Popa. Towards a theory of schema-mapping optimization. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 33–42, January 2008. doi:10.1145/1376916.1376922.

16 Jan Foniok, Jaroslav Nesetril, and Claude Tardif. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Eur. J. Comb.*, 29(4):881–899, 2008.

17 Maurice Funk, Jean Christoph Jung, and Carsten Lutz. Actively learning el-concepts and queries in the presence of an ontology. Manuscript, 2020.

18 Georg Gottlob and Pierre Senellart. Schema mapping discovery from data instances. *J. ACM*, 57(2):6:1–6:37, 2010. doi:10.1145/1667053.1667055.

19 David Haussler. Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4:7–40, 1989.

20 Pavol Hell and Jaroslav Nešetřil. The Core of a Graph. *Discrete Mathematics*, 109:117–126, 1992.

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

22 Kouichi Hirata. On the hardness of learning acyclic conjunctive queries. In Hiroki Arimura, Sanjay Jain, and Arun Sharma, editors, *Algorithmic Learning Theory, 11th International*

Conference, ALT 2000, Sydney, Australia, December 11-13, 2000, Proceedings, volume 1968 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

23 Wojtek Kazana and Luc Segoufin. First-order queries on classes of structures with bounded expansion. *Logical Methods in Computer Science*, Volume 16, Issue 1, February 2020. URL: https://lmcs.episciences.org/6156.

24 Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, page 61–75, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1065167.1065176.

25 Carsten Lutz and Frank Wolter. The data complexity of description logic ontologies. *Logical Methods in Computer Science*, 13, November 2016. doi:10.23638/LMCS-13(4:7)2017.

26 Maarten Marx. Narcissists, stepmothers and spies. In Ian Horrocks and Sergio Tessaris, editors, *Proceedings of the 2002 International Workshop on Description Logics (DL2002), Toulouse, France, April 19-21, 2002*, volume 53 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2002. URL: http://ceur-ws.org/Vol-53/narcists.pdf.

27 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion iii. restricted graph homomorphism dualities. *European Journal of Combinatorics*, 29(4):1012–1024, 2008. Homomorphisms: Structure and Highlights. doi:10.1016/j.ejc.2007.11.019.

28 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion i. decompositions. *European Journal of Combinatorics*, 29(3):760–776, 2008. doi:10.1016/j.ejc.2006.07.013.

29 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity (Graphs, Structures, and Algorithms)*, volume 28. Springer, January 2012. doi:10.1007/978-3-642-27875-4.

30 Jaroslav Nešetřil and Claude Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *Journal of Combinatorial Theory, Series B*, 80(1):80–97, 2000. doi:10.1006/jctb.2000.1970.

31 Jaroslav Nešetřil and Claude Tardif. Short answers to exponentially long questions: Extremal aspects of homomorphism duality. *SIAM J. Discret. Math.*, 19(4):914–920, August 2005. doi:10.1137/S0895480104445630.

32 Ana Ozaki. Learning description logic ontologies: Five approaches. where do they stand? *KI - Künstliche Intelligenz*, April 2020. doi:10.1007/s13218-020-00656-9.

33 Slawek Staworko and Piotr Wieczorek. Learning twig and path queries. In Alin Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 140–154. ACM, 2012.

34 Balder ten Cate, Laura Chiticariu, Phokion Kolaitis, and Wang-Chiew Tan. Laconic schema mappings: Computing the core with sql queries. *Proc. VLDB Endow.*, 2(1):1006–1017, August 2009. doi:10.14778/1687627.1687741.

35 Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In Marcelo Arenas and Martín Ugarte, editors, *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, volume 31 of *LIPIcs*, pages 161–176. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

36 Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. Learning schema mappings. *ACM Trans. Database Syst.*, 38(4):28:1–28:31, 2013. doi:10.1145/2539032.2539035.

37 Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. Active learning of GAV schema mappings. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 355–368. ACM, 2018. doi:10.1145/3196959.3196974.

38 Yaacov Y. Weiss and Sara Cohen. Reverse engineering spj-queries from examples. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 151–166. ACM, 2017.

39 Ross Willard. Testing expressibility is hard. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 9–23. Springer, 2010.