Extremal Fitting Problems for Conjunctive Queries

Balder ten Cate
ILLC, Universiteit van Amsterdam
Netherlands
b.d.tencate@uva.nl

Victor Dalmau
Universitat Pompeu Fabra
Spain
victor.dalmau@upf.edu

Maurice Funk
Universität Leipzig
ScaDS.AI Center Dresden/Leipzig
Germany
mfunk@informatik.uni-leipzig.de

Carsten Lutz
Universität Leipzig
ScaDS.AI Center Dresden/Leipzig
Germany
clu@informatik.uni-leipzig.de

ABSTRACT

The *fitting problem* for conjunctive queries (CQs) is the problem to construct a CQ that fits a given set of labeled data examples. When a fitting CQ exists, it is in general not unique. This leads us to proposing natural refinements of the notion of a fitting CQ, such as *most-general fitting CQ*, *most-specific fitting CQ*, and *unique fitting CQ*. We give structural characterizations of these notions in terms of (suitable refinements of) homomorphism dualities, frontiers, and direct products, which enable the construction of the refined fitting CQs when they exist. We also pinpoint the complexity of the associated existence and verification problems, and determine the size of fitting CQs. We study the same problems for UCQs and for the more restricted class of tree CQs.

CCS CONCEPTS

• **Information systems** → **Query languages**; • **Computing methodologies** → **Supervised learning**.

KEYWORDS

Conjunctive Queries, Database Queries, Data Examples, Fitting, Homomorphism Dualities

ACM Reference Format:
Balder ten Cate, Victor Dalmau, Maurice Funk, and Carsten Lutz. 2023. Extremal Fitting Problems for Conjunctive Queries. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3584372.3588655

1 INTRODUCTION

The *fitting problem* for conjunctive queries (CQs) is the problem to construct a CQ q that fits a given set of labeled data examples, meaning that q returns all positive examples as an answer while returning none of the negative examples. This fundamental problem has a long history in database research. It lies at the heart of the classic *Query-By-Example* paradigm that aims to assist users in query formation and query refinement, and has been intensively studied for CQs [4, 33, 36] and other types of queries (e.g., [3, 7, 18]). The fitting problem is also central to *Inductive Logic Programming* [19, 22], where CQs correspond to the basic case of non-recursive single-rule Datalog programs, and has close connections to fitting problems for *schema mappings* [2, 10]. More recent motivation comes from *automatic feature generation in machine learning with relational data* [5, 29]. Here, the CQ fitting problem arises because a CQ that separates positive from negative examples in (a sufficiently large subset of) a labeled dataset is a natural contender for being added as an input feature to the model [5]. In addition, there has been significant recent interest in fitting CQs and other queries in knowledge representation, typically in the presence of an ontology [9, 21, 26, 27, 32, 35].

When a fitting CQ exists, in general it need not be unique up to equivalence. In fact, there may be infinitely many pairwise non-equivalent fitting CQs. However, the fitting CQs form a *convex set*: whenever two CQs q_1, q_2 fit a set of labeled examples, then the same holds for every CQ q with $q_1 \subseteq q \subseteq q_2$, where "$\subseteq$" denotes query containment. Maximal elements of this convex set can be viewed as "most-general" fitting CQs, while minimal elements can be viewed as "most-specific" fitting CQs. The set of all most-general and all most-specific fitting CQs (when they exist), can thus be viewed as natural representatives of the entire set of fitting CQs, c.f. the version-space representation theorem used in machine learning [34, Chapter 2.5]. In the context of automatic feature generation mentioned above, it would thus be natural to compute all extremal fitting CQs and add them as features, especially when infinitely many fitting CQs exist. Likewise, in query refinement tasks where the aim is to construct a modified query that excludes unwanted answers or includes missing answers (cf. [36]), it is also natural to ask for a most-general, respectively, most-specific fitting query.

In this paper we embark on a systematic study of extremal fitting CQs. To the best of our knowledge, we are the first to do so. We show that the intuitive concepts of most-general and most-specific fitting CQs can be formalized in multiple ways. We give structural characterizations of each notion, study the associated verification, existence, and computation problems, and establish upper and lower bounds on the size of extremal fitting CQs. The characterizations link "weakly most-general" fittings to the notion of homomorphism frontiers, "complete bases" of most-general fittings to (a certain relativized version of) homomorphism dualities, and most-specific fittings to direct products. We use the structural characterizations to obtain effective algorithms and pinpoint the exact complexity of

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06.
https://doi.org/10.1145/3584372.3588655

	Verification	Existence	Construction and size
Any Fitting	DP-c (Thm 3.1)	coNExpTime-c [10, 37]	In ExpTime [37]; Exp size lower bound (Thm 3.26)
Most-Specific	NExpTime-c (Thm 3.7, 3.23)	coNExpTime-c [10, 37]	In ExpTime [37]; Exp size lower bound (Thm 3.26)
Weakly Most-General	NP-c (Thm 3.12)	ExpTime-c (Thm 3.13, 3.24)	In 2ExpTime (Thm 3.13); Exp size lower bound (Thm 3.26)
Basis of Most-General	NExpTime-c (Thm 3.17, 3.23)	NExpTime-c (Thm 3.17, 3.23)	In 3ExpTime (Thm 3.18); 2Exp size lower bound (Thm 3.27)
Unique	NExpTime-c (Thm 3.21, 3.23)	NExpTime-c (Thm 3.21, 3.23)	In ExpTime [37]; Exp size lower bound (Thm 3.26)

Table 1: Summary of results for CQs

	Verification	Existence	Construction and size
Any Fitting	DP-complete (Thm 4.5)	coNP-complete (Thm 4.5)	in PTime (Thm 4.5
Most-Specific	DP-complete (Thm 4.5)	coNP-complete (Thm 4.5)	in PTime (Thm 4.5)
Most-General	HomDual-equivalent (Thm 4.7)	NP-c (Thm 4.5)	in 2ExpTime (Thm 4.5)
Unique	HomDual-equivalent (Thm 4.7)	HomDual-equivalent (Thm 4.7)	in PTime (Thm 4.5)

Table 2: Summary of results for UCQs

	Verification	Existence	Construction and size
Any Fitting	PTime (Thm 5.2)	ExpTime-c [21]	In 2ExpTime (Thm 5.4); 2Exp size lower bound (Thm 5.21)
Most-Specific	ExpTime-c (Thm 5.7, 5.20)	ExpTime-c (Thm 5.7, 5.20)	In 2ExpTime (Thm 5.8)
Weakly Most-General	PTime (Thm 5.11)	ExpTime-c (Thm 5.12, 5.20)	In 2ExpTime (Thm 5.12)
Basis of Most-General	ExpTime-c (Thm 5.16, 5.20)	ExpTime-c (Thm 5.18, 5.20)	In 3ExpTime (2Exp upper bound on size of members) (Thm 5.18)
Unique	ExpTime-c (Thm 5.13, 5.20)	ExpTime-c (Thm 5.13, 5.20)	in 2ExpTime (Thm 5.4)

Table 3: Summary of results for tree CQs

the decision and computation problems mentioned above, and to establish size bounds. Our algorithms use a combination of techniques from automata theory and from the literature on constraint satisfaction problems. We perform the same study for two other natural classes of database queries, namely *unions of conjunctive queries* (UCQs) and acyclic connected unary CQs, from now on referred to as *tree CQs*. For the latter class, which holds significance as it corresponds to the concept language of the description logic $\mathcal{ELI}$ that is prominent in knowledge representation, we restrict our attention to relation symbols of arity one and two. The main complexity results and size bounds for CQs, UCQs, and tree CQs are summarized in Tables 1, 2, and 3. Note that, since the classical (non-extremal) fitting problem for CQs is already coNExpTime-complete [10, 37], it is not surprising that many of the problems we consider here turn out to be of similarly high complexity. We will comment on possible strategies for taming the complexity of these problems in Sect. 6.

All proofs are provided in the appendix of the long version [12].

Related Work. The fitting problem for CQs and UCQs, as well as for bounded-treewidth CQs and UCQs, was studied in [2, 4, 10, 37]. Note that, from a fitting point of view, the *GAV schema mappings* and *LAV schema mappings* studied in [2] correspond in a precise way to UCQs and CQs, respectively, cf. [10]. The fitting problem for tree CQs (equivalently, $\mathcal{ELI}$*-concept expressions*) was studied in [21]. The fitting problem for CQs is also closely related to the ILP consistency problem for Horn clauses, studied in [22], although the latter differs in assuming a bound on the size of clauses.

The notion of a *most-specific fitting query* appears in several places in this literature, largely because of the fact that some of the canonical fitting algorithms naturally produce such fittings. We are not aware of any prior work studying the verification or construction of most-general fitting queries or unique fitting queries, although [11] studies the inverse problem, namely the existence and construction of uniquely characterizing examples for a query, and we build on results from [11].

The problem of deriving queries from data examples has also been studied from the perspective of computational learning theory, cf. the related work sections in [11, 14].

2 PRELIMINARIES

Schema, Instance, CQ, Homomorphism, Core. A *schema* (or relational signature) is a finite set of relation symbols $\mathcal{S} = \{R_1, \ldots, R_n\}$, where each relation symbol R_i has an associated arity $\text{arity}(R_i) \geq 1$. A *fact* over $\mathcal{S}$ is an expression $R(a_1, \ldots, a_n)$, where $a_1, \ldots, a_n$ are *values*, $R \in \mathcal{S}$, and $\text{arity}(R) = n$. An *instance over* $\mathcal{S}$ is a finite set I of facts over $\mathcal{S}$. The *active domain* of I (denoted $\text{adom}(I)$) is the set of all values occurring in facts of I.

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 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. The variables in $\mathbf{x}$ are called *answer variables*, and the other variables *existential variables*. Each answer variable is required to occur in at least one conjunct α_i. This requirement is known as the *safety* condition. A CQ of arity 0 is called a *Boolean CQ*.

If q is a k-ary CQ and I is an instance over the same schema as q, we denote by $q(I)$ the set of all k-tuples of values from the active domain of I that satisfy the query q in I. We write $q \subseteq q'$ if q and q' are queries over the same schema, and of the same arity, and $q(I) \subseteq q'(I)$ holds for all instances I. We say that q and q' are *logically equivalent* (denoted $q \equiv q'$) if $q \subseteq q'$ and $q' \subseteq q$ both hold.

Given two instances I, J over the same schema, a *homomorphism* $h : I \to J$ is a map from $\text{adom}(I)$ to $\text{adom}(J)$ that preserves all facts. When such a homomorphism exists, we say that I "homomorphically maps to" J and write $I \to J$. We say that I and J are *homomorphically equivalent* if $I \to J$ and $J \to I$.

It is well known that every instance I has a unique (up to isomorphism) minimal subinstance to which it is homomorphically equivalent, known as the *core* of I. Furthermore, two instances are homomorphically equivalent iff their cores are isomorphic.

Pointed Instance, Canonical Instance, Canonical CQ, UNP. A *pointed instance* for schema $\mathcal{S}$ is a pair $(I, \mathbf{a})$ where I is an instance over $\mathcal{S}$, and $\mathbf{a}$ is a tuple of values. The values in $\mathbf{a}$ are typically elements of $\text{adom}(I)$, but we also admit here values from outside of $\text{adom}(I)$ as this allows us to simplify some definitions and proofs. If the tuple $\mathbf{a}$ consists of k values, then we call $(I, \mathbf{a})$ a *k-ary* pointed instance. We refer to $\mathbf{a}$ as the *distinguished elements* of the pointed instance.

The definition of a homomorphism naturally extends to pointed instances. More precisely a homomorphism $h : (I, \mathbf{a}) \to (J, \mathbf{b})$ is a map from $\text{adom}(I) \cup \{\mathbf{a}\}$ to $\text{adom}(J) \cup \{\mathbf{b}\}$ that maps every fact of I to a fact of J, and that maps every distinguished element a_i to the corresponding distinguished element b_i.

There is a natural correspondence between k-ary CQs over a schema $\mathcal{S}$ and k-ary pointed instances over $\mathcal{S}$. In one direction, the *canonical instance* of a CQ $q(\mathbf{x})$ is the pointed instance $\widehat{q} = (I_q, \mathbf{x})$, where the domain of I_q is the set of variables occurring in q and the facts of I_q are the conjuncts of q. Note that every distinguished element of $\widehat{q}$ does indeed belong to the active domain (i.e. occurs in a fact), due to the safety condition of CQs. Conversely, the *canonical CQ* of a pointed instance $(I, \mathbf{a})$ with $\mathbf{a} = a_1, \ldots, a_k$ is the CQ $q(x_{a_1}, \ldots, x_{a_k})$ that has a variable x_a for every value $a \in \text{adom}(I)$, and a conjunct for every fact of A. Here, we assume that all distinguished elements belong to the active domain.

We write $q \to q'$ when there is a homomorphism $h : \widehat{q} \to \widehat{q'}$ and $q \to (I, \mathbf{a})$ when $\widehat{q} \to (I, \mathbf{a})$. By the classic Chandra-Merlin Thm. [17], then, a tuple $\mathbf{a}$ belongs to $q(I)$ if and only if $q \to (I, \mathbf{a})$ holds; and $q \subseteq q'$ holds if and only if $q' \to q$.

A pointed instance $(I, \mathbf{a})$, with $\mathbf{a} = a_1, \ldots, a_k$, has the *Unique Names Property (UNP)* if $a_i \neq a_j$ for all $i \neq j$.

Disjoint Union, Direct Product. Let $(I, \mathbf{a})$ and $(J, \mathbf{a})$ be pointed instances over the same schema $\mathcal{S}$ with the UNP, where both pointed instances have the same tuple of distinguished elements. Furthermore, assume that $\text{adom}(I) \cap \text{adom}(J) \subseteq \{\mathbf{a}\}$. Then the *disjoint union* $(I, \mathbf{a}) \uplus (J, \mathbf{a})$ is the pointed instance $(I \cup J, \mathbf{a})$, where the facts of $I \cup J$ are the union of the facts of I and J. This construction generalizes to arbitrary pairs of k-ary pointed instances with the UNP, by taking suitable isomorphic copies of the input instances (to ensure that they have the same tuple of distinguished elements, and are disjoint otherwise). This operation also naturally generalizes to finite sets of k-ary pointed instances with the UNP.

The *direct product* of two k-ary pointed instances $(I, \mathbf{a})$ and $(J, \mathbf{b})$, where $\mathbf{a} = \langle a_1, \ldots a_k \rangle$ and $\mathbf{b} = \langle b_1, \ldots, b_k \rangle$ is the k-ary pointed instance $(I \times J, \langle (a_1, b_1), \ldots, (a_k, b_k) \rangle)$, where $I \times J$ consists of all facts $R((c_1, d_1), \ldots, (c_n, d_n))$ such that $R(c_1, \ldots, c_n)$ is a fact of I and $R(d_1, \ldots, d_n)$ is a fact of J. The same operation of direct product can also be applied to CQs: for CQs $q_1(\mathbf{x}), q_2(\mathbf{y})$ (over the same schema and of the same arity), $q_1 \times q_2$ is the canonical CQ of the direct product of pointed instances $(I_{q_1}, \mathbf{x}) \times (I_{q_2}, \mathbf{y})$. However, this yields a well-defined CQ only when the designated elements of $(I_{q_1}, \mathbf{x}) \times (I_{q_2}, \mathbf{y})$ belong to the active domain, which is not necessarily the case. This construction extends naturally to finite sets of pointed instances (where the direct product of an empty set of pointed instances is, by convention, the pointed instance $(I, \langle a, \ldots, a \rangle)$ where I consists of all possible facts over the singleton domain $\{a\}$).

Data Example, Fitting Problem. A k-ary *data example* for schema $\mathcal{S}$ (for $k \geq 0$) is a pointed instance $e = (I, \mathbf{a})$ where $\mathbf{a}$ is a k-tuple of values from $\text{adom}(I)$. A data example $(I, \mathbf{a})$ is said to be a *positive example* for a query q (over the same schema and of the same arity) if $\mathbf{a} \in q(I)$, and a *negative example* otherwise. By a *collection of labeled examples* we mean a pair $E = (E^+, E^-)$ of finite sets of data examples. The size of a data example e (as measured by the number of facts) is denoted by $|e|$, and the combined size of a set of data examples E by $||E|| = \Sigma_{e \in E} |e|$.

We say that q *fits* E if each data example in E^+ is a positive example for q and each data example in E^- is a negative example for q. The *fitting problem* (for CQs) is the problem, given as input a collection of labeled examples, to decide if a fitting CQ exists.

A special case is where the input examples involve a single database instance I, and hence can be given jointly as (I, S^+, S^-), where S^+, S^- are sets of tuples. We focus on the general version of the fitting problem here, but note that the aforementioned special case typically carries the same complexity (cf. [10, Thm. 2]).

Frontiers, Dualities, C-Acyclicity, Degree. A *frontier* for a CQ is, intuitively, a finite complete set of minimal weakenings of q. Formally, a finite set of CQs $\{q_1, \ldots, q_n\}$ is a frontier for a CQ q, with respect to a class C of CQs, if:

(1) for all $i \leq n$, $q_i \to q$ and $q \not\to q_i$, and
(2) for all $q' \in C$ such that $q' \to q$ and $q \not\to q'$, it holds that $q' \to q_i$ for some $i \leq n$.

If C is the class of all CQs, we simply call $\{q_1, \ldots, q_n\}$ a frontier for q.

Another related concept is that of *homomorphism dualities*. A pair of finite sets of data examples (F, D) is a homomorphism duality if

$$\{e \mid e \text{ is a data example and } e \to e' \text{ for some } e' \in D\} =$$
$$\{e \mid e \text{ is a data example and } e' \not\to e \text{ for all } e' \in F\}.$$

Homomorphism dualities have been studied extensively in the literature on combinatorics and constraint satisfaction, and elsewhere (e.g., [6, 20, 30]).

Frontiers and homomorphism dualities were studied in [11, 20]. Their existence was characterized in terms of a structural property called *c-acyclicity*: the *incidence graph* of a CQ q is the bipartite multi-graph consisting of the variables and the atoms of q, and such that there is a distinct edge between a variable and an atom for each occurrence of the variable in the atom. A CQ q is *c-acyclic* if every cycle in the incidence graph (including every self-loop and every cycle of length 2 consisting of different edges that connect the same pair of nodes) passes through an answer variable of q.

Theorem 2.1 ([1, 11]). *For all CQs q the following are equivalent:*

(1) q has a frontier,
(2) there exists a homomorphism duality $(\{q\}, D)$,
(3) the core of the canonical instance of q is c-acyclic.

Furthermore, for any fixed $k \geq 0$, a frontier for a k-ary c-acyclic CQs can be computed in polynomial time, and a set D as in (2) can be computed in exponential time.

By the *degree* of a CQ q we mean the maximum degree of variables in the incidence graph of q (i.e., the maximum number of occurrences of a variable in q).

3 THE CASE OF CONJUNCTIVE QUERIES

In this section, we study the fitting problem for CQs. We first review results for the case where the fitting CQ needs not satisfy any further properties. After that, we introduce and study extremal fitting CQs, including most-general, most-specific, and unique fittings. For these, we first concentrate on characterizations and upper bounds, deferring lower bounds to Sect. 3.5

To simplify presentation, when we speak of a CQ q in the context of a collection of labeled examples E, we mean that q ranges over CQs that have the same schema and arity as the data examples in E.

3.1 Arbitrary Fitting CQs

We first consider the verification problem for fitting CQs: *given a collection of labeled examples E and a CQ q, does q fit E?* This problem naturally falls in the complexity class DP (i.e., it can be expressed as the intersection of a problem in NP and a problem in coNP). Indeed:

Theorem 3.1. *The verification problem for fitting CQs is DP-complete. The lower bound holds for a schema consisting of a single binary relation, a fixed collection of labeled examples, and Boolean CQs.*

The *existence* problem for fitting CQs (*given a collection of labeled examples E, is there a CQ that fits E?*) was studied in [10, 37].

Theorem 3.2 ([10, 37]). *The existence problem for fitting CQs is coNExpTime-complete. The lower bound holds already for Boolean CQs over a fixed schema consisting of a single binary relation.*

Theorem 3.3 ([37]). *If any CQ fits a collection of labeled examples $E = (E^+, E^-)$, then the canonical CQ of the direct product $\Pi_{e \in E^+}(e)$ is well-defined and fits E.*

When we are promised that a fitting CQ exists, we can construct one in (deterministic) single exponential time. We will see in Sect. 3.5 that this is optimal, as there is a matching lower bound.

3.2 Most-Specific Fitting CQs

There are two natural ways to define *most-specific* fitting CQs:

Definition 3.4.

- A CQ q is a **strongly most-specific fitting** CQ for a collection of labeled examples E if q fits E and for every CQ q' that fits E, we have $q \subseteq q'$.
- A CQ q is a **weakly most-specific fitting** CQ for a collection of labeled examples E if q fits E and for every CQ q' that fits E, $q' \subseteq q$ implies $q \equiv q'$.

It follows from Thm. 3.3 that the above two notions coincide:

Proposition 3.5. *For all CQs q and collections of labeled examples $E = (E^+, E^-)$, the following are equivalent:*

(1) q is strongly most-specific fitting for E,
(2) q is weakly most-specific fitting for E,
(3) q fits E and q is homomorphically equivalent to the canonical CQ of $\Pi_{e \in E^+}(e)$. [1]

In light of Prop. 3.5, we simply speak of *most-specific fitting CQs*, dropping "weak" and "strong".

[1] In particular, in this case, the canonical CQ of $\Pi_{e \in E^+}(e)$ is well-defined.

Example 3.6. Let $\mathcal{S} = \{R, P\}$, where R is a ternary relation and P is a unary relation. Consider the collection of labeled examples $E = (E^+ = \{I_1, I_2\}, E^- = \{I_3\})$, where $I_1 = \{R(a, a, b), P(a)\}$, $I_2 = \{R(c, d, d), P(c)\}$, and $I_3 = \emptyset$. The Boolean CQs $q_1 :\text{-} \exists xyz\, R(x, y, z)$ and $q_2 :\text{-} \exists xyz(R(x, y, z) \wedge P(x))$ both fit E, but q_2 is more specific than q_1. Indeed, q_2 is most-specific fitting for E, as it is homomorphically equivalent to the canonical query of $I_1 \times I_2$.

It follows from Prop. 3.5 and Thm. 3.2 that the existence problem for most-specific fitting CQs coincides with that for arbitrary fitting CQs, and hence, is coNExpTime-complete; and that we can construct in exponential time a CQ q (namely, the canonical CQ of $\Pi_{e \in E^+}(e)$), with the property that, if there is a most-specific fitting CQ, then q is one. For the *verification* problem, finally, Thm. 3.3, with Thm. 3.1, implies:

Theorem 3.7. *The verification problem for most-specific fitting CQs is in NExpTime.*

3.3 Most-General Fitting CQs

For *most-general fitting* CQs, there are again two natural definitions.

Definition 3.8.

- A CQ q is a **strongly most-general fitting** CQ for a collection of labeled examples E if q fits E and for all CQs q' that fit E, we have $q' \subseteq q$.
- A CQ q is a **weakly most-general fitting** CQ for a collection of labeled examples E if q fits E and for every CQ q' that fits E, $q \subseteq q'$ implies $q \equiv q'$

Unlike in the case of most-specific fitting CQs, as we will see, these two notions do *not* coincide. In fact, there is a third:

Definition 3.9. A finite set of CQs $\{q_1, \ldots, q_n\}$ is a **basis of most-general fitting CQs** for E if each q_i fits E and for all CQs q' that fit E, we have $q' \subseteq q_i$ for some $i \leq n$. If, in addition, no strict subset of $\{q_1, \ldots, q_n\}$ is a basis of most-general fitting CQs for E, we say that $\{q_1, \ldots, q_n\}$ is a *minimal* basis.

Each member of a minimal basis is indeed guaranteed to be weakly most-general fitting. The same does not necessarily hold for non-minimal bases. We could have included this as an explicit requirement in the definition, but we decided not to, in order to simplify the statement of the characterizations below.

It is easy to see that minimal bases are unique up to homomorphic equivalence. Also, a strongly most-general fitting CQ is simply a basis of size 1. We will therefore consider the notions of *weakly most-general fitting CQs* and *basis of most-general fitting CQs*, only.

Example 3.10. Let $\mathcal{S} = \{R, P, Q\}$, where R is a binary relation and P, Q are unary relations. The following examples pertain to Boolean CQs. Let K_2 be the 2-element clique, i.e., $K_2 = \{R(a, b), R(b, a)\}$. Furthermore, let I_P, I_Q, and I_{PQ} be the instances consisting of the set of facts $\{P(a)\}$, $\{Q(a)\}$, and $\{P(a), Q(a)\}$, respectively.

(1) The collection of labeled examples $(E^+ = \emptyset, E^- = \{I_{PQ}\})$ has a strongly most-general fitting CQ, namely $q :\text{-} \exists xy(R(x, y))$.
(2) The collection of labeled examples $E = (E^+ = \emptyset, E^- = \{I_P, I_Q\})$ has a basis of most-general fitting CQs of size two, consisting of $q_1 :\text{-} \exists xy(R(x, y))$ and $q_2 :\text{-} \exists xy(P(x) \wedge Q(y))$. In particular, each of these two CQs is weakly most-general fitting for E.

(3) The collection of labeled examples $E = (E^+ = \emptyset, E^- = \{K_2\})$ does not have a weakly most-general fitting CQ. Indeed, a CQ q fits E iff q is not two-colorable, i.e., q, viewed as a graph, contains a cycle of odd length. Take a fitting CQ q and let k be the size of the smallest cycle in q of odd length. For C_{3k} the (odd) cycle of length $3k$, we have $q \subseteq C_{3k}$ and $C_{3k} \not\subseteq q$, and C_{3k} fits E.
(4) The collection of labeled examples $(E^+ = \emptyset, E^- = \{K_2, I_P, I_Q\})$ has a weakly most-general fitting CQ, namely $q :\text{-} \exists xy(P(x) \wedge Q(y))$. By the same reasoning as in (3), there is no basis of most-general fitting CQs.

Weakly most-general fitting CQs. As it turns out, weakly most-general fitting CQs can be characterized in terms of frontiers.

Proposition 3.11. *The following are equivalent for all collections of labeled examples $E = (E^+, E^-)$ and all CQs q:*

(1) q is weakly most-general fitting for E,
(2) q fits E, q has a frontier and every element of the frontier has a homomorphism to an example in E^-,
(3) q fits E and $\{q \times q_e \mid e \in E^- \text{ and } q \times q_e \text{ is a well-defined CQ}\}$ is a frontier for q,

where q_e is the canonical CQ of e.

Using Thm. 2.1, we can now show:

Theorem 3.12. *Fix $k \geq 0$. The verification problem for weakly most-general fitting k-ary CQs is NP-complete. In fact, it remains NP-complete even if the examples are fixed suitably and, in addition, the input query is assumed to fit the examples.*

Theorem 3.13. *The existence problem for weakly most-general fitting CQs is in ExpTime. Moreover, if such a CQ exists, then*

(1) there is one of doubly exponential size and
(2) we can produce one in time $2^{2^{poly(n)}} + poly(m)$ where $n = ||E||$ and m is the size of the smallest weakly most-general fitting CQ.

The proof of Thm. 3.13 uses tree automata. More precisely, we show that, given a collection of labeled examples $E = (E^+, E^-)$, (i) if there is a weakly most-general fitting CQ for E, then there is one that is c-acyclic and has a degree at most $||E^-||$; and (ii) we can construct in ExpTime a non-deterministic tree automaton $\mathfrak{A}_E$ that accepts precisely the (suitably encoded) c-acyclic weakly most-general fitting CQs for E of degree at most $||E^-||$.

Bases of most-general fitting CQs. In the same way that the weakly most-general fitting CQs are characterized in terms of frontiers, *bases of most-general fitting CQs* admit a characterization in terms of *homomorphism dualities*. To spell this out, we need a refinement of this concept, *relativized* homomorphism dualities.

Definition 3.14 (Relativized homomorphism dualities). A pair of finite sets of data examples (F, D) forms a homomorphism duality relative to a data example p, if for all data examples e with $e \to p$, the following are equivalent:

(1) e homomorphically maps to a data example in D,
(2) No data example in F homomorphically maps to e.

Proposition 3.15. *For all collections of labeled examples $E = (E^+, E^-)$, the following are equivalent for all CQs $q_1, \ldots, q_n$:*

(1) $\{q_1, \ldots, q_n\}$ is a basis of most-general fitting CQs for E,
(2) each q_i fits E and $(\{e_{q_1}, \ldots, e_{q_n}\}, E^-)$ is a homomorphism duality relative to p,

where $p = \Pi_{e \in E^+}(e)$ and e_{q_i} is the canonical instance of q_i.

While homomorphism dualities have been studied extensively, we are not aware of the relativized variant having been considered.

Theorem 3.16.

(1) The following is NP-complete: given a finite set of data examples D and a data example p, is there a finite set of data examples F such that (F, D) is a homomorphism duality relative to p?
(2) Given a finite set of data examples D and a data example p, if there is a finite set of data examples F such that (F, D) is a homomorphism duality relative to p, then we can compute in 2ExpTime such a set F, where each $e \in F$ is of size $2^{O(||D||^2 \cdot \log ||D|| \cdot |p|)}$.

Thm. 3.16(1) was proved in [30] and [8] for non-relativized dualities and where D consists of a single instance without distinguished elements. Our proof, given in the appendix, extends the one in [8]. As a consequence, we get:

Theorem 3.17. *The existence and verification problems for bases of most-general fitting CQs is in NExpTime.*

Theorem 3.18. *Let $E = (E^+, E^-)$ be a collection of labeled examples, for which a basis of most-general fitting CQs exists. Then we can compute a minimal such basis in 3ExpTime, consisting of CQs of size $2^{poly(||E^-||) \cdot 2^{O(||E^+||)}}$.*

3.4 Unique fitting CQs

By a *unique fitting CQ* for a collection of labeled examples E, we mean a fitting CQ q with the property that *every CQ that fits E is logically equivalent to q*.

Example 3.19. Let $\mathcal{S}$ consist of a single binary relation R, and let I be the instance consisting of the facts $R(a, b)$, $R(b, a)$, and $R(b, b)$. Let $E = (E^+, E^-)$, where $E^+ = \{(I, b)\}$ and $E^- = \{(I, a)\}$.

The query $q(x) :\text{-} R(x, x)$ is a unique fitting CQ for E. Indeed, q fits E, and it is easy to see that if $q'(x)$ is any CQ that fits E, then q' must contain the conjunct $R(x, x)$ (in order to fit E^-). From this, it is easy to see that q and q' admit homomorphisms to each other.

Proposition 3.20. *For every CQ q and collection of labeled examples $E = (E^+, E^-)$ the following are equivalent:*

(1) q is a unique fitting CQ for E,
(2) q is a most-specific and weakly most-general fitting CQ for E,
(3) q is homomorphically equivalent to $\Pi_{e \in E^+}(q_e)$ and $\{q \times e \mid e \in E^- \text{ and } q \times e \text{ is a well-defined CQ}\}$ is a frontier for q.

Our previous results on most-specific fitting CQs and weakly most-general fitting CQs now immediately imply:[2]

Theorem 3.21. *The verification and existence problems for unique fitting CQs are in NExpTime. When a unique fitting CQ exists, it can be computed in exponential time.*

[2]Indeed, the existence of a unique fitting CQ for $E = (E^+, E^-)$ can be tested simply by checking that $\Pi_{e \in E^+}(e)$ is weakly most-general fitting for E.

3.5 Lower bounds

The lower bound proofs below involve reductions from the *Product Homomorphism Problem* (PHP) [10]. The PHP takes as input a set of instances $I_1, \ldots, I_n$ and an instance J, and asks whether the direct product $I_1 \times \cdots \times I_n$ admits a homomorphism to J. This problem is NExpTime-complete [10, 37]. We need a refinement of this:

Theorem 3.22. *Let $\mathcal{S}$ consist of a single binary relation. There is a fixed k for which the following problem is NExpTime-complete: given k-ary pointed $\mathcal{S}$-instances $(I_1, \boldsymbol{a}_1) \ldots, (I_n, \boldsymbol{a}_n)$ and $(J, \boldsymbol{b})$ with the UNP, where $(J, \boldsymbol{b})$ is c-acyclic, is it the case that $\Pi_i(I_i, \boldsymbol{a}_i) \to (J, \boldsymbol{b})$?*

Using this we obtain the following results:

Theorem 3.23. *The following problems are NExpTime-hard:*

(1) The verification problem for most-specific fitting CQs.
(2) The verification problem for unique fitting CQs.
(3) The existence problem for unique fitting CQs.
(4) The verification problem for bases of most-general fitting CQs.
(5) The existence problem for bases of most-general fitting CQs.

Each problem is NExpTime-hard already for a fixed schema and arity, and, in the case of the verification problems, when restricted to inputs where the input CQ fits the examples, or, in the case of the existence problems, when restricted to inputs where a fitting CQ exists.

For the existence of weakly most-general fitting CQs, we prove an ExpTime lower bound by adapting a reduction from the word problem for certain alternating Turing machines used in [24] to prove hardness of a product simulation problem for transition systems. Unlike the previous reductions, it does not apply to the restricted case where a fitting CQ is promised to exist.

Theorem 3.24. *The existence problem for weakly most-general fitting CQs is ExpTime-hard.*

The same proof also shows that Thm. 3.24 holds for *tree CQs*, which we will introduce and study in Sect. 5.

The following results provide *size* lower bounds:

Theorem 3.25. *Fix a schema consisting of a single binary relation. For $n > 0$, we can construct a collection of Boolean data examples of combined size polynomial in n such that a fitting CQ exists, but not one of size less than 2^n.*

We can prove something stronger (but not for a fixed schema):

Theorem 3.26. *For $n \geq 0$, we can construct a schema with $O(n)$ unary and binary relations and a collection of labeled examples of combined size polynomial in n such that*

(1) There is a unique fitting CQ.
(2) Every fitting CQ contains at least 2^n variables.

Theorem 3.27. *For $n \geq 0$, we can construct a schema with $O(n)$ unary and binary relations and a collection of labeled examples of combined size polynomial in n such that*

(1) There is a basis of most-general fitting CQs.
(2) Every such basis contains at least 2^{2^n} CQs.

4 THE CASE OF UCQS

A k-ary *union of conjunctive queries* (UCQ) over a schema $\mathcal{S}$ is an expression of the form $q_1 \cup \cdots \cup q_n$, where $q_1, \ldots, q_n$ are k-ary CQs over $\mathcal{S}$. Each q_i is called a *disjunct* of q. Logically, q is interpreted as the disjunction of $q_1, \ldots, q_n$. For two UCQs q, q', we say that q *maps homomorphically to* q' (written: $q \to q'$) if, for every disjunct q'_i of q', there is a disjunct q_j of q such that $q_j \to q'_i$. Under this definition, as for CQs we have $q \to q'$ precisely if $q' \subseteq q$. All the notions and problems considered in Sect. 3 now naturally generalize to UCQs.

Example 4.1. Consider a schema consisting of the unary relations P, Q, R, and let $k = 0$. Let E consist of positive examples $\{P(a), Q(a)\}$ and $\{P(a), R(a)\}$, and negative examples $\{P(a)\}$ and $\{Q(a), R(a)\}$. Clearly, the UCQ $\exists x P(x) \wedge Q(x) \cup \exists x P(x) \wedge R(x)$ fits. Indeed, it can be shown that this UCQ is unique fitting for E. However, there is no fitting CQ for E, as the direct product of the positive examples maps to the first negative example.

We next give characterizations for most-specific fitting UCQs, most-general fitting UCQs, and unique fitting UCQs, in the style of the characterization for CQs provided in Sect. 3. For most-general fitting UCQs, the weak and the strong version turn out to coincide, unlike for CQs.

Proposition 4.2. *(Implicit in [2].) For all collections of labeled examples $E = (E^+, E^-)$ and UCQs q, the following are equivalent:*

(1) q is a strongly most-specific fitting UCQ for E,
(2) q is a weakly most-specific fitting UCQ for E,
(3) q fits E and is homomorphically equivalent to $\bigcup_{e \in E^+} q_e$.

Proposition 4.3. *For all collections of labeled examples $E = (E^+, E^-)$ and UCQs $q = q_1 \cup \cdots \cup q_n$, the following are equivalent:*

(1) q is a strongly most-general fitting UCQ for E,
(2) q is a weakly most-general fitting UCQ for E,
(3) q fits E^+ and $(\{e_{q_1}, \ldots, e_{q_n}\}, E^-)$ is a homomorphism duality.

Proposition 4.4. *For all collections of labeled examples $E = (E^+, E^-)$ and UCQs q, the following are equivalent:*

(1) q is a unique fitting UCQ for E,
(2) q fits E and the pair (E^+, E^-) is a homomorphism duality,
(3) q is homomorphically equivalent to $\bigcup_{e \in E^+} q_e$ and (E^+, E^-) is a homomorphism duality.

Based on these characterizations we obtain:

Theorem 4.5.

(1) The existence problem for fitting UCQs (equivalently, for most-specific fitting UCQs) is coNP-complete; if a fitting UCQ exists, a most-specific fitting UCQ can be computed in PTime.
(2) The existence problem for most-general fitting UCQs is NP-complete; if a most-general fitting UCQ exists, one can be computed in 2ExpTime.
(3) The verification problem for fitting UCQs is DP-complete.
(4) The verification problem for most-specific fitting UCQs is DP-complete.

In order to state the remaining complexity results, let HomDual be the problem of testing if a given pair (F, D) is a homomorphism duality. The precise complexity of this problem is not known, but we prove the following.

Proposition 4.6. *HomDual is in ExpTime and NP-hard.*

The upper bound in Prop. 4.6 is based on the observation that, in order for (F, D) to be a homomorphism duality, each $e \in F$ must be c-acyclic. For the lower bound, we use an argument that was also used in [30] to show that FO definability of a CSP is NP-hard: we reduce from 3-SAT.

Theorem 4.7. *The following problems are computationally equivalent (via polynomial conjunctive reductions) to HomDual:*

(1) The existence problem for unique fitting UCQs,
(2) The verification problem for unique fitting UCQs,
(3) The verification problem for most-general fitting UCQs.

As we mentioned, there is a PTime-computable "canonical candidate" fitting UCQ, namely $\bigcup_{e \in E^+}(q_e)$. That is, computing a fitting UCQ under the promise that one exists, is in PTime. While this can be viewed as a positive result, it is somewhat disappointing as the UCQ in question does nothing more than enumerate the positive examples. It does not "compress" or "generalize from" the input examples in a meaningful way. This, it turns out, is unavoidable:

Theorem 4.8 ([13]). *There does* not *exist an "efficient Occam algorithm" for UCQs, i.e., a PTime algorithm taking as input a collection of labeled examples E for which a fitting UCQ is promised to exists and producing a fitting UCQ of size $O(m)^\alpha \cdot poly(n)$ where m is the size of the input, n is the size of the smallest fitting UCQ, and $\alpha < 1$.*

5 THE CASE OF TREE CQS

We now consider *tree CQs*, that is, unary CQs that are acyclic and connected. Moreover, we concentrate on schemas that consist only of unary and binary relations, which we refer to as *binary* schemas. Note that this type of schema is at the core of prominent web data formalisms such as RDF and OWL. Apart from being natural per se, the class of tree CQs holds significance as it correspond to concept expressions in the description logic $\mathcal{ELI}$. Table 3 summarizes our complexity results on tree CQs.

As we shift from unrestricted CQs to tree CQs, simulations take on the role of homomorphisms. In addition, unraveling a CQ or an instance into a (finite or infinite) tree turns out to be a central operation.

Simulation. Given two instances I, J over the same binary schema, a *simulation of I in J* is a relation $S \subseteq \mathsf{adom}(I) \times \mathsf{adom}(J)$ that satisfies the following properties:

(1) if $A(a) \in I$ and $(a, a') \in S$, then $A(a') \in J$;
(2) if $R(a, b) \in I$ and $(a, a') \in S$, then there is an $R(a', b') \in J$ with $(b, b') \in S$.
(3) if $R(a, b) \in I$ and $(b, b') \in S$, then there is an $R(a', b') \in J$ with $(a, a') \in S$.

We write $(I, a) \preceq (J, b)$ if there exists a simulation S of I in J with $(a, b) \in S$. A simulation of a tree CQ $q(x)$ in an instance I is a simulation of I_q in I, and we write $q \preceq (I, a)$ as shorthand for $(I_q, x) \preceq (I, a)$, and likewise for $(I, a) \preceq q$. It is well-known that if I is a tree, then $(I, a) \preceq (J, b)$ iff there is a homomorphism h from I to J with $h(a) = b$. We thus have the following.

Lemma 5.1. *For all instances I, $a \in adom(I)$, and tree CQs $q(x)$, $I \models q(a)$ iff $q \preceq (I, a)$.*

Unraveling. Let I be an instance over a binary schema. A *role* is a binary relation symbol R or its *converse* R^-. For an instance I, we may write $R^-(a, b) \in I$ to mean $R(b, a) \in I$. A *path* in I is a sequence $p = a_1 R_1 \cdots R_{k-1} a_k$, $k \geq 1$, where $a_1, \ldots, a_k \in \mathsf{adom}(I)$ and $R_1, \ldots, R_{k-1}$ are roles such that $R_i(a_i, a_{i+1}) \in I$ for $1 \leq i < k$. We say that p is of *length* k, *starts* at a_1 and *ends* at a_k. The *unraveling of I at $a \in \mathsf{adom}(I)$* is the instance U with active domain $\mathsf{adom}(U)$ that consists of all paths starting at a. It contains the fact

- $A(p)$ for every path $p \in \mathsf{adom}(U)$ that ends with some $b \in \mathsf{adom}(I)$ such that $A(b) \in I$, and
- $R(p, pRb)$ for every path $pRb \in \mathsf{adom}(U)$.

For all $m \geq 1$, the *m-finite unraveling of I at a* is the (finite) restriction of I to all paths of length at most m.

5.1 Arbitrary Fitting Tree CQs

By Lem. 5.1, we can decide verification of arbitrary fitting tree CQs by checking the (non-)existence of simulations to positive and negative examples. Since the existence of simulations can be decided in PTime [25], we obtain the following.

Theorem 5.2. *The verification problem for fitting tree CQs is in PTime.*

The following was proved in [21] in the setting of the description logic $\mathcal{ELI}$, see Thm. 12 of that paper and its proof.

Theorem 5.3 ([21]). *The existence problem for fitting tree CQs is ExpTime-complete. The lower bound already holds for a fixed schema.*

We actually reprove the upper bound in Thm. 5.3 using an approach based on (two-way alternating) tree automata. We gain from this the following result regarding the size of fitting tree CQs.

Theorem 5.4. *If any tree CQ fits a collection of labeled examples $E = (E^+, E^-)$, then we can produce a DAG representation of a fitting tree CQ with a minimal number of variables in single exponential time and the size of such a tree CQ is at most double exponential.*

A matching lower bound is given in Sect. 5.5.

5.2 Most-Specific Fitting Tree CQs

We may define a strong and a weak version of most-specific fitting tree CQs, in analogy with Def. 3.4. We then observe the following counterpart of Prop. 3.5.

Proposition 5.5. *For all tree CQs q and collections of labeled examples $E = (E^+, E^-)$, the following are equivalent:*

(1) q is a weakly most-specific fitting for E,
(2) q is a strongly most-specific fitting for E,
(3) q fits E and $\Pi_{e \in E^+}(e) \preceq q$.

We thus simply speak of *most-specific* fittings tree CQs. Unlike in the non-tree case, the existence of most-specific fitting tree CQs does not coincide with the existence of arbitrary fitting tree CQs.

Example 5.6. Consider the single positive example $R(a, a)$ and the empty set of negative examples. Then $q(x) \text{ :- } R(x, y)$ is a fitting tree CQ and $p(x) \text{ :- } R(x, x)$ is a most-specific fitting CQ, but there is no most-specific fitting tree CQ. In fact, any m-finite unraveling of $p(x)$ is a fitting tree CQ and there is no (finite) fitting tree CQ that is more specific than all these.

In [28] it is shown that verification and existence of a most-specific fitting tree CQ are in ExpTime and PSpace-hard when there are only positive examples, but no negative examples. We extend the upper bounds to the case with negative examples.

Theorem 5.7. *Verification and existence of most-specific fitting tree CQs is in ExpTime.*

The upper bound for verification follows from Prop. 5.5, and the upper bound for existence follows from Prop. 5.5 and the results in [28]. However, we reprove the latter using tree automata to show the following.

Theorem 5.8. *If a collection of labeled examples $E = (E^+, E^-)$ admits a most-specific tree CQ fitting, then we can construct a DAG representation of such a fitting with a minimal number of variables in single exponential time and the size of such a tree CQ is at most double exponential.*

The proof of Thm. 5.8 comes with a characterization of most-specific fitting tree CQs in terms of certain initial pieces of the unraveling of $\prod_{e \in E^+}(e)$.

5.3 Weakly Most-General and Unique Fitting Tree CQs

We define weakly and strongly most-general tree CQs in the obvious way and likewise for bases of most-general fitting tree CQs and unique fitting tree CQs, see Def. 3.8 and 3.9. The following example illustrates that the existence of weakly most-general tree CQs does not coincide with the existence of weakly most-general CQs.

Example 5.9. Let $(E^+ = \emptyset, E^- = \{\{P(a_0)\}, \{R(a_0, a_0)\}\}$. Then there are no weakly most-general fitting tree CQs. To see this, let $q(x)$ be a tree CQ that fits the examples. Clearly, $q(x)$ must contain both an R-atom and a P atom. Let n be the shortest distance, in the graph of q, from x to some y that satisfies P, and let π be the path from x to y, written as a sequence of roles R and R^-. If π is empty, then the query $R(x, y) \wedge R(y, x) \wedge P(x)$ is homomorphically strictly weaker than q, but still fits. If π is non-empty, then the query $x(\pi; \pi^-; \pi)y \wedge P(y)$ is homomorphically weaker than q, but fits. Thus q is not weakly most-general.

However, weakly most-general fitting CQs exist that are not tree CQs. In fact, we obtain a complete basis of most-general fitting CQs of size 3 by taking the CQs $q(x) \text{ :- } \exists yzu(\alpha(x) \wedge R(y, z) \wedge P(u))$ where $\alpha(x)$ is $P(x)$ or $\exists v R(x, v)$ or $\exists v R(v, x)$, serving purely to make the CQ safe.

As in the case of unrestricted CQs, we may characterize weakly most-general fitting tree CQs using frontiers. The following is an immediate consequence of the definition of frontiers.

Proposition 5.10. *The following are equivalent for all collections of labeled examples $E = (E^+, E^-)$ and tree CQs q:*

(1) q is a weakly most-general fitting for E,
(2) q fits E and every element of the frontier for q w.r.t. tree CQs simulates to an example in E^-.

As every tree CQ is c-acyclic, it has a frontier that can be computed in polynomial time. We have the choice of using the same frontier construction as in the proofs for Sect. 3.3 or one that is tailored towards trees and 'only' yields a frontier w.r.t. tree CQs [11]. Both constructions need only polynomial time and, together with Prop. 5.10, yield a PTime upper bound for the verification problem.

Theorem 5.11. *Verification of weakly most-general fitting tree CQs is in PTime.*

For the existence problem, we choose the frontier construction from [11] and then again use an approach based on tree automata. We also obtain the same results regarding the size and computation of weakly most-general fitting tree CQs as in Sect. 5.1 and 5.2.

Theorem 5.12. *Existence of weakly most-general fitting tree CQs is in ExpTime. Moreover, if a collection of labeled examples $E = (E^+, E^-)$ admits a weakly most-general tree CQ fitting, then we can construct a DAG representation of such a fitting with a minimal number of variables in single exponential time and the size of such a tree CQ is at most double exponential.*

For uniquely fitting tree CQs, we observe that a fitting tree CQ is a unique fitting iff it is both a most-specific and a weakly most-general fitting. This immediately gives an ExpTime upper bound for verification, from the ExpTime upper bounds for verifying most-specific and weakly most-general tree CQs. We obtain an ExpTime upper bound for the existence of uniquely fitting tree CQs by combining the automata constructions for these two cases.

Theorem 5.13. *Verification and existence of unique fitting tree CQs is in ExpTime.*

We remark that Thm. 5.4 clearly also applies to unique fitting tree CQs: if a unique fitting tree CQ exists, then the algorithm from the proof of Thm. 5.4 must compute it.

5.4 Bases of Most General Fitting Tree CQs

In Sect. 3, we have characterized bases of most-general fitting CQs in terms of relativized homomorphism dualities. Here, we do the same for tree CQs, using simulation dualities instead.

Definition 5.14 (Relativized simulation dualities). A pair of finite sets of data examples (F, D) forms a *simulation duality* if, for all data examples e, the following are equivalent:

(1) $e \preceq e'$ for some $e \in D$,
(2) $e' \not\preceq e$ for all $e' \in F$.

We say that (F, D) forms a simulation duality *relative to a data example p* if the above conditions hold for all e with $e \preceq p$.

Proposition 5.15. *For all collections of labeled examples $E = (E^+, E^-)$, the following are equivalent, for $p = \Pi_{e \in E^+}(e)$:*

(1) $\{q_1, \ldots, q_n\}$ is a basis of most-general fitting tree CQs for E,
(2) each q_i fits E and $(\{q_1, \ldots, q_n\}, E^-)$ is a simulation duality relative to p.

We use Prop. 5.15 and the fact any homomorphism duality (F, D) where F consists only of trees is also a simulation duality to show:

Theorem 5.16. *The verification problem for bases of most-general fitting tree CQs is in ExpTime.*

For the existence problem, we use the following characterization: let D be a finite collection of data examples. A tree CQ q is a *critical tree obstruction* for D if $q \not\preceq e$ for all $e \in D$ and every tree CQ q' that can be obtained from q by removing subtrees satisfies $q' \preceq e$ for some $e \in D$.

Proposition 5.17. *Let D be a finite set of data examples and $\widehat{e}$ a data example. Then the following are equivalent:*

(1) there is a finite set of tree data examples F such that (F, D) is a simulation duality relative to $\widehat{e}$,
(2) there is a finite number of critical tree obstructions q for D that satisfy $q \to \widehat{e}$ (up to isomorphism).

We use Prop. 5.2 to provide a reduction to the infinity problem for tree automata. This also yields bounds for the construction and size of bases of most-general fitting tree CQs

Theorem 5.18. *The existence problem for bases of most-general fitting tree CQs is in ExpTime. Moreover, if a collection of labeled examples E has a basis of most-general fitting tree CQs, then it has such a basis in which every tree CQ has size at most double exponential in $||E||$.*

5.5 Lower Bounds

All the complexity upper bounds stated for tree CQs above are tight. We establish matching ExpTime lower bounds by a polynomial time reduction from the product simulation problem into trees (with one exception).

Product Simulation Problem Into Trees. The *product simulation problem* asks, for finite pointed instances $(I_1, a_1), \ldots, (I_n, a_n)$ and (J, b), whether $\Pi_{1 \leq i \leq n}(I_i, a_i) \preceq (J, b)$. A variant of this problem was shown to be ExpTime-hard in [24] where simulations are replaced with $\downarrow$-simulations, meaning that the third condition of simulations is dropped, and certain transition systems are used in place of instances. This result was adapted to database instances in [21]. Here, we consider instead the *product simulation problem into trees* where the target instance J is required to be a tree (and full simulations are used in place of $\downarrow$-simulations). We prove ExpTime-hardness by a non-trivial reduction from the $\downarrow$-simulation problem studied in [21].

Theorem 5.19. *The product simulation problem into trees is* ExpTime*-hard, even for a fixed schema.*

This improves a PSpace lower bound from [28] where, however, *all* involved instances were required to be trees. It is easy to prove an ExpTime upper bound by computing the product and then deciding the existence of a simulation in polynomial time [25].

Theorem 5.20. *The verification problem and the existence problem are ExpTime-hard for:*

(1) weakly most-general fitting tree CQs,
(2) most-specific fitting tree CQs,
(3) unique fitting tree CQs, and
(4) complete bases of most-general fitting tree CQs.

Each problem is ExpTime-hard already for a fixed schema and arity, and, in the case of the verification problems, when restricted to inputs where the input CQ fits the examples, or, in the case of the existence problems, when restricted to inputs where a fitting CQ exists.

Points (2) to (4) of Thm. 5.20 are proved by reductions from the product simulation problem into trees. Point (1) is proved simultaneously with Thm. 3.24 by adapting a reduction from the word problem for alternating Turing machines used in [24].

We also establish a double exponential lower bound on the size of (arbitrary) fitting tree CQs.

Theorem 5.21. *For all $n \geq 0$, there is a collection of labeled examples of combined size polynomial in n such that a fitting tree CQ exists and the size of every fitting tree CQ is at least 2^{2^n}. This even holds for a fixed schema.*

We do not currently have a similar lower bound for any of the other types of fitting tree CQs listed in Table 3.

6 CONCLUSION

The characterizations and complexity results we presented, we believe, give a fairly complete picture of extremal fitting problems for CQs, UCQs, and tree CQs. Similar studies could be performed, of course, for other query and specification languages (e.g., graph database queries, schema mappings). In particular, the problem of computing fitting queries has received considerable interest in knowledge representation, where, additionally, background knowledge in the form of an ontology is considered. The existence of a fitting $\mathcal{ELI}$ concept (corresponding to a tree CQ) is undecidable in the presence of an $\mathcal{ELI}$ ontology [21], but there are more restricted settings, involving e.g. $\mathcal{EL}$ concept queries, that are decidable and have received considerable interest [9, 21, 31].

Since the non-extremal fitting problem for CQs is already coNExpTime-complete [10, 37], it is not surprising that many of our complexity bounds are similarly high. In [4], it was shown that the (non-extremal) fitting problem for CQs can be made tractable by a combination of two modifications to the problem: (i) "desynchronization", which effectively means to consider UCQs instead of CQs, and (ii) replacing homomorphism tests by k-consistency tests, which effectively means to restrict attention to queries of bounded treewidth. Similarly, in our results we also see improved complexity bounds when considering UCQs and tree CQs. While we have not studied unions of tree CQs in this paper, based on results in [4] one may expect that they will exhibit a further reduction in the complexity of fitting. We leave this as future work. Another way to reduce the complexity is to consider size-bounded versions of the fitting problem, an approach that also has learning-related benefits [15].

A question that we have not addressed so far is what to do if an extremal fitting query of interest does not exist. For practical purposes, in such cases (and possibly in general) it may be natural to consider relaxations where the fitting query is required to be, for instance, most-general, only as compared to other queries *on some given (unlabeled) dataset*. It is easy to see that, under this relaxation, a basis of most-general fitting queries always exists.

It would also be interesting to extend our extremal fitting analysis to allow for approximate fitting, for instance using a threshold based approach as in [5] or an optimization-based approach as in [16, 23].

ACKNOWLEDGMENTS

Balder ten Cate is supported by the European Union's Horizon 2020 research and innovation programme (MSCA-101031081), Victor Dalmau is supported by the MiCin (PID2019-109137GB-C/AEI/10.13039/501100011033), and Carsten Lutz by the DFG Collaborative Research Center 1320 EASE.

REFERENCES

[1] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. 2011. Characterizing Schema Mappings via Data Examples. *ACM Trans. Database Syst.* 36, 4 (2011), 23:1–23:48. https://doi.org/10.1145/2043652.2043656

[2] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. 2011. Designing and Refining Schema Mappings via Data Examples. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1989323.1989338

[3] Marcelo Arenas, Gonzalo I. Diaz, and Egor V. Kostylev. 2016. Reverse Engineering SPARQL Queries. In *Proceedings of the 25th International Conference on World Wide Web* (Montréal, Québec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 239–249. https://doi.org/10.1145/2872427.2882989

[4] Pablo Barceló and Miguel Romero. 2017. The Complexity of Reverse Engineering Problems for Conjunctive Queries. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy (LIPIcs, Vol. 68)*, Michael Benedikt and Giorgio Orsi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:17. https://doi.org/10.4230/LIPIcs.ICDT.2017.7

[5] Pablo Barceló, Alexander Baumgartner, Victor Dalmau, and Benny Kimelfeld. 2021. Regularizing conjunctive features for classification. *J. Comput. System Sci.* 119 (2021), 97–124. https://doi.org/10.1016/j.jcss.2021.01.003

[6] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.* 39, 4 (2014), 33:1–33:44. https://doi.org/10.1145/2661643

[7] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. 2015. Learning Path Queries on Graph Databases. In *EDBT*. 109–120. https://doi.org/10.5441/002/edbt.2015.11

[8] Raimundo Briceño, Andrei Bulatov, Víctor Dalmau, and Benoît Larose. 2021. Dismantlability, connectedness, and mixing in relational structures. *J. of Comb. Theory, Ser. B* 147 (2021), 37–70. https://doi.org/10.1016/j.jctb.2020.10.001

[9] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. 2016. DL-Learner - A framework for inductive learning on the Semantic Web. *J. Web Semant.* 39 (2016), 15–24. https://doi.org/10.1016/j.websem.2016.06.001

[10] Balder ten Cate and Víctor Dalmau. 2015. The Product Homomorphism Problem and Applications. In *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium (LIPIcs, Vol. 31)*, Marcelo Arenas and Martín Ugarte (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 161–176. https://doi.org/10.4230/LIPIcs.ICDT.2015.161

[11] Balder ten Cate and Victor Dalmau. 2022. Conjunctive Queries: Unique Characterizations and Exact Learnability. *ACM Trans. Database Syst.* 47, 4 (2022), 14:1–14:41. https://doi.org/10.1145/3559756

[12] Balder ten Cate, Victor Dalmau, Maurice Funk, and Carsten Lutz. 2022. Extremal Fitting Problems for Conjunctive Queries. https://doi.org/10.48550/arXiv.2206.05080 arXiv:2206.05080 [cs.DB]

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

[14] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. 2022. On the non-efficient PAC learnability of acyclic conjunctive queries. https://doi.org/10.48550/arXiv.2208.10255 arXiv:2208.10255 [cs.DB]

[15] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. 2023. SAT-Based PAC Learning of Description Logic Concepts. Forthcoming.

[16] Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. 2017. Approximation Algorithms for Schema-Mapping Discovery from Data Examples. *ACM Trans. Database Syst.* 42, 2 (2017), 12:1–12:41. https://doi.org/10.1145/3044712

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

[18] Sara Cohen and Yaacov Y. Weiss. 2016. The Complexity of Learning Tree Patterns from Example Graphs. *ACM Trans. Database Syst.* 41, 2 (2016), 14:1–14:44. https://doi.org/10.1145/2890492

[19] Andrew Cropper, Sebastijan Dumančič, Richard Evans, and Stephen H. Muggleton. 2022. Inductive logic programming at 30. *Mach. Learn.* 111, 1 (2022), 147–172. https://doi.org/10.1007/s10994-021-06089-1

[20] Jan Foniok, Jaroslav Nesetril, and Claude Tardif. 2008. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Eur. J. Comb.* 29, 4 (2008), 881–899. https://doi.org/10.1016/j.ejc.2007.11.017

[21] Maurice Funk, Jean Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. 2019. Learning Description Logic Concepts: When can Positive and Negative Examples be Separated?. In *Proceedings of IJCAI 2019*. 1682–1688. https://doi.org/10.24963/ijcai.2019/233

[22] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 1999. On the Complexity of Some Inductive Logic Programming Problems. *New Generation Comput.* 17, 1 (1999), 53–75. https://doi.org/10.1007/BF03037582

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

[24] David Harel, Orna Kupferman, and Moshe Y. Vardi. 2002. On the Complexity of Verifying Concurrent Transition Systems. *Inf. Comput.* 173, 2 (2002), 143–161. https://doi.org/10.1006/inco.2001.2920

[25] Monika Rauch Henzinger, Thomas A. Henzinger, and Peter W. Kopke. 1995. Computing Simulations on Finite and Infinite Graphs. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*. IEEE Computer Society, 453–462. https://doi.org/10.1109/SFCS.1995.492576

[26] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. 2020. Logical Separability of Incomplete Data under Ontologies. In *Proceedings of KR 2020*, D. Calvanese, E. Erdem, and M. Thielscher (Eds.). 517–528. https://doi.org/10.24963/kr.2020/52

[27] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. 2021. Separating Data Examples by Description Logic Concepts with Restricted Signatures. In *Proceedings of KR 2021*, M. Bienvenu, G. Lakemeyer, and E. Erdem (Eds.). 390–399. https://doi.org/10.24963/kr.2021/37

[28] Jean Christoph Jung, Carsten Lutz, and Frank Wolter. 2020. Least General Generalizations in Description Logic: Verification and Existence. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*. 2854–2861. https://doi.org/10.1609/aaai.v34i03.5675

[29] Benny Kimelfeld and Christopher Ré. 2018. A Relational Framework for Classifier Engineering. *ACM SIGMOD Record* 47 (2018), 6–13. https://doi.org/10.1145/3277006.3277009

[30] Benoit Larose, Cynthia Loten, and Claude Tardif. 2007. A Characterisation of First-Order Constraint Satisfaction Problems. *Log. Methods Comput. Sci.* 3, 4 (2007). https://doi.org/10.2168/LMCS-3(4:6)2007

[31] Jens Lehmann and Christoph Haase. 2009. Ideal Downward Refinement in the $\mathcal{EL}$ Description Logic. In *Inductive Logic Programming, 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers (Lecture Notes in Computer Science, Vol. 5989)*, Luc De Raedt (Ed.). Springer, 73–87. https://doi.org/10.1007/978-3-642-13840-9_8

[32] Jens Lehmann and Pascal Hitzler. 2010. Concept learning in description logics using refinement operators. *Mach. Learn.* 78, 1-2 (2010), 203–250. https://doi.org/10.1007/s10994-009-5146-2

[33] Hao Li, Chee-Yong Chan, and David Maier. 2015. Query from Examples: An Iterative, Data-Driven Approach to Query Construction. *Proc. VLDB Endow.* 8, 13 (2015), 2158–2169. https://doi.org/10.14778/2831360.2831369

[34] Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, New York.

[35] Giuseppe Rizzo, Nicola Fanizzi, and Claudia d'Amato. 2020. Class expression induction as concept space exploration: From DL-FOIL to DL-FOCL. *Future Gener. Comput. Syst.* 108 (2020), 256–272. https://doi.org/10.1016/j.future.2020.02.071

[36] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. 2014. Query reverse engineering. *The VLDB Journal* 23, 5 (2014), 721–746. https://doi.org/10.1007/s00778-013-0349-3

[37] Ross Willard. 2010. Testing Expressibility Is Hard. In *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6308)*, David Cohen (Ed.). Springer, 9–23. https://doi.org/10.1007/978-3-642-15396-9_4