Semantic Optimization in Tractable Classes of Conjunctive Queries

Citation for published version:
Barceló, P, Pieris, A & Romero, M 2017, 'Semantic Optimization in Tractable Classes of Conjunctive Queries' ACM SIGMOD Record, vol. 46, no. 2, pp. 5-17. DOI: 10.1145/3137586.3137588

Digital Object Identifier (DOI):
10.1145/3137586.3137588

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
ACM SIGMOD Record

Semantic Optimization in Tractable Classes of Conjunctive Queries[*]

Pablo Barceló
Center for Semantic Web Research &
DCC, University of Chile
pbarcelo@dcc.uchile.cl

Andreas Pieris
School of Informatics
University of Edinburgh
apieris@inf.ed.ac.uk

Miguel Romero
Center for Semantic Web Research &
DCC, University of Chile
mromero@dcc.uchile.cl

ABSTRACT

This paper reports on recent advances in semantic query optimization. We focus on the core class of conjunctive queries (CQs). Since CQ evaluation is NP-complete, a long line of research has concentrated on identifying fragments of CQs that can be efficiently evaluated. One of the most general such restrictions corresponds to bounded generalized hypertreewidth, which extends the notion of acyclicity. Here we discuss the problem of reformulating a CQ into one of bounded generalized hypertreewidth. Furthermore, we study whether knowing that such a reformulation exists alleviates the cost of CQ evaluation. In case a CQ cannot be reformulated as one of bounded generalized hypertreewidth, we discuss how it can be approximated in an optimal way. All the above issues are examined both for the constraint-free case, and the case where constraints, in fact, tuple-generating and equality-generating dependencies, are present.

1. INTRODUCTION

Conjunctive queries (CQs) are one of the most fundamental classes of database queries (see, e.g., [2, 17, 33, 41, 49]). In particular, CQs correspond to select-project-join queries in relational algebra and to select-from-where queries in SQL. However, CQ evaluation is not an easy task, especially over large volumes of data. This has led to a flurry of activity for developing heuristics that alleviate CQ evaluation in practice.

One important method of this kind is CQ optimization. Recall that query optimization is a basic database task that amounts to transforming a query into one that is more efficient to evaluate. The database theory community has developed several principled methods for optimization of CQs, many of which are based on *static-analysis* tasks such as containment [2]. In a nutshell, such methods compute a *minimal* equivalent version of a CQ, where minimality refers to the number of atoms. As argued by Abiteboul, Hull, and Vianu [2], this provides a theoretical notion of "true optimality" for the reformulation of a CQ, as opposed to practical considerations based on heuristics. For each CQ q, the minimal equivalent CQ is its *core* q' [39]. Although the static analysis tasks that support CQ minimization are NP-complete [17], this is not a major problem for most real-life applications, as the input (the CQ) is small.

It is known, on the other hand, that semantic information about the data, in the form of constraints, can be used to enrich query optimization by guiding the query transformation process. This is often referred to as *semantic query optimization* [15]. In the aforementioned analysis of CQ minimization, however, constraints play no role, as CQ equivalence is defined over *all* databases. Adding constraints yields a refined notion of CQ equivalence, which holds over those databases that satisfy a given set of constraints only. Minimization of CQs under this refined notion thus provides a principled approach to semantic query optimization [22].

An important shortcoming of the results on CQ minimization (under constraints) mentioned above is that there is no theoretical guarantee that the minimized version of a CQ is in fact easier to evaluate (recall that, in general, CQ evaluation is NP-complete [17]). We know, on the other hand, quite a bit about classes of CQs which can be evaluated in polynomial time: These are the ones that admit a suitable *tree decomposition of small width* [18, 20, 34, 37]. Our proposal is to study the fundamental problem of semantic optimization in such tractable classes of CQs; i.e., whether a set of constraints can be used to reformulate a CQ as one of small width, and if so, what is the cost of computing and evaluating such a reformulation. Following Abiteboul et al., this would provide us with a theoretical guarantee of "true efficiency" for those reformulations.

Due to its relative importance in the database literature and mature theoretical status, we concentrate on the classes of CQs of bounded generalized hypertreewidth (see [32] for a recent survey). In particular,

[*]Part of this work was done while Romero was visiting the Simons Institute for the Theory of Computing. Barceló and Romero are funded by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. Pieris is supported by the EPSRC Programme Grant EP/M025268/.

CQs of generalized hypertreewidth one correspond to the oldest and most studied tractability condition for CQs: *acyclicity* [49]. In terms of constraints, we consider *tuple-generating dependecies* (tgds) and *equality-generating dependencies* (egds), which subsume all real-life database constraints; in particular, tgds extend *inclusion dependencies*, while egds extend *functional dependencies*. Tgds and egds are widely applied in data integration [42], data exchange [24], and ontology-based data access [12], as a tool for expressing rich semantic constraints among different relations in the database. Due to this fact, they can be used to enhance the semantic optimization process studied here.

In this survey, we present several recent results that serve as a theoretical framework for the problem of how to obtain maximal benefit from semantic optimization in classes of CQs of bounded generalized hypertreewidth. We focus on the following questions:

1. For which classes of constraints is the reformulation problem decidable? Also, in these decidable classes, what is the complexity of the problem?

2. How semantic optimization in tractable classes of CQs can be used to tackle the ultimate problem of evaluating CQs more efficiently?

Potential impact. While most of the CQs encountered in practical situations are of low hypertreewidth [32], the work presented here is relevant due to the following:

1. Evaluating a CQ q of generalized hypertreewidth k, for $k \geq 1$, over a database $\mathcal{D}$ takes time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [34]. Even if k is small, say $k = 3$, finding an equivalent CQ q' of smaller hypertreewidth might improve the complexity of evaluation (especially when the database $\mathcal{D}$ is large).

2. Assume that q is of hypertreewidth $k > 1$. In the case that an equivalent CQ of hypertreewidth $k' < k$ cannot be found, the tools presented here provide an *approximation* of q of hypertreewidth k'. Such an approximation is a CQ q' of hypertreewidth k' that is *maximally contained* in q. That is, q' returns sound (but not necessarily complete) answers to q over those databases that satisfy the constraints, and there is no CQ q'' of hypertreewidth k' that gets "closer" to q than q'. Evaluating such an approximation q' might be convenient for obtaining quick and sound answers to q when exact evaluation is infeasible or is taking too long.

CQs of bounded hypertreewidth require specialized algorithms for their implementation. Although these specialized algorithms are not implemented in commercial database systems, there is currently an important body of research trying to integrate them into some query optimizers; see, e.g., [1, 3, 5, 29, 30]. This suggests that semantic optimization techniques based on hypertreewidth have the potential to provide new query optimization techniques of practical impact.

Organization. Preliminaries are in Section 2. In Section 3, we study when a CQ can be reformulated as one of bounded generalized hypertreewidth in the absence of constraints, and how such a reformulation helps query evaluation. The extension of such an investigation to the case where constraints are available is considered in the next two sections: Section 4 deals with reformulation and Section 5 with evaluation. Approximations are studied in Section 6. Final remarks are in Section 7.

2. PRELIMINARIES

Databases. A *schema* is a finite set of relation symbols, each one of which has an associated arity $n > 0$. A *database* $\mathcal{D}$ over a schema σ is a finite set of atoms of the form $R(\bar{a})$, where R is a relation symbol in σ of arity $n > 0$ and $\bar{a}$ is an n-ary tuple of constants. We write $\mathcal{D}$ also for the set of elements mentioned in $\mathcal{D}$.

Conjunctive queries. A *conjunctive query* (CQ) q over a schema σ is a rule of the form:

$$\mathsf{Ans}(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m), \qquad (1)$$

such that (a) each $R_i(\bar{x}_i)$ is an atom over σ, for $1 \leq i \leq m$, (b) $\bar{x}$ is a sequence of variables taken from the $\bar{x}_i$'s, and (c) Ans is a distinguished relation symbol that represents the answer of q. We write $q(\bar{x})$ to denote that $\bar{x}$ is the sequence of variables that appear in such an answer.

As usual, the evaluation of a CQ q of the form (1) over a database $\mathcal{D}$ is obtained by computing the join of the atoms in the set $\{R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)\}$, and then projecting only the variables $\bar{x}$ in the answer of q. For the purposes of this paper, it is convenient to formally define this in terms of the notion of *homomorphism* from q to $\mathcal{D}$. Recall that these are the mappings h from the set of variables in q to the elements of $\mathcal{D}$ such that $R_i(h(\bar{x}_i)) \in \mathcal{D}$ for each $1 \leq i \leq m$. The *evaluation of* $q(\bar{x})$ *over* $\mathcal{D}$, denoted $q(\mathcal{D})$, consists then of those tuples $h(\bar{x})$ such that h is a homomorphism from q to $\mathcal{D}$.

Example 1. We describe a social network using a schema σ with two binary relation symbols: $\mathsf{Friends}$ and Likes. The first one establishes when two persons are friends, while the second one establishes when a person likes a post. Suppose that we want to retrieve all the pairs of mutual friends that have some post they like in common. We can express this as a CQ $q(x, y)$ defined as $\mathsf{Ans}(x, y) \leftarrow \mathsf{Friends}(x, y), \mathsf{Likes}(x, z), \mathsf{Likes}(y, z)$. □

Tractable classes of CQs. The *evaluation problem* for CQs is defined as follows: Given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ of constants in $\mathcal{D}$, check if $\bar{t} \in q(\mathcal{D})$. This problem is NP-complete [17], but becomes tractable for several syntactically defined subclasses of CQs. The oldest such tractability condition corresponds to *acyclicity* [10, 31]. Intuitively, a CQ q is acyclic if its atoms can be arranged in the form of a tree while preserving the connectivity of its variables. More formally, q is acyclic if it admits a *join tree*, that is, a tree T whose nodes are the atoms of q, and for each variable x that appears in q it is the case that the set of nodes in which x is mentioned defines a connected subtree of T. We denote by AC the class of acyclic CQs. Yannakakis's seminal work established that AC defines a tractable class of CQs; in fact, the queries in this class can be evaluated in linear time $O(|\mathcal{D}| \cdot |q|)$, where $|\mathcal{D}|$ and $|q|$ are the size of the database $\mathcal{D}$ and the CQ q, respectively [49].

Example 2. The CQ in Example 1 is not acyclic. In any way we arrange its atoms as the nodes of a tree, we will lose the connectivity for at least one variable. On the other hand, the CQ

$$\mathsf{Ans}(x, y) \leftarrow \mathsf{Friends}(x, y), \mathsf{Likes}(x, z), \mathsf{Likes}(y, z'),$$

which is obtained from q by "breaking" the join on variable z, is in AC. This is witnessed by the join tree whose root is the atom $\mathsf{Friends}(x, y)$, and the atoms $\mathsf{Likes}(x, z)$ and $\mathsf{Likes}(y, z')$ are its children. Such CQ retrieves pairs of mutual friends each one of which likes some post. □

It has been observed, however, that a significant proportion of the CQs that appear in practice are not acyclic, but are in some sense *mildly acyclic* (see, e.g., [32]). This motivated the search for notions that represent the degree of acyclicity of a CQ, and for efficient evaluation algorithms for CQs with low degree of acyclicity. The degree of acyclicity of a CQ in this context is traditionally known as its *width*. Such width can be defined by using the notion of *generalized hypertree decomposition*, which extends the notion of join tree by allowing each node of the tree to be associated with several atoms of the query. The formal definition follows.

A generalized hypertree decomposition of a CQ $q := \mathsf{Ans}(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ is a tuple (T, λ, χ), where T is a tree, λ is a mapping that assigns a subset of the variables in q to each node t of T, and χ is a mapping that assigns a subset of the atoms $\{R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)\}$ to each node t of T, such that:

1. For each $1 \leq i \leq m$, the variables in $\bar{x}_i$ are contained in $\lambda(t)$, for some $t \in T$.

2. For each variable x in q, the set of nodes t of T for which x occurs in $\lambda(t)$ is connected.

3. For each $t \in T$, the variables in $\lambda(t)$ are "covered" by the atoms in $\chi(t)$; i.e., $\lambda(t) \subseteq \bigcup_{R_i(\bar{x}_i) \in \chi(t)} \bar{x}_i$.

For a generalized hypertree decomposition (T, λ, χ), its *width* is defined as the maximal size of a set of the form $\chi(t)$ over all nodes t of T. The *generalized hypertreewidth* of a CQ q is the minimum width over all generalized hypertree decompositions of q. We denote by $\mathsf{GHW}(k)$, for $k \geq 1$, the set of CQs of generalized hypertreewidth bounded by k. The notion of bounded generalized hypertreewidth subsumes acyclicity; in particular, $\mathsf{AC} = \mathsf{GHW}(1)$ [34].

Example 3. Recall that the CQ in Example 1 is not acyclic, i.e., is not in $\mathsf{GHW}(1) = \mathsf{AC}$. It is, however, in $\mathsf{GHW}(2)$. The generalized hypertree decomposition of width two that witnesses this fact has only one node t such that $\lambda(t) = \{x, y, z\}$ and $\chi(t) = \{\mathsf{Likes}(x, z), \mathsf{Likes}(y, z)\}$. □

It can be shown, by using tools based on the *existential pebble game* [19], that CQs of bounded generalized hypertreewidth can be evaluated in polynomial time.

THEOREM 1. *Fix $k \geq 1$. The evaluation problem for the CQs in $\mathsf{GHW}(k)$ can be solved in polynomial time.*

At this point, it should be stressed that in the above theorem we assume that the input query already falls in $\mathsf{GHW}(k)$, for some fixed $k \geq 1$. However, one can claim that this is not a realistic assumption. In general, the input query is an arbitrary CQ for which we do not know a priori whether it falls in $\mathsf{GHW}(k)$. In this case, we should first check whether it belongs to $\mathsf{GHW}(k)$, and, if this is the case, then proceed with the actual evaluation. This brings us to the *recognizability problem for* $\mathsf{GHW}(k)$, that is, checking if a given CQ q is in $\mathsf{GHW}(k)$. It is known that for $k = 1$ (i.e., for acyclic queries), the above problem can be solved in linear time [48]. However, for $k > 1$, it becomes NP-complete [27]. This implies that, given a CQ q, we can check in time $2^{|q|^c}$, for some integer $c \geq 1$, whether q belongs to $\mathsf{GHW}(k)$, and, if this is the case, then a generalized hypertree decomposition of q of width k is constructed. Now, having such a hypertree decomposition in place, we can evaluate q over the input database $\mathcal{D}$ in time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [34].

Summing up, there is an integer $c \geq 1$ such that, given a CQ q, a database $\mathcal{D}$, and a tuple of constants $\bar{t}$, checking if q belongs to $\mathsf{GHW}(k)$, and, if this is the case, then check if $\bar{t} \in q(\mathcal{D})$, can be carried out in time:

$$2^{|q|^c} + O(|\mathcal{D}|^{k+1} \cdot |q|).$$

The fact that we spend exponential time in the size of the query for checking whether q belongs to $\mathsf{GHW}(k)$ is not a big practical drawback since this check corresponds to a static analysis task, i.e., it only depends on the size of the "small" CQ. For such tasks, a single-exponential time procedure is considered to be acceptable, and it is actually the norm in many cases including database and verification problems; see, e.g., [2, 45, 47].

A restriction of the notion of generalized hypertreewidth, which ensures tractability of recognizition, known as *hypertreewidth*, has been also studied in the literature [32, 34]. However, due to the nature of the problems that we consider here, it is convenient to use the less restrictive notion of generalized hypertreewidth, even at the extra cost of recognition.

CQ containment and equivalence. Two notions that are crucial for query optimization purposes are CQ containment and equivalence. Let q and q' be CQs. Then, q is *contained* in q', denoted $q \subseteq q'$, if $q(\mathcal{D}) \subseteq q'(\mathcal{D})$, for every database $\mathcal{D}$. Further, q is *equivalent* to q', denoted $q \equiv q'$, if $q \subseteq q'$ and $q' \subseteq q$ (or, equivalently, if they return the same answers over every database, i.e., $q(\mathcal{D}) = q'(\mathcal{D})$ for each database $\mathcal{D}$). Interestingly, containment is polynomially equivalent to CQ evaluation. Given a CQ q, let $\mathcal{D}_q$ be the so-called *canonical database* of q obtained from q by replacing each variable x in q with a new constant $c(x)$. Then:

PROPOSITION 2. [17] *Let $q(\bar{x}), q'(\bar{x}')$ be CQs. It holds that $q \subseteq q'$ iff $c(\bar{x}) \in q'(\mathcal{D}_q)$.*

The above proposition implies that CQ containment and equivalence are NP-complete problems [17].

The core of a CQ. In CQ minimization one is interested in finding a minimal CQ (in terms of number of joins) that is equivalent to a given CQ q. Such a minimal equivalent CQ always corresponds to a *core* of q [38]. This is a CQ q' that is obtained by deleting atoms from q and the following hold: (a) $q \equiv q'$, and (b) q is not equivalent to any CQ q'' that is obtained by removing one or more atoms from q'. In other words, a core of q is a minimally equivalent CQ that is obtained by removing atoms from q. Clearly, a core of a CQ q always exists. Moreover, all cores of q are isomorphic [38], and, therefore, we can talk about *the* core of q.

3. REFORMULATION IN THE ABSENCE OF CONSTRAINTS

It is instructive to start our investigation by focussing on semantic optimization in classes of bounded generalized hypertreewidth in the absence of constraints. We concentrate on the following problems:

- **Reformulation:** Fix $k \geq 1$. Given a CQ q, is it the case that it can be reformulated as a CQ that falls in $\mathsf{GHW}(k)$ that yields the same answers over all databases? More formally, is there a CQ q' in $\mathsf{GHW}(k)$ such that $q \equiv q'$?
- **Evaluation:** In case the latter holds, can we efficiently perform query evaluation for q? Notice that this is not a priori obvious: Although we know that the reformulation q' of q can be efficiently evaluated (since it is in $\mathsf{GHW}(k)$), the cost of computing such a reformulation might be prohibitively high.

3.1 Reformulation of CQs in $\mathsf{GHW}(k)$

We denote by $\mathsf{Equiv}(\mathsf{GHW}(k))$ the class of CQs q that are equivalent to some CQ $q' \in \mathsf{GHW}(k)$. In particular, the CQs in $\mathsf{Equiv}(\mathsf{GHW}(1))$ are known as *semantically acyclic* [6, 9]. It is easy to show that semantic acyclicity is incomparable to the notion of bounded generalized hypertreewidth, i.e., for each $k \geq 1$, there is a semantically acyclic CQ that is not in $\mathsf{GHW}(k)$.

Here we study the following problem for $k \geq 1$:

PROBLEM :	$\mathsf{Reformulation}(\mathsf{GHW}(k))$
INPUT :	A CQ q.
QUESTION :	Is q in $\mathsf{Equiv}(\mathsf{GHW}(k))$?

The main tool that we exploit in the investigation of the above problem is a simple characterization of the classes $\mathsf{Equiv}(\mathsf{GHW}(k))$, for $k \geq 1$, in terms of the core:

PROPOSITION 3 (IMPLICIT IN [9]). *Fix $k \geq 1$. For each CQ q the following are equivalent:*

- *$q \in \mathsf{Equiv}(\mathsf{GHW}(k))$.*
- *The core of q is in $\mathsf{GHW}(k)$.*

The upward implication is trivial: if the core q' of q is in $\mathsf{GHW}(k)$, then $q \in \mathsf{Equiv}(\mathsf{GHW}(k))$ since $q \equiv q'$ by definition. The downward implication states that if a CQ q is equivalent to some CQ $q' \in \mathsf{GHW}(k)$, then q' can always be assumed to be the core of q. The proof of this fact for the case $k = 1$ can be found in [9], but the same ideas can be used to show that this holds for any $k \geq 1$. It is worth noticing that similar techniques have also been used to prove analogous results for the notion of *bounded treewidth* [20]. This is yet another tractability-ensuring condition for CQ evaluation, which is particularly well-suited for the case when the arity of the schema is fixed [36].

Since the core of a CQ q is obtained by removing atoms from q, and such a core is equivalent to q, Proposition 3 implies the following small query property:

PROPOSITION 4. *Fix $k \geq 1$ and let q be a CQ. If $q \in \mathsf{Equiv}(\mathsf{GHW}(k))$, then there exists a CQ $q' \in \mathsf{GHW}(k)$, where $|q'| \leq |q|$, such that $q \equiv q'$.*

The above small query property allows us not only to establish that $\mathsf{Reformulation}(\mathsf{GHW}(k))$ is decidable, but also to pinpoint its exact complexity. Given a CQ q, here is a simple procedure that decides whether $q \in$

$\mathsf{Equiv}(\mathsf{GHW}(k))$: Guess a CQ q' of size at most $|q|$, and verify that (a) $q' \in \mathsf{GHW}(k)$, and (b) $q \equiv q'$. Since, as mentioned before, both (a) and (b) can be carried out in NP, we conclude that the whole procedure can be performed in NP. A matching lower bound can be proved using more elaborate techniques [20]. From the above discussion we get that:

THEOREM 5. *For each fixed $k \geq 1$, the problem* $\mathsf{Reformulation}(\mathsf{GHW}(k))$ *is* NP*-complete.*

3.2 Evaluation of CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))$

Does knowing that a CQ q can be reformulated as a CQ q' in $\mathsf{GHW}(k)$ alleviate the cost of query evaluation? As for CQs in $\mathsf{GHW}(k)$, it can be shown, by exploiting techniques based on the existential pebble game [19], that CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))$, for some fixed $k \geq 1$, can be evaluated in polynomial time.

THEOREM 6. *Fix $k \geq 1$. The evaluation of CQs in* $\mathsf{Equiv}(\mathsf{GHW}(k))$ *can be solved in polynomial time.*

In the previous theorem, we assume that the input query falls in $\mathsf{Equiv}(\mathsf{GHW}(k))$, for a fixed $k \geq 1$. However, as already discussed in Section 2 for $\mathsf{GHW}(k)$, this is not a realistic assumption. In a practical context, we should first check whether the input query belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$, and, if this is the case, then proceed with the actual evaluation. From Theorem 5 (and the underlying algorithm) we know that, given a CQ q, we can check in time $2^{|q|^c}$, for some integer $c \geq 1$, whether q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$. Now, if this is the case, then a CQ q' such that $q \equiv q'$ and $|q'| \leq |q|$ that belongs to $\mathsf{GHW}(k)$, and a generalized hypertree decomposition of q' of width k are constructed; actually, q' is the core of q. Having q' and its decomposition in place, we can evaluate it over the input database $\mathcal{D}$ in time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [34]. Summing up:

COROLLARY 7. *Fix $k \geq 1$. There is an integer $c \geq 1$ such that, given a CQ q, a database $\mathcal{D}$, and a tuple of constants $\bar{t}$, checking whether q belongs to* $\mathsf{Equiv}(\mathsf{GHW}(k))$*, and, if this is the case, then check whether $\bar{t} \in q(\mathcal{D})$, can be solved in time:*

$$2^{|q|^c} \; + \; O(|\mathcal{D}|^{k+1} \cdot |q|).$$

This bound simply states that, after a preprocessing step that takes single-exponential time in the size of the CQ q to check whether q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$, an evaluation step that takes polynomial time is performed. While the running time of the procedure is not polynomial in the combined size of the database $\mathcal{D}$ and the CQ q, it can still be considered as efficient for the reasons already explained in Section 2.

4. REFORMULATION IN THE PRESENCE OF CONSTRAINTS

As said above, in the constraint-free case, a CQ q is equivalent to one in $\mathsf{GHW}(k)$ iff its core is in $\mathsf{GHW}(k)$. Hence, the only reason why q is not in $\mathsf{GHW}(k)$ in the first place is because it has not been minimized (since CQ minimization reduces to computing the core). Thus, semantic optimization in $\mathsf{GHW}(k)$ is not really different from usual CQ minimization. The presence of constraints, on the other hand, yields a more interesting notion of semantic optimization. This is because constraints can be applied on CQs to produce $\mathsf{GHW}(k)$ reformulations of them. Let us show this via an example.

Example 4. Consider a database that stores information about customers, records, and musical styles. The relation $\mathsf{Interest}$ contains pairs (c, s) such that the customer c is interested in the style s. The relation Class contains pairs (r, s) such that the record r is of style s. Finally, the relation Owns contains a pair (c, r) when the customer c owns the record r. Consider now a CQ $q(x, y)$ defined as follows:

$$\mathsf{Ans}(x, y) \leftarrow \mathsf{Owns}(x, y), \mathsf{Class}(y, z), \mathsf{Interest}(x, z).$$

The above query asks for pairs (c, r) such that the customer c owns the record r and has expressed interest in at least one of the styles with which r is associated. It can easily be proved that q is the core of itself but it is not acyclic. Therefore, Proposition 3 implies that q is not equivalent to an acyclic CQ (without constraints).

Assume now that the record store keeps a full list of interests for its customers, based on the styles of the records each customer has bought in the past. In other words, the database satisfies the constraint τ defined as:

$$\forall x \forall y \forall z \big(\mathsf{Owns}(x, y), \mathsf{Class}(y, z) \rightarrow \mathsf{Interest}(x, z)\big).$$

Having this information in place, we can reformulate $q(x, y)$ as the following acyclic CQ $q'(x, y)$:

$$\mathsf{Ans}(x, y) \leftarrow \mathsf{Owns}(x, y), \mathsf{Class}(y, z).$$

Notice that q and q' are in fact equivalent over every database that satisfies the constraint τ. □

Before we proceed further, let us introduce the classes of constraints that we consider in this paper, and some basics on CQ equivalence under constraints.

Constraints. We consider the two most important classes of database constraints; namely:

1. *Tuple-generating dependencies* (tgds), i.e., expressions of the form $\forall \bar{x} \forall \bar{y} \big(\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})\big)$, where ϕ and ψ are conjuntions of atoms. Notice that the constraint in Example 4 is a tgd. Tgds

subsume the central class of *inclusion dependencies* (IDs) [23]. For example, assuming that R and P are binary relations, the ID $R[1] \subseteq P[2]$, which simply states that the set of values occurring in the first attribute of R is a subset of the set of values in the second attribute of P, is expressed via the tgd $\forall x \forall y (R(x,y) \rightarrow \exists z \, P(z,x))$.

2. *Equality-generating dependencies* (egds), i.e., expressions of the form $\forall \bar{x}\big(\phi(\bar{x}) \rightarrow y = z\big)$, where ϕ is a conjunction of atoms and y, z are variables in $\bar{x}$. Egds subsume the important classes of *keys* and *functional dependencies* (FDs). For example, assuming that R is a ternary relation, the FD $R : \{1\} \rightarrow \{3\}$, i.e., the first attribute of R functionally determines the third attribute of R, is expressed via the egd $\forall x \forall y \forall z \forall y' \forall z' (R(x,y,z) \wedge R(x,y',z') \rightarrow z = z')$. Notice that FDs that have more than one attribute in the right-hand side, are expressed via a *set* of egds.

A database $\mathcal{D}$ *satisfies* a tgd of the above form if the following holds: for each tuple $(\bar{a}, \bar{b})$ of elements such that all atoms in $\phi(\bar{a}, \bar{b})$ are in $\mathcal{D}$, there is a tuple $\bar{c}$ of elements such that all atoms in $\psi(\bar{a}, \bar{c})$ are in $\mathcal{D}$. Analogously, $\mathcal{D}$ satisfies an egd $\forall \bar{x}(\phi(\bar{x}) \rightarrow y = z)$ if, for each tuple $\bar{a}$ of elements such that all atoms in $\phi(\bar{a})$ are in $\mathcal{D}$, the elements in $\bar{a}$ that correspond to variables y and z are the same. Finally, $\mathcal{D}$ satisfies a set Σ of constraints if it satisfies every tgd and egd in Σ.

CQ equivalence under constraints. In semantic optimization, one uses the information provided by the constraints to replace a given CQ q by an equivalent one that is easier to evaluate. However, in this context the right notion of equivalence corresponds to the one that is measured over those databases that satisfy the constraints only (as we know that our datasets satisfy such constraints). We formalize this as follows. Let q, q' be CQs and Σ a set of constraints. Then *q is equivalent to q' under* Σ, denoted $q \equiv_\Sigma q'$, if and only if $q(\mathcal{D}) = q'(\mathcal{D})$ for each database $\mathcal{D}$ that satisfies Σ. The notion of containment is defined analogously, and we write $q \subseteq_\Sigma q'$.

If Σ consists only of egds, then deciding $q \equiv_\Sigma q'$ remains NP-complete. This is obtained by applying the well-known *chase procedure* [44] on q and q', respectively, and then checking for equivalence of the resulting queries. When applied on a CQ q, the chase procedure *fires* each egd on the canonical database $\mathcal{D}_q$ of q, equating variables if needed in order to restore consistency. The procedure finishes when the resulting database satisfies all egds in Σ.

On the other hand, the situation is more difficult if Σ consists of tgds. Although a characterization based on the chase procedure exists, the result of the chase under a set of tgds is, in general, infinite. Hence, this tool no longer provides a decision procedure. This is not surprising since checking CQ containment/equivalence under arbitrary sets of tgds is undecidable [11]. This negative result has motivated a long search for practical restrictions of the class of tgds with decidable CQ containment (and, thus, equivalence) problems. Such restrictions are often classified into four main paradigms:

1. *Full tgds:* These are tgds without existentially quantified variables, i.e., of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow \psi(\bar{x}))$. In the database literature, such sets are particularly important as they correspond to queries expressible in the widely studied *Datalog* language [2]. CQ containment is decidable for sets of full tgds since the chase always terminates.

2. *Guardedness:* A tgd is *guarded* if its body $\phi(\bar{x}, \bar{y})$ contains an atom, called the *guard*, that contains all the variables in $(\bar{x} \cup \bar{y})$. Although the chase under guarded tgds does not necessarily terminate, CQ containment is decidable in this case since the result of such a chase has finite treewidth [12]. A crucial subclass of guarded tgds is the class of *linear* tgds [13], i.e., tgds with only one atom in the body, which subsume inclusion dependencies.

3. *Non-recursiveness:* A set Σ of tgds is *non-recursive* if its predicate graph contains no directed cycles. (Non-recursive sets of tgds are also known as *acyclic* [24, 43], but we reserve this term for CQs). This class ensures the termination of the chase, and thus decidability of CQ containment.

4. *Stickiness:* The goal of stickiness is to capture joins among variables that are not expressible via guarded tgds, but without forcing the chase to terminate. The definition is based on an inductive marking procedure that marks the variables that could violate a particular semantic property of the chase [14]. Decidability of CQ containment is obtained by applying *query rewriting* techniques.

The complexity of CQ containment and equivalence varies from class to class. It is EXPTIME-complete for full tgds [21] and sticky sets of tgds [21], 2EXPTIME-complete for guarded tgds [12], PSPACE-complete for linear tgds [13, 40], and NEXPTIME-complete for non-recursive sets of tgds [43]. Fixing the schema, or even its arity, yields better complexities in most of the cases.

4.1 Reformulation with tgds

One of the main tasks of our work is to study the problem of checking if a CQ q can be reformulated as a CQ in $\mathsf{GHW}(k)$, for a fixed $k \geq 1$, over those databases that satisfy a set Σ of tgds. We formally define such a reformulation problem below. Given a set Σ of constraints, we write $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$ for the set of CQs

q for which there exists a CQ q' in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. We assume in the following that $\mathbb{C}$ is a class of sets of tgds (e.g., guarded, non-recursive, sticky, etc.):

PROBLEM :	$\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$
INPUT :	A CQ q and a finite set $\Sigma \in \mathbb{C}$.
QUESTION :	Is q in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$?

One might be tempted to think, in view of the Example 4, that when a reformulation $q' \in \mathsf{GHW}(k)$ of q under a set Σ of tgds exists, then such a q' is not very "big", or at least its size is bounded by $|q| + |\Sigma|$. This would allow us to state a small query property for the reformulation problem, and thus establish its decidability. As explained next, this is not the case since the decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ depends not only on the decidability of CQ containment under sets of tgds in $\mathbb{C}$, but also on other considerations.

Undecidable cases. Under mild syntactic assumptions on its input, $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is as hard as CQ containment under $\mathbb{C}$, i.e, we can obtain decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ only for those $\mathbb{C}$'s for which CQ containment is decidable [6]. At this point, one might think that some version of the converse also holds, i.e., $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is reducible to CQ containment under sets of tgds in $\mathbb{C}$. This would imply the decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ for any class $\mathbb{C}$ of sets of tgds for which CQ containment is decidable (in particular, for the full, guarded, non-recursive, and sticky sets of tgds). The next result shows that the picture is more complicated than this, as the reformulation problem is undecidable even if we focus on the class $\mathsf{AC} = \mathsf{GHW}(1)$ of acyclic CQs and the class $\mathcal{F}$ of sets of full tgds:

THEOREM 8. [6] $\mathsf{Reformulation}(\mathsf{GHW}(1), \mathcal{F})$ *is undecidable.*

The question that comes up is whether the reformulation problem is decidable for the remaining classes of tgds, i.e., guarded, non-recursive and sticky, that have a decidable CQ containment problem, and if yes, what is the exact complexity of the problem.

Decidable cases. The next proposition is the main tool that we use to show the decidability of the reformulation problem under certain classes of tgds:

PROPOSITION 9. *Fix $k \geq 1$. Let q and q' be CQs over schema σ such that $q' \in \mathsf{GHW}(k)$ and $q' \subseteq q$. There is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k+1) \cdot a_\sigma$, where a_σ is the maximum arity over all predicates of σ.*

PROOF (SKETCH). Since, by hypothesis, $q'(\bar{x}') \subseteq q(\bar{x})$, it is the case from Proposition 2 that there exists a homomorphism h from q to $\mathcal{D}_{q'}$ such that $h(\bar{x}) = c(\bar{x}')$. Also, since $q' \in \mathsf{GHW}(k)$, it admits a generalized hypertree decomposition (T, λ, χ) of width k. Assume that $\alpha_1, \ldots \alpha_n$ are the atoms of q. By definition, the atoms $h(\alpha_1), \ldots, h(\alpha_n)$ are covered by some nodes $v_1, \ldots, v_m$, where $m \leq n$, of T. In other words, for each $i \in \{1, \ldots, n\}$, there exists $j \in \{1, \ldots, m\}$ such that the elements occurring in $h(\alpha_i)$ form a subset of $\lambda(v_j)$. Consider now the subtree T_q of T consisting of $v_1, \ldots, v_m$ and their ancestors. From T_q we extract the tree $F = (V, E)$ defined as follows:

- V consists of all the root and leaf nodes of T_q, and all the inner nodes of T_q with at least two children.
- For $v, u \in V$, $(v, u) \in E$ iff u is a descendant of v in T_q, and the only nodes of V that occur on the unique simple path from v to u in T_q are v and u.

Our intention is to construct the desired CQ q'' by transforming the atoms occurring in F into a CQ. Let $\mathcal{J} = \bigcup_{v \in V} \chi(v)$. Notice that F is *not* necessarily a generalized hypetree decomposition of $\mathcal{J}$. Indeed, it may be the case that there exists an atom $R(\bar{t})$ in $\mathcal{J}$, but there is no node v in F such that $\bar{t} \subseteq \lambda(v)$. Interestingly, we can transform F into a generalized hypertree decomposition (F, λ', χ') of some instance $\mathcal{J}'$, by renaming some of the terms in $\mathcal{J}$, and then exploit $\mathcal{J}'$ for constructing q''. For example, a node v in F labeled by $\lambda(v) = \{t_1, t_2\}$ and $\chi(v) = \{R(t_1, t'), P(t_2, t', t'')\}$ will be transformed into a node labeled by $\lambda'(v) = \{t_1, t_2, \#_1, \#_2\}$ and $\chi'(v) = \{R(t_1, \#_1), P(t_2, \#_1, \#_2)\}$, where $\#_1$ and $\#_2$ are fresh constants. The set $\mathcal{J}'$ is then defined as $\{h(\alpha_1), \ldots, h(\alpha_n)\} \cup \bigcup_{v \in V} \chi'(v)$.

It is not difficult to verify that (F, λ', χ') is a generalized hypertree decomposition of $\mathcal{J}'$ of width k. Moreover, by construction, F has at most $2 \cdot |q|$ nodes, and each such node is labeled (via χ') with at most k atoms. Thus, $|\mathcal{J}'| \leq 2 \cdot |q| \cdot k + |q| = |q| \cdot (2k+1)$. Therefore, by transforming $\mathcal{J}'$ into a CQ, we obtain a query in $\mathsf{GHW}(k)$ of size at most $|\mathcal{J}'| \cdot a_\sigma \leq |q| \cdot (2k+1) \cdot a_\sigma$; let $q''(\bar{x}')$ be this query. Observe that $c(\bar{x}') \in q(\mathcal{D}_{q''})$ since the homomorphism h maps q to $\mathcal{D}_{q''}$. This fact allows us to conclude that $q'' \subseteq q$ by Proposition 2. Furthermore, there exists a homomorphism, obtained by reversing the renaming substitutions applied during the construction of (F, λ', χ'), that maps q'' to $\mathcal{D}_{q'}$. This allows us to show that $c(\bar{x}') \in q''(\mathcal{D}_{q'})$, and, therefore, that $q' \subseteq q''$ from Proposition 2. Consequently, q'' is the desired CQ, and Proposition 9 follows. □

We write $\mathcal{G}$, $\mathcal{L}$, $\mathcal{NR}$ and $\mathcal{S}$ for the classes of guarded, linear, non-recursive, and sticky sets of tgds, respectively. By exploiting Proposition 9, we can establish a small query property, analogous to Proposition 4 for the constraint-free case, for all the above classes. We define, for each $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$, a function $f_{\mathbb{C}}$ from the set of pairs (Σ, q), where $\Sigma \in \mathbb{C}$ and q is a CQ, to the natural numbers, which will be used to bound the size of the "small" query. Given a set Σ of tgds and a CQ q, let (i) $p_{\Sigma,q}$ be the number of predicates in Σ and q, (ii) $a_{\Sigma,q}$ the maximum arity over all those predicates, and (iii) b_Σ the maximum number of atoms in the body of a tgd in Σ. Then:

$$f_{\mathbb{C}}(\Sigma, q) \quad = \quad \begin{cases} |q|, & \mathbb{C} = \mathcal{G}, \\ |q|, & \mathbb{C} = \mathcal{L}, \\ |q| \cdot (b_\Sigma)^{p_{\Sigma,q}}, & \mathbb{C} = \mathcal{NR}, \\ p_{\Sigma,q} \cdot (a_{\Sigma,q} \cdot |q| + 1)^{a_{\Sigma,q}}, & \mathbb{C} = \mathcal{S}. \end{cases}$$

We can now state the following small query property:

PROPOSITION 10. *Fix $k \geq 1$. Let Σ be a finite set of tgds in $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$ and q a CQ . If $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, then there is a CQ $q' \in \mathsf{GHW}(k)$, where $|q'| \leq f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$, such that $q \equiv_\Sigma q'$.*

PROOF (SKETCH). We first focus on the class $\mathcal{G}$; the same proof applies for linear tgds since $\mathcal{L} \subseteq \mathcal{G}$. By hypothesis, $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, which means that there exists a CQ $q'(\bar{x}')$ in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. Let q_Σ and q'_Σ, respectively, be the CQs that are obtained by applying the chase procedure over the atoms of q and q', respectively, using the tgds of Σ. Notice that such queries might contain an *infinite* number of atoms. The notion of evaluation, as well as generalized hypertree decomposition and generalized hypetreewidth, naturally extend to such infinite queries. It is well-known that $q \subseteq_\Sigma q'$ iff $q_\Sigma \subseteq q'$; see, e.g., [12].

Guarded tgds enjoy a property called *generalized hypetreewidth preserving chase*. This means that if we chase a CQ in $\mathsf{GHW}(k)$ using guarded tgds, the resulting query also falls in $\mathsf{GHW}(k)$. Therefore, $q'_\Sigma \in \mathsf{GHW}(k)$. It can easily be checked that Proposition 9 holds even if the left-hand side query q' is infinite. Since $q'_\Sigma \subseteq q$, there exists a CQ $q'' \in \mathsf{GHW}(k)$ such that $q'_\Sigma \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $q'' \subseteq q$, we have that $q'' \subseteq_\Sigma q$. Moreover, $q \subseteq_\Sigma q''$. This follows from the fact that $q \subseteq_\Sigma q'$ (by hypothesis) and $q' \subseteq_\Sigma q''$ (since $q'_\Sigma \subseteq q''$). We conclude that $q \equiv_\Sigma q''$.

We now focus on non-recursive sets of tgds. It is not difficult to verify that this class does not enjoy the generalized hypetreewidth preserving chase property, and, thus, we cannot apply the same argument as for guarded tgds. However, $\mathcal{NR}$ enjoys some other crucial property, which is very useful for our purposes. Given a CQ q, and a set $\Sigma \in \mathcal{NR}$, we can construct a *union of CQs* (UCQ) Q such that: for every $q'(\bar{x}')$, it holds that $q' \subseteq_\Sigma q$ iff $c(\bar{x}') \in Q(\mathcal{D}_{q'})$. Moreover, the *height* of such a rewriting Q, that is, the maximal size of its disjuncts, is at most $f_{\mathcal{NR}}(\Sigma, q)$; for more details see [35].[1] We can now explain how Proposition 10 is obtained for $\mathcal{NR}$.

Since $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, there is a CQ $q'(\bar{x}')$ in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. As $\mathcal{NR}$ is UCQ rewritable, there is a UCQ Q such that $c(\bar{x}') \in Q(\mathcal{D}_{q'})$, i.e., there exists a CQ q_r (one of the disjuncts of Q) such that $c(\bar{x}') \in q_r(\mathcal{D}_{q'})$. Hence, $q' \subseteq q_r$ by Proposition 2. From Proposition 9, there is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq q'' \subseteq q_r$ and $|q''| \leq |q_r| \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $|q_r| \leq f_{\mathcal{NR}}(\Sigma, q)$, we have $|q''| \leq f_{\mathcal{NR}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$. We claim now that $q \equiv_\Sigma q''$. In fact, $q \subseteq_\Sigma q'$ by hypothesis, and thus $q \subseteq_\Sigma q''$ (since $q' \subseteq q''$). On the other hand, $q_r \subseteq_\Sigma q$ (otherwise, Q would not be a UCQ rewriting), and since $q'' \subseteq q_r$, we get that $q'' \subseteq_\Sigma q$.

Finally, for the class of sticky sets of tgds, we follow the same approach as for non-recursive sets of tgds. The class $\mathcal{S}$ is UCQ rewritable, and, given a set $\Sigma \in \mathcal{S}$ and a CQ q, the height of each UCQ rewriting of q and Σ is at most $f_{\mathcal{S}}(\Sigma, q)$; for more details see [35]. □

Since CQ containment is decidable for any class $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$, Proposition 10 provides a decision procedure for $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$. Given a CQ q and a finite set $\Sigma \in \mathbb{C}$, this procedure is as follows:

1. Guess a CQ q' that falls in $\mathsf{GHW}(k)$ of size at most $f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$; and
2. Verify that $q \equiv_\Sigma q'$.

By exploiting known results on the complexity of CQ containment for the classes of tgds under consideration (see above), and carefully analyzing the time and space complexity of the above procedure, we obtain worst-case optimal upper bounds, apart from the case of sticky sets of tgds for which the complexity remains open. The lower bounds are inherited from CQ containment. Then:

THEOREM 11. *Fix $k \geq 1$ and a class of tgds $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$. $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is*

- 2EXPTIME-*complete for* $\mathbb{C} = \mathcal{G}$.
- PSPACE-*complete for* $\mathbb{C} = \mathcal{L}$.
- NEXPTIME-*complete for* $\mathbb{C} = \mathcal{NR}$.
- *in* NEXPTIME *and* EXPTIME-*hard for* $\mathbb{C} = \mathcal{S}$.

If we assume that the underlying schema is fixed, then in all cases the complexity becomes NP-complete. Better complexity results can be obtained in the case of $\mathcal{G}$, $\mathcal{L}$ and $\mathcal{S}$ if the arity of the schema is fixed, while for $\mathcal{NR}$ it remains NEXPTIME-hard; for details see [6].

4.2 Reformulation with egds

The reformulation problem under egds is quite challenging, and not very well-understood up to date. Although the CQ containment problem under egds can easily be shown to be decidable (as said before, it is

[1] Let us clarify that the work [35] does not explicitly consider the class $\mathcal{NR}$. However, the rewriting algorithm in that paper works also for non-recursive sets of tgds.

NP-complete), currently we do not even know the decidability status of the reformulation problem under the simple class of egds that correspond to keys.

A positive, yet very challenging result in this direction has been recently obtained by Figueira [25]. It states that the reformulation problem is decidable for the class of *unary* FDs, denoted $\mathbb{UFD}$, when restricted to schemas consisting of unary and binary relations. Recall that unary FDs are FDs of the form $R : A \rightarrow B$, where the cardinality of A is one. The following holds:

THEOREM 12. [25] *Fix $k \geq 1$. Given a finite set $\Sigma \in \mathbb{UFD}$ over a schema with unary and binary relations, and a CQ q, we can decide in double-exponential time if there exists a CQ $q' \in \mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$.*

Let us clarify that in [25] the above result is shown for CQs of bounded treewidth. However, the proof adapts to CQs of bounded generalized hypetreewidth [26].

5. THE EVALUATION PROBLEM

As we observed earlier, in the absence of constraints the property of being equivalent to a CQ in $\mathsf{GHW}(k)$, for $k \geq 1$, has a positive impact on query evaluation. We observe here that, at least partially, this good behavior extends to the notion of being equivalent to a CQ in $\mathsf{GHW}(k)$, for $k \geq 1$, under the decidable classes of constraints we identified in the previous section. In particular, evaluation of CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for sets Σ of constraints in such classes can be solved by a *fixed-parameter tractable* (fpt) algorithm, assuming the parameter to be $|q| + |\Sigma|$. Recall that this means that the problem can be solved in time $O(|\mathcal{D}|^c \cdot f(|q| + |\Sigma|))$, for $c \geq 1$ and $f : \mathbb{N} \rightarrow \mathbb{N}$ a computable function. This is an improvement over general CQ evaluation for which no fpt algorithm is believed to exist; see, e.g., [28, 46].

Fix $k \geq 1$ and a class $\mathbb{C}$ of constraints. We study the following problem in this section:

PROBLEM :	$\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$
INPUT :	$\Sigma \in \mathbb{C}$, a CQ $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, a database $\mathcal{D}$ such that $\mathcal{D}$ satisfies Σ, and a tuple $\bar{t}$ of elements in $\mathcal{D}$.
QUESTION :	Is $\bar{t} \in q(\mathcal{D})$?

5.1 Evaluation under tgds

Recall that Theorem 6 establishes that the evaluation problem for CQs that can be reformulated in the class $\mathsf{GHW}(k)$ in the absence of constraints is feasible in polynomial time. As stated next, this good behavior extends to the class $\mathcal{G}$ of sets of guarded tgds:

PROPOSITION 13. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathcal{G})$ *is feasible in polynomial time, for each fixed $k \geq 1$.*

The proof of Proposition 13 for the case $k = 1$ can be found in [6]. A slight modification of this proof yields the result for any $k \geq 1$. We do not know if this good behavior extends to the classes $\mathcal{NR}$ and $\mathcal{S}$. We can prove, nevertheless, that the problem in question retains some good properties; in fact, it is fixed-parameter tractable under such classes:

PROPOSITION 14. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{NR}, \mathcal{S}\}$.* $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$ *is fixed-parameter tractable.*

$\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$, as in the constraint-free case, makes the unrealistic assumption that we know in advance that the CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for a given set Σ of tgds in $\mathbb{C}$. To study the more realistic scenario in which we want to first check if this is the case, and then, if so, check whether $\bar{t} \in q(\mathcal{D})$, we have to return to the guess-and-check procedure from Section 4.1. This procedure checks in double-exponential time if a CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for any set of tgds $\Sigma \in \mathbb{C}$. More importantly, in case that $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$ it also yields an equivalent CQ q' in $\mathsf{GHW}(k)$ of at most exponential size in $|q| + |\Sigma|$. We can then compute and evaluate such a query q' on $\mathcal{D}$, and return $q(\mathcal{D}) = q'(\mathcal{D})$. We know that the latter can be done in time $O(|\mathcal{D}|^{k+1} \cdot |q'|)$, which is $|\mathcal{D}|^{k+1} \cdot 2^{O(|q|+|\Sigma|)}$. Summing up:

COROLLARY 15. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$. Given a CQ q, a set Σ of tgds in $\mathbb{C}$, a database $\mathcal{D}$ satisfying Σ, and a tuple $\bar{t}$ in $\mathcal{D}$, the problem of checking if q is in* $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$*, and, if this is the case, then check whether $\bar{t} \in q(\mathcal{D})$, can be solved in time:*

$$2^{2^{O(|q|+|\Sigma|)}} + |\mathcal{D}|^{k+1} \cdot 2^{O(|q|+|\Sigma|)}.$$

Notice that Proposition 14 follows directly from this result. The algorithm presented above, however, can hardly be claimed to be practical. In fact, it requires a preprocessing step for computing an equivalent reformulation of q under Σ that takes double-exponential time. Although this is a static analysis task, a double-exponential time procedure is too costly in practice even for small q and Σ. Thus, it would be useful to develop heuristics that lower the complexity of this task to at least single-exponential time. A notable exception is the class of linear tgds since, in this case, the guess-and-check algorithm from Section 4.1 takes exponential time to check if a CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for $\Sigma \in \mathcal{L}$.

5.2 Evaluation under egds

Following the same approach as above, we can prove that $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{UFD})$ is fixed-parameter tractable, when restricted to schemas with unary and binary relations. This is because, again, the procedure that checks reformulation for a CQ q under a set $\Sigma \in \mathbb{UFD}$, used in the proof of Theorem 12, yields an equivalent CQ q' in $\mathsf{GHW}(k)$ in case that such a q' exists. Importantly enough, this fixed-parameter tractable algorithm works without the unrealistic assumption that q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for the given set Σ.

Notably, it follows from techniques in [6] that fixed-parameter tractability of evaluation extends to the whole class $\mathbb{EGD}$ of sets of egds. Moreover, for the class $\mathbb{FD}$ of FDs it is even possible to obtain tractability:

PROPOSITION 16. *Fix $k \geq 1$. It holds that:*

1. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{EGD})$ *is fixed-parameter tractable.*
2. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{FD})$ *can be solved in polynomial time.*

In contrast to the case of $\mathbb{UFD}$, though, the evaluation algorithms underlying Proposition 16 require knowing in advance that $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for the given set Σ of egds. However, checking whether such a promise holds for q might be an undecidable problem.

6. APPROXIMATIONS

Let $\mathbb{C}$ be any of the decidable classes of finite sets of tgds we study in this paper (i.e., $\mathcal{G}, \mathcal{NR}$, or $\mathcal{S}$). Then, for any CQ q and set Σ of constraints in $\mathbb{C}$, our techniques yield the *maximally contained* CQs q' in $\mathsf{GHW}(k)$ under Σ.[2] Following the recent database literature, such q's correspond to the $\mathsf{GHW}(k)$*-approximations of q under* Σ; see, e.g., [7, 8, 9]. Computing and evaluating the $\mathsf{GHW}(k)$-approximations of q might help finding "quick" (i.e., fixed-parameter tractable) answers to it when exact evaluation is infeasible.

We define the notion of $\mathsf{GHW}(k)$-approximation of q under Σ below, following the idea that such approximations correspond to its maximally contained CQs in $\mathsf{GHW}(k)$ under Σ:

Definition 1. ($\mathsf{GHW}(k)$-approximations) Fix $k \geq 1$. Let q be a CQ and Σ a finite set of tgds. A $\mathsf{GHW}(k)$*-approximation of q under* Σ is a CQ $q' \in \mathsf{GHW}(k)$ that satisfies the following two conditions:

- **Soundness:** q' only retrieves sound answers with respect to q; in other words, $q' \subseteq_\Sigma q$.
- **Maximality:** There is no CQ q'' in $\mathsf{GHW}(k)$ that approximates q better in terms of containment; i.e., for every $q'' \in \mathsf{GHW}(k)$ it is the case that:
$$q' \subseteq_\Sigma q'' \subseteq_\Sigma q \implies q' \equiv_\Sigma q''.$$

Notice that whenever q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, i.e., there is a CQ $q' \in \mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$, then the unique $\mathsf{GHW}(k)$-approximation of q under Σ is q' itself. That is, the notion of $\mathsf{GHW}(k)$-approximation provides a suitable extension of the notion of $\mathsf{GHW}(k)$-reformulation. We show in the following example that computing an approximation might be useful when exact reformulation is impossible.

Example 5. Recall the database given in Example 4 whose schema is $\{\mathsf{Interest}, \mathsf{Class}, \mathsf{Owns}\}$. Suppose we additionally have a relation $\mathsf{Incompatible}$ that contains pairs (s_1, s_2) whenever style s_1 is incompatible with style s_2. Consider now the query that retrieves all the customers c that own a record r from a style s in which he/she is interested, and also c has shown interest in at least two incompatible styles. This query can be expressed by the following CQ $q(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,y), \mathsf{Class}(y,z),\\ & \mathsf{Interest}(x,z), \mathsf{Interest}(x,z_1), \mathsf{Interest}(x,z_2),\\ & \mathsf{Incompatible}(z_1,z_2).\end{aligned}$$

As in Example 4, suppose that the database satisfies the constraint $\tau := \forall x \forall y \forall z \big(\mathsf{Owns}(x,y), \mathsf{Class}(y,z) \rightarrow \mathsf{Interest}(x,z)\big)$. As it turns out, q cannot be reformulated in $\mathsf{GHW}(1)$. The intuition is that, although we can remove atom $\mathsf{Interest}(x,z)$ as in Example 4, the cycle over variables $\{x, z_1, z_2\}$ is still present. Nevertheless, we can approximate q in $\mathsf{GHW}(1)$ via the CQ $q'(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,y), \mathsf{Class}(y,z),\\ & \mathsf{Interest}(x,w), \mathsf{Incompatible}(w,w).\end{aligned}$$

Note that q' is obtained by removing $\mathsf{Interest}(x,z)$ from q, and identifying the variables z_1 and z_2 with w. Interestingly, this example also shows that approximations can improve in the presence of constraints. Indeed, a possible approximation of q, ignoring τ, is $q''(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,t), \mathsf{Class}(t,t), \mathsf{Interest}(x,t),\\ & \mathsf{Interest}(x,w), \mathsf{Incompatible}(w,w).\end{aligned}$$

It is easy to verify that $q'' \subsetneq_\Sigma q'$. □

6.1 Approximations in the absence of constraints

As in the case of the reformulation problem, it is instructive to start by studying approximations in the absence of constraints. We call $\mathsf{GHW}(k)$-approximations of q under $\Sigma = \varnothing$ simply $\mathsf{GHW}(k)$-approximations of q. As shown in [7], $\mathsf{GHW}(k)$-approximations have good properties in this context that justify its application. In particular, they always exist, are of polynomial size, and can be computed in single-exponential time in the size of the CQ q. For brevity, we write $\mathsf{Approx}(q, \mathsf{GHW}(k))$ for the set of all the $\mathsf{GHW}(k)$-approximations of q (up to equivalence). The following holds:

[2] As said, the decidability of reformulation under egds is not well-understood. Thus, we concentrate on tgds in this section.

THEOREM 17. *Fix $k \geq 1$. Then:*

1. *Every CQ q has a $\mathsf{GHW}(k)$-approximation.*
2. *Given a CQ q, there is an exponential time algorithm that computes the set $\mathsf{Approx}(q, \mathsf{GHW}(k))$.*
3. *For each CQ q, each CQ in $\mathsf{Approx}(q, \mathsf{GHW}(k))$ is of polynomial size.*

The proof of the above result relies on Proposition 9. Recall that the latter proposition states that for every CQ q and CQ $q' \in \mathsf{GHW}(k)$ such that $q' \subseteq q$, we can find a CQ $q'' \in \mathsf{GHW}(k)$ that approximates q at least as well as q', i.e., $q' \subseteq q'' \subseteq q$, and its size is polynomially bounded by that of q. Let us now explain how Theorem 17 follows from Proposition 9.

First, observe that for every CQ q there is at least one q' in $\mathsf{GHW}(k)$ of polynomial size such that $q' \subseteq q$. Simply take a single variable x and add a tuple $R(x, \dots, x)$ for each symbol R in the underlying schema σ. The resulting CQ q' is in $\mathsf{GHW}(1)$, and thus in $\mathsf{GHW}(k)$ for each $k \geq 1$. Moreover, there is a homomorphism from q to the canonical database $\mathcal{D}_{q'}$ of q': just map each variable of q to $c(x)$. Thus, $(c(x), \dots, c(x)) \in q(\mathcal{D}_{q'})$, and hence $q' \subseteq q$ from Proposition 2. It is clear that q' is of polynomial size. Let $t_\sigma : \mathbb{N} \to \mathbb{N}$ be the polynomial such that $t_\sigma(n) = \max\{|q'|, n \cdot (2k+1) \cdot a_\sigma\}$. (Recall that a_σ is the maximum arity of a relation in σ).

Consider now the set $\mathsf{Cont}(q, \mathsf{GHW}(k))$ of CQs q' in $\mathsf{GHW}(k)$ over σ of size at most $t_\sigma(|q|)$ such that $q' \subseteq q$. From the above discussion, this set is nonempty. Let us consider the set $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ consisting of the $\subseteq$-maximal elements of $\mathsf{Cont}(q, \mathsf{GHW}(k))$. We claim that $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ consists of all the $\mathsf{GHW}(k)$-approximations of q (up to equivalence). We first show that each $\mathsf{GHW}(k)$-approximation q' of q is equivalent to some CQ $q'' \in \mathsf{Maximal}(q, \mathsf{GHW}(k))$. Consider such a $\mathsf{GHW}(k)$-approximation q' of q. By definition, $q' \in \mathsf{GHW}(k)$ and $q' \subseteq q$, and, thus, from Proposition 9 there is a CQ $q^* \in \mathsf{GHW}(k)$ such that $q' \subseteq q^* \subseteq q$ and the size of q^* is at most $t_\sigma(|q|)$. Therefore, $q^* \in \mathsf{Cont}(q, \mathsf{GHW}(k))$, and there is a CQ $q'' \in \mathsf{Maximal}(q, \mathsf{GHW}(k))$ such that $q' \subseteq q^* \subseteq q'' \subseteq q$. By definition, $q'' \in \mathsf{GHW}(k)$ and, thus, $q' \equiv q''$ since q' is a $\mathsf{GHW}(k)$-approximation of q. The proof that each CQ q' in $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ is, in fact, a $\mathsf{GHW}(k)$-approximation of q follows a similar reasoning.

Notice that $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ contains at least one CQ (since $\mathsf{Cont}(q, \mathsf{GHW}(k))$ is nonempty). Thus, each CQ q has at least one $\mathsf{GHW}(k)$-approximation. This yields item (1) of Theorem 17. For item (2), it is sufficient to observe that the set $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ can be computed in single-exponential time. This is done by simply enumerating all CQs q' of size at most $t_\sigma(|q|)$, and for each one of them checking the following: (a) $q' \in \mathsf{GHW}(k)$, (b) $q' \subseteq q$, and (c) there is no $q'' \in \mathsf{GHW}(k)$ such that $q' \subsetneq q'' \subseteq q$ and the size of q'' is at most $t_\sigma(|q|)$. Each one of these steps can be carried out in single-exponential time.

Evaluation of approximations. Let us look at the problem of evaluating the $\mathsf{GHW}(k)$-approximations of q, i.e., given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ in $\mathcal{D}$, checking whether $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q. Since each such a q' is contained in q, we can then be sure that $\bar{t}$ also belongs to $q(\mathcal{D})$.

As explained above, the set $\mathsf{Approx}(q, \mathsf{GHW}(k))$ of $\mathsf{GHW}(k)$-approximations of q can be computed in single-exponential time. Hence, checking if $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q can be carried out by a fixed-parameter tractable algorithm in time:

$$2^{r(|q|)} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{r'(|q|)},$$

for polynomials $r, r' : \mathbb{N} \to \mathbb{N}$. Notably, unless P = NP this problem cannot be solved in polynomial time:

PROPOSITION 18. *Fix $k \geq 1$. Given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ in $\mathcal{D}$, checking if $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q is* NP*-complete.*

Let us end up by explaining more in detail why it might be convenient, in some cases, to evaluate the approximations of a CQ q as as way to obtain quick answers when exact evaluation is infeasible or is taking too long. Suppose, in particular, that q cannot be reformulated as a CQ in $\mathsf{GHW}(k)$. Hence, it must be the case that $q \in \mathsf{GHW}(k')$ for some $k' > k$. Let us assume that a generalized hypertree decomposition of q of width k' is available to us. We can then use this decomposition to solve the exact evaluation problem for q over $\mathcal{D}$ in time $O(|\mathcal{D}|^{k'+1} \cdot |q|)$. Still, in the realistic case in which $\mathcal{D}$ is too large – in particular, when $2^{r(|q|)} + 2^{r'(|q|)} \ll |\mathcal{D}|$ – we have that evaluating the $\mathsf{GHW}(k)$-approximations of q over $\mathcal{D}$ in time $2^{r(|q|)} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{r'(|q|)}$ can be considerably faster than evaluating q itself in time $O(|\mathcal{D}|^{k'+1} \cdot |q|)$.

Number of approximations. Theorem 17 establishes a single-exponential upper bound on the number of $\mathsf{GHW}(k)$-approximations that a CQ can have. As established next, this is optimal even for the case $k = 1$.

PROPOSITION 19. [7] *There is a family $\{q_n\}_{n \geq 1}$ of CQs such that each CQ q_n is of size at most $O(n)$ and has $\Omega(2^n)$ non-equivalent $\mathsf{GHW}(1)$-approximations.*

6.2 Approximations with tgds

Let us now study $\mathsf{GHW}(k)$-approximations under sets Σ of tgds. Our main result establishes that if Σ comes from one of the well-behaved classes of sets of tgds we study in the paper (i.e., $\mathcal{G}$, $\mathcal{NR}$, or $\mathcal{S}$), then the $\mathsf{GHW}(k)$-approximations under Σ continue to have good properties in terms of existence and computation. For brevity, we write $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$ for the set of all the $\mathsf{GHW}(k)$-approximations of q under Σ (up to equivalence). The following holds:

THEOREM 20. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$:*

1. *Every CQ q has a $\mathsf{GHW}(k)$-approximation under Σ, where $\Sigma \in \mathbb{C}$.*
2. *Given a CQ q and a set $\Sigma \in \mathbb{C}$, there is a double-exponential time algorithm that computes the set $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$.*
3. *For each CQ q and set $\Sigma \in \mathbb{C}$, each CQ in $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$ is of exponential size.*

As for the case of Theorem 17, we prove Theorem 20 by exploiting a small query property:

PROPOSITION 21. *Fix $k \geq 1$ and assume that $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$. There is a polynomial $t : \mathbb{N} \to \mathbb{N}$ such that for each CQs q, q' and set $\Sigma \in \mathbb{C}$, if $q' \in \mathsf{GHW}(k)$ and $q' \subseteq_\Sigma q$, then there is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$ and $|q''| \leq 2^{t(|q|+|\Sigma|)}$.*

The explanation of how Theorem 20 follows from Proposition 21 mimics the explanation of how Theorem 17 follows from Proposition 9. Let us note that Proposition 21 follows from the proof of Proposition 10, and moreover, we can refine the upper bound for $|q''|$ to be $f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $f_{\mathcal{G}}(\Sigma, q)$ is polynomial, we can obtain an improved version of Theorem 20, for the case of guarded tgds, stating that the approximations are of polynomial size.

The comparison of Theorem 17 and Theorem 20 shows that the addition of constraints does not come for free: (1) Computing the set of approximations under sets of tgds in $\mathbb{C}$ takes double-exponential time, as opposed to the single-exponential time procedure obtained in the absence of them. (2) Approximations in the presence of non-recursive and sticky sets of tgds can be of exponential size, while they are polynomial in their absence.

Evaluation of approximations. From Theorem 20, we obtain that evaluating $\mathsf{GHW}(k)$-approximations under Σ, where Σ is a set of tgds in $\mathbb{C}$, can be solved in time:

$$2^{2^{r(|q|+|\Sigma|)}} + |\mathcal{D}|^{k+1} \cdot 2^{2^{r'(|q|+|\Sigma|)}},$$

for suitable polynomials $r, r' : \mathbb{N} \to \mathbb{N}$. That is, this problem is fixed-parameter tractable. On the other hand, the double-exponential dependence on $|q| + |\Sigma|$ is impractical. It would be important then to develop heuristics that find at least some of these approximations in at most single-exponential time on the size of q and Σ.

7. FINAL REMARKS

We have not only surveyed, but also provided a common framework for recent results about semantic optimization in the classes $\mathsf{GHW}(k)$ – of CQs of bounded generalized hypertreewidth – under tgds or egds. Surprisingly, there are cases where CQ containment is decidable, while reformulation is undecidable. Such cases include the class of full tgds. We have then focussed on the main classes of tgds for which CQ containment is decidable, and do not subsume full tgds, i.e., guarded, non-recursive and sticky sets of tgds. For all these classes we have explained why the reformulation problem is decidable, and provided several complexity results. Regarding egds, we have presented a deep result that establishes the decidability of the reformulation problem under unary FDs over binary schemas.

We have also considered the problem of evaluating a query that can be reformulated in $\mathsf{GHW}(k)$ over a database that satisfies certain constraints. In all cases, when the refomulation problem is decidable such an evaluation problem can be solved by a fixed-parameter tractable procedure. This procedure is "realistic", as it also checks whether the query satisfies the reformulation requirements. By lifting this condition, one can further show that the aforementioned evaluation problem remains fixed-parameter tractable under any sets of egds, and even tractable for sets of guarded tgds or FDs.

Finally, we explained how the techniques developed for studying the reformulation problem also yield the $\mathsf{GHW}(k)$-approximations of a query when an exact reformulation in $\mathsf{GHW}(k)$ cannot be found. Such approximations can be used to "quickly" find sound answers to the query when its exact evaluation is infeasible.

Interestingly, all the complexity results on reformulation under tgds presented in the paper continue to hold for a more *liberal* version of reformulation under constraints that is based on unions of CQs. In such case we are given a UCQ Q and a finite set Σ of tgds, and the question is whether there is a union Q' of CQs in $\mathsf{GHW}(k)$ that is equivalent to Q under Σ. Moreover, when such a reformulation exists we obtain that evaluation, as above, is fixed-parameter tractable.

Many challenging problems remain open, the most noticeable being the decidability status of reformulation under egds/FDs. For egds, we have some indications that the problem is undecidable; in fact, that the existing proof of undecidability for the reformulation problem under full tgds can be recast in terms of egds. For FDs we have no understanding whatsoever at this stage.

So far, decidability results for reformulation have been obtained separately for tgds, on the one hand, and egds, on the other. But in practice tgds and egds often appear together. The decidability boundary for CQ containment in the presence of both types of constraints is delicate [16], but some restricted decidable instances of the problem have been identified [4]. It deserves to be explored whether such restrictions also yield decidability for the reformulation problem studied here.

8. REFERENCES

[1] Christopher R. Aberger, Susan Tu, Kunle Olukotun, and Christopher Ré. Emptyheaded: A relational engine for graph processing. In *SIGMOD*, pages 431–446, 2016.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] F. Afrati, M. Joglekar, C. Ré, S. Salihoglu, and J. D. Ullman. GYM: A multiround join algorithm in mapreduce. CoRR, abs/1410.4156, 2014.

[4] Antoine Amarilli and Michael Benedikt. Finite open-world query answering with number restrictions. In *LICS*, pages 305–316, 2015.

[5] K. Amroun, Z. Habbas, and W. Aggoune-Mtalaa. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *VLDB*, 5(10):968–979, 2012.

[6] Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic acyclicity under constraints. In *PODS*, pages 343–354, 2016.

[7] Pablo Barceló, Leonid Libkin, and Miguel Romero. Efficient approximations of conjunctive queries. *SIAM J. Comput.*, 43(3):1085–1130, 2014.

[8] Pablo Barceló, Reinhard Pichler, and Sebastian Skritek. Efficient evaluation and approximation of well-designed pattern trees. In *PODS*, pages 131–144, 2015.

[9] Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Semantic acyclicity on graph databases. In *SIAM J. Comput.*, 2016.

[10] Catriel Beeri, Ronald Fagin, David Maier, Alberto O. Mendelzon, Jeffrey D. Ullman, and Mihalis Yannakakis. Properties of acyclic database schemes. In *STOC*, pages 355–362, 1981.

[11] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *ICALP*, pages 73–85, 1981.

[12] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[13] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[14] Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[15] Upen S. Chakravarthy, John Grant, and Jack Minker. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst.*, 15(2):162–207, 1990.

[16] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies. *SIAM J. of Comput.*, 14:671–677, 1985.

[17] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[18] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000.

[19] Hubie Chen and Víctor Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *CP*, pages 167–181, 2005.

[20] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP*, pages 310–326, 2002.

[21] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[22] Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[23] Ronald Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.

[24] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[25] Diego Figueira. Semantically acyclic conjunctive queries under functional dependencies. In *LICS*, pages 847–856, 2016.

[26] Diego Figueira, 2017. Personal communication.

[27] Wolfgang Fischl, Georg Gottlob, and Reinhard Pichler. General and fractional hypertree decompositions: Hard and easy cases. *CoRR*, abs/1611.01090, 2016.

[28] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.

[29] L. Ghionna, L. Granata, G. Greco, and F. Scarcello. Hypertree decompositions for query optimization. In *ICDE*, pages 36–45, 2007.

[30] L. Ghionna, L., G. Greco, and F. Scarcello. H-DB: A hybrid quantitative-structural SQL optimizer. In *CIKM*, pages 2573–2576, 2011.

[31] Nathan Goodman and Oded Shmueli. Tree queries: A simple class of relational queries. *ACM Trans. Database Syst.*, 7(4):653–677, 1982.

[32] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: Questions and answers. In *PODS*, pages 57–74, 2016.

[33] Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001.

[34] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

[35] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Query rewriting and optimization for ontological databases. *ACM Trans. Database Syst.*, 2014.

[36] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007.

[37] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 11(1):4:1–4:20, 2014.

[38] Pavol Hell and Jaroslav Nešetřil. The core of a graph. *Discrete Mathematics*, 109:117–126, 1992.

[39] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[40] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[41] Paris C. Kanellakis. Elements of relational database theory. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 1073–1156. 1990.

[42] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[43] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari. From classical to consistent query answering under existential rules. In *AAAI*, pages 1546–1552, 2015.

[44] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.

[45] S. Malik and L. Zhang. Boolean satisfiability: from theoretical hardness to practical success. *CACM*, 52(68):76–82, 2009.

[46] Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *J. Comput. Syst. Sci.*, 58(3):407–427, 1999.

[47] A. Robinson and A. Voronkov. *Handbook of Automated Reasoning*. The MIT Press, 2001.

[48] Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

[49] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.