Efficient Approximations of Conjunctive Queries

Pablo Barcelo
Department of Computer Science, Universidad de Chile
pbarcelo@dcc.uchile.cl

Leonid Libkin
School of Informatics, University of Edinburgh
libkin@inf.ed.ac.uk

Miguel Romero
Department of Computer Science, Universidad de Chile
miromero@ing.uchile.cl

ABSTRACT

When finding exact answers to a query over a large database is infeasible, it is natural to approximate the query by a more efficient one that comes from a class with good bounds on the complexity of query evaluation. In this paper we study such approximations for conjunctive queries. These queries are of special importance in databases, and we have a very good understanding of the classes that admit fast query evaluation, such as acyclic, or bounded (hyper)treewidth queries.

We define approximations of a given query Q as queries from one of those classes that disagree with Q as little as possible. We mostly concentrate on approximations that are guaranteed to return correct answers. We prove that for the above classes of tractable conjunctive queries, approximations always exist, and are at most polynomial in the size of the original query. This follows from general results we establish that relate closure properties of classes of conjunctive queries to the existence of approximations. We also show that in many cases, the size of approximations is bounded by the size of the query they approximate. We establish a number of results showing how combinatorial properties of queries affect properties of their approximations, study bounds on the number of approximations, as well as the complexity of finding and identifying approximations. We also look at approximations that return all correct answers and study their properties.

Categories and Subject Descriptors. H.2.3 [**Database Management**]: Languages—*Query Languages*; G.2.2 [**Discrete Mathematics**]: *Graph algorithms*

Keywords. Conjunctive queries, query evaluation, query approximation, tractability, acyclic queries, treewidth, hypertree width, graphs, homomorphisms.

1. INTRODUCTION

The idea of finding approximate solutions to problems for which computing exact solutions is impossible or infeasible is ubiquitous in computer science. It is common in database research too: approximate query answering techniques are used for evaluating queries over extremely large databases or for queries with very high inherent complexity, see, e.g., [10, 11, 14, 21, 26]. By analyzing the structure of both the database and the query one can often find a reasonable approximation of the answer, sometimes with performance guarantees. Approximate techniques are relevant even for problems whose complexity is viewed as acceptable for regular-size databases, since finding precise answers may become impossible for large data sets we often deal with these days.

To approximate a query, we must have a good understanding of the complexity of query evaluation, in order to find an approximation that is guaranteed to be efficient. For one very common class of queries – *conjunctive*, or select-project-join queries – we do have a very good understanding of their complexity. In fact, we know which classes of conjunctive queries (CQs from now on) are easy to evaluate [8, 15, 16, 17, 22, 32]. Given the importance of conjunctive queries, and our good understanding of them, we would like to initiate a study of their approximations. We do it from the *static analysis* point of view, i.e., independently of the input database: for a query Q, we want to find another query Q' that will be much faster than Q, and whose output would be close to the output of Q on *all* databases. Such analysis is essential when a query is repeatedly evaluated on a very large database (say, in response to frequent updates), and when producing approximations based on both data and queries may be infeasible.

The complexity of checking whether a tuple $\bar{a}$ belongs to the output of a CQ Q on a database D is of the order $|D|^{O(|Q|)}$, where $|\cdot|$ measures the size (of a database or a query) [3, 31]. In fact, the problem is known to be NP-complete, when its input consists of D as well as Q (even for Boolean CQs). In other words, the *combined complexity* of CQs is intractable [7]. Of course the *data complexity* of CQs is low, but having $O(|Q|)$ as the exponent may be prohibitively high for very large datasets.

This observation led to an extensive study of classes of CQs for which the combined complexity is tractable. The first result of this kind by Yannakakis [32] showed tractability for *acyclic* CQs. That was later extended to queries of *bounded treewidth* [8, 12, 22]; this notion captures tractability for classes of CQs defined in terms of their graphs [17]. For classes of CQs defined in terms of their hypergraphs, the corresponding notion guaranteeing tractability is *bounded hypertree width* [16], which includes acyclicity as a special case. All these conditions can be tested in polynomial time [5, 13, 16].

The question we address is whether we can approximate a CQ Q by a CQ Q' from one of such classes so that Q and Q' would disagree as little as possible. Assume, for example, that we manage to find an approximation of Q by an *acyclic* CQ Q', for which checking whether $\bar{a} \in Q'(D)$ is done in time $O(|D| \cdot |Q'|)$ [32]. Then we replaced the original problem of complexity $|D|^{O(|Q|)}$ with that of complexity

$$O\big(f(|Q|) + |D| \cdot s(|Q|)\big)$$

where $s(\cdot)$ measures the size of the resulting approximation, and $f(\cdot)$ is the complexity of finding one.

Thus, assuming that the complexity measures f and s are acceptable, the combined complexity of running Q' is much better than for Q. Hence, if the quality of the approximation Q is good too, then we may prefer to run the much faster query Q' instead of Q, especially in the case of very large databases. Thus, we need to answer the following questions:

- What are the acceptable bounds for constructing approximations, i.e., the functions f and s above?
- What types of guarantees do we expect from approximations?

For the first question, if Q' is of the same size as Q, or even if it polynomially increases the size, this is completely acceptable, as the exponent $O(|Q|)$ is now replaced by the factor $s(|Q|)$. For the complexity f of static computation (i.e., transforming Q to Q'), a single exponential is typically acceptable. Indeed, this is the norm in many static analysis and verification questions [27, 30], and small exponents (like $2^{O(|Q|)}$ or $2^{O(|Q| \log |Q|)}$ we shall mainly encounter) are significantly smaller than $|D|^{|Q|}$ if $|D|$ is large. Thus, in terms of their complexity, our desiderata for approximations are:

1. the approximating query should be at most polynomially larger than Q – and ideally, bounded by the size of Q; and
2. the complexity of finding an approximating query should not exceed single-exponential.

As for the guarantees we expect from approximations, in general they can be formulated in two different ways. By doing it qualitatively we state that an approximation is a query that cannot be improved in terms of how much it disagrees with the query it approximates. Alternatively, to do it quantitatively, we define a measure of disagreement between two queries, and look for approximations whose measure of disagreement with the query they approximate is below a certain threshold.

Here we develop the qualitative approach to approximating CQs. For a given Q, we compare queries from some good (tractable) class $\mathcal{C}$ by how much they disagree with Q: to do so, we define an ordering $Q_1 \sqsubseteq_Q Q_2$ saying, intuitively, that Q_2 disagrees with Q less often than Q_1 does. Then the best queries with respect to the ordering are our approximations from the class $\mathcal{C}$.

Furthermore, we require the approximations to return correct results. This approach is standard in databases (for instance, the standard approximation of query results in the settings of query answering using views and data integration is the notion of maximally contained rewriting [2, 18, 24]). However, we shall also briefly discuss what happens if we relax this requirement.

Our main goal is to explore approximations of arbitrary CQs by tractable CQs. We begin by studying queries on graphs (as the essential machinery needs to be developed for them), and then extend results to arbitrary databases. It turns out that approximations are guaranteed to exist for all the tractable classes of CQs mentioned earlier, which makes the notion worth studying.

In general, the structure of approximations will depend heavily on combinatorial properties of the (tableau of the) query Q we try to approximate. Consider, for instance, a Boolean query $Q_1() \text{:-} E(x,y), E(y,z), E(z,x)$ over graphs. Its best acyclic approximation is $Q_1'() \text{:-} E(x,x)$, which is contained in every Boolean graph query and thus provides us with little information. It turns out that this will be the case whenever the tableau of the query is not a bipartite graph. Let $P_m(x_0, \dots, x_m)$ be the CQ stating that $x_0, \dots, x_m$ form a path of length m, i.e., $E(x_0, x_1), \dots, E(x_{m-1}, x_m)$. If we now look at $Q_2() \text{ :- } P_3(x,y,z,u), P_3(x',y',z',u'), E(x,z'), E(y,u')$ (which has a cycle with variables x, y, z', u'), then it has a nontrivial acyclic approximation $Q_2'() \text{:-} P_4(x,y,z,u,v)$. What changed is that the tableau of Q_2 is bipartite, which guarantees the existence of nontrivial approximations. This example provides a flavor of the results we establish. Of course Boolean queries on graphs are just a useful test case, but they tell us for which classes of queries it makes sense to look for meaningful approximations.

We now provide a quick summary of the results of the paper. As mentioned earlier, we first study queries over graphs and then lift results to arbitrary queries.

Results for graph queries For a query Q, we are interested in approximations Q' from a good class $\mathcal{C}$. The classes we consider are acyclic queries [32] and queries of

Class of queries	Type of approximation	Existence of approximation	size of approximation	time to compute approximation
Graph queries	Acyclic	always exists	at most $\|Q\|$	single-exponential
	Treewidth k			
Arbitrary queries	Acyclic		polynomial in $\|Q\|$	
	Hypertreewidth k			

Figure 1: Summary of results on approximations for conjunctive queries Q

fixed treewidth k, which capture the notion of tractability of CQs over graphs [17]. The first two rows in Figure 1 summarize some of our results: within both classes, approximations exist for all queries (this will follow from a general existence result that relates closure properties of classes of graphs to the existence of approximations), they do not increase the complexity of the query, and can be constructed in single-exponential time, thus satisfying all our desiderata for approximating queries.

We prove several additional results as well. We study the structure of approximations, and relate graph-theoretic properties of tableaux of queries with properties of their approximations (showing, for instance, a close relationship between $(k+1)$-colorability of the tableau and the existence of interesting treewidth-k approximations). For Boolean queries, we show a finer trichotomy result for acyclic approximations, and also prove that such approximations are guaranteed to reduce the number of joins. As for the number of non-equivalent approximations, there are finitely many of them, in fact at most exponentially many in $|Q|$.

We provide further complexity analysis, showing that the problem of checking whether Q' is an acyclic (or treewidth-k) approximation of Q is complete for the class DP (this class, defined formally later, is "slightly" above both NP and coNP [29]). DP-completeness results appeared in the database literature in connection with computing cores of structures [9]; our result is of a very different nature because it holds even when both Q and Q' are minimized (i.e., their tableaux are cores).

We also look at overapproximations which return all correct answers (and perhaps more). There the situation is quite different: acyclic overapproximations need not exist even for Boolean CQs. We provide some sufficient conditions for the non-existence of overapproximations, and show than when overapproximations of a query Q do exist, they are unique (up to equivalence), and have strictly fewer joins than Q itself.

Results for arbitrary queries There are two ways of getting tractable classes of CQs over arbitrary databases, depending on whether one formulates conditions in terms of the *graph* of a query Q, or its *hypergraph*. For graph-based notions, it is known that bounded treewidth characterizes tractability [17]. For them, results for graph queries extend to arbitrary queries.

For hypergraph-based notions, we have the original notion of acyclicity from [32] and its more recent extension to the notion of bounded *hypertree width* [16]; it is known that hypertree width 1 coincides with acyclicity. We again prove a general existence result for approximations. However, the closure conditions imposed on classes of hypergraphs are becoming more involved, and it actually requires an effort to prove that they hold for classes of bounded hypertree width. We show that it is still possible to find approximations in single exponential time. As for their sizes, the need not be bounded by $|Q|$, but they remain polynomial in $|Q|$, with polynomial depending only on the vocabulary (schema). Thus, as the summary table in Figure 1 shows, in this case too, our desiderata for approximations are met.

Regarding techniques required to prove these results, we mainly work with tableaux of queries, and characterize approximations via preorders based on the existence of homomorphisms. Thus, we make a heavy use of techniques from the theory of graph homomorphisms [19]. Besides graph theory and combinatorics, these are commonly used in constraint satisfaction [23], but recently they were applied in database theory as well [6, 25].

Organization Basic notations are given in Section 2. In Section 3 we define the notion of approximations. Section 4 studies queries over graphs, concentrating on acyclic and bounded treewidth approximations. In Section 5 we look at arbitrary databases, concentrating on acyclic and bounded hypertree width approximations. Overapproximations are studied in Section 6, and conclusions are given in Section 7. Due to space limitations, complete proofs are in the appendix.

2. NOTATIONS

Graphs and digraphs Both graphs and digraphs are defined as pairs $G = \langle V, E\rangle$, where V is a set of nodes (vertices) and E is a set of edges. For graphs, an edge is a set $\{u, v\}$, where $u, v \in V$; for digraphs, an edge is a pair (u, v), i.e., it has an orientation from u to v. If $u = v$, we have a (undirected or directed) loop.

If $G = \langle V, E\rangle$ is a directed graph, then G^{u} is the underlying undirected graph: $G^{\mathrm{u}} = \langle V, \{\{u, v\} \mid (u, v) \in E\}\rangle$. We denote by K_m the complete graph on m vertices: $K_m = \langle\{u_1, \ldots, u_m\}, \{\{u_i, u_j\} \mid i \neq j,\ i, j \leq m\}\rangle$, and by $K_m^{\rightleftarrows}$ the complete digraph on m vertices, i.e., $K_m^{\rightleftarrows} = \langle\{u_1, \ldots, u_m\}, \{(u_i, u_j) \mid i \neq j,\ i, j \leq m\}$, so that edges go in both directions. Note that $(K_m^{\rightleftarrows})^{\mathrm{u}} = K_m$.

Graph homomorphisms and cores Given two graphs (directed or undirected) $G_1 = \langle V_1, E_1\rangle$ and $G_2 = \langle V_2, E_2\rangle$, a *homomorphism* between them is a map $h : V_1 \to V_2$ such that $h(e)$ is in E_2 for every edge $e \in E_1$. Of course by $h(e)$ we mean $\{h(u), h(v)\}$ if $e = \{u, v\}$ and $(h(u), h(v))$ if $e = (u, v)$. The image of h is the (di)graph $\mathrm{Im}(h) = \langle h(V_1), \{h(e) \mid e \in E_1\}\rangle$. If there is a homomorphism h from G_1 to G_2, we write $G_1 \to G_2$ or $G_1 \xrightarrow{h} G_2$.

A graph G is a *core* if there is no homomorphism $G \to G'$ into a proper subgraph G' of G. A subgraph G' of G is a *core of* G if G' is a core and $G \to G'$. It is well known that all cores of a graph are isomorphic and hence we can speak of the core of a graph, denoted by $\mathrm{core}(G)$. We say that two graphs G and G' are *homomorphically equivalent* if both $G \to G'$ and $G' \to G$ hold. Homomorphically equivalent graphs have the same core, i.e., $\mathrm{core}(G)$ and $\mathrm{core}(G')$ are isomorphic.

We shall also deal with graphs with distinguished vertices. Let G, G' be (di)graphs and $\bar{u}$, $\bar{u}'$ tuples of vertices in G and G', respectively, of the same length. Then we write $(G, \bar{u}) \to (G', \bar{u}')$ if there is a homomorphism $h : G \to G'$ such that $h(\bar{u}) = \bar{u}'$. With this definition, the notion of core naturally extends to graphs with distinguished vertices.

We write $G \rightleftarrows\!\!\!\!/\; G'$ if $G \to G'$, but $G' \to G$ does not hold.

Databases (relational structures) While the case of graphs is crucial for understanding the main concepts, we shall also state results for conjunctive queries over arbitrary relational structures. A *vocabulary* (often called a *schema* in the database context) is a set σ of relation names $R_1, \ldots, R_l$, each relation R_i having an arity n_i. A *relational structure*, or a database, of vocabulary σ is $\mathcal{D} = \langle U, R_1^{\mathcal{D}}, \ldots, R_l^{\mathcal{D}}\rangle$, where U is a finite set, and each $R_i^{\mathcal{D}}$ is an n_i-ary relation over U, i.e., a subset of U^{n_i}. We usually omit the superscript $^{\mathcal{D}}$ if it clear from the context. We also assume (as is normal in database theory) that U is the active domain of $\mathcal{D}$, i.e., the set of all elements that occur in relations $R_i^{\mathcal{D}}$'s.

Both directed and undirected graphs, for example, are relational structures of the vocabulary that contains a single binary relation E. For digraphs, it is the edge relation; for graphs, it contains pairs (u, v) and (v, u) for each edge $\{u, v\}$.

We often deal with databases together with a tuple of distinguished elements, i.e., $(\mathcal{D}, \bar{a})$, where $\bar{a}$ is a k-tuple of elements of the active domain, for some $k > 0$. Technically, these are structures of vocabulary σ expanded with k extra constant symbols, interpreted as $\bar{a}$.

Homomorphisms of structures are defined in the same way as for graphs: for $\mathcal{D}_1 = \langle U_1, (R_i^{\mathcal{D}_1})_{i\le l}\rangle$ and $\mathcal{D}_2 = \langle U_2, (R_i^{\mathcal{D}_2})_{i\le l}\rangle$, a homomorphism $h : \mathcal{D}_1 \to \mathcal{D}_2$ is a map from U_1 to U_2 so that $h(\bar{t}) \in R_i^{\mathcal{D}_2}$ for every n_i-ary tuple $\bar{t} \in R_i^{\mathcal{D}_1}$, for all $i \le l$. As before, we write $\mathcal{D}_1 \to \mathcal{D}_2$ in this case. For databases with tuples of distinguished elements we have $(\mathcal{D}_1, \bar{a}_1) \to (\mathcal{D}_2, \bar{a}_2)$ if the homomorphism h in addition satisfies $h(\bar{a}_1) = \bar{a}_2$.

The notion of a core for relational structures (with distinguished elements) is defined just as for graphs, using homomorphisms of structures.

Conjunctive queries and tableaux A conjunctive query (CQ) over a relational vocabulary σ is a logical formula in the $\exists, \wedge$-fragment of first-order logic, i.e., a formula of the form $Q(\bar{x}) = \exists \bar{y} \bigwedge_{j=1}^{m} R_{i_j}(\bar{x}_{i_j})$, where each R_{i_j} is a symbol from σ, and $\bar{x}_{i_j}$ a tuple of variables among $\bar{x}, \bar{y}$ whose length is the arity of R_{i_j}. These are often written in a rule-based notation

$$Q(\bar{x}) \coloneq R_{i_1}(\bar{x}_{i_1}), \ldots, R_{i_m}(\bar{x}_{i_m}). \tag{1}$$

The *number of joins* in the CQ (1) is $m - 1$. Given a database $\mathcal{D}$, the answer $Q(\mathcal{D})$ to Q is $\{\bar{a} \mid \mathcal{D} \models Q(\bar{a})\}$. If Q is a Boolean query (a sentence), the answer *true* is, as usual, modeled by the set containing the empty tuple, and the answer *false* by the empty set.

A CQ Q is *contained* in a CQ Q', written as $Q \subseteq Q'$, if $Q(\mathcal{D}) \subseteq Q'(\mathcal{D})$ for every database $\mathcal{D}$.

With each CQ $Q(\bar{x})$ of the form (1) we associate its *tableau* $(T_Q, \bar{x})$, where T_Q is the body of Q viewed as a σ-database; i.e., it contains tuples $\bar{x}_{i_j}$'s in relations R_{i_j}'s, for $j \le m$. If Q is a Boolean CQ, then its tableau is just the σ-structure T_Q.

Many key properties of CQs can be stated in terms of homomorphisms of tableaux. For example, $\bar{a} \in Q(\mathcal{D})$ iff $(T_Q, \bar{x}) \to (\mathcal{D}, \bar{a})$. For CQs $Q(\bar{x})$ and $Q'(\bar{x}')$ with the same number of free variables, $Q \subseteq Q'$ iff $(T_{Q'}, \bar{x}') \to (T_Q, \bar{x})$. Hence, the combined complexity of CQ evaluation and the complexity of CQ containment are in NP (in fact, both are NP-complete [7]).

3. THE NOTION OF APPROXIMATION

We now explain the main idea of approximations. Suppose $\mathcal{C}$ is a class of conjunctive queries (e.g., acyclic, or of bounded treewidth). We are given a query Q not in this class, and we want to approximate it within $\mathcal{C}$. For that, we define an ordering $\sqsubset_Q$ on queries in $\mathcal{C}$: the meaning of $Q_1 \sqsubset_Q Q_2$ is that "Q_2 approximates Q better than Q_1 does", i.e., Q_2 agrees with Q more often than Q_1. Then we look for maximal elements with respect to $\sqsubset_Q$ as good approximations of Q. As explained earlier, we typically are interested in queries that are guaranteed to return correct results.

We now formalize this. For queries $Q(\bar{x})$ and $Q'(\bar{x}')$, a database $\mathcal{D}$ and a tuple $\bar{a}$, we say that Q *agrees with* Q' on $(\mathcal{D}, \bar{a})$ if either $\bar{a}$ belongs to both $Q(\mathcal{D})$ and $Q'(\mathcal{D})$, or to none of them. Then, for CQs $Q(\bar{x}), Q_1(\bar{x}_1), Q_2(\bar{x}_2)$,

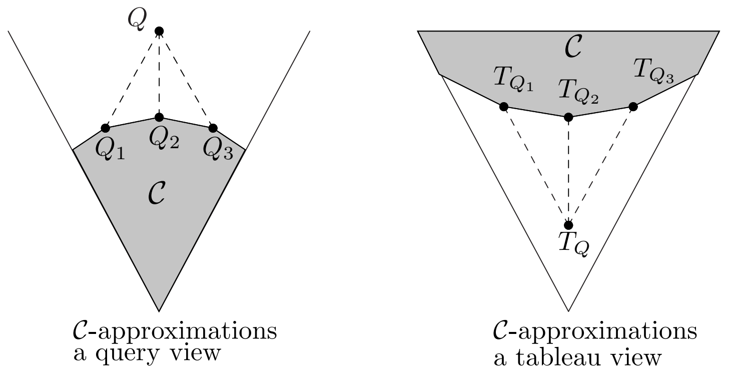


Figure 2: $\mathcal{C}$-approximations: an illustration

with $\bar{x}, \bar{x}_1, \bar{x}_2$ of the same length, we define

$$Q_1 \sqsubseteq_Q Q_2 \overset{\text{def}}{=} \forall(\mathcal{D}, \bar{a}) \left(\begin{array}{c} Q_1 \text{ agrees with } Q \text{ on } (\mathcal{D}, \bar{a}) \\ \Downarrow \\ Q_2 \text{ agrees with } Q \text{ on } (\mathcal{D}, \bar{a}) \end{array} \right)$$

That is, Q_2 approximates Q at least as well as Q_1 does. Then Q_2 approximates Q better than Q_1 does if $Q_1 \sqsubset_Q Q_2$, i.e., $Q_1 \sqsubseteq_Q Q_2$ and $Q_2 \not\sqsubseteq_Q Q_1$.

Definition 3.1. **(Approximations)** *Given a class $\mathcal{C}$ of CQs and a query Q, a query $Q' \in \mathcal{C}$ such that $Q' \subseteq Q$ is a* $\mathcal{C}$-approximation *of Q if there is no query $Q'' \in \mathcal{C}$ with $Q'' \subseteq Q$ such that $Q' \sqsubset_Q Q''$.*

In other words, Q' is an approximation of Q if it is guaranteed to return correct results and no other query approximates Q better than Q'.

We next show that the use of the ordering $\sqsubseteq_Q$ in the definition can be replaced by a containment test:

Proposition 3.2. *Given a CQ $Q(\bar{x})$ and a CQ $Q'(\bar{x}') \in \mathcal{C}$ with the same number of free variables, the following are equivalent:*

1. *Q' is a $\mathcal{C}$-approximation of Q;*
2. *$Q' \subseteq Q$, and there is no $Q'' \in \mathcal{C}$ such that $Q' \subset Q'' \subseteq Q$;*
3. *$(T_Q, \bar{x}) \to (T_{Q'}, \bar{x}')$, and there is no $Q''(\bar{x}'') \in \mathcal{C}$ such that $(T_Q, \bar{x}) \to (T_{Q''}, \bar{x}'') \not\rightleftarrows (T_{Q'}, \bar{x}')$.*

Before describing the classes in which we shall try to approximate CQs, we present a useful view of approximations via orderings on queries and tableaux.

Approximations via ordering Both CQs and their tableaux come naturally equipped with two preorders: containment of CQs, and the existence of homomorphisms between tableaux. These preorders are dual to each other [7]: $Q \subseteq Q' \Leftrightarrow T_{Q'} \to T_Q$. These relations are reflexive and transitive but not antisymmetric (as we may have different equivalent queries), hence they are preorders. They become partial orders when restricted to cores, or minimized CQs. Indeed, if both $T_{Q'} \to T_Q$ and $T_Q \to T_{Q'}$ hold, then $T_{Q'}$ and T_Q are homomorphically equivalent and thus have the same core (which happens to be the tableau of the minimized version of Q). The preorder $\to$ and its restriction to cores have been actively studied over graphs, digraphs, and relational structures [19], and we shall heavily use their properties in our proofs.

With this view, we can visualize the result of Proposition 3.2 as shown in Fig. 2. The $\mathcal{C}$-approximations of Q are the "closest" elements of class $\mathcal{C}$ that are below Q in the $\subseteq$ ordering. If we switch to the tableau view, then approximations are the closest elements of $\mathcal{C}$ which are above the tableau of Q in the $\to$ ordering.

Good classes of queries We look for approximations within tractable classes of CQs, which include acyclic queries, as well as queries of bounded treewidth and hypertree width [8, 12, 16, 17, 22, 32]. We now define the first two (hypertree width is defined in Section 5).

We first need the notion of tree decompositions of hypergraphs of queries. Recall that a hypergraph $\mathcal{H} = \langle V, \mathcal{E} \rangle$ has a set of vertices V and a set of hyperedges $\mathcal{E}$; each hyperedge is a subset of V. For a CQ Q, its hypergraph $\mathcal{H}(Q)$ has all the variables used in Q as vertices; the hyperedges are sets of variables that appear in the same atom. For example, for the query with the body $R(x, y, z), R(x, v, v), E(v, z)$, the hyperedges are $\{x, y, z\}$, $\{x, v\}$, and $\{v, z\}$.

A *tree decomposition* of a hypergraph $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is a tree T together with a map $f : T \to 2^V$ that associates a set of vertices in V with each node of T such that

1. each hyperedge from $\mathcal{E}$ is contained in one of the sets $f(u)$ for $u \in T$; and
2. for every $v \in V$, the set $\{u \in T \mid v \in f(u)\}$ is a connected subset of T.

The *width* of a decomposition is $\max_{u \in T} |f(u)| - 1$, and the *treewidth* of $\mathcal{H}$ is the minimum width of its tree decompositions. If $\mathcal{H}$ is a tree (or a forest) to start with, then its treewidth is 1. We refer to the classes of hypergraphs of treewidth at most k as $\mathsf{TW}(k)$, and, slightly abusing notation, we use $\mathsf{TW}(k)$ to also denote the classes of CQs (and their tableaux) whose hypergraphs have treewidth at most k.

A hypergraph is *acyclic* if there is a tree decomposition (T, f) of it such that every $f(u)$ is a hyperedge. A CQ is acyclic if its hypergraph is acyclic. We use AC to denote the class of acyclic hypergraphs (and also acyclic CQs, and their tableaux). For queries over graphs, we have $\mathsf{AC} = \mathsf{TW}(1)$. In general the notions of bounded treewidth and acyclicity are incompatible (see, e.g., [12]).

Remark The notion of approximation is based on the semantics of queries, while the good classes are defined purely by syntactic means. It may happen that a query Q not from a good class $\mathcal{C}$ is equivalent to some query Q' from $\mathcal{C}$. In that case, $\mathcal{C}$-approximations of Q are simply equivalent to Q.

4. APPROXIMATING QUERIES OVER GRAPHS

In this section we look at queries over graphs. That is, the vocabulary σ has a single binary relation $E(\cdot,\cdot)$, interpreted as a directed graph. Given a CQ $Q(\bar{x})$, its tableau $(T_Q, \bar{x})$ is a digraph as well, with a distinguished tuple of elements $\bar{x}$. Thus, we shall define classes of queries in terms of classes $\mathcal{C}$ of graphs: a CQ Q is a *$\mathcal{C}$-query* iff the graph T_Q is in $\mathcal{C}$.

The standard tractable classes of acyclic CQs and treewidth-k CQs do arise in this way (we shall explain this shortly). But first we prove a very general result on the existence of approximations, which shows good behavior of those for many classes of queries.

Theorem 4.1. *Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. Then every CQ Q that has at least one $\mathcal{C}$-query contained in it also has a $\mathcal{C}$-approximation.*

Moreover, the number of non-equivalent $\mathcal{C}$-approximations of Q is at most exponential in the size of Q, and every $\mathcal{C}$-approximation of Q is equivalent to one which has at most as many joins as Q.

We give a simple proof to illustrate why techniques based on the homomorphism orderings are useful to us.

Proof. Given $Q(\bar{x})$ which is not a $\mathcal{C}$-query, let $H^{\mathcal{C}}(Q)$ be the set of all $\mathcal{C}$-queries whose tableaux are of the form $(\mathrm{Im}(h), h(\bar{x}))$, where h is a homomorphism defined on $(T_Q, \bar{x})$. All such queries are contained in Q. Up to equivalence (renaming of variables) there are finitely many elements in $H^{\mathcal{C}}(Q)$. Moreover, it is nonempty. Indeed, there is a $\mathcal{C}$-query $Q'(\bar{x}')$ with $Q' \subseteq Q$ and hence $(T_Q, \bar{x}) \xrightarrow{h} (T'_Q, \bar{x}')$ for some h (thus $h(\bar{x}) = \bar{x}'$). By the closure under subgraphs we know that $(\mathrm{Im}(h), \bar{x}')$ is a tableau of a $\mathcal{C}$-query. Now consider minimal elements, with respect to the preorder $\rightarrow$, in the set $H^{\mathcal{C}}(Q)$. We claim that they are $\mathcal{C}$-approximations of Q. Indeed let $(\mathrm{Im}(h_0), \bar{x}')$ be one such element, with $\bar{x}' = h_0(\bar{x})$. If it is not a $\mathcal{C}$-approximation, then we have $(T_Q, \bar{x}) \xrightarrow{g} (T, \bar{x}'') \xrightarrow{g_1} (\mathrm{Im}(h_0), \bar{x}')$ for some homomorphisms g and g_1 such that $(\mathrm{Im}(h_0), \bar{x}') \not\rightarrow (T, \bar{x}'')$, with $T \in \mathcal{C}$. Hence we have $(T_Q, \bar{x}') \xrightarrow{g} (\mathrm{Im}(g), \bar{x}'') \xrightarrow{g_1} (\mathrm{Im}(h_0), \bar{x}')$, as well as $(\mathrm{Im}(h_0), \bar{x}') \not\rightarrow (\mathrm{Im}(g), \bar{x}'')$, and $\mathrm{Im}(g)$ is in $\mathcal{C}$ since $\mathcal{C}$ is closed under taking subgraphs. Hence $(\mathrm{Im}(g), \bar{x}'')$ is in $H^{\mathcal{C}}(Q)$, and by the minimality of $(\mathrm{Im}(h_0), \bar{x}')$ we conclude that it is equivalent to $(\mathrm{Im}(g), \bar{x}'')$, and thus is a $\mathcal{C}$-approximation.

If $Q'(\bar{x}')$ is a $\mathcal{C}$-approximation, then $(T_Q, \bar{x}) \xrightarrow{h} (T_{Q'}, \bar{x}')$ and thus $(T_Q, \bar{x}) \xrightarrow{h} (\mathrm{Im}(h), \bar{x}')$, with $\mathrm{Im}(h)$ being a subgraph of $T_{Q'}$, and thus in $\mathcal{C}$. By Proposition 3.2 this implies that $(\mathrm{Im}(h), \bar{x}')$ and $(T_{Q'}, \bar{x}')$ are homomorphically equivalent, and $(\mathrm{Im}(h), \bar{x}')$ is a $\mathcal{C}$-approximation equivalent to Q'. Hence, all $\mathcal{C}$-approximations can be chosen to be of the form $(\mathrm{Im}(h), \bar{x}')$, which shows that there are at most exponentially many of them, and that they need not have more joins than Q. $\square$

Approximations for acyclic and treewidth-k queries We now explain how acyclic graph queries and treewidth-k graph queries appear as $\mathcal{C}$-queries for appropriately chosen classes $\mathcal{C}$. An undirected graph can be viewed as a hypergraph (in which all hyperedges are of cardinality 1 or 2), and thus a graph query Q is acyclic, or of treewidth-k, if T_Q^{u}, the underlying graph of its tableau, is acyclic (as a hypergraph), or has treewidth at most k. We still refer to these as AC and $\mathsf{TW}(k)$. Notice that acyclicity is not the graph acyclicity since loops are allowed: for instance, the undirected graph with an edge between nodes x and y and a loop on x is acyclic, since the hypergraph with hyperedges $\{x, y\}$ and $\{x\}$ is acyclic. Also we formulate these conditions in terms of undirected graphs: for instance, the query $Q() \text{:-} E(x, y), E(y, x)$ is acyclic, since T_Q^{u} contains a single edge between x and y. Basically, acyclicity disallows cycles of length 3 or more.

There is a *trivial* query that belongs to AC and all $\mathsf{TW}(k)$'s that every other CQ Q contains. Indeed, let $K_1^{\circlearrowleft}$ be a single-element loop, i.e., a graph with a single node x and a loop (x, x) on that node. Then, for each query $Q(\bar{x})$ with m free variables, we have, via a constant homomorphism: $(T_Q, \bar{x}) \rightarrow (K_1^{\circlearrowleft}, (x, \ldots, x))$, and thus $Q'(x, \ldots, x) \text{ :- } E(x, x)$ is contained in Q.

This, together with the closure of AC and $\mathsf{TW}(k)$ under taking subgraphs, gives us:

Corollary 4.2. *Every CQ Q over graphs has an acyclic approximation, as well as a treewidth-k approximation, for each $k > 0$.*

Size and number of approximations Let $\mathcal{C}$-$\mathsf{APPR}(Q)$ be the set of all $\mathcal{C}$-approximations of Q. For example, AC-$\mathsf{APPR}(Q)$ is the class of acyclic approximations of Q. These sets are nonempty when $\mathcal{C}$ is AC or $\mathsf{TW}(k)$. They are infinite, but for a simple reason: each CQ has infinitely many equivalent CQs.

It is well known though [7] that each CQ $Q(\bar{x})$ has a unique (up to renaming of variables) equivalent minimal query: in fact, this is the query whose tableau is $\text{core}(T_Q, \bar{x})$. It is obtained by the standard process of minimization of CQs. We thus denote by $\mathcal{C}\text{-}\mathsf{APPR}_{\min}(Q)$ the set of all minimizations of $\mathcal{C}$-approximations of Q.

From Corollary 4.2 and Theorem 4.1 we obtain:

Corollary 4.3. *For every CQ Q, both $\mathsf{AC}\text{-}\mathsf{APPR}_{\min}(Q)$ and $\mathsf{TW}(k)\text{-}\mathsf{APPR}_{\min}(Q)$ are finite nonempty sets of queries. The number of queries in those sets is at most exponential in the size of Q, and each one has at most as many joins as Q. Moreover, a query from $\mathsf{AC}\text{-}\mathsf{APPR}_{\min}(Q)$ or $\mathsf{TW}(k)\text{-}\mathsf{APPR}_{\min}(Q)$ can be constructed in single-exponential time in $|Q|$.*

Hence, acyclic and treewidth-k approximations fulfill the criteria from the introduction: they always exist, they are not more complex than the original query, and they can be found with reasonable complexity.

As for the complexity of finding an approximation, one can easily see that the algorithm that simply checks homomorphisms on T_Q and selects one whose image is minimal with respect to $\rightarrow$ runs in time $2^{O(n \cdot \log n)}$, where n is the number of variables in Q. We shall discuss the complexity in more detail in Subsection 4.1.1.

As for the number of elements of $\mathcal{C}\text{-}\mathsf{APPR}_{\min}(Q)$, a simple upper bound is $2^{n \cdot \log n}$ (a better bound is the nth Bell number [4]). This raises the question whether the exponential number of approximating queries can be witnessed. We prove that this is the case.

Proposition 4.4. *There is a family $(Q_n)_{n>0}$ of Boolean CQs over graphs such that the number of variables and joins in Q_n's grows linearly with n, and $|\mathsf{AC}\text{-}\mathsf{APPR}_{\min}(Q_n)| \geq 2^n$ for all $n > 0$.*

4.1 Acyclic approximations

We now study acyclic approximations in more detail. We begin with the case of Boolean queries, when the tableau of a query is just a graph, and show a trichotomy theorem for them, classifying approximations based on graph-theoretic properties of the tableau. Then we extend results to arbitrary queries. After that we study the complexity of acyclic approximations, and consider the case of connected queries (i.e., queries whose tableaux are connected graphs).

Boolean queries These queries are of the form $Q() \text{:-} \dots$ and thus produce yes/no answers; their tableaux are simply directed graphs T_Q. We already talked about them in the introduction, and mentioned that for nontrivial approximations, the tableau must be *bipartite*. Recall that a digraph G is bipartite if $G \rightarrow K_2^{\rightleftarrows}$, i.e., G is 2-colorable: its nodes can be split into two disjoint subsets A and B so that all edges have endpoints in different subsets.

Recall the example from the introduction: the cyclic query $Q_1() \text{:-} E(x,y), E(y,z), E(z,x)$ had a *trivial* acyclic approximation $Q^{\text{triv}}() \text{:-} E(x,x)$ (which is contained in every Boolean graph query). The reason for that was T_{Q_1} was not bipartite. In the introduction, we saw an example of a query with a bipartite tableaux that had a nontrivial approximation stating the existence of a path of length 4. Note that every query whose tableau is bipartite will contain the *trivial bipartite* query $Q_2^{\text{triv}}() \text{:-} E(x,y), E(y,x)$, whose tableau is $K_2^{\rightleftarrows}$. For some cyclic queries, e.g., $Q_3() \text{:-} E(x,y), E(y,z), E(z,u), E(x,u)$, this trivial query is the only acyclic approximation. This behavior is caused by the cycle being unbalanced. We next define this concept [19], and then state the trichotomy result.

An *oriented cycle* is a digraph with vertices $u_1, \dots, u_n$ and n edges such that either (u_i, u_{i+1}) or (u_{i+1}, u_i) is an edge, for each $i < n$, and either (u_1, u_n) or (u_n, u_1) is an edge. We shall refer to edges (u_i, u_{i+1}) and (u_n, u_1) as *forward edges* and to edges (u_{i+1}, u_i) and (u_1, u_n) as *backward edges*. An oriented cycle is *balanced* if the number of forward edges equals the number of backward edges, and a digraph is balanced if every oriented cycle in it is balanced.

Theorem 4.5. *Let Q be a Boolean CQ over graphs. Then, if its tableau T_Q:*

- *is not bipartite, then Q has only the trivial acyclic approximation Q^{triv};*
- *is bipartite but not balanced, then Q's only acyclic approximation is the trivial bipartite query Q_2^{triv};*
- *is bipartite and balanced, then none of Q's acyclic approximations is trivial, and none contains two subgoals of the form $E(x,y), E(y,x)$.*

Of course when we talk about the only approximation, we mean it up to query equivalence. Note that the conditions used in the theorem – being bipartite and balanced – can be checked in polynomial time [19, 33].

As a corollary to the proof of this result, we obtain:

Corollary 4.6. *Let Q be a Boolean cyclic CQ over graphs. Then all minimized acyclic approximations of Q have strictly fewer joins than Q.*

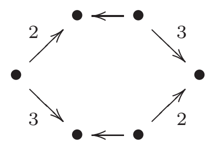
The most interesting case, according to the above theorem, is that of queries whose tableaux are bipartite and balanced. But then a natural question is whether under such restrictions, CQs are already tractable. It turns out that they are not, so it does make perfect sense to approximate queries from that class.

Proposition 4.7. *The combined complexity of evaluating Boolean CQs over graphs whose tableaux are bipartite and balanced is* NP*-complete.*

We conclude our investigation of Boolean CQs with a remark on a subclass of acyclic approximations with special properties. A query Q' is a *tight $\mathcal{C}$-approximation* of Q if it a $\mathcal{C}$-approximation of Q and there is no query Q'' such that $Q' \subset Q'' \subset Q$. It is not clear a priori whether, and for which classes $\mathcal{C}$, such approximations exist. The results of [28] (reformulated in terms of tableaux of queries) imply that if a tight $\mathcal{C}$-approximation Q' of a query Q is minimized and connected, then Q' is acyclic. Hence, tightness forces the approximating query to be acyclic. The next question is whether acyclic tight approximations exist. We can show that this is the case.

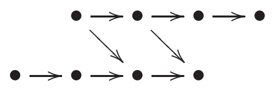
Proposition 4.8. *There is an infinite family of nonequivalent Boolean CQs Q_n, Q'_n, for $n > 0$, so that Q'_n is a tight acyclic approximation of Q_n.*

Example 4.9. Consider a Boolean query Q whose tableau is the graph below, in which number k above an edge represents a path of length k:



This graph is bipartite and balanced, so Theorem 4.5 tells us that is has nontrivial acyclic approximations. In fact it can be shown that Q has a unique (up to equivalence) acyclic approximation Q', whose tableau is the path of length 4 (i.e., the query $Q'() \text{:-} P_4(x, y, u, z, v)$ mentioned in the Introduction).

The same Q' serves as a tight acyclic approximation to the query whose tableau is:



This is exactly the query Q_2 from the Introduction, for which, as stated there, Q' is an acyclic approximation.

Non-Boolean queries For CQs with free variables, it is still true that those whose tableaux are bipartite have nontrivial acyclic approximations. However, now some queries with non-bipartite tableaux may have approximations whose bodies do not trivialize to just $E(x, x)$. For example, consider a query $Q(x, y) \text{:-} E(x, y), E(y, z), E(z, x)$. It can be shown easily that $Q'(x, y) \text{:-} E(x, y), E(y, x), E(x, x)$ is an acyclic approximation of it; the tableau of Q' is $K_2^{\rightleftarrows}$ with a loop on one of the nodes (recall that the definition of query acyclicity refers to tree decompositions of the query hypergraphs, so Q' is indeed acyclic).

What distinguishes the case of bipartite tableaux now when we look at queries with free variables is that they do not have subgoals of the form $E(x, x)$ in approximations. That is, we have the following dichotomy:

Theorem 4.10. *Let $Q(\bar{x})$ be a cyclic CQ over graphs. If its tableau T_Q*

- *is not bipartite, then all of Q's acyclic approximations have a subgoal of the form $E(x, x)$.*
- *is bipartite, then Q has an acyclic approximation without a subgoal of the form $E(x, x)$.*

We already know that acyclic approximations of an arbitrary query Q can be constructed in single-exponential time and are bounded by the size of Q. For Boolean queries we saw that acyclic approximations also have strictly fewer joins than Q. With free variables, the number of joins may sometimes be the same as for Q itself.

Proposition 4.11. *There is a non-Boolean cyclic CQ over graphs such that all of its minimized acyclic approximations have exactly as many joins as Q.*

4.1.1 Complexity

We have seen that a (minimized) acyclic approximation can be found in single-exponential time. Of course this is expected given NP-hardness of most static analysis tasks related to CQs. To do a more detailed analysis of complexity, we formulate it as the following decision problem:

PROBLEM:	ACYCLIC APPROXIMATION
INPUT:	a cyclic CQ Q, an acyclic CQ Q'.
QUESTION:	Is Q' an acyclic approximation of Q?

To solve ACYCLIC APPROXIMATION, we need to check two things:

1. $Q' \subseteq Q$; and
2. there is no acyclic Q'' such that $Q' \subset Q'' \subseteq Q$.

The first subproblem is solvable in NP. Checking whether there is an acyclic query Q'' not equivalent to Q' with $Q' \subseteq Q'' \subseteq Q$ is solvable in NP too. This means $T_Q \to T_{Q''} \rightleftarrows\!\!\!\!\!/\; T_{Q'}$ and hence such Q'', if it exists can always be chosen not to exceed the size of Q. Therefore, one can guess $T_{Q''}$ and all homomorphisms in NP. Furthermore, since both $T_{Q''}$ and $T_{Q'}$ are acyclic, checking that $T_{Q'} \not\to T_{Q''}$ can be done in polynomial time. Thus, the second subproblem is solvable in coNP.

Hence, to solve ACYCLIC APPROXIMATION, we need to solve an NP subproblem and a coNP subproblem. This means that the problem is in complexity class DP: this is the class of problems (languages) which are intersections of an NP language and a coNP language [29]. It turns out that the problem is also hard for the class.

Theorem 4.12. *The problem* ACYCLIC APPROXIMATION *is* DP*-complete. It remains* DP*-complete even if Q' is fixed and both Q and Q' are Boolean and minimized (i.e., their tableaux are cores).*

DP-completeness appeared in database literature in connections with cores: checking whether $G' = \text{core}(G)$, for two graphs G and G', is known to be DP-complete [9]. The source of DP-completeness in our case is completely different, as hardness applies even if the tableaux of queries are cores to start with, and the proof, which is quite involved, uses techniques different from those in [9].

4.2 Bounded treewidth queries

We have already seen that treewidth-k approximations of a CQ Q always exist, that they cannot exceed the size of Q, and can be constructed in single-exponential time. There is an analog of the dichotomy for acyclic queries, in which bipartiteness (i.e., being 2-colorable) is replaced by $(k+1)$-colorability for $\mathsf{TW}(k)$.

Theorem 4.13. *Let Q be a CQ over graphs. If its tableau T_Q*

- *is not $(k+1)$-colorable, then all of its $\mathsf{TW}(k)$-approximations have a subgoal of the form $E(x,x)$;*
- *is $(k+1)$-colorable, then Q has a $\mathsf{TW}(k)$-approximation without a subgoal of the form $E(x,x)$.*

Recall that a Boolean CQ $Q^{\text{triv}}() \text{:–} E(x,x)$ is a trivial (acyclic, or treewidth-k) approximation of every Boolean CQ. In the acyclic case, 2-colorability (or bipartiteness) of T_Q was equivalent to the existence of nontrivial approximations. This result extends to treewidth-k.

Corollary 4.14. *A Boolean CQ Q over graphs has a nontrivial $\mathsf{TW}(k)$-approximation iff its tableau T_Q is $(k+1)$-colorable.*

Note the big difference in the complexity of testing for the existence of nontrivial approximations: while it is in PTIME in the acyclic case, the problem is already NP-complete for $\mathsf{TW}(2)$.

If a Boolean CQ Q has a nontrivial $\mathsf{TW}(k)$-approximation, then the query Q^{triv}_{k+1} with the tableau $K^{\rightleftarrows}_{k+1}$ is contained in Q. For $k=1$, we had a necessary and sufficient condition for such a query to be an approximation (it was the PTIME-testable condition of not being balanced, see Theorem 4.5). For $\mathsf{TW}(k)$, we do not have such a characterization, but we do know that even for $\mathsf{TW}(2)$, the criterion will be much harder than for the acyclic case due to the following.

Proposition 4.15. *For every $k > 1$, testing, for a Boolean CQ Q over graphs, whether Q^{triv}_{k+1} is a $\mathsf{TW}(k)$-approximation of Q is* NP*-hard.*

Thus, while the behavior of acyclic and treewidth-k approximations for $k > 1$ is in general similar, testing conditions that guarantee certain properties of approximations is harder even for treewidth-2, compared to the acyclic case.

Finally, we note that the analog of the ACYCLIC APPROXIMATION problem for treewidth k (i.e., checking if Q' is a $\mathsf{TW}(k)$-approximation of Q) remains DP-complete for all $k \geq 1$. Indeed, the proof of the upper bound for the acyclic case applies to bounded treewidth, and the lower bound is already established for $k=1$.

4.2.1 Connected queries

A query Q is connected if its tableau T_Q is a connected graph. In general, a CQ Q is equivalent to the conjunction of connected CQs whose tableaux are the connected components of T_Q. We now show that approximations of arbitrary queries can be obtained from those for their connected components.

Let $Q(\bar{x})$ be a CQ. It can be represented in the form

$$Q(\bar{x}) = \bigwedge_{i=1}^{m} Q_i(\bar{x}_i) \ \wedge \ \bigwedge_{l=1}^{r} q_l \qquad (2)$$

where all subqueries Q_is and q_ls have connected tableaux and no two share a variable – essentially these are the connected components of the tableau of Q. The queries Q_is have free variables, and the q_ls are Boolean.

The propositions below are stated not only for acyclic and treewidth-k approximations, but generally for $\mathcal{C}$-approximations when $\mathcal{C}$ satisfies the following conditions: closure under subgraphs and disjoint unions, and $K_1^{\circlearrowleft} \in \mathcal{C}$. Note that both AC and $\mathsf{TW}(k)$ are such.

Proposition 4.16. *For a CQ $Q(\bar{x})$ as in (2), let $Q'_i(\bar{y}_i)$ be $\mathcal{C}$-approximations of $Q_i(\bar{x}_i)$ for $i \leq m$. Let Q_* be $\bigwedge\{q_l \mid \exists \bar{y}_i Q'_i(\bar{y}_i) \not\subseteq q_l \text{ for all } i\}$, and let Q'_* be its $\mathcal{C}$-approximation. Then $Q'(\bar{x}) = \bigwedge_{i=1}^{m} Q'_i(\bar{y}_i) \ \wedge \ Q'_*$ is a $\mathcal{C}$-approximation of Q.*

This does not yet fully reduce the unconnected case to the connected case, as we still need to construct approximations of Boolean queries (i.e., Q'_* from the above proposition). Consider a Boolean CQ $Q = Q_1 \wedge \ldots \wedge Q_m$, where all the Q_i's are connected, and no Q_i, Q_j have a variable in common if $i \neq j$.

Proposition 4.17. *For a Boolean CQ Q as above, each $\mathcal{C}$-approximation is equivalent to one of the form $Q'_1 \wedge \ldots \wedge Q'_m$, where each Q'_i is a $\mathcal{C}$-approximation of Q_i.*

The proof also provides a criterion when such a conjunction is an approximation (details are in the appendix). Thus, Propositions 4.16 and 4.17 together fully reduce the case of arbitrary queries to connected ones.

5. APPROXIMATING ARBITRARY QUERIES

We now switch to queries over arbitrary vocabularies. For them, tractability restrictions could be either *graph-based* and *hypergraph-based*. For the graph-based notions, one deals with the graph of query Q, denoted by

$G(Q)$. The nodes of $G(Q)$ are variables used in Q. If there is an atom $R(x_1, \ldots, x_n)$ in Q, then $G(Q)$ has undirected edges $\{x_i, x_j\}$ for all $1 \leq i < j \leq n$. Note that for graph queries, we have $G(Q) = T_Q^{\mathrm{u}}$.

For hypergraph-based notions, we put restrictions of the hypergraph $\mathcal{H}(Q)$ of Q. Recall that its nodes again are variables used in Q, and its hyperedges correspond to the atoms of Q, i.e., for each atom $R(x_1, \ldots, x_n)$ in Q, we have a hyperedge $\{x_1, \ldots, x_n\}$.

Restrictions on queries are imposed as follows. If $\mathcal{C}$ is a class of graphs, or hypergraphs, then a query Q is

- a *graph-based $\mathcal{C}$-query* if $G(Q) \in \mathcal{C}$, and
- a *hypergraph-based $\mathcal{C}$-query* if $\mathcal{H}(Q) \in \mathcal{C}$.

In general, these are incompatible: there are graph-based classes that are not hypergraph-based, and vice versa [12].

5.1 Graph-based classes

For graph-based queries, it is easy to transfer results from queries over graphs to queries over arbitrary schemas. We state the result below only for the classes of tractable graph-based queries, but a general existence theorem, extending Theorem 4.1, is true as well.

Tractability of CQ answering with respect to graph-based classes of queries was fully characterized in [17]: given a class $\mathcal{C}$, query answering for graph-based $\mathcal{C}$-queries is tractable iff $\mathcal{C} \subseteq \mathsf{TW}(k)$ for some k.

We call a CQ Q' a *graph-based $\mathcal{C}$-approximation* of Q if it is an approximation of Q in the class of graph-based $\mathcal{C}$-queries. Then we have an analog of the existence of approximation results from Section 4.

Theorem 5.1. *Every CQ Q has a graph-based $\mathsf{TW}(k)$-approximation, for every $k \geq 1$, with at most as many joins as Q. Moreover, such an approximation can be found in single-exponential time.*

5.2 Hypergraph-based classes

We now look at *hypergraph-based $\mathcal{C}$-approximations*, i.e., approximations in the class of hypergraph-based $\mathcal{C}$ queries.

The oldest tractability criterion for CQs, acyclicity [32], is a hypergraph-based notion (see the definition in Section 3). An analog of bounded treewidth for hypergraphs was defined in [16]; that notion of bounded *hypertree width* properly extended acyclicity and led to tractable classes of CQs over arbitrary vocabularies.

Our first goal, therefore, is to have a general result about the existence of approximations that will apply to both acyclicity and bounded hypertree width (to be defined formally shortly).

Note we cannot trivially lift the closure condition used in Theorem 4.1 for hypergraphs, since even acyclic hypergraphs are not closed under taking subhypergraphs. Indeed, take a hypergraph $\mathcal{H}$ with hyperedges $\{a, b, c\}, \{a, b\}, \{b, c\}, \{a, c\}$. It is acyclic: the decomposition has $\{a, b, c\}$ associated with the root of the tree, and two-element edges with the children of the root. However, it has cyclic subhypergraphs, for instance, one that contains its two-element edges.

The closure conditions we use instead are:

- *Closure under induced subhypergraphs.* If $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is in $\mathcal{C}$ and $\mathcal{H}'$ is an induced subhypergraph, then $\mathcal{H}' \in \mathcal{C}$. Recall that an induced subhypergraph is one of the form $\langle V', \{e \cap V' \mid e \in \mathcal{E}\} \rangle$.
- *Closure under edge extensions:* if $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is in $\mathcal{C}$ and $\mathcal{H}'$ is obtained by adding new vertices V' to one hyperedge $e \in \mathcal{E}$, where V' is disjoint from V, then $\mathcal{H}' \in \mathcal{C}$.

We shall see that these will be satisfied by the classes of hypergraphs of interest to us. The analog of the previous existence results can now be stated as follows.

Theorem 5.2. *Let $\mathcal{C}$ be a class of hypergraphs closed under induced subhypergraphs and edge extensions. Then every CQ Q that has at least one hypergraph-based $\mathcal{C}$-query contained in it, has a hypergraph-based $\mathcal{C}$-approximation.*

Moreover, the number of non-equivalent hypergraph based $\mathcal{C}$-approximations of Q is at most exponential in the size of Q, and every such approximation is equivalent to one which has at most $O(n^{m-1})$ variables and at most $O(n^m)$ joins, where n is the number of variables in Q, and m is the maximum arity of a relation in the vocabulary.

It is straightforward to check that the class of acyclic hypergraphs satisfies both closure conditions, and that any constant homomorphism on a query Q produces an acyclic query. Thus,

Corollary 5.3. *For every vocabulary σ, there exist two polynomials p_σ and r_σ such that every CQ Q over σ has a hypergraph-based acyclic approximation of size at most $p_\sigma(|Q|)$ that can be found in time $2^{r_\sigma(|Q|)}$.*

Next, we extend these results to hypertree width. First we recall the definitions [16]. A *hypertree decomposition* of a hypergraph $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is a triple $\langle T, f, c \rangle$, where T is a rooted tree, f is a map from T to 2^V and c is a map from T to $2^{\mathcal{E}}$, such that

- (T, f) is a tree decomposition of $\mathcal{H}$.
- $f(u) \subseteq \bigcup c(u)$ holds for every $u \in T$.
- $\bigcup c(u) \cap \bigcup \{f(t) \mid t \in T_u\} \subseteq f(u)$ holds for every $u \in T$, where T_u refers to the subtree of T rooted at u.

The *width* of a hypertree decomposition $\langle T, f, c\rangle$ is $\max_{u \in T} |c(u)|$. The *hypertree width* $hw(\mathcal{H})$ of $\mathcal{H}$ is the minimum width over all its hypertree decompositions. We denote by $\mathsf{HTW}(k)$ the class of hypergraphs with hypertree width at most k, and slightly abusing notation, the class of CQ's or tableaux whose hypergraphs have hypertree width at most k.

The key result of [16] is that for each fixed k, CQs from $\mathsf{HTW}(k)$ can be evaluated in polynomial time with respect to combined complexity. It is also shown in [16] that a hypergraph $\mathcal{H}$ is acyclic iff its hypertree width is 1. That is, $\mathsf{AC} = \mathsf{HTW}(1)$.

To apply the existence result, we need to check the closure conditions for hypergraphs of fixed hypertree width. It turns out they are satisfied.

Lemma 5.4. *For each k, the class $\mathsf{HTW}(k)$ is closed under induced subhypergraphs and edge extensions.*

This gives us the desired result about the existence of approximations within $\mathsf{HTW}(k)$ for every k.

Corollary 5.5. *For every vocabulary σ, there exist two polynomials p_σ and r_σ such that every CQ Q over σ has a hypergraph-based $\mathsf{HTW}(k)$-approximation of size at most $p_\sigma(|Q|)$ that can be found in time $2^{r_\sigma(|Q|)}$, for every $k \geq 1$.*

Example 5.6. Consider a Boolean query

$$Q() \;\text{:-}\; R(x_1, x_2, x_3), R(x_3, x_4, x_5), R(x_5, x_6, x_1)$$

over a schema with one ternary relation. If we had a binary relation instead and omitted the middle attribute, we would obtain a query whose tableau is a cycle of length 3, thus having only trivial approximations. However, going beyond graphs lets us find nontrivial acyclic approximations. In fact this query has 3 non-equivalent acyclic approximations (all queries below are minimized):

- With fewer joins than Q:

$$Q_1'() \;\text{:-}\; R(x, y, x).$$

- With as many joins as Q:

$$Q_2'() \;\text{:-}\; R(x_1, x_2, x_3), R(x_3, x_4, x_2), R(x_2, x_5, x_1).$$

- With more joins than Q:

$$\begin{aligned} Q_3'() \;\;\text{:-}\;\; & R(x_1, x_2, x_3), R(x_3, x_4, x_5), \\ & R(x_5, x_6, x_1), R(x_1, x_3, x_5). \end{aligned}$$

6. EXTENSIONS

So far approximations were required to produce only correct answers, i.e., no false positives. In general, one can change and/or relax this assumption. One obvious way is to look for approximations that produce all correct answers, and perhaps something else, i.e., no false negatives. We refer to them as *overapproximations*.

Formally, for a CQ Q not in a class $\mathcal{C}$ of CQs, a query $Q' \in \mathcal{C}$ such that $Q \subseteq Q'$ is a $\mathcal{C}$*-overapproximation* of Q if there is no query $Q'' \in \mathcal{C}$ with $Q \subseteq Q''$ such that $Q' \sqsubset_Q Q''$. In other words, Q' is an overapproximation of Q if it is guaranteed to return all results of Q and no other such query approximates Q better than Q'.

As for approximations we have studied, the use of the ordering $\sqsubseteq_Q$ can be replaced by containment tests.

Proposition 6.1. *Given a CQ Q, a CQ $Q' \in \mathcal{C}$ satisfying $Q \subseteq Q'$ is a $\mathcal{C}$-overapproximation of Q iff there is no $Q'' \in \mathcal{C}$ such that $Q \subseteq Q'' \subset Q'$.*

Recall that for acyclic approximations of Boolean graph queries, we had two cases which resulted in trivial approximations Q^{triv} and Q_2^{triv}, whose tableaux are the single-element loop $K_1^{\circlearrowleft}$ and the graph $K_2^{\rightleftarrows}$. These may serve as acyclic overapproximations as well, but in general we have both existence and nonexistence results.

Proposition 6.2. *Let Q be a cyclic Boolean CQ over graphs. Then:*

- *If T_Q contains $K_1^{\circlearrowleft}$ as a subgraph, then Q^{triv} is the unique, up to equivalence, acyclic overapproximation of Q.*
- *If T_Q contains $K_2^{\rightleftarrows}$ as a subgraph, but does not contain $K_1^{\circlearrowleft}$, then Q_2^{triv} is the unique, up to equivalence, acyclic overapproximation of Q.*
- *If T_Q contains neither $K_1^{\circlearrowleft}$ nor $K_2^{\rightleftarrows}$ as a subgraph, then:*
  1. *When T_Q is not a balanced graph, Q has no acyclic overapproximations.*
  2. *When T_Q is a balanced graph, Q may or may not have acyclic overapproximations.*

Even though we do not yet have a criterion for the existence of acyclic approximations of Q when the tableau of Q is balanced, we can say a lot about their structure and size when they exist: they will be closely related to spanning forests of T_Q, and will be bounded by the size of Q.

Theorem 6.3. *If a cyclic Boolean CQ Q has an acyclic overapproximation, then, up to equivalence, it has exactly one, whose tableau is homomorphically equivalent to a subgraph of T_Q (in fact, to a spanning forest of T_Q).*

Hence, if an acyclic overapproximation exists, its minimization will have fewer joins than Q, because it will be the core of a subgraph of T_Q. Note that T_Q may have multiple spanning forests, but only one of them (up to homomorphic equivalence) corresponds to the acyclic overapproximation.

Even more generally, one can define arbitrary approximations of Q that do not impose any conditions on Q' except it being maximal with respect $\sqsubseteq_Q$. We leave their study to future work, and conclude by the analysis of complexity of the relation $\sqsubseteq_Q$.

Proposition 6.4. *The following problem is* NP*-complete: given three CQs Q, Q_1, Q_2, is $Q_1 \sqsubseteq_Q Q_2$? It remains* NP*-complete if Q_1, Q_2 are restricted to be from the class* AC *of acyclic queries, or from* $\mathsf{TW}(k)$ *for $k \geq 1$.*

Note that even the upper bound in this proposition is not straightforward, as the definition of $\sqsubseteq_Q$ involves universal quantification over databases, and then checking whether CQs evaluate to true or false over them. Even if we could prove that it suffices to consider databases of at most polynomial size, just parsing the definition would have given us a Π_2^p upper bound. The proof instead establishes structural properties of $\sqsubseteq_Q$ based on the properties of the $\rightarrow$ ordering.

7. CONCLUSIONS

We have primarily concentrated on approximations of conjunctive queries that are guaranteed to return correct answers. Given the importance of acyclic CQs and very good complexity bounds for them, we have focused on acyclic approximations, but we also provided results on approximations within classes of bounded treewidth and bounded hypertree width. We have proved the existence of approximations, and showed they can be found with an acceptable computational overhead, and that their sizes are at most polynomial in the size of the original query, and sometimes are bounded by the size of the original query.

Here we dealt with the qualitative approach to approximations; in the future we would like to study quantitative guarantees as well, by defining measures showing how different from the original query approximations are. One approach is to find probabilistic guarantees for approximations. Note that such guarantees have been studied for queries from expressive languages (e.g., with fixed points or infinitary connectives) [1, 20], with typical results showing that queries are equivalent to those from simpler logics (e.g, FO) almost everywhere. One possibility is to specialize these results to much weaker logics, e.g., to CQs and their tractable subclasses.

Acknowledgments We thank Wenfei Fan and Claudio Gutierrez for their comments. Partial support provided by Fondecyt grant 1110171, EPSRC grants G049165 and F028288, and FET-Open Project FoX, grant agreement 233599. Part of this work was done when the first and the third authors visited Edinburgh, and the second author visited Santiago.

8. References

[1] S. Abiteboul, K. Compton, and V. Vianu. Queries are easier than you thought (probably). In *ACM Symp. on Principles of Database Systems*, 1992, ACM Press, pages 23–32.

[2] S. Abiteboul, O. Duschka. Complexity of answering queries using materialized views. In *PODS 1998*, pages 254–263.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[4] M. Aigner. *Combinatorial Theory.* Springer, 1997.

[5] H. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25 (1996), 1305–1317.

[6] B. ten Cate, P. Kolaitis, W.-C. Tan. Database constraints and homomorphism dualities. In *CP 2010*, pages 475–490.

[7] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM Symp. on Theory of Computing*, 1977, pages 77–90.

[8] C. Chekuri, A. Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.* 239(2): 211-229 (2000).

[9] R. Fagin, Ph. Kolaitis, L. Popa. Data exchange: getting to the core. *ACM TODS* 30 (2005), 90–101.

[10] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu. Graph pattern matching: from intractable to polynomial time. *PVLDB* 3(1): 264-275 (2010).

[11] R. Fink, D. Olteanu. On the optimal approximation of queries using tractable propositional languages. In *ICDT 2011*, pages 174–185.

[12] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49 (2002), 716–752.

[13] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[14] M. Garofalakis and P. Gibbons. Approximate query processing: taming the terabytes. In *VLDB'01.*

[15] G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48 (2001), 431–498.

[16] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *JCSS*, 64 (2002), 579–627.

[17] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *ACM Symp. on Theory of Computing*, 2001, pages 657–666.

[18] A. Halevy. Answering queries using views: A survey. *VLDB J.* 10(4):270-294 (2001).

[19] P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.

[20] L. Hella, Ph. Kolaitis, and K. Luosto. Almost everywhere equivalence of logics in finite model theory. *Bull. of Symbolic Logic*, 2 (1996), 422–443.

[21] Y. Ioannidis. Approximations in database systems. In *ICDT'03*, pages 16–30.

[22] P. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS* 61(2):302-332 (2000).

[23] P. Kolaitis and M. Vardi. A logical approach to constraint satisfaction. In *Finite Model Theory and Its Applications*, Springer 2007, pages 339–370.

[24] M. Lenzerini. Data integration: a theoretical perspective. In *PODS'02*, pages 233–246.

[25] L. Libkin. Incomplete information and certain answers in general data models. In *PODS 2011*, pages 59–70.

[26] Q. Liu. Approximate query processing. *Encyclopedia of Database Systems*, 2009, pages 113–119.

[27] S. Malik and L. Zhang. Boolean satisfiability: from theoretical hardness to practical success. *CACM* **52**(8), 76–82, 2009.

[28] J. Nešetřil, C. Tardif. Duality theorems for finite structures (Characterising gaps and good characterisations). *J. Combin. Theory, Ser. B* 80(1):80-97 (2000).

[29] C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *JCSS*, 28 (1986), 244–259.

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

[31] M. Vardi. On the complexity of bounded-variable queries. In *PODS'95*, pages 266–276.

[32] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. Conf. on Very Large Databases*, 1981, pages 82–94.

[33] D. West. *Introduction to Graph Theory.* Prentice Hall, 2001.