Efficient Approximations of Conjunctive Queries

Pablo Barceló[1] Leonid Libkin[2] Miguel Romero[1]

[1] Department of Computer Science, Universidad de Chile
[2] School of Informatics, University of Edinburgh

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 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.

The technical toolkit of the paper comes from the theory of graph homomorphisms, as we mainly work with tableaux of queries and characterize approximations via preorders based on the existence of homomorphisms. In particular, most of our results can be also interpreted as approximation or complexity results for directed graphs.

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 databases too: approximate query answering techniques help evaluate queries over extremely large databases or queries with very high inherent complexity, see, e.g., [14, 15, 18, 29, 34]. By analyzing the structure of both the database and the query one often finds 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 [11, 19, 20, 23, 30, 43]. Given the importance of CQs, 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 $\mathcal{D}$ is of the order $|\mathcal{D}|^{O(|Q|)}$, where $|\cdot|$ measures the size of a database or a query [3, 42]. In fact, the problem is known to be NP-complete, when its input consists of $\mathcal{D}$ as well as Q (even for Boolean CQs). In other words, the *combined complexity* of CQs is intractable [10]. Of course the *data complexity* of CQs is low (more precisely, it belongs to the circuit complexity class AC_0), 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 [43] showed tractability for *acyclic* CQs. That was later extended to queries of *bounded treewidth* [11, 16, 30]; this notion captures tractability for classes of CQs defined in terms of their graphs [23]. For classes of CQs defined in terms of their hypergraphs, the corresponding notions guaranteeing tractability are *bounded hypertree width* [20] and bounded *generalized* hypertree width [21], which include acyclicity as a special case. All these conditions can be tested in polynomial time [8, 17, 20], except for bounded generalized hypertree width [22].

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 $\bar{a} \in Q'(\mathcal{D})$ is done in time $O(|\mathcal{D}| \cdot |Q'|)$ [43]. Then we replaced the original problem of complexity $|\mathcal{D}|^{O(|Q|)}$ with that of complexity

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

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 [35, 39], and modest exponential functions (like $2^{O(|Q|)}$ or $2^{O(|Q|\log|Q|)}$ we shall mainly encounter) are significantly smaller than $|\mathcal{D}|^{|Q|}$ if $|\mathcal{D}|$ 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

Class of queries	Type of approximation	Existence of approximation	Size of approximation	Time to compute approximation
CQs in terms of underlying graph	Treewidth 1	always exists	at most $\lvert Q\rvert$	single-exponential in $\lvert Q\rvert$
	Treewidth k			
CQs in terms of underlying hypergraph	Acyclic		polynomial in $\lvert Q\rvert$	
	Hypertreewidth k			

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

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 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, 24, 32].

Our goal is to explore approximations of arbitrary CQs by tractable CQs. We shall see that approximations are guaranteed to exist for all the tractable classes of CQs mentioned earlier, which makes the notion worth studying.

The structure of approximations depends heavily on combinatorial properties of the (tableau of the) query Q we 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 bipartite. Let $P_m(x_0,\ldots,x_m)$ be the CQ stating that $x_0,\ldots,x_m$ form a path of length m, i.e., $E(x_0,x_1),\ldots,E(x_{m-1},x_m)$. 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

$$Q_2'()\text{:-}P_4(x',x,y,z,u)$$

What changed is that the tableau of Q_2 is bipartite, which guarantees the existence of nontrivial approximations.

Going beyond graph vocabularies allows us to find more approximations. Consider again Q_1 above, replace binary relation E with a ternary relation R, and introduce fresh variables in the middle positions, i.e., look at the query $Q()\text{:-}R(x,u,y),R(y,v,z),R(z,w,x)$. This query does have several nontrivial acyclic approximations: for instance, $Q'()\text{:-}R(x,u,y),R(y,v,u),R(u,w,x)$ is one.

These examples provides a flavor of the results we establish. We now provide a quick summary of the results of the paper. Recall that there are two ways of getting tractable classes of CQs over arbitrary vocabularies, depending on whether one formulates conditions in terms of the *graph* of a query Q, or its *hypergraph*. We first study approximations in tractable classes of CQs defined in terms of its graph, and then in those defined in terms of its hypergraph.

Results for classes of CQs in terms of its graph For a query Q, we are interested in approximations Q' from a good class $\mathcal{C}$ defined in terms of its underlying graph. The classes we consider are queries of bounded treewidth k, which capture the notion of tractability of CQs in the case of graph-based classes [23]. The first two rows in Figure 1 summarize some of our results: 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 (Corollary 4.2 and 4.3), thus satisfying all our desiderata for approximating queries.

We also show that there are at most exponentially many non-equivalent approximations of treewidth-k for a CQ Q, and that the exponential number of approximations can be witnessed even for CQs over graphs (Proposition 4.4).

We then provide further complexity analysis, showing that if the problem of computing a treewidth-k approximation can be solved in polynomial time then P = NP (Proposition 4.11). We also study the decision problem of checking whether Q' is a treewidth-k approximation of Q. We show that this problem is complete for the class DP (this class, defined formally later, is "slightly" above both NP and coNP [37]), and that DP-hardness holds even for treewidth-1 and queries over graphs (Theorem 4.12). DP-completeness results appeared in the database literature in connection with computing cores of structures [13]; our result is of different nature because it holds even when both Q and Q' are minimized (i.e., their tableaux are cores).

Finally, in Section 5, we study the structure of approximations over graphs. We show a close relationship between $(k+1)$-colorability of the tableau and the existence of interesting treewidth-k approximations. For Boolean queries, we prove a finer trichotomy result for acyclic approximations (recall that for CQs over graphs, acyclicity and treewidth-1 coincide), which also shows that such approximations are guaranteed to reduce the number of joins.

Results for classes of CQs in terms of its hypergraph For hypergraph-based notions, we have the original notion of acyclicity from [43] and its more recent extensions to the notions of bounded *hypertree width* [20] and bounded *generalized* hypertree width [21]. It is known that hypertree width 1 coincides with acyclicity, and that both are contained in generalized hypertree width 1. 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, they need not be bounded by $|Q|$, but they remain polynomial in $|Q|$, with polynomial depending only on the vocabulary (Corollary 6.5). 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 graphs and homomorphisms [25]. In particular we heavily use the notion of *core*: a graph G that cannot be homomorphically mapped into a proper subgraph. Besides graph theory and combinatorics, techniques from graph homomorphisms are commonly used in constraint satisfaction [31], but recently they were applied in database theory as well [9, 13, 33].

Many of our results therefore can be interpreted as results about directed graphs (digraphs) and their homomorphisms. We say that an acyclic digraph T is an acyclic approximation of a digraph G if there is a homomorphism from G to T, but whenever this homomorphism goes via another acyclic digraph T' (i.e., we have homomorphisms from G to T' and from T' to T) then there is also a homomorphism from T to T'. Our results then imply the following, that might be of independent interest in graph theory:

- Every digraph G has an acyclic approximation.
- The size of the core of an acyclic approximation does not exceed the size of G (hence there could be at most exponentially many of those).
- There are examples of digraphs witnessing exponential number of nonisomorphic cores of acyclic approximations.
- Testing whether T is an acyclic approximation of G is DP-complete (even if both T and G are cores).
- In fact, even checking for the existence of an exact homomorphism from G to T (i.e., such that there is no homomorphism from G to a proper subgraph of T) is DP-complete.

Organization Basic notations are given in Section 2. In Section 3 we define the notion of approximations. Section 4 studies approximations in graph-based classes of queries, concentrating on bounded treewidth CQs. Section 5 concentrates on queries on graphs and studies important structural properties of approximations in such scenario. In Section 6 we look at approximations in hypergraph-based classes of queries, concentrating on acyclic and bounded (generalized) hypertree width CQs. Conclusions are in Section 7. This paper is a full version based on two conference papers [6] and [7].

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 (nodes) 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 nodes: $K_m = \langle\{u_1, \dots, u_m\}, \{\{u_i, u_j\} \mid i \neq j,\ i, j \leq m\}\rangle$, and by $K_m^{\rightleftarrows}$ the complete digraph on m nodes, i.e., $K_m^{\rightleftarrows} = \langle\{u_1, \dots, u_m\}, \{(u_i, u_j) \mid i \neq j,\ i, j \leq m\}\rangle$, so that edges go in both directions. Note that $(K_m^{\rightleftarrows})^{\mathrm{u}} = K_m$.

Databases (relational structures) A *vocabulary* (often called a *schema* in the database context) is a set σ of relation names $R_1, \dots, R_l$, each relation R_i having an arity n_i. A *relational structure*, or a database, of vocabulary σ is $\mathcal{D} = \langle U, R_1^{\mathcal{D}}, \dots, R_l^{\mathcal{D}}\rangle$, 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 is 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 usually do not distinguish between a (di)graph and the structure that it represents it.

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 and cores Given databases $\mathcal{D}_1 = \langle U_1, (R_i^{\mathcal{D}_1})_{i\leq l}\rangle$ and $\mathcal{D}_2 = \langle U_2, (R_i^{\mathcal{D}_2})_{i\leq l}\rangle$, 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 $i \leq l$. The *image* of h is the structure $\mathrm{Im}(h) = \langle h(U_1), (R'_i)_{i\leq l}\rangle$, where $R'_i = \{h(\bar{t}) \mid \bar{t} \in R_i^{\mathcal{D}_1}\}$, for each $i \leq l$. If there is a homomorphism h from $\mathcal{D}_1$ to $\mathcal{D}_2$, we write $\mathcal{D}_1 \to \mathcal{D}_2$ or $\mathcal{D}_1 \xrightarrow{h} \mathcal{D}_2$. 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 database $\mathcal{D}_1$ is *contained* in $\mathcal{D}_2$ if $R_i^{\mathcal{D}_1} \subseteq R_i^{\mathcal{D}_2}$ for each $i \leq \ell$. It is strictly contained if $R_i^{\mathcal{D}_1} \subset R_i^{\mathcal{D}_2}$ for some $i \leq \ell$. A database $\mathcal{D}$ is a *core* if there is no homomorphism $\mathcal{D} \to \mathcal{D}'$ into a database $\mathcal{D}'$ that is strictly contained in $\mathcal{D}$. A database $\mathcal{D}'$ that is strictly contained in $\mathcal{D}$ is a *core of* $\mathcal{D}$ if $\mathcal{D}'$ is a core and $\mathcal{D} \to \mathcal{D}'$. It is well known that all cores of a database are isomorphic [25] and hence we can speak of the core of a database $\mathcal{D}$, denoted by $\mathrm{core}(\mathcal{D})$. We say that two databases $\mathcal{D}$ and $\mathcal{D}'$ are *homomorphically equivalent* if both $\mathcal{D} \to \mathcal{D}'$ and $\mathcal{D}' \to \mathcal{D}$ hold. Homomorphically equivalent databases have the same core, i.e., $\mathrm{core}(\mathcal{D})$ and $\mathrm{core}(\mathcal{D}')$ are isomorphic.

We write $\mathcal{D} \rightleftarrows\!\!\!\!\!/\; \mathcal{D}'$ if $\mathcal{D} \to \mathcal{D}'$, but $\mathcal{D}' \to \mathcal{D}$ does not hold. (i.e., when $\mathcal{D} \to \mathcal{D}'$ but $\mathcal{D}$ and $\mathcal{D}'$ are not homomorphically equivalent).

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 $\mathcal{D}$. As usual, we write $Q \subset Q'$ if $Q \subseteq Q'$ but it is not the case that $Q' \subseteq Q$. The queries Q and Q' are *equivalent*, denoted by $Q \equiv Q'$, if both $Q \subseteq Q'$ and $Q' \subseteq Q$.

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 $j \leq m$. 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 [10]).

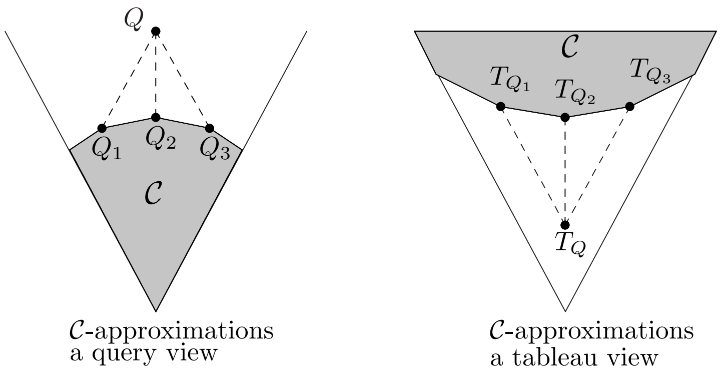


Figure 2: $\mathcal{C}$-approximations: an illustration

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 (hyper)treewidth). We are given a query Q not in this class, and we want to approximate it within $\mathcal{C}$. As explained earlier, we are interested in queries that are guaranteed to return correct results only. Thus, we are looking for a CQ in $\mathcal{C}$ that is *maximally contained* in Q.

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 $Q'' \in \mathcal{C}$ such that $Q' \subset Q'' \subseteq Q$.*

In other words, Q' is an approximation of Q in $\mathcal{C}$ if it is guaranteed to return correct results and no other query in $\mathcal{C}$ approximates Q better than Q'. It can be equivalently proved that Q' is a $\mathcal{C}$-approximation of Q if $Q' \subseteq Q$ and no query $Q'' \in \mathcal{C}$ "agrees" with Q more than Q' does [6]. The latter means that for every database $\mathcal{D}$ and tuple $\bar{a}$ of elements of $\mathcal{D}$, if $\bar{a} \in Q(\mathcal{D})$ but $\bar{a} \notin Q'(\mathcal{D})$ then also $\bar{a} \notin Q''(\mathcal{D})$.

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 [10]: $Q \subseteq Q' \Leftrightarrow T_{Q'} \to T_Q$. 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 → and its restriction to cores have been actively studied over graphs, digraphs, and relational structures [25], and we shall heavily use their properties in our proofs.

With this view, we can visualize Definition 3.1 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 → 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 (generalized) hypertree width

[11, 16, 20, 21, 23, 30, 43]. We now define the first two (hypertree width is defined in Section 6).

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 nodes 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 nodes; 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 nodes 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., [16]).

4 Graph-based Approximations

We start by looking for approximations within classes of CQs defined in terms of its graph, which include queries of bounded treewidth [11, 16, 23]. This condition fully characterizes tractability of CQ answering with respect to graph-based classes of queries [23] (under a certain complexity-theoretic assumption): 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.

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 $1 \leq i < j \leq n$. We define classes of queries in terms of classes $\mathcal{C}$ of graphs: a CQ Q is a *graph-based $\mathcal{C}$-query* if and only if $G(Q)$ is in $\mathcal{C}$.

The standard tractable classes of treewidth-k CQs do arise in this way. Indeed, $\mathsf{TW}(k)$ is the class of queries defined by the class $\mathcal{C}$ of graphs that have treewidth at most k (when viewed as a hypergraph). 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.

Several results from this section apply even for queries over graphs and treewidth 1 (e.g, size and complexity lower bounds). Recall that in such context it is the case that $\mathsf{AC} = \mathsf{TW}(1)$, and hence those results apply as well for the hypergraph-based notion of acyclicity.

4.1 Existence of approximations

We prove a very general result on the existence of approximations, which shows good behavior of those for many classes of queries. For this, we shall need two mild conditions only: The class $\mathcal{C}$

of graphs is closed under taking subgraphs, and there is at least one graph-based $\mathcal{C}$-query that is contained in the query Q that is being approximated.

Theorem 4.1 *1. Let $\mathcal{C}$ be a class of graphs closed under taking subgraphs. Then every CQ Q that has at least one graph-based $\mathcal{C}$-query contained in it also has a graph-based $\mathcal{C}$-approximation.*

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

Proof: Given a query $Q(\bar{x})$, let $H^{\mathcal{C}}(Q)$ be the set of all graph-based $\mathcal{C}$-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 there are finitely many elements in $H^{\mathcal{C}}(Q)$. Moreover, it is nonempty. Indeed, there is a graph-based $\mathcal{C}$-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 graph-based $\mathcal{C}$-query.

Now consider minimal elements, with respect to the preorder $\to$, in the (tableaux of) set $H^{\mathcal{C}}(Q)$. We claim that they are graph-based $\mathcal{C}$-approximations of Q. Indeed let $(\mathrm{Im}(h_0), \bar{x}')$ be the tableau of one such element, with $\bar{x}' = h_0(\bar{x})$. If it is not a graph-based $\mathcal{C}$-approximation, then there exists a graph-based $\mathcal{C}$-query, whose tableau is $(T, \bar{x}'')$, such that $(T_Q, \bar{x}) \xrightarrow{g} (T, \bar{x}'') \xrightarrow{g_1} (\mathrm{Im}(h_0), \bar{x}')$ for some homomorphisms g and g_1 such that $(\mathrm{Im}(h_0), \bar{x}') \not\to (T, \bar{x}'')$. 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 $(\mathrm{Im}(h_0), \bar{x}') \not\to (\mathrm{Im}(g), \bar{x}'')$, and $(\mathrm{Im}(g), \bar{x}'')$ is the tableau of a graph-based $\mathcal{C}$-query since $\mathcal{C}$ is closed under taking subgraphs. Hence, $(\mathrm{Im}(g), \bar{x}'')$ is the tableau of a CQ in $H^{\mathcal{C}}(Q)$ and $(\mathrm{Im}(g), \bar{x}'') \rightleftarrows\!\!\!\!/\ (\mathrm{Im}(h_0), \bar{x}')$, which contradicts the minimality of $(\mathrm{Im}(h_0), \bar{x}')$.

If $Q'(\bar{x}')$ is a graph-based $\mathcal{C}$-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 substructure of $T_{Q'}$, and $(\mathrm{Im}(h), \bar{x}')$ the tableau of a graph-based $\mathcal{C}$-query. It follows that $(\mathrm{Im}(h), \bar{x}')$ and $(T_{Q'}, \bar{x}')$ are homomorphically equivalent, and the CQ with tableau $(\mathrm{Im}(h), \bar{x}')$ is a graph-based $\mathcal{C}$-approximation equivalent to Q'. Hence, all graph-based $\mathcal{C}$-approximations can be chosen to have a tableau 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. □

Approximations for treewidth-k queries There is a *trivial* query that belongs to all $\mathsf{TW}(k)$'s that every other CQ Q contains. Indeed, let Q_{trivial} be the query on a single variable x that is obtained by taking the conjunction of all atoms of the form $R(x, \ldots, x)$, for R a relation symbol in the vocabulary. Then, for each query $Q(\bar{x})$ with m free variables, we have, via a constant homomorphism: $(T_Q, \bar{x}) \to (T_{Q_{\text{trivial}}}, (x, \ldots, x))$, and thus Q_{trivial} is contained in Q.

This, together with Theorem 4.1 and the closure of $\mathsf{TW}(k)$ under taking subgraphs, gives us the following:

Corollary 4.2 *Every CQ Q has a $\mathsf{TW}(k)$-approximation, for each $k > 0$.*

4.2 Size and number of approximations

Let $\mathcal{C}$-$\mathsf{APPR}(Q)$ be the set of all graph-based $\mathcal{C}$-approximations of Q. For each $k \geq 1$, this set is nonempty when $\mathcal{C}$ is $\mathsf{TW}(k)$. It is also infinite, but for a simple reason: each CQ has infinitely many equivalent CQs.

It is well known though [10] 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 $\mathcal{C}$-$\mathsf{APPR}_{\min}(Q)$ the set of all minimizations of graph-based $\mathcal{C}$-approximations of Q.

From Corollary 4.2 and Theorem 4.1 we obtain:

Corollary 4.3 *For every CQ Q and $k \geq 1$, $\mathsf{TW}(k)$-$\mathsf{APPR}_{\min}(Q)$ is a finite nonempty set 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 $\mathsf{TW}(k)$-$\mathsf{APPR}_{\min}(Q)$ can be constructed in single-exponential time in $|Q|$.*

Hence, 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.

Note that the exponential bound in Corollary 4.3 is not due to the minimization procedure which actually happens to be polynomial for queries of fixed treewidth. In general, there is a simple algorithm for finding approximations that just checks homomorphisms on T_Q and selects one whose image is minimal with respect to $\rightarrow$; it 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 4.3.

As for the number of elements of $\mathcal{C}$-$\mathsf{APPR}_{\min}(Q)$, 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 even for queries over graphs and treewidth 1.

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 $|\mathsf{TW}(1)\text{-}\mathsf{APPR}_{\min}(Q_n)| \geq 2^n$ for all $n > 0$.*

Proof: We need some definitions and results from [25]. An *oriented path* $P = (u_0, ..., u_n)$ is a digraph with nodes $u_0, \ldots, u_n$ and n edges such that either (u_i, u_{i+1}) or (u_{i+1}, u_i) is an edge, for each $0 \leq i < n$. We shall refer to edges (u_i, u_{i+1}) as *forward* edges and to edges (u_{i+1}, u_i) as *backward* edges. The node u_0 is the *initial* node of the oriented path and u_n the *terminal* one. Typically, we depict an oriented path P as an edge uv labeled with P, representing that the initial node of P is u and the terminal node is v. An *oriented cycle* is an oriented path whose initial and terminal node coincide.

We define the *net length* of P to be the number of forward edges minus the number of backward edges of P. Often we write oriented paths as strings in $\{0,1\}^*$, where 0 represents a forward edge and 1 represents a backward edge. For example, $P = 001$ means that P is the oriented path with two forward edges followed by a backward edge.

A digraph G is *balanced* if each one of its oriented cycles has net length 0, that is, the number of forward edges equals the number of backward edges. For a balanced digraph G and a node v in G, we define the *level* of v to be

$$\max \{\text{net length of } P \mid P \text{ is an oriented path in } G \text{ with terminal node } v\}.$$

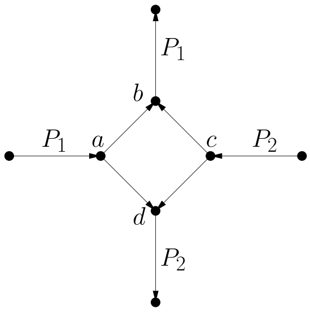


Figure 3: The digraph D.

It can be proved that the level of v is finite since G is balanced [25]. We define the *height* of G, denoted $hg(G)$, to be the maximum level of a node in G.

The following is an important lemma from [25]:

Lemma 4.5 *Let G and H be two balanced digraphs of the same height. Then any homomorphism from G to H preserves the levels of nodes.*

We now prove Proposition 4.4. Consider the oriented paths $P_1 = 001000$ and $P_2 = 000100$. It is straightforward to check that P_1 and P_2 are incomparable cores (i.e, $P_1 \not\to P_2$ and $P_2 \not\to P_1$). We define the digraph D as follows: Consider the digraph $\langle V, E\rangle$ such that $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, d), (c, b), (c, d)\}$. Add disjoint copies of P_1 and P_2 and identify the initial node of the copy of P_1 and P_2, with b and d, respectively. Then add two new disjoint copies of P_1 and P_2, and identify the terminal node of the copy of P_1 and P_2, with a and c, respectively. The resulting digraph D is depicted in Figure 3.

We also define D_{ac} and D_{bd} as the digraphs obtained from D by identifying a with c, and b with d, respectively. This is shown in Figure 4. Note that both D_{ac} and D_{bd} are balanced, and have height 9.

Claim 4.6 *D_{ac} and D_{bd} are incomparable cores.*

Proof: We first prove that D_{ac} is a core. Assume otherwise. Then $D_{ac} \xrightarrow{h} D_{ac}$, where h is not surjective. Since D_{ac} is balanced, Lemma 4.5 tells us that h preserves levels. Figure 4 shows the different levels. Observe that the only node in D_{ac} with level 4 is e, thus $h(e) = e$. Note also that $h(x_1)$ is either x_1 or x_3. But $h(x_1) = x_3$ implies $P_1 \to P_2$, which is impossible. It follows that $h(x_1) = x_1$. Similarly, we have $h(x_3) = x_3$. Using the same argument, we have $h(b) = b$, otherwise $h(b) = d$ and $P_1 \to P_2$. Similarly, $h(d) = d$. Finally, we must have $h(x_2) = x_2$ and $h(x_4) = x_4$. It follows that h is surjective, which is a contradiction. Analogously, we have that D_{bd} is a core.

We prove next that D_{ac} and D_{bd} are incomparable. Assume otherwise. Suppose first that $D_{ac} \xrightarrow{h} D_{bd}$. Observe that h preserves levels, since D_{ac} and D_{bd} have the same height. It follows that $h(e)$ is either a or c. If $h(e) = a$, then $h(x_3) = y_1$. It follows that $P_2 \to P_1$. Similarly, if $h(e) = c$, it follows that $P_1 \to P_2$. In any case, we have a contradiction with the fact that P_1 and P_2 are incomparable. The case $D_{bd} \not\to D_{ac}$ is analogous. □

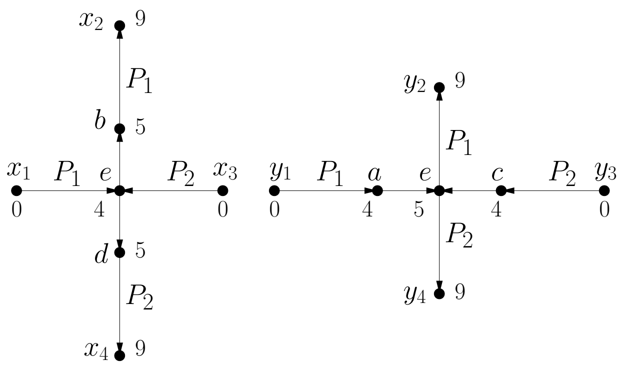


Figure 4: D_{ac} and D_{bd}, and some of its levels.

For $n \geq 1$, we define G_n to be the digraph constructed as follows: Take the union of n disjoint copies of the digraph D. For each $1 \leq i < n$, add an edge from the terminal node of the copy of P_2 which starts in d in the i-th copy of D to the initial node of the copy of P_1 which ends in a in the $(i+1)$-th copy of D. Figure 5 shows a graphical depiction of G_3.

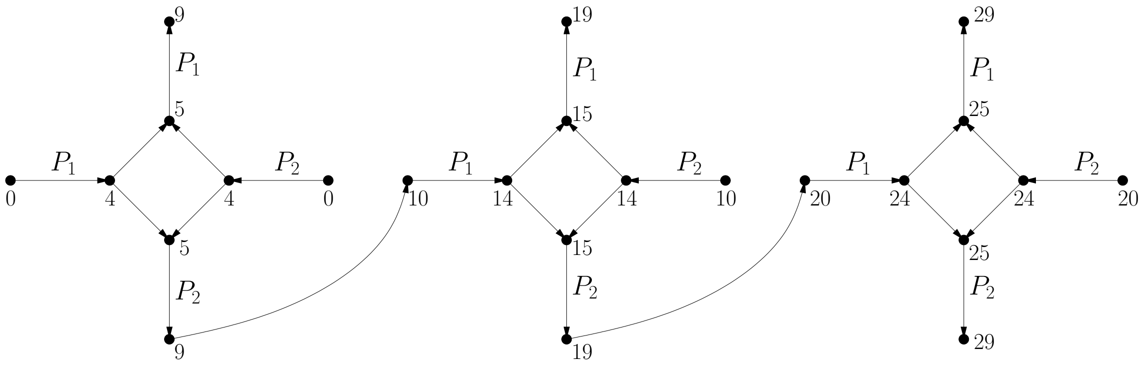


Figure 5: The digraph G_3 and some of its levels.

Consider a string $s \in \{V, H\}^n$ for $n \geq 1$. We define G_n^s to be the digraph obtained from G_n by identifying in the i-th copy of D, $1 \leq i \leq n$, the nodes a and c if $s_i = V$ (that is, turning the i-th copy of D in G_n into D_{ac}) and b and d if $s_i = H$ (that is, turning the i-th copy of D in G_n into D_{bd}).

Claim 4.7 *Let $s, s' \in \{V, H\}^n$ for $n \geq 1$ and assume that $s \neq s'$. Then G_n^s and $G_n^{s'}$ are incomparable cores.*

Proof: We first prove that G_n^s is a core for each $s \in \{V, H\}^n$. Assume otherwise. Then $G_n^s \xrightarrow{h} G_n^s$, where h is not surjective. Observe that G_n^s is balanced and no two nodes in different copies of D_{ac} or D_{bd} in G_n^s can share a level, as shown in Figure 5. It follows from Lemma 4.5 that h maps the i-th copy of D_{ac} or D_{bd} in G_n^s to itself. Thus, h is surjective in such copy (because from Claim 4.6 both D_{ac} and D_{bd} are cores), and, therefore, h itself is surjective. This is a contradiction.

Now, assume that $G_n^s \xrightarrow{h} G_n^{s'}$. Since G_n^s and $G_n^{s'}$ have the same height, we have from Lemma 4.5 that h preserves levels. Again, we have that h maps the i-th copy of D_{ac} or D_{bd} in G_n^s to the i-th copy of D_{ac} or D_{bd} in $G_n^{s'}$, respectively. Since there exists $1 \leq j \leq n$ such that the j-th components of s and s' are different, we have that h maps D_{ac} to D_{bd}, or vice versa. This contradicts Claim 4.6 which states that D_{ac} and D_{bd} are incomparable. □

We state a claim about databases that will be useful in this proof and in other proofs as well:

Claim 4.8 *Let $\mathcal{D}$ and $\mathcal{D}'$ be two databases over the same schema. If $\mathcal{D} \xrightarrow{h} \mathcal{D}'$ and $h(a) = h(b)$ for a and b in $\mathcal{D}$, then $\mathcal{D}^* \rightarrow \mathcal{D}'$, where $\mathcal{D}^*$ is the database obtained from $\mathcal{D}$ by identifying a and b with a new element c.*

Proof: Just map c to $h(a) = h(b)$ and all other elements to themselves. □

For each $n \geq 1$ and $s \in \{V, H\}^n$, let Q_n be the CQ whose tableau is G_n and Q_n^s the CQ whose tableau is G_n^s. Observe that each Q_n^s has treewidth 1.

Claim 4.9 *Q_n^s is a $\mathsf{TW}(1)$-approximation of Q_n, for each $n \geq 1$ and $s \in \{V, H\}^n$.*

Proof: By contradiction, suppose that there exists a CQ $Q'' \in \mathsf{TW}(1)$ such that $G_n \xrightarrow{h} T_{Q''} \not\rightleftarrows G_n^s$. Consider the i-th copy of D in G_n. If we restrict h to such copy it must be the case that $h(a) = h(c)$ or $h(b) = h(d)$. Otherwise $h(a)$, $h(b)$, $h(c)$ and $h(d)$ forms an oriented cycle in $T_{Q''}$. With each i such that $1 \leq i \leq n$ we associate a label $t_i \in \{V, H\}$ in the following way: If the restriction of h to the i-th copy of D in G_n satisfies $h(a) = h(c)$ then $t_i = V$, otherwise $t_i = H$. We then define a word t in $\{V, H\}^n$ as $t_1 t_2 ... t_n$. Using Claim 4.8 we have that $G_n^t \rightarrow T_{Q''}$, and, by composition, $G_n^t \rightarrow G_n^s$. Using Claim 4.7 we have $t = s$, and then $G_n^s \rightarrow T_{Q''}$, which is a contradiction. □

We conclude the proof of the proposition now. Observe that for each $n \geq 1$ and $s \in \{V, H\}^n$ it is the case that $Q_n^s \in \mathsf{TW}(1)\text{-}\mathsf{APPR}_{\min}(Q_n)$ (because G_n^s is a core). Therefore, $|\mathsf{TW}(1)\text{-}\mathsf{APPR}_{\min}(Q_n)| \geq 2^n$, for all $n \geq 1$. Furthermore, observe that the number of variables in Q_n is $28n$ and the number of joins is the number of edges in G_n minus 1, that is, $29n - 2$. Therefore, the family $(Q_n)_{n \geq 0}$ satisfies the required conditions. □

Reinterpretation of results for graphs Let G be an arbitrary digraph and T an acyclic digraph. We say that T is an *acyclic approximation* of G if $G \rightarrow T$ and there is no acyclic T' such that $G \rightarrow T'$ and $T' \not\rightleftarrows T$.

Then our results imply the following:

Corollary 4.10

- *Every digraph G has an acyclic approximation.*
- *The number of nonisomorphic cores of acyclic approximations of G is at most $2^{n \cdot \log n}$, where n is the number of vertices of G.*
- *There exists a family G_n, $n > 0$ of cyclic digraphs so that the number of edges and vertices in G_ns grows linearly with n, and each G_n has at least 2^n nonisomorphic cores of acyclic approximations.*

4.3 Complexity of computation and identification

We have seen that a (minimized) treewidth-k approximation can be found in single-exponential time. Of course this is expected given NP-hardness of most static analysis tasks related to CQs. It is also in accordance with the following proposition that states that if treewidth-k approximations could be computed in polynomial time then P = NP.

Proposition 4.11 *Assume there exists a polynomial time algorithm that, given a CQ Q, computes a $\mathsf{TW}(k)$-approximation of Q, for a fixed $k \geq 1$. Then* P = NP.

Proof: For each fixed $k \geq 1$, the following problem is NP-complete [12]: Given a Boolean CQ Q over graphs, check if Q is equivalent to a CQ in $\mathsf{TW}(k)$. We prove next that, for each fixed $k \geq 1$, this problem can be decided in polynomial time if we assume the existence of a polynomial time algorithm $\mathcal{A}$ that computes $\mathsf{TW}(k)$-approximations for CQs over graphs.

Let $\mathcal{A}(Q)$ be the output of $\mathcal{A}$ on input a CQ Q. We claim that $Q \subseteq \mathcal{A}(Q)$ if and only if Q is equivalent to a CQ Q' in $\mathsf{TW}(k)$ (i.e. $Q \equiv Q'$). Assume first that $Q \equiv Q'$, for some $Q' \in \mathsf{TW}(k)$. Since $\mathcal{A}(Q)$ is $\mathsf{TW}(k)$-approximation of Q (and, thus, of Q'), it is the case that $\mathcal{A}(Q) \subseteq Q' \subseteq Q$, and, hence, $Q' \equiv \mathcal{A}(Q)$. It follows that $Q \subseteq \mathcal{A}(Q)$, which was to be proved. On the other hand, if $Q \subseteq \mathcal{A}(Q)$ then $Q \equiv \mathcal{A}(Q)$ (because $\mathcal{A}(Q)$ is a $\mathsf{TW}(k)$-approximation of Q, and hence it is contained in Q). We conclude that Q is equivalent to a CQ Q' that belongs to $\mathsf{TW}(k)$.

Since checking whether $Q \subseteq \mathcal{A}(Q)$ corresponds to evaluating the bounded treewidth query $\mathcal{A}(Q)$ in the tableau of Q, which can be done in polynomial time, we conclude that the whole procedure can be done in polynomial time. This finishes the proof of the proposition. □

To do a more detailed analysis of complexity for approximations, we study the following associated decision problem:

Problem:	Treewidth-k Approximation
Input:	a CQ Q, a treewidth-k CQ Q'.
Question:	Is Q' a treewidth-k approximation of Q?

To solve Treewidth-k Approximation, we need to check two things:

1. $Q' \subseteq Q$; and
2. there is no $Q'' \in \mathsf{TW}(k)$ such that $Q' \subset Q'' \subseteq Q$.

The first subproblem is solvable in NP. Checking whether there is a query $Q'' \in \mathsf{TW}(k)$ not equivalent to Q' with $Q' \subseteq Q'' \subseteq Q$ is solvable in NP too. This means $T_Q \to T_{Q''} \not\rightleftarrows T_{Q'}$ 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 $T_{Q'}$ have treewidth at most k, 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 Treewidth-k Approximation, we need to solve an NP subproblem and a coNP subproblem. This means that the problem is in the complexity class DP: this is the class of problems (languages) which are intersections of an NP language and a coNP language [37]. It turns out that the problem is also hard for the class, even when $k = 1$.

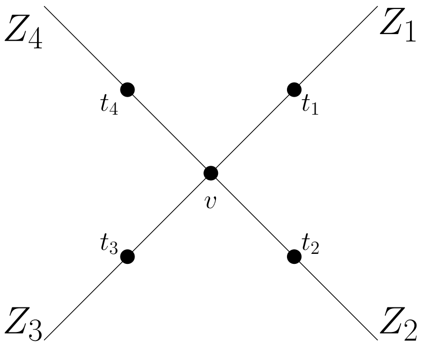


Figure 6: The structure of T.

Theorem 4.12 *The problem* Treewidth-1 Approximation *is* DP*-complete. It remains* DP*-complete even if Q and Q' are CQs over graphs, 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 [13]. 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 [13]. We only sketch the main ideas behind the proof here, since the complete proof is technical and lengthy. The complete proof can be found in the appendix.

Proof (Sketch) of Theorem 4.12: The class $\mathsf{TW}(1)$ over the vocabulary of graphs contains all acyclic directed graphs, i.e. the directed graphs whose underlying undirected graph contains no cycles. It thus suffices to show that the following problem is DP-complete:

Problem:	Graph Acyclic Approximation
Input:	a digraph G, an acyclic digraph T.
Question:	Is $G \to T$ and there is no acyclic digraph A such that $G \to A \rightleftarrows\!\!\!\!/\; T$?

In order to prove this, we consider the Exact Four Colorability problem: Given a graph G, decide if G is 4-colorable but not 3-colorable. It is known that this problem is DP-complete [40]. We provide a polynomial time reduction from Exact Four Colorability to Graph Acyclic Approximation.

Using techniques from [26], we construct two digraphs T and $\tilde{T}$. As Figure 6 shows, T consists of digraphs Z_1, Z_2, Z_3 and Z_4, whose only vertex in common is v. Each of the distinguished vertices t_1, t_2, t_3 and t_4 represents a possible color. The digraph $\tilde{T}$ has two distinguished vertices p and q.

The key properties of T and $\tilde{T}$ are the following:

1. If $h : \tilde{T} \to T$ is an homomorphism, then $h(p), h(q) \in \{t_1, t_2, t_3, t_4\}$ and $h(p) \neq h(q)$.
2. For any pair (t, t') of elements of $\{t_1, t_2, t_3, t_4\}$ with $t \neq t'$, there exists an homomorphism $h : \tilde{T} \to T$ such that $h(p) = t$ and $h(q) = t'$.
3. If $h : \tilde{T} \to T$ is an homomorphism and $h(p) = t_i$ or $h(q) = t_i$ for $1 \leq i \leq 4$, then the subgraph Z_i is contained in the homomorphic image of h.

4. For any pair (t, t') of elements of $\{t_1, t_2, t_3\}$ with $t \neq t'$, there exists an homomorphism $h : \tilde{T} \to T$ such that $h(p) = t$, $h(q) = t'$ and the homomorphic image of h is contained in $Z_1 \cup Z_2 \cup Z_3$.

Now, given a graph G we define $\varphi(G)$ to be the digraph obtained from G by replacing each edge $\{a, b\}$ with a fresh copy of $\tilde{T}$, where we identify p with a and q with b. This is the first step of our construction. Using properties (1)-(4) it is straightforward to prove the following:

Claim 4.13 *G is 4-colorable but not 3-colorable if and only if $\varphi(G) \to T$ and there is no homomorphism from $\varphi(G)$ to a proper subgraph of T.*

Indeed, if G is 4-colorable, we can consider the set $\{t_1, t_2, t_3, t_4\}$ to be the colors. Using property (2) it follows that $\varphi(G) \to T$. If there is an homomorphism h from $\varphi(G)$ to a proper subgraph of T, then there exists $t^* \in \{t_1, t_2, t_3, t_4\}$ such that $h(a) \neq t^*$ for all $a \in G$ (notice that each vertex of G can be viewed as a vertex of $\varphi(G)$); otherwise, by property (3), h would be surjective. Therefore, since the image of h over G contains at most 3 elements from $\{t_1, t_2, t_3, t_4\}$, using property (1) it follows that G is 3-colorable. This proves the forward direction. For the backward direction, notice that $\varphi(G) \to T$ implies that G is 4-colorable, again using property (1). Finally, if G is 3-colorable, we can consider the set $\{t_1, t_2, t_3\}$ to be the colors. Using property (4) define the standard homomorphism h from $\varphi(G) \to T$. Observe that h is not surjective as it is contained in $Z_1 \cup Z_2 \cup Z_3$.

As a corollary we obtain that the following problem is also DP-complete:

Problem:	Exact Acyclic Homomorphism
Input:	a digraph G, an acyclic digraph T.
Question:	Is $G \to T$ and $G \not\to S$ for every proper subgraph S of T?

At this point, $\varphi(G)$ does not provide a reduction from Exact Four Colorability to Graph Acyclic Approximation: It could be possible that there is no homomorphism from $\varphi(G)$ to a proper subgraph of T, but still there is an acyclic digraph A such that $\varphi(G) \to A \not\rightleftarrows T$. In other words, it is possible that the instance $(\varphi(G), T)$ belongs to Exact Acyclic Homomorphism but not to Graph Acyclic Approximation. We explain next how this problem can be avoided.

We modify $\varphi(G)$ in order to have the following two properties: (†) $(\varphi(G), T)$ is still a reduction from Exact Four Colorability to Exact Acyclic Homomorphism and (††) $(\varphi(G), T) \in$ Exact Acyclic Homomorphism if and only if $(\varphi(G), T) \in$ Graph Acyclic Approximation. This immediately implies DP-completness of Graph Acyclic Approximation. Our modification of $\varphi(G)$ satisfies the following property:

(P) *Suppose that A is an acyclic digraph, h an homomorphism from $\varphi(G)$ to A, g a surjective homomorphism from A to T, and for each $t \in \{t_1, t_2, t_3, t_4\}$ there exists a vertex u of G such that $g \circ h(u) = t$. Then there exists an homomorphism r from T to A.*

Observe that property (P) implies property (††): For the forward implication, asssume that $(\varphi(G), T) \in$ Exact Acyclic Homomorphism and, by contradiction, suppose that there exists an acyclic digraph A such that $\varphi(G) \to A \not\rightleftarrows T$. Let h be a homomorphism from $\varphi(G)$ to A and g a homomorphism from A to T. Notice that g must to be surjective, otherwise $g \circ h$ would be a homomorphism from $\varphi(G)$ to a proper subgraph of T. Moreover, it must be the case that for each $t \in \{t_1, t_2, t_3, t_4\}$ there exists a vertex u of G such that $g \circ h(u) = t$. Assume otherwise. Then using property (1) we can conclude that G is 3-colorable, and therefore that there exists a homomorphism from $\varphi(G)$ to a proper subgraph T (considering $\{t_1, t_2, t_3\}$ to be the colors and then using property (4)). Thus, using property (P) it follows that $T \to A$, which is a contradiction. The backward direction is trivially true, as the existence of a homomorphism h from $\varphi(G)$ to a proper subgraph of T implies the existence of an acyclic digraph A such that $\varphi(G) \to A \not\rightleftarrows T$: it suffices to consider the homomorphic image $\text{Im}(h)$ of $\varphi(G)$ in T. Notice that $\text{Im}(h)$ is acyclic, $\varphi(G) \to \text{Im}(h) \to T$ and $T \not\to \text{Im}(h)$, since T is a core.

The idea of the construction is to consider a digraph Q^* such that Z_1, Z_2, Z_3 and Z_4 are acyclic approximations of Q^*. Moreover, Q^* has an "initial vertex" x and a "terminal vertex" y. Now, we add to $\varphi(G)$ a fresh node v_0, and for each vertex u in $\varphi(G)$ that corresponds to a vertex of G we add a fresh copy of Q^*, identifying x with v_0 and y with u.

Using the structure of Q^* it can be shown that $\varphi(G)$ is still a reduction from Exact Four Colorability, i.e., that satisfies property (a). Moreover, it can be shown that it satisfies property (P). The intuition behind this is as follows: if A is an acyclic digraph as in property (P), we can choose u_1, u_2, u_3 and u_4 from G such that $g \circ h(u_i) = t_i$, for each $1 \leq i \leq 4$. Each u_i has an associated copy Q_i^* of Q^*. Each of the Q_i^*s must be mapped via h to an acyclic subgraph A_i of A. At the same time, each A_i must be mapped via g to Z_i. Thus, since Z_i is actually an acyclic approximation of Q^*, there exists a homomorphism r_i from Z_i to A_i, for each $1 \leq i \leq 4$. These homomorphisms can be combined to define a homomorphism r from T to A.

Our construction of $\varphi(G)$ does not necessarily yield a digraph that is a core. However, by applying a more involved construction we can force $\varphi(G)$ also to be a core. The technical details can be found in the appendix. □

5 Queries over Graphs

In this section we look more closely at queries over graphs. That is, the vocabulary σ has a single binary relation $E(\cdot, \cdot)$, interpreted as a directed graph. In this restricted scenario we prove several results about the structure of approximations in graph-based classes.

5.1 Acyclic approximations

We study acyclic (or, equivalently, treewidth-1) approximations in detail for graph queries. 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. Note that a query is acyclic if and only if its tableau has no oriented cycles of length 3 or more.

5.1.1 Boolean queries

These queries are of the form $Q() \text{:-} \ldots$ 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 $G \to \overleftrightarrow{K_2}$, 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 $Q^{\mathrm{triv}}()\text{:--}E(x,x)$ (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 $Q_2^{\mathrm{triv}}()\text{:--}E(x,y), E(y,x)$, whose tableau is $K_2^{\rightleftarrows}$. For some bipartite 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 [25], and then state the trichotomy result.

We now recall several notions from [25] that have been already introduced in the proof of Proposition 4.4. An oriented cycle is a digraph with nodes $u_1, \dots, u_n$ 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. Edges (u_i, u_{i+1}) and (u_n, u_1) are *forward edges* and edges (u_{i+1}, u_i) and (u_1, u_n) are *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.

Now we can state a trichotomy result for acyclic approximations of Boolean CQs over graphs.

Theorem 5.1 *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} (up to equivalence);*
- *is bipartite but not balanced, then Q's only acyclic approximation (up to equivalence) 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 $E(x,y), E(y,x)$.*

Proof: Suppose the tableau T_Q is not bipartite and let Q' be an acyclic approximation of Q. If $T_{Q'}$ has no loops then, by acyclicity, it is bipartite; hence $T_Q \to T_{Q'} \to K_2^{\rightleftarrows}$, which contradicts non-bipartiteness of T_Q. Hence $T_{Q'}$ has a loop, and Q' is equivalent to Q^{triv}.

Let T_Q be bipartite and not balanced, and let Q' be an acyclic approximation of Q. We prove that $T_{Q'}$ is homomorphically equivalent to $K_2^{\rightleftarrows}$. Note that $T_{Q'}$ has no loops: otherwise $T_Q \to K_2^{\rightleftarrows} \not\rightleftarrows T_{Q'}$, and Q' is not an approximation. Thus, $T_{Q'}$ is bipartite and $T_{Q'} \to K_2^{\rightleftarrows}$. For the converse, assume $K_2^{\rightleftarrows} \not\to T_{Q'}$. Then $K_2^{\rightleftarrows}$ is not subgraph of $T_{Q'}$. Since Q' is acyclic, this implies that $T_{Q'}$ is balanced, and the following claim shows that $T_Q \to T_{Q'}$ implies that T_Q is balanced too, which is a contradiction.

Claim 5.2 *Balanced digraphs are closed under inverse homomorphisms. That is, if $G \to H$ and H is balanced, then G is balanced.*

Proof: We use the following characterization of balanced digraphs from [25]: G is balanced if and only if $G \to \vec{P}_k$, for some $k \geq 1$, where $\vec{P}_k$ denotes the directed path of length k. Now, suppose $G \to H$ and H is balanced. Then H is homomorphic to a directed path, and, by composition, G is homomorphic to such directed path as well. Therefore, G is balanced. □

Finally, let T_Q be bipartite and balanced, and let Q' be an acyclic approximation of Q. As above, we see that Q' is not equivalent to the trivial CQ. We now prove that $K_2^{\rightleftarrows}$ is not a subgraph of $T_{Q'}$, implying the result. Assume otherwise; since T_Q is balanced, we have that $T_Q \rightarrow \vec{P}_k$ for some k [25]. Since $\vec{P}_k \not\rightleftarrows K_2^{\rightleftarrows}$, we have $T_Q \rightarrow \vec{P}_k \not\rightleftarrows T_{Q'}$, which contradicts the minimality of $T_{Q'}$. □

Note that the conditions used in the theorem – being bipartite and balanced – can be checked in polynomial time [25, 44].

As a corollary to the proof of the previous theorem, we obtain:

Corollary 5.3 *Let Q be a Boolean cyclic CQ over graphs. Then all minimized acyclic approximations of Q have strictly fewer joins than Q.*

Proof: Let Q' be a minimized acyclic approximation of Q. It suffices to show that the tableau $T_{Q'}$ has strictly fewer edges than the tableau T_Q. We denote by $|E(T_{Q'})|$ and $|T_{Q'}|$ the numbers of edges and nodes in $T_{Q'}$, respectively.

If T_Q is not bipartite, or if is bipartite but not balanced, using Theorem 5.1 and the fact that T_Q has at least 3 edges (T_Q is cyclic), we have the result.

Now, if T_Q is bipartite and balanced we know that $K_2^{\rightleftarrows} \not\subseteq T_{Q'}$ (Theorem 5.1) and since $T_{Q'}$ is a core (Q' is a minimized CQ), we know that $T_{Q'}$ is a homomorphic image of T_Q, via some h, that is, $T_Q \xrightarrow{h} T_{Q'}$, where $\mathrm{Im}(h) = T_{Q'}$. It suffices to show that there are two edges in T_Q which are mapped via h to the same edge in $T_{Q'}$. Since T_Q is cyclic there exists a connected component of T_Q which is cyclic, namely H (connected in the sense that H^{u} is connected). Let h' be the restriction of h to H. Note that $\mathrm{Im}(h')$ is connected and acyclic, and since $K_2^{\rightleftarrows} \not\subseteq \mathrm{Im}(h')$, it follows that $|E(\mathrm{Im}(h'))| = |\mathrm{Im}(h')| - 1 \leq |H| - 1$. Finally, observe that $|E(H)| > |H| - 1$, otherwise since H is connected, then H would be acyclic, which is a contradiction. Thus, $|E(\mathrm{Im}(h'))| < |E(H)|$ and h' maps two edges to one edge in $T_{Q'}$. In particular, h maps two edges to one edge in $T_{Q'}$. □

Reformulating our results in terms of graphs, we obtain:

Corollary 5.4 *For every cyclic digraph G and its acyclic approximation T, the core of T has strictly fewer edges than G. Moreover, T is not homomorphically equivalent to a single loop iff G is bipartite.*

Theorem 5.1 says that the most interesting case, for graph queries, is when the tableau is bipartite and balanced (as we already mentioned in the introduction, for relations of higher arity, such restrictions need not be imposed). A natural question is whether CQs with such tableaux are still intractable (i.e., whether it still makes sense to approximate them). We prove next that this is the case.

Proposition 5.5 *The combined complexity of evaluating Boolean CQs over graphs whose tableaux are bipartite and balanced is* NP*-complete.*

Proof: Note that any balanced digraph is bipartite, thus bipartite and balanced digraphs are exactly balanced digraphs. It suffices to prove (due to the correspondence between CQs and tableaux) that the following equivalent problem is NP-complete: Given a balanced digraph G and a digraph H, check if $G \rightarrow H$.

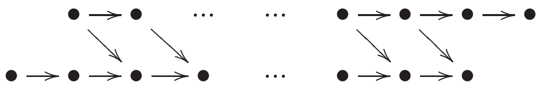
It is known that there exists an oriented tree T (undirected tree plus orientation in the edges) such that the following problem is NP-complete [25]: Given a digraph G, decide if

$G \to T$. This problem remains NP-complete even for G balanced, since a digraph homomorphic to T must be balanced (using Claim 5.2 and the fact that any oriented tree is balanced). Thus the result follows and, in fact, the problem is NP-complete even if H is a fixed oriented tree. □

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 is 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 [36] (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 5.6 *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.*

Proof: Let G_k be the digraph constructed as follows. Take two disjoint copies of a directed path of length k, the first on nodes $x_0, x_1, \ldots, x_k$, and the second on nodes $y_0, y_1, \ldots, y_k$. Then add edges $(x_0, y_2), (x_1, y_3), \ldots, (x_i, y_{i+2}), \ldots, (x_{k-2}, y_k)$. The picture is shown below.



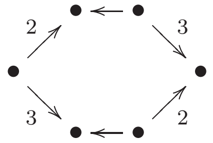
It can then be shown that $\vec{P}_{k+1}$, the path of length $k+1$, has two properties, as long as $k \geq 3$:

1. $G_k \to \vec{P}_{k+1}$; and
2. there is no digraph G such that $G_k \not\rightleftarrows G \not\rightleftarrows \vec{P}_{k+1}$.

The first property is immediate. To show the second, we use the standard construction for gaps in the lattice of digraphs [36]: we take $\vec{P}_{k+1}$ and compute its dual F_k (following the procedure in [36] or in Theorem 3.35 in [25]), that has the following property: for every digraph H, either $H \to F_k$ or $\vec{P}_{k+1} \to H$ holds. Then we take the digraph $F_k \times \vec{P}_{k+1}$. Of course $F_k \times \vec{P}_{k+1} \to \vec{P}_{k+1}$; the results of [36] further tell us that there is no digraph between $F_k \times \vec{P}_{k+1}$ and $\vec{P}_{k+1}$ in the $\to$ ordering. We then compute the core of $F_k \times \vec{P}_{k+1}$, which happens to be G_k. We omit the tedious calculations.

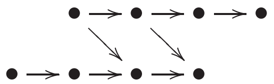
With the properties 1 and 2 above established, we simply take Q_n to be the query whose tableau is G_{n+2} (as the properties above are guaranteed starting with G_3) and Q'_n to be the CQ whose tableau is $\vec{P}_{n+3}$. □

Example 5.7 Consider a Boolean query Q whose tableau is the digraph below, in which number k above an edge represents a path of length k:



This digraph is bipartite and balanced, so Theorem 5.1 tells us that it 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 $Q'() \text{:–} P_4(x', x, y, z, u)$ 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. □

5.1.2 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 5.8 *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 $E(x, x)$.*

Proof: Suppose that the tableau T_Q of Q is not bipartite and let Q' be an acyclic approximation of Q. Suppose that $T_{Q'}$ has no loops. Then $T_{Q'}$ is bipartite (since $T_{Q'}^{\mathrm{u}}$ is acyclic), implying that $T_{Q'} \rightarrow K_2^{\rightleftarrows}$. Since $T_Q \rightarrow T_{Q'}$, we have that $T_Q \rightarrow K_2^{\rightleftarrows}$ as well. If follows that T_Q is bipartite, which is impossible. Therefore, $T_{Q'}$ has loops, i.e, Q' has a subgoal of the form $E(x, x)$.

Now, suppose that the tableau T_Q of Q is bipartite. Using an argument similar to the proof of Theorem 4.1 we shall prove that there exists an acyclic approximation of Q without a subgoal of the form $E(x, x)$, i.e, whose tableau has no loops. Let $\mathcal{A}_Q$ be the set of all digraphs with distinguished elements $(H, \bar{u})$ such that H^{u} is acyclic and has no loops, $(T_Q, \bar{x}) \rightarrow (H, \bar{u})$ and $|H| \leq |T_Q| + 1$ (where $|G|$ is the number of nodes of the digraph G). Clearly, $\mathcal{A}_Q$ is finite. Moreover, it is not empty, since there exists a homomorphism h from T_Q to $K_2^{\rightleftarrows}$ (this follows from the fact that T_Q is bipartite) and this implies that $(K_2^{\rightleftarrows}, h(\bar{x}))$ is contained in $\mathcal{A}_Q$. We can pick a minimal element (with respect to $\rightarrow$) $(H', \bar{u}')$ from $\mathcal{A}_Q$. We shall show that $Q'(\bar{u}')$, the CQ whose tableau is $(H', \bar{u}')$, is an acyclic approximation of Q. Suppose not, then there exists an acyclic CQ $Q''(\bar{x}'')$ such that $(T_Q, \bar{x}) \xrightarrow{g} (T_{Q''}, \bar{x}'') \not\rightleftarrows (H', \bar{u}')$. Observe that $(\mathrm{Im}(g), \bar{x}'')$ has no loops (otherwise, $(T_{Q''}, \bar{x}'') \not\rightarrow (H', \bar{u}')$), $\mathrm{Im}(g)^{\mathrm{u}}$ is acyclic (since $\mathrm{Im}(g)$ is subgraph of $T_{Q''}$), $(T_Q, \bar{x}) \rightarrow (\mathrm{Im}(g), \bar{x}'')$ and $|\mathrm{Im}(g)| \leq |T_Q|$. Therefore, $(\mathrm{Im}(g), \bar{x}'')$ is contained in $\mathcal{A}_Q$. Finally, note that $(\mathrm{Im}(g), \bar{x}'') \not\rightleftarrows (H', \bar{u}')$, which contradicts the minimality of $(H', \bar{u}')$ in $\mathcal{A}_Q$. Thus, Q' is an acyclic approximation of Q and since $T_{Q'} = H'$ has no loops, we have the result. □

Notice that the previous theorem actually generalizes Theorem 5.1, since a boolean CQ is trivial if and only if its tableau has a loop. However, when the query is not Boolean the latter is not necessarily true.

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

Proof: Consider the following query:

$$Q(x_1, x_2, x_3) :- E(x_1, x_2), E(x_2, x_3), E(x_3, x_4), E(x_4, x_1).$$

This query is minimized. Its tableau, which we denote by (G, x_1, x_2, x_3), contains an oriented cycle on nodes x_1, x_2, x_3, x_4, with x_1, x_2, x_3 being distinguished nodes.

Let $G' = \langle V', E' \rangle$ be a digraph containing nodes x_1', x_2', x_3' (not necessarily distinct) so that (G', x_1', x_2', x_3') is a tableau of an acyclic approximation of Q. We know that (G', x_1', x_2', x_3') is an image of some homomorphism h defined on G. Note that if the homomorphism h were one-to-one, then all the edges of G would be present in G' and thus G' would be cyclic. Hence, $|V'| \leq 3$.

By definition, $h(x_i) = x_i'$ for $1 \leq i \leq 3$. Consider first a homomorphism h so that x_i's, for $1 \leq i \leq 3$, are distinct. Then there are three possibilities where x_4 could be mapped:

- If $h(x_4) = x_1'$ or $h(x_4) = x_3'$, then $\mathrm{Im}(h)$ is a cyclic digraph, contradicting the assumption.
- If $h(x_4) = x_2'$, we get a digraph consisting of two copies of $\overrightleftarrows{K_2}$, i.e., a graph G_0 with nodes x_i', $1 \leq i \leq 3$, and edges $(x_1', x_2'), (x_2', x_1')$ as well as $(x_2', x_3'), (x_3', x_2')$. The corresponding query $Q_0(x_1', x_2', x_3')$:–$E(x_1', x_2'), E(x_2', x_1'), E(x_2', x_3'), E(x_3', x_2')$ has the same 3 joins as the original query.

Next we see what happens when h collapses some of x_i's, for $1 \leq i \leq 3$. First we look at the cases when h collapses two of those. Suppose we collapse x_1 and x_2, i.e., $h(x_1) = h(x_2)$ (and thus $x_1' = x_2'$) and $x_3' \neq x_1'$. There are three possibilities for x_4:

- If $h(x_4) = x_1' = x_2'$, then $\mathrm{Im}(h)$ is the digraph G_1 with nodes x_1', x_3' and edges $(x_1', x_3'), (x_3', x_1'), (x_1', x_1')$, with distinguished nodes (x_1', x_1', x_3'). It is routine to check that $(G_0, x_1', x_2', x_3') \not\rightleftarrows (G_1, x_1', x_1', x_3')$, and hence the result of this homomorphism is not an acyclic approximation.
- If $h(x_4) = x_3'$, then the image of h is a digraph with 4 edges, hence corresponding to a query with 3 joins, same as in the original Q.
- If $h(x_4)$ is different from x_1', x_2', x_3', then $\mathrm{Im}(h)$ has a cycle.

The case when x_2 and x_3 are collapsed to the same node by h is completely symmetric. Now assume that h collapses x_1 and x_3, i.e., $x_1' = x_3'$. Again there are three cases.

- If $h(x_4) = x'_2$, then the image of h is (G_2, x'_1, x'_2, x'_1) where G_2 is a copy of $K_2^{\rightleftarrows}$ on x'_1, x'_2. In this case again we easily verify $(G_0, x'_1, x'_2, x'_3) \not\rightleftarrows (G_2, x'_1, x'_2, x'_1)$, meaning that the latter cannot be an acyclic approximation.

- If $h(x_4) = x'_1$, then the image of the original tableau is the same digraph as in the previous case, plus a loop on x'_1. Hence, the same argument as above shows that it cannot be an acyclic approximation.

- Otherwise, if $h(x_4)$ is different from x'_1, x'_2, x'_3, then $\mathrm{Im}(h)$ is a union of two copies of $K_2^{\rightleftarrows}$, and thus it has the same 3 joins as the original query.

Finally, if h collapses all nodes (say to x'_1), there are two possibilities.

- If $h(x_4) = x'_1$, then we end up with a trivial query with the tableau $(K_1^{\circlearrowleft}, x'_1, x'_1, x'_1)$, where $K_1^{\circlearrowleft}$ is the digraph that consists of a directed loop on a single element, and clearly (G_0, x'_1, x'_2, x'_3) $\not\rightleftarrows$ $(K_1^{\circlearrowleft}, x'_1, x'_1, x'_1)$.

- If $h(x_4) = x'_4 \neq x'_1$, then $\mathrm{Im}(h)$ consists of a copy of $K_2^{\rightleftarrows}$ on x'_1 and x'_4, as well as a loop on x'_1. In this case we again easily verify that $(G_0, x'_1, x'_2, x'_3) \not\rightleftarrows (\mathrm{Im}(h), x'_1, x'_1, x'_1)$.

Hence, in all the cases when the number of joins is reduced, the resulting query is not an acyclic approximation, which proves the proposition. □

5.2 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 5.10 *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 $E(x,x)$.*

Proof: We use similar arguments to the proof of Theorem 5.8. We also use the well-known result that each digraph without loops of treewidth at most k is $(k+1)$-colorable, or, equivalently, is homomorphic to $K_{k+1}^{\rightleftarrows}$.

Assume first that the tableau T_Q of Q is not $(k+1)$-colorable and let Q' be a $\mathsf{TW}(k)$-approximation of Q. Assume for the sake of contradiction that $T_{Q'}$ has no loops. Then $T_{Q'}$ is $(k+1)$-colorable, implying that $T_{Q'} \rightarrow K_{k+1}^{\rightleftarrows}$. Since $T_Q \rightarrow T_{Q'}$, we have that $T_Q \rightarrow K_{k+1}^{\rightleftarrows}$ as well. If follows that T_Q is $(k+1)$-colorable, which is a contradiction. Therefore, $T_{Q'}$ have loops, i.e, it has a subgoal of the form $E(x,x)$.

Assume, on the other hand, that the tableau T_Q of Q is $(k+1)$-colorable and let $\mathcal{A}_Q$ be the set of all digraphs with distinguished elements $(H, \bar{u})$ such that H^{u} has treewidth at most k and has no loops, $(T_Q, \bar{x}) \rightarrow (H, \bar{u})$ and $|H| \leq |T_Q| + k$. Clearly, $\mathcal{A}_Q$ is finite. Moreover, it is not empty, since there exists a homomorphism h from T_Q to $K^{\rightleftarrows}_{k+1}$ (T_Q is $(k+1)$-colorable) and this implies that $(K^{\rightleftarrows}_{k+1}, h(\bar{x}))$ is contained in $\mathcal{A}_Q$. We pick a minimal element (with respect to $\rightarrow$) $(H', \bar{u}')$ from $\mathcal{A}_Q$. We shall show that $Q'(\bar{u}')$, the CQ whose tableau is $(H', \bar{u}')$, is a $\mathsf{TW}(k)$-approximation of Q. Suppose not, then there exists a CQ $Q''(\bar{x}'')$ with treewidth at most k such that $(T_Q, \bar{x}) \xrightarrow{g} (T_{Q''}, \bar{x}'') \not\rightleftarrows (H', \bar{u}')$. Observe that $(\mathrm{Im}(g), \bar{x}'')$ has no loops (otherwise, $(T_{Q''}, \bar{x}'') \not\rightarrow (H', \bar{u}')$), $\mathrm{Im}(g)^{\mathrm{u}}$ has treewidth at most k (since $\mathrm{Im}(g)$ is subgraph of $T_{Q''}$), $(T_Q, \bar{x}) \rightarrow (\mathrm{Im}(g), \bar{x}'')$ and $|\mathrm{Im}(g)| \leq |T_Q|$. Therefore, $(\mathrm{Im}(g), \bar{x}'')$ is contained in $\mathcal{A}_Q$. Finally, note that $(\mathrm{Im}(g), \bar{x}'') \not\rightleftarrows (H', \bar{u}')$, which is a contradiction with the minimality of $(H', \bar{u}')$ in $\mathcal{A}_Q$. Thus, Q' is a $\mathsf{TW}(k)$-approximation of Q and since $T_{Q'} = H'$ has no loops, we have the result. □

Recall that a Boolean CQ $Q^{\mathrm{triv}}() \text{:–} E(x,x)$ 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 5.11 *A Boolean CQ Q over graphs has a nontrivial $\mathsf{TW}(k)$-approximation iff its tableau T_Q is $(k+1)$-colorable.*

Proof: Let Q' be a $\mathsf{TW}(k)$-approximation of Q. Assume first that the tableau T_Q of Q is not $(k+1)$-colorable. Using Theorem 5.10 we know that $T_{Q'}$ has loops. Therefore, Q' is equivalent to Q^{triv}. Now, assume that the tableau T_Q of Q is $(k+1)$-colorable. If $T_{Q'}$ has loops we have that $T_Q \rightarrow K^{\rightleftarrows}_{k+1} \not\rightleftarrows T_{Q'}$, which is a contradiction with the minimality of $T_{Q'}$ because $K^{\rightleftarrows}_{k+1}$ is of treewidth k. Then $T_{Q'}$ has no loops, i.e., is not equivalent to Q^{triv}. □

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 5.1). 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 5.12 *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.*

Proof: Let $k > 1$. We shall prove that there is a polynomial reduction from the $(k+1)$-coloring problem to our problem. Let G be a graph. The reduction returns the CQ $\varphi(G)$ whose tableau is $\vec{G} + K^{\rightleftarrows}_{k+1}$, where $+$ denotes disjoint union and $\vec{G}$ is the directed version of G obtained by replacing each undirected edge $\{a, b\}$ in G with both (a, b) and (b, a). We prove next that G is $(k+1)$-colorable iff Q^{triv}_{k+1} is a $\mathsf{TW}(k)$-approximation of $\varphi(G)$.

Suppose that G is $(k+1)$-colorable, then $\vec{G} \rightarrow K^{\rightleftarrows}_{k+1}$, implying that the tableau of $\varphi(G)$ is homomorphically equivalent to $K^{\rightleftarrows}_{k+1}$, the tableau of Q^{triv}_{k+1}. Therefore, $\varphi(G)$ is equivalent to Q^{triv}_{k+1}, and, thus, Q^{triv}_{k+1} is a $\mathsf{TW}(k)$-approximation of $\varphi(G)$.

Suppose, on the other hand, that Q^{triv}_{k+1} is a $\mathsf{TW}(k)$-approximation of $\varphi(G)$. In particular, we have that $T_{\varphi(G)} \rightarrow T_{Q^{\mathrm{triv}}_{k+1}}$, implying that $\vec{G} \rightarrow K^{\rightleftarrows}_{k+1}$.Therefore, G is $(k+1)$-colorable. □

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.

5.3 Graphs vs higher-arity relations

We now contrast graph queries with those that use relations of higher arity, to demonstrate that higher arity gives a lot more freedom for finding interesting approximations. Since we deal with graph-based classes of queries, we shall be looking for the strongest, i.e., $\mathsf{TW}(1)$-approximations. Suppose we have an arbitrary query Q with n variables. Then the maximum possible treewidth of Q is $n-1$. We look for approximation that decrease the treewidth in the strongest possible way. That is, we say that Q' is a *strong treewidth approximation* of Q if Q' is a $\mathsf{TW}(1)$-approximation of Q, and Q has the maximum possible treewidth > 1 (i.e., its treewidth is the number of variables minus 1).

For this subsection we deal with Boolean queries, and assume that the vocabulary consists of one m-ary relation R; when $m = 2$, we deal with graphs. In fact, in the case of graphs the notion of strong treewidth approximation trivializes: if Q' is a strong treewidth approximation of Q, then Q' is equivalent to the trivial query Q^{triv}. Indeed, if Q is of maximum possible treewidth > 1, then $G(Q)$ is K_n (perhaps with some loops), and hence for $n > 2$ it is not bipartite, implying triviality of $\mathsf{TW}(1)$-approximation.

However, when $m > 2$, there are many possible strong treewidth approximations, even in cases that appear to be close to the cases of graphs admitting only trivial approximations. So for now, fix $m > 2$ and assume that Q' is a strong treewidth approximation of Q. First observe that Q' can have at most 2 variables: indeed, since $G(Q)$ is a K_n (perhaps with loops), then if $G(Q')$ has at least 3 nodes, it would have a triangle and hence be of treewidth at least 2. So we call a Boolean query Q' over an m-ary relation R a *potential strong treewidth approximation* if $G(Q')$ has at most 2 nodes.

Proposition 5.13 *Let Q' be a potential strong treewidth approximation. Assume that Q' is non-trivial. Then, for every $n > m$, there is a conjunctive query Q with n variables such that Q' is a strong treewidth approximation of Q. Moreover, if the query Q' has k atoms, then Q can be chosen to have at most $k + \frac{n(n-1)}{2} - 1$ atoms.*

Proof: Observe that in every atom in Q' one variable occurs at least twice. Assume first that there is an atom in which some variable occurs exactly twice, say, an atom $R(x, \ldots, x, y, y)$. Then in Q we put atoms $R(x_1, \ldots, x_1, x_i, x_j)$ for all $2 \leq i \leq j \leq n$, assuming Q has variables $x_1, \ldots, x_n$. For every other atom $R(x, \ldots, x, y, \ldots, y)$ with r occurrences of y we put in Q the atom $R(x_1, \ldots, x_1, x_2, \ldots, x_{r+1})$ (of course variables can occur in an arbitrary order; we simply replace all the occurrences of x with x_1 and r occurrences of y with $x_2, \ldots, x_{r+1}$, in the same order in which the y's occur in the atom). The construction ensures that $G(Q)$ is K_n and it is easy to verify that Q has at most $k + n - 2 + \frac{(n-1)(n-2)}{2} = k + \frac{n(n-1)}{2} - 1$ atoms, as only one atom in Q' generates multiple atoms in Q. The mapping sending x_1 to x and every x_i with $i > 1$ to y is a homomorphism showing $Q' \subseteq Q$. If there were $\mathsf{TW}(1)$-approximation Q'' of Q with $Q' \subset Q'' \subset Q$, then $G(Q'')$ would be K_2 (perhaps with loops), so a homomorphism of the tableau of Q'' into the tableau of Q' can only be the identity, or swapping the roles of variables x and y. Using this one easily verifies that Q' is an approximation.

If we do not have an atom with exactly two occurrences of a variable, then pick an atom with a minimum number p of repetitions of a variable, say $R(x, \ldots, x, y, \ldots, y)$, where y occurs p times. Then we replace it by putting in Q atoms $R(x_1, \ldots, x_1,\ x_2, \ldots, x_{p-1},\ x_i, x_j)$ with $p \leq i < j \leq n$ (where x_1s correspond to the positions of x). In addition, we put in Q atoms $R(x_1, \ldots, x_1,\ x_i, \ldots, x_i,\ x_i, x_i)$ whenever $2 \leq i \leq n$. The proof then is the same (only the number of atoms in Q gets smaller). □

Recall that for Boolean graph queries ($m = 2$), $\mathsf{TW}(1)$-approximations strictly decrease the number of joins. Beyond graphs, however, this need not be the case.

Proposition 5.14 *For every $k \geq 3$, one can find a relation symbol R of arity $m > 2$, and two minimized conjunctive queries Q and Q' over R with the same number of joins such that Q' is a strong treewidth approximation of Q.*

Proof: We take m, the arity of R, to be equal to k. In Q, the first three atoms are $R(x_1, x_2, x_3, x_4, \ldots, x_k), R(x_2, x_1, x_{k+1}, x_4, \ldots, x_k)$, and $R(x_3, x_{k+1}, x_1, x_4, \ldots, x_k)$. The next $k - 3$ atoms are of the form $R(x_j, x_j, \ldots, x_j, x_1, x_j, \ldots, x_j)$, where x_1 appears in the jth position; here $4 \leq j \leq k$. In Q', we have k atoms of the form $R(x, y, \ldots, y),\ R(y, x, y, \ldots, y),\ \ldots,\ R(y, \ldots, y, x)$, i.e., x appears once, every time in a different position. It is straightforward to verify all three conditions of the proposition. □

A slight drawback of the previous result is that it requires relations of high arity. But we can show that already for *ternary* relations, the behavior of strong treewidth approximations is drastically different from the graph case.

Consider the query $Q_{tr}()$:–$R(x, y), R(y, z), R(z, x)$; it states that a graph contains a triangle. Its graph is not bipartite, since $G(Q) = K_3$, and hence it only has trivial $\mathsf{TW}(1)$-approximation.

We now look at ternary relations R. We call an instance R of a ternary relation an *almost-triangle* if there is an element that belongs to every triple of R, and when it is removed from every triple, the resulting pairs form a triangle. For instance, $(4, 1, 2), (4, 2, 3), (4, 3, 1)$ is an almost-triangle: when we remove 4 for each of the triples, we end up with the pairs $(1, 2), (2, 3), (3, 1)$ which form a triangle.

Proposition 5.15 *There is a minimized conjunctive query Q over a ternary relation R that uses 4 variables, has maximum treewidth 3, such that:*

- *the tableau T_Q of Q is an almost-triangle; and*
- *Q has a strong treewidth approximation Q' with the same number of joins as Q.*

Proof: We define Q over variables x_1, x_2, x_3, x_4 as follows:

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

Its tableau is an almost triangle (just remove x_1). Furthermore $G(Q) = K_4$ and thus it has treewidth 3. It is also minimized.

We then look at

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

It is routine to verify that Q' is a strong treewidth approximation of Q satisfying all conditions of the proposition. □

Thus, indeed, the behavior of treewidth approximations is already drastically different for the case of ternary relations, compared to graphs.

6 Hypergraph-based Approximations

We now switch to study approximations in tractable classes defined by restricting the hypergraph $\mathcal{H}(Q)$ of a CQ Q: The nodes of $\mathcal{H}(Q)$ 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 $\{x_1, \ldots, x_n\}$. If $\mathcal{C}$ is a class of hypergraphs, then a query Q is a *hypergraph-based $\mathcal{C}$-query* if $\mathcal{H}(Q) \in \mathcal{C}$. In general, graph-based and hypergraph-based classes of CQs are incompatible: there are graph-based classes that are not hypergraph-based, and vice versa [16].

The oldest tractability criterion for CQs, acyclicity [43], is a hypergraph-based notion (see the definition in Section 3). It corresponds to the class of CQs Q such that $\mathcal{H}(Q)$ belongs to the class AC of acyclic hypergraphs, or, in other words, Q is a hypergraph-based AC-query. Analogs of bounded treewidth for hypergraphs were defined in [20, 21]; those notions of bounded *hypertree width* and *generalized hypertree width* properly extended acyclicity and led to tractable classes of CQs over arbitrary vocabularies.

We look at *hypergraph-based $\mathcal{C}$-approximations*, i.e., approximations in the class of hypergraph-based $\mathcal{C}$ queries. Our first goal is to have a general result about the existence of approximations that will apply to both acyclicity and bounded (generalized) 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 $\langle V', \{e \cap V' \mid e \in \mathcal{E}\} \rangle$.

  For instance, take again the hypergraph $\mathcal{H}$ with hyperedges $\{a, b, c\}, \{a, b\}, \{b, c\}, \{a, c\}$. Then the only induced subhypergraph of $\mathcal{H}$ that contains all of its two-element edges is $\mathcal{H}$ itself.

- *Closure under edge extensions:* if $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is in $\mathcal{C}$ and $\mathcal{H}'$ is obtained by adding new nodes 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 6.1 *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.

Proof: We make use of the following claim:

Claim 6.2 *Let $\mathcal{C}$ be a class of hypergraphs closed under induced subhypergraphs and edge extensions. Let $Q(\bar{x})$ be a CQ and $Q'(\bar{x}')$ be a hypergraph-based $\mathcal{C}$-query, both over vocabulary σ, such that $Q' \subseteq Q$. Let us denote by n be the number of variables in Q, by ℓ the number of relation symbols in σ, and by m the maximum arity of a symbol in σ. Then there exists a hypergraph-based $\mathcal{C}$-query Q'', with at most $n + (m-1)^2 n^{m-1}$ variables and at most $\ell \cdot n^m$ joins, such that $Q' \subseteq Q'' \subseteq Q$.*

Proof: Since $Q' \subseteq Q$ it is the case that $(T_Q, \bar{x}) \xrightarrow{h} (T_{Q'}, \bar{x}')$. Consider the database T over σ constructed as follows: Let U be the active domain of $\text{Im}(h)$. For each relation symbol $R \in \sigma$, add to R^T all tuples $\bar{t} \in R^{T_{Q'}}$ such that $U_{\bar{t}} \subseteq U$, where $U_{\bar{t}}$ is the set of elements that occur in $\bar{t}$. Notice that $\mathcal{H}(T)$ is not necessarily an induced subhypergraph of $\mathcal{H}(T_{Q'})$, as there might be hyperedges e in $\mathcal{H}(T_{Q'})$ that contain elements in U but $e \cap U$ is not a hyperedge in $\mathcal{H}(T)$. As such, it is not possible to apply the first closure condition of $\mathcal{C}$ in order to infer that $\mathcal{H}(T)$ belongs to $\mathcal{C}$. A straightforward alternative seems to consider the subhypergraph $\mathcal{H}_U$ of $\mathcal{H}(T_{Q'})$ induced by U. The problem then is that there might be hyperedges in $\mathcal{H}_U$ that do not belong to $\mathcal{H}(T_{Q'})$, and, thus, it is not clear how to define from $\mathcal{H}_U$ a query Q'' that contains Q', which is one of the properties we are looking for. We will have to follow a different strategy based on the second closure property of $\mathcal{C}$ – closure under edge extensions – to build the desired query Q''.

A nonempty subset X of U is an *extended subset* if (1) there is no tuple $\bar{t}$ in some relation of T such that $U_{\bar{t}} = X$, and (2) there exists a tuple $\bar{s}$ in some relation of $T_{Q'}$ such that $U_{\bar{s}} \not\subseteq U$ and $U_{\bar{s}} \cap U = X$. That is, the extended subsets X of U are precisely the hyperedges of $\mathcal{H}_U$ that are not hyperedges of $\mathcal{H}(T_{Q'})$, and, thus, of $\mathcal{H}(T)$. Using this notion we define a database T' that is obtained by extending T as follows: For each extended subset X of U, choose an arbitrary tuple $\bar{s}_X$ and an arbitrary symbol S_X in σ such that (i) $\bar{s}_X$ belongs to the interpretation of S_X in $T_{Q'}$, (ii) $U_{\bar{s}_X} \not\subseteq U$ and (iii) $U_{\bar{s}_X} \cap U = X$ (we know this tuple exists by definition of extended subset). Add to the interpretation of S_X in T' the tuple $\bar{s}'_X$ that is obtained from $\bar{s}_X$ by renaming all elements in $U_{\bar{s}_X} \setminus U = \{z_1, z_2, ..., z_r\}$ by new elements $\{z'_1, z'_2, ..., z'_r\}$.

Observe that the hypergraph $\mathcal{H}(T')$ is in $\mathcal{C}$ since it is obtained from $\mathcal{H}(T_{Q'})$ in the following way: Take the induced subhypergraph $\mathcal{H}_U$ of $\mathcal{H}(T_{Q'})$ whose node set is U, and then for each extended subset $X \subseteq U$ do an edge extension in X (recall that each extended subset X of U is a hyperedge of $\mathcal{H}_U$). Using the closure properties of $\mathcal{C}$ and the fact that $\mathcal{H}(T_{Q'}) \in \mathcal{C}$, it follows that $\mathcal{H}(T') \in \mathcal{C}$.

Notice that h is a homomorphism from $(T_Q, \bar{x})$ to $(T', h(\bar{x})) = (T', \bar{x}')$ since $\text{Im}(h)$ is contained in T'. Thus, $(T_Q, \bar{x}) \rightarrow (T', \bar{x}')$. We prove next that $(T', \bar{x}') \rightarrow (T_{Q'}, \bar{x}')$. In fact, let T^* be the database that is obtained from T by adding, for each extended subset X of U, the tuple $\bar{s}_X$ to the interpretation of S_X, where S_X and $\bar{s}_X$ are as defined before for X. By definition, T^* is contained in $T_{Q'}$, and hence, the identity mapping is a homomorphism from $(T^*, \bar{x}')$ to $(T_{Q'}, \bar{x}')$. Notice that T' is (up to isomorphism) the database that is obtained from T^* by taking each tuple of the form $\bar{s}_X$ in T^* and replacing each element z in $\bar{s}_X$ that does not belong to U with a fresh element z'. In particular, two different facts of T' can only share elements that belong to U. It can be easily seen then that the mapping $h : T' \to T_{Q'}$ defined as the identity on U and as $h(z') = z$ for each element z' in T' that does not belong to U and replaced element z in T^*, is a homomorphism. Further, $\bar{x}'$ belongs to T, and, thus, to U, and hence h is the identity on $\bar{x}'$. We conclude that $(T_Q, \bar{x}) \to (T', \bar{x}') \to (T_{Q'}, \bar{x}')$.

Recall that for each extended subset X of U it is the case that there exists a tuple $\bar{s}$ in some relation of $T_{Q'}$ such that $U_{\bar{s}} \nsubseteq U$ and $U_{\bar{s}} \cap U = X$. Thus, $|U_{\bar{s}} \setminus X| \geq 1$. Since $|U_{\bar{s}}| \leq m$ (because $|U_{\bar{s}}|$ is bounded by the maximum arity of a tuple in $T_{Q'}$, and, thus, by the maximum arity m of a symbol in σ), we conclude that for each extended subset X of U it is the case that $|X| \leq m - 1$. Furthermore, for each extended subset X of size $i \geq 1$ we add at most $m - i$ fresh elements to the domain of T to construct T'. It follows that the number of elements in T' that do not belong to T is at most

$$\sum_{i=1}^{m-1} \binom{|U|}{i} (m-i) \leq (m-1)^2 \cdot |U|^{m-1}$$

Therefore, it is the case that T' has at most $|U| + (m-1)^2 \cdot |U|^{m-1}$ elements. Since $|U|$ is bounded by n (which is the number of variables in Q) the number of elements of T' is bounded by $n + (m-1)^2 n^{m-1}$.

In addition, the number of facts in T' equals the number of facts of T plus one fact for each extended subset X of U. Since for each such X there is no tuple $\bar{t}$ in a relation of T with $U_{\bar{t}} = X$, the number of facts in T' is bounded by the number of facts of the form $R(\bar{t})$, where $R \in \sigma$ and $\bar{t}$ is a tuple of elements in U. This number is bounded by $l \cdot |U|^m$, that is, by $\ell \cdot n^m$.

We conclude that the CQ Q'' whose tableau is $(T', \bar{x}')$ satisfies all the required conditions of the claim. □

We continue now with the proof of Theorem 6.1. We start with the first part of it. Let $Q(\bar{x})$ be a CQ over σ that has at least one hypergraph-based $\mathcal{C}$-query contained in it, namely $Q'(\bar{x}')$. Let l be the number of relations and m the maximum arity of a relation in σ, respectively. Define $H^{\mathcal{C}}(Q)$ as the set of all CQs that are hypergraph-based $\mathcal{C}$-queries contained in Q and have at most $|Q| + (m-1)^2 \cdot |Q|^{m-1}$ variables and at most $l \cdot |Q|^m$ joins, where $|Q|$ denotes the number of variables of Q. Clearly, $H^{\mathcal{C}}(Q)$ is finite (up to renaming of variables). From Claim 6.2, it follows that $H^{\mathcal{C}}(Q)$ is not empty. Take a maximal element $\tilde{Q}$ in $H^{\mathcal{C}}(Q)$ (with respect to $\subseteq$). We claim that $\tilde{Q}$ is a hypergraph-based $\mathcal{C}$-approximation of Q. Assume for the sake of contradiction that this is not the case. Then there exists a hypergraph-based $\mathcal{C}$-query P such that $\tilde{Q} \subset P \subseteq Q$. Using Claim 6.2, there exists a hypergraph-based $\mathcal{C}$-query P' such that $\tilde{Q} \subset P \subseteq P' \subseteq Q$ and P' is in $H^{\mathcal{C}}(Q)$. Since $\tilde{Q} \subset P'$ and $P' \in H^{\mathcal{C}}(Q)$, we have a contradiction with the maximality of $\tilde{Q}$. Thus, $\tilde{Q}$ is a hypergraph-based $\mathcal{C}$-approximation of Q.

It easily follows from Claim 6.2 that each hypergraph-based $\mathcal{C}$-approximation of Q is equivalent to a CQ in $H^{\mathcal{C}}(Q)$. The second part of Theorem 6.1 follows directly from this observation. □

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 6.3 *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 $2^{r_\sigma(|Q|)}$.*

In the specific case of acyclicity, this result (in fact with linear bounds) can also be derived from the results of [5]. The general closure-based approach we have here allows us to extend it further, to both hypertree width and generalized hypertree width. First we recall the definitions [21]. A *generalized 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 $u \in T$.

A *hypertree decomposition* of $\mathcal{H}$ [20] is a generalized hypertree decomposition that satisfies, in addition, the following property:

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

The *width* of a (generalized) hypertree decomposition $\langle T, f, c\rangle$ is $\max_{u\in T} |c(u)|$. The (generalized) *hypertree width* of $\mathcal{H}$ is the minimum width over all its (generalized) 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 CQs or tableaux whose hypergraphs have hypertree width at most k. Similarly, we denote by $\mathsf{GHTW}(k)$ the class of hypergraphs, CQs and tableaux of generalized hyper treewidth at most k. Obviously, $\mathsf{HTW}(k) \subseteq \mathsf{GHTW}(k)$, for each $k \geq 1$. It is shown in [20] that a hypergraph $\mathcal{H}$ is acyclic iff its hypertree width is 1. That is, $\mathsf{AC} = \mathsf{HTW}(1)$, and, thus, $\mathsf{AC} \subseteq \mathsf{GHTW}(1)$.

The key result of [20] is that for each fixed k, CQs from $\mathsf{HTW}(k)$ can be evaluated in polynomial time with respect to combined complexity. Notably, the same holds for the class $\mathsf{GHTW}(k)$, for each fixed k [21]. There is, however, a crucial difference between the two notions: Verifying whether $\mathsf{HTW}(\mathcal{H}) \leq k$, for a given hypergraph $\mathcal{H}$ and a fixed $k \geq 1$, can be solved in polynomial time [20], while verifying whether $\mathsf{GHTW}(\mathcal{H}) \leq k$, for any $k \geq 3$, is NP-complete [22].

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 6.4 *For each k, the class $\mathsf{HTW}(k)$ is closed under induced subhypergraphs and edge extensions. The same holds for the class $\mathsf{GHTW}(k)$.*

Proof: We only prove it for $\mathsf{HTW}(k)$. It will be clear from the proof that the same argument applies to $\mathsf{GHTW}(k)$. Let $\mathcal{H} = \langle V, \mathcal{E}\rangle$ be a hypergraph of hypertree width at most k and $\langle T, f, c\rangle$ a hypertree decomposition of $\mathcal{H}$ of width at most k.

We consider edge extensions first. Let $\mathcal{H}'$ be a hypergraph obtained from $\mathcal{H}$ by extending hyperedge e with new nodes V'. Thus, the set of nodes of $\mathcal{H}'$ is $V \cup V'$ and its set of hyperedges is $(\mathcal{E} \setminus \{e\}) \cup \{e'\}$. First, pick an arbitrary node u_e in T that satisfies $e \subseteq f(u_e)$ and is at minimal distance to the root (such element u_e exists because $\langle T, f\rangle$ is a tree decomposition of $\mathcal{H}$). Let T' be the tree that is obtained from T by adding a new child z to u_e in T. We then define a function $f' : T' \to 2^{(V\cup V')}$ as follows: $f'(u) = f(u)$ for each node u in T' that is neither z nor u_e, $f'(z) = e'$, and, finally, $f'(u_e) = f(u_e) \cup V'$, if $e \in c(u_e)$, and $f'(u_e) = f(u_e)$ otherwise. We also define a function $c' : T' \to 2^{(\mathcal{E}\setminus\{e\})\cup\{e'\})}$ such that $c'(z) = \{e'\}$ and for each $u \in T'$ that is not z it is the case that $c'(u) = c(u)$, if $e \notin c(u)$, and $c'(u) = (c(u) \setminus \{e\}) \cup \{e'\}$, otherwise. Clearly, the width of $\langle T', f', c'\rangle$ is at most k. We prove next that $\langle T', f', c'\rangle$ is a hypertree decomposition of $\mathcal{H}'$.

First, we shall prove that $\langle T', f' \rangle$ is a tree decomposition of $\mathcal{H}'$. Consider a hyperedge d in $\mathcal{H}'$. If $d \neq e'$ then there exists a node u in T such that $d \subseteq f(u)$ (because $\langle T, f \rangle$ is a tree decomposition of $\mathcal{H}$). By definition, d also belongs to $f'(u)$. If $d = e'$ then $d \subseteq f'(z)$. This shows that each hyperedge of $\mathcal{H}'$ is contained in $f'(u')$, for some node u' in T'. Consider now a node v of $\mathcal{H}'$, i.e. $v \in V \cup V'$. If $v \notin e' = e \cup V'$ then, by definition, each node u of T' such that $v \in f'(u)$ belongs to T and satisfies that $v \in f(u)$ iff $v \in f'(u)$. Thus, since $\langle T, f \rangle$ is a tree decomposition of $\mathcal{H}$, we conclude that $\{u \in T' \mid v \in f'(u)\}$ is a connected subset of T'. If $v \in e$ then a similar argument shows that $\{u \in T' \mid v \in f'(u)\}$ is exactly

$$\{u \in T \mid v \in f(u)\} \ \cup \ \{z\}.$$

Since z is a leaf of u_e and, by definition, $v \in f(u_e)$, we conclude from the fact that $\langle T, f \rangle$ is a tree decomposition of $\mathcal{H}$ that $\{u \in T' \mid v \in f'(u)\}$ is a connected subset of T'. Finally, if $v \in V'$ then $\{u \in T' \mid v \in f'(u)\}$ is the single node z, if $e \notin c(u_e)$, or the single edge $\{u_e, z\}$, if $e \in c(u_e)$, both of which are connected subsets of T'. We conclude that $\langle T', f' \rangle$ is a tree decomposition of $\mathcal{H}'$.

We prove next that $f'(u) \subseteq \bigcup c'(u)$, for each node u in T'. First of all, by definition $f'(z) = e' \subseteq \bigcup c'(z) = \{e'\}$. Also, since $\langle T, f, c \rangle$ is a hypertree decomposition of $\mathcal{H}$, we have that $f(u_e) \subseteq \bigcup c(u_e)$, and, thus, $f'(u_e) \subseteq \bigcup c'(u_e)$ (no matter whether e is or is not in $c(u_e)$). For u different from z and u_e, we have by definition that $f'(u) = f(u)$, and, by hypothesis, $f(u) \subseteq \bigcup c(u)$. But for each u in T' that is neither z nor u_e we have by definition that $\bigcup c(u) \subseteq \bigcup c'(u)$, and, thus, $f'(u) \subseteq \bigcup c'(u)$.

We prove finally that $\bigcup c'(u) \cap \bigcup \{f'(t) \ \mid \ t \in T'_u\} \subseteq f'(u)$ holds for every $u \in T'$. Consider first a node u in T' such that $u_e \notin T'_u$. If $u = z$ then the property holds trivially since $c'(z) = \{e'\}$, $f'(z) = e'$ and $\{f'(t) \ \mid \ t \in T'_z\} = \{f'(z)\}$. Suppose then that $u \neq z$. By definition, $f(x) = f'(x)$ for each $x \in T'_u$, and, thus, no element in V' belongs to $f'(x)$, for some $x \in T'_u$. It follows that

$$\bigcup c'(u) \cap \bigcup \{f'(t) \mid t \in T'_u\} \ = \ \bigcup c(u) \cap \bigcup \{f(t) \mid t \in T_u\}.$$

But the latter is contained in $f(u)$ because $\langle T, f, c \rangle$ is a hypertree decomposition of $\mathcal{H}$, and, thus, in $f'(u)$ because $f(u) = f'(u)$.

Consider now a node u in T' such that $u_e \in T'_u$. Assume first that $u = u_e$. If $e \in c(u_e)$ we have by definition that

$$\bigcup c'(u) \cap \bigcup \{f'(t) \mid t \in T'_u\} \ = \ V' \cup \big(\bigcup c(u) \cap \bigcup \{f(t) \mid t \in T_u\}\big).$$

But since $\langle T, f, c \rangle$ is a hypertree decomposition of $\mathcal{H}$ we have that the latter is contained in $V' \cup f(u)$, which by definition is $f'(u)$. If $e \notin c(u_e)$, then $\bigcup c'(u) = \bigcup c(u)$ and, thus, $\bigcup c'(u)$ does not contain any element in V'. We conclude that

$$\bigcup c'(u) \cap \bigcup \{f'(t) \mid t \in T'_u\} \ = \ \bigcup c(u) \cap \bigcup \ \{f(t) \mid t \in T_u\}.$$

But the latter is contained in $f(u)$ because $\langle T, f, c \rangle$ is a hypertree decomposition of $\mathcal{H}$, and, therefore, in $f'(u)$ because $f(u_e) = f'(u_e)$. Finally, assume that $u \neq u_e$. Then necessarily $e \notin c(u)$. Assume otherwise. Then since $e \subseteq f(u_e)$ we would have that $e \subseteq \bigcup c(u) \cap f(u_e)$, and, thus, in $\bigcup c(u) \cap \bigcup \{f(t) \mid t \in T_u\}$. But $\langle T, f, c \rangle$ is a hypertree decomposition of $\mathcal{H}$, and, thus, $e \subseteq f(u)$, which is a contradiction with the minimality of u_e with respect to distance to the root of T. Then, $e \notin c(u)$, implying that $V' \cap \bigcup c'(u) = \emptyset$. Therefore,

$$\bigcup c'(u) \cap \bigcup \{f'(t) \mid t \in T'_u\} \ = \ \bigcup c(u) \cap \bigcup \{f(t) \mid t \in T_u\}.$$

But the latter is contained in $f(u)$ because $\langle T, f, c\rangle$ is a hypertree decomposition of $\mathcal{H}$, and, therefore, in $f'(u)$ because $f(u) = f'(u)$. We conclude then that $\langle T', f', c'\rangle$ is a hypertree decomposition of $\mathcal{H}'$.

We consider now closure under induced subhypergraphs. Let $\mathcal{H}' = \langle V', \{e \cap V' \mid e \in \mathcal{E}\}\rangle$ be an induced subhypergraph of $\mathcal{H}$, for $V' \subseteq V$. Let also T', f' and c' be defined as follows: $T' = T$, and for each $u \in T'$ it is the case that $f'(u) = f(u) \cap V'$, and $c'(u) = \{e \cap V' \mid e \in c(u)\}$. Clearly, the width of $\langle T', f', c'\rangle$ is at most k. We prove next that $\langle T', f', c'\rangle$ is a hypertree decomposition of $\mathcal{H}'$.

We prove first that $\langle T', f'\rangle$ is a tree decomposition of $\mathcal{H}'$. Consider a hyperedge d in $\mathcal{H}'$. Then, $d = e \cap V'$, for some $e \in \mathcal{E}$. Since $\langle T, f\rangle$ is a tree decomposition of $\mathcal{H}$ there is $u \in T$ such that $e \subseteq f(u)$. It follows that $d = e \cap V' \subseteq f(u) \cap V' = f'(u)$. Let v be a node in $\mathcal{H}'$, i.e. $v \in V'$. It is easy to see that $\{u \in T' \mid v \in f'(u)\}$ is exactly the same set than $\{u \in T \mid v \in f(u)\}$. Since $\langle T, f\rangle$ is a tree decomposition of $\mathcal{H}$ we conclude that $\{u \in T' \mid v \in f'(u)\}$ is a connected subset of T'.

For each $u \in T'$ we have by definition that $f'(u) = f(u) \cap V'$, and, thus, $f'(u) \subseteq \bigcup c(u) \cap V'$ (because $f(u) \subseteq \bigcup c(u)$ from the fact that $\langle T, f, c\rangle$ is a hypertree decomposition of $\mathcal{H}$). But the latter is precisely $\bigcup c'(u)$, which implies that $f'(u) \subseteq \bigcup c'(u)$, for each $u \in T'$. In addition, $\bigcup c'(u) \cap \bigcup\{f'(t) \mid t \in T'_u\}$ is by definition equal to the set $\bigcup c(u) \cap \bigcup\{f(t) \mid t \in T_u\} \cap V'$. Since $\langle T, f, c\rangle$ is a hypertree decomposition of $\mathcal{H}$ the latter is contained in $f(u) \cap V'$, which is $f'(u)$ by definition. We conclude that $\langle T', f', c'\rangle$ is a hypertree decomposition of $\mathcal{H}'$. □

Note that the query Q_{trivial} from Section 4.1 is in $\mathsf{HTW}(k)$ and $\mathsf{GHTW}(k)$. This gives the desired result about the existence of approximations within $\mathsf{HTW}(k)$ for every k.

Corollary 6.5 *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 or $\mathsf{GHTW}(k)$-approximation of size at most $p_\sigma(|Q|)$ that can be found in time $2^{r_\sigma(|Q|)}$, for every $k \geq 1$.*

Example 6.6 Consider a Boolean query

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

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:
$$Q'_1() \;{:\!-}\; R(x, y, x).$$

- With as many joins as Q:
$$Q'_2() \;{:\!-}\; R(x_1, x_2, x_3), R(x_3, x_4, x_2), R(x_2, x_6, x_1).$$

- With more joins than Q:
$$Q'_3(){:\!-}R(x_1, x_2, x_3), R(x_3, x_4, x_5), R(x_5, x_6, x_1), R(x_1, x_3, x_5).$$

□

7 Conclusions

We have 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.

In this work we only 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, 27], 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.

We also plan to study other notions of approximations that are not constrained to return correct result only. These include *overapproximations*, that return all correct results, and arbitrary approximations that simply look for maximal elements in the preorder $\sqsubseteq_Q$. Even the most basic problems related to those notions of approximations, such as existence and complexity, seem challenging.

Acknowledgments We thank Wenfei Fan, Claudio Gutierrez, Luc Segoufin, Thomas Schwentick, and the referees of conference versions [6, 7] 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.

References

[1] S. Abiteboul, K. Compton, and V. Vianu. Queries are easier than you thought (probably). In *Proceedings of the 11th ACM Symposium on Principles of Database Systems*, PODS'92, pages 23–32, 1992.

[2] S. Abiteboul, O. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, PODS'98, pages 254–263, 1998.

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

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

[5] V. Bárány, G. Gottlob, M. Otto. Querying the guarded fragment. In *Proceedings of IEEE Logic in Computer Science*, LICS'10, pages 1–10.

[6] P. Barceló, L. Libkin, M. Romero. Efficient approximations of conjunctive queries. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems*, PODS'12, pages 249-260.

[7] P. Barceló, L. Libkin, M. Romero. On low treewidth approximations of conjunctive queries. In *Proceedings of 6th Alberto Mendelzon International Workshop on Foundations of Data Management*, AMW'12, CEUR Workshop Proceedings 866, 2012, pages 91-101.

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

[9] B. ten Cate, Ph. Kolaitis, W.-C. Tan. Database constraints and homomorphism dualities. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming*, CP'10, pages 475–490, 2010.

[10] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, STOC'77, pages 77–90, 1977.

[11] C. Chekuri, A. Rajaraman. Conjunctive query containment revisited. *Theoretical Computer Science*, 239(2), pages 211-229, 2000.

[12] V. Dalmau, P. G. Kolaitis, M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP* 2002, pages 310–326.

[13] R. Fagin, Ph. Kolaitis, L. Popa. Data exchange: getting to the core. *ACM Transactions on Database Systems*, 30, pages 90–101, 2005.

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

[15] R. Fink, D. Olteanu. On the optimal approximation of queries using tractable propositional languages. In *Proceedings of the 14th International Conference on Database Theory*, ICDT'11, pages 174–185, 2011.

[16] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *Journal of the ACM*, 49, pages 716–752, 2002.

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

[18] M. Garofalakis and P. Gibbons. Approximate query processing: taming the terabytes. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB'01.

[19] G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 48, pages 431–498, 2001.

[20] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64, pages 579–627, 2002.

[21] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66, pages 775–808, 2003.

[22] G. Gottlob, Z. Miklos, and T. Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. *Journal of the ACM*, 56(6), 2009.

[23] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings on 33rd ACM Symposium on Theory of Computing*, STOC'01, pages 657–666, 2001.

[24] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4), pages 270-294, 2001.

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

[26] P. Hell, J. Nešetřil, and X. Zhu. Complexity of tree homomorphisms. *Discrete Applied Mathematics* 70(1): 23-36 (1996).

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

[28] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5), pages 279-295, 1997.

[29] Y. Ioannidis. Approximations in database systems. In *Proceedings of the 9th International Conference on Database Theory*, ICDT'03, pages 16–30, 2003.

[30] Ph. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2), pages 302-332, 2000.

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

[32] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st Symposium on Principles of Database Systems*, PODS'02, pages 233–246, 2002.

[33] L. Libkin. Incomplete information and certain answers in general data models. In *Proceedings of the 30th Symposium on Principles of Database Systems*, PODS'11, pages 59–70, 2011.

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

[35] S. Malik and L. Zhang. Boolean satisfiability: from theoretical hardness to practical success. *Communications of the ACM*, 52(8), pages 76–82, 2009.

[36] J. Nešetřil, C. Tardif. Duality theorems for finite structures (Characterising gaps and good characterisations). *Journal of Combinatorial Theory, Ser. B*, 80(1), pages 80-97, 2000.

[37] C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28, pages 244–259, 1996.

[38] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, FOCS'77, pages 46-57, 1997.

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

[40] J. Rothe. Exact complexity of exact-four-colorability. *Inf. Process. Lett.* 87(1): 7-12 (2003).

[41] Y. Sagiv, M. Yannakakis. Equivalences among relational expressions with the inion and difference operators. *Journal of the ACM* 27(4), pages 633-655, 1980.

[42] M. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, PODS'95, pages 266–276, 1995.

[43] M. Yannakakis. Algorithms for acyclic database schemes. In *Proceedings of 7th International Conference on Very Large Databases*, VLDB'81, pages 82–94, 1981.

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

8 Appendix: Proof of Theorem 4.12

The class $\mathsf{TW}(1)$ over graphs contains all *acyclic* directed graphs, i.e. directed graphs whose underlying undirected graph contains no cycles. It thus suffices to show that the following problem is DP-complete:

Problem:	Graph Acyclic Approximation
Input:	a digraph G, an acyclic digraph T.
Question:	Is $G \to T$ and there is no acyclic digraph T' such that $G \to T' \rightleftarrows\!\!\!\!\!/\;\; T$?

The Exact Four Colorability problem is defined as follows: Given a graph G, decide if G is 4-colorable but not 3-colorable. It is known that this problem is DP-complete [40]. We define a polynomial time reduction from Exact Four Colorability to Graph Acyclic Approximation.

We use several notions, such as oriented paths, cycles, heights and levels, that were defined in the proof of Proposition 4.4. The proof requires some preparation. Consider the oriented paths $P_i = 0^{i+1}10^{11-i}$, for each $1 \leq i \leq 9$. Observe that all these oriented paths are incomparable cores and have net length 11. We have the following:

Claim 8.1 *For each $1 \leq i < j \leq 9$, there exists an oriented path P_{ij} such that $P_{ij} \to P_i$, $P_{ij} \to P_j$ and $P_{ij} \not\to P_k$ for each $1 \leq k \leq 9$ with $k \neq i$ and $k \neq j$.*

Proof: We take $P_{ij} = 0^{i+1}100^{j-i}10^{11-j}$. Using Lemma 4.5 it can be verified that the conditions are satisfied. □

Claim 8.2 *For each $1 \leq i < j < k \leq 9$, there exists an oriented path P_{ijk} such that $P_{ijk} \to P_i$, $P_{ijk} \to P_j$, $P_{ijk} \to P_k$ and $P_{ijk} \not\to P_\ell$ for each $1 \leq \ell \leq 9$ with $\ell \neq i$, $\ell \neq j$ and $\ell \neq k$.*

Proof: We take $P_{ijk} = 0^{i+1}100^{j-i}100^{k-j}10^{11-k}$. Using Lemma 4.5 it can be verified that the conditions are satisfied. □

Next we define a digraph Q^* (depicted in Figure 7) as follows: Consider the balanced cycle $(a_1, a_2, ..., a_8, a_1)$ defined by the string 01010101. For each $1 \leq i \leq 8$, we add a disjoint copy of P_i. If i is odd, we identify a_i with the terminal node of P_i; otherwise, we identify a_i with the

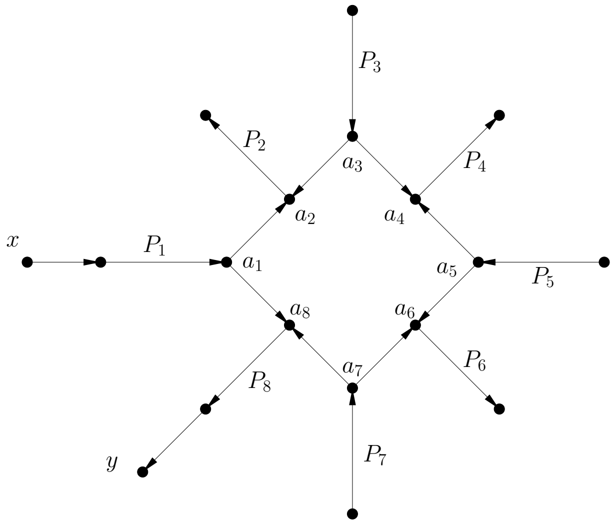


Figure 7: The digraph Q^*.

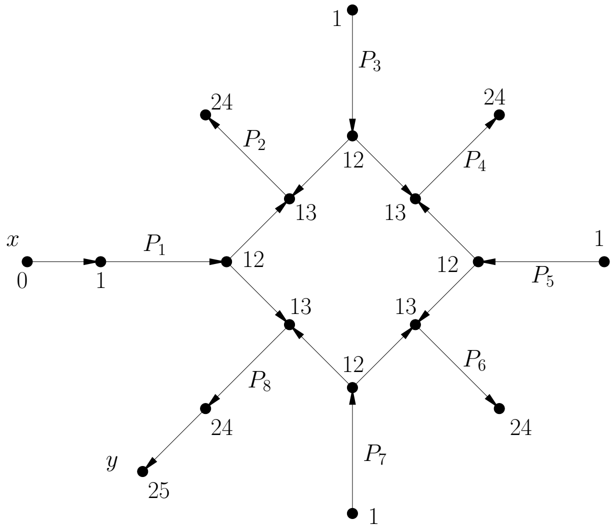


Figure 8: The digraph Q^* and some of its levels.

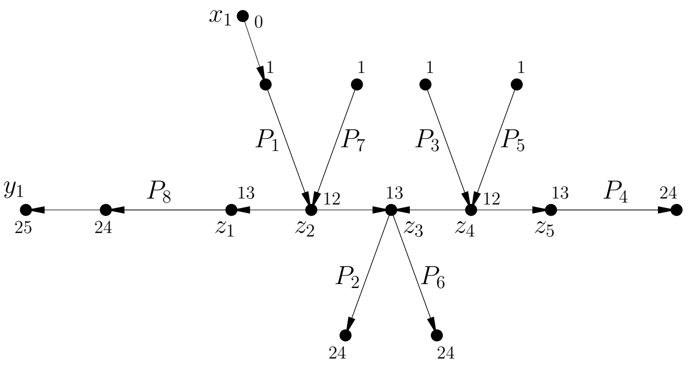


Figure 9: The digraph T_1 and some of its levels.

initial node of P_i. Finally, we add two new nodes x and y and two new edges: one from x to the initial node of the copy of P_1, and another one from the terminal node of the copy of P_8 to y. The resulting digraph Q^* and its levels are depicted in Figure 8. Notice that Q^* is balanced and that its height, $hg(Q^*)$, is 25. Moreover, the nodes x and y are the only ones in Q^* with level 0 and 25, respectively.

We also define acyclic digraphs T_1, T_2, T_3 and T_4 as follows: Each T_i, for $1 \leq i \leq 4$, is obtained from Q^* by identifying some specific nodes. For T_1 we identify a_1, a_2 and a_3 with a_7, a_6 and a_5, respectively. In the case of T_2 we identify a_8, a_1 and a_2 with a_6, a_5 and a_4, respectively. For T_3 we identify a_7, a_8 and a_1 with a_5, a_4 and a_3, respectively. Finally, for T_4 we identify a_6, a_7 and a_8 with a_4, a_3 and a_2. Note that for each $1 \leq i \leq 4$, $hg(T_i) = 25$ and the nodes x_i and y_i are the only ones in T_i with level 0 and 25, respectively, as depicted in Figures 9 and 10.

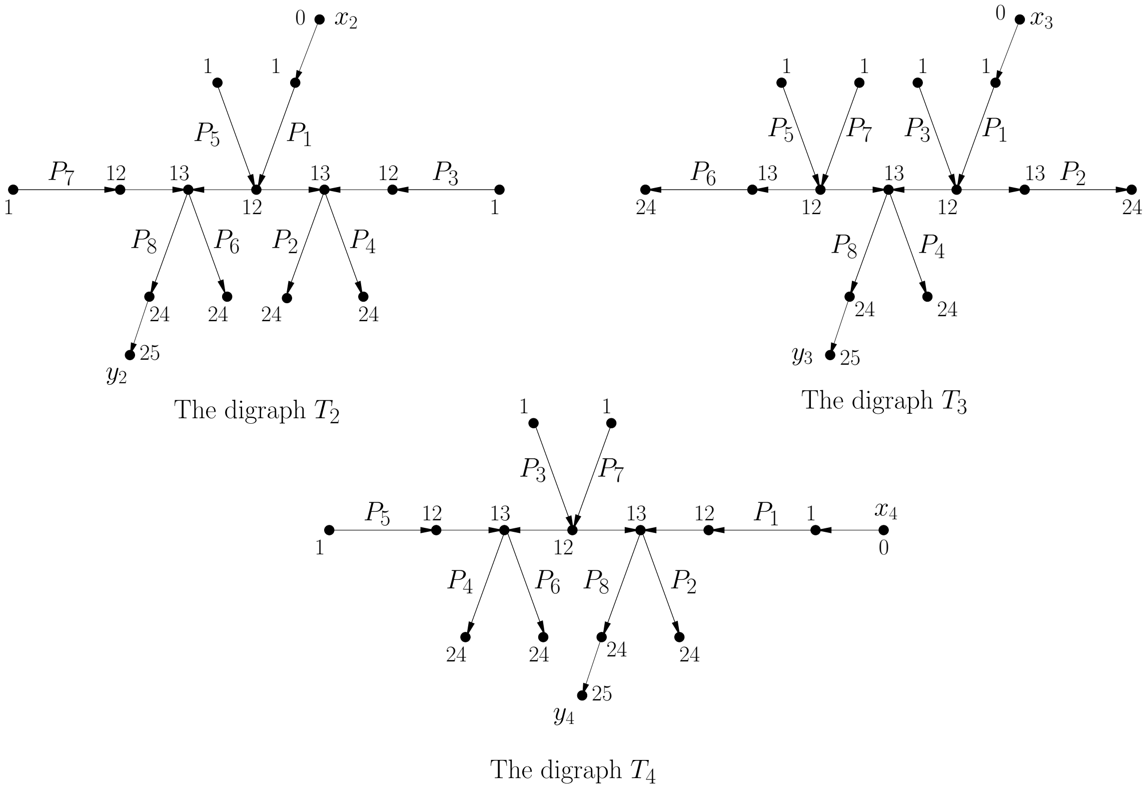
Since the T_i's are acyclic, they are by definition balanced. Using Lemma 4.5 and the incomparability of the P_i's, it easily follows that T_1, T_2, T_3 and T_4 are incomparable cores. Furthermore, observe that $Q^* \xrightarrow{h_i} T_i$ for all $1 \leq i \leq 4$, where h_i is the homomorphism naturally defined by the identification of nodes we used to construct the T_i's. Note that for each $1 \leq i \leq 4$, h_i is a surjective homomorphism, i.e., $\text{Im}(h_i) = T_i$. Even more, we have the following:

Claim 8.3 *For each $1 \leq i \leq 4$, h_i is the unique homomorphism from Q^* to T_i. In particular, any homomorphism from Q^* to T_i is surjective, for each $1 \leq i \leq 4$.*

Proof: Let h be a homomorphism that witnesses $Q^* \rightarrow T_1$. We shall prove that $h = h_1$. Since Q^* and T_1 are balanced, we have that h preserves levels (Lemma 4.5). Thus, either $h(a_8) = z_1$, $h(a_8) = z_3$ or $h(a_8) = z_5$. If $h(a_8) = z_3$ then $P_8 \rightarrow P_2$ or $P_8 \rightarrow P_6$, which is is impossible since the P_i's are incomparable. Similarly, if $h(a_8) = z_5$ then $P_8 \rightarrow P_4$, which is also not possible. It follows that $h(a_8) = z_1 = h_1(a_8)$. Using the same argument we can prove that $h(c) = h_1(c)$, for each other element c of Q^*. For T_2, T_3 and T_4 the argument is analogous. □

A key property of the T_i's is the following:

Claim 8.4 *For each $1 \leq i \leq 4$, $Q^* \rightarrow T_i$ and there is no acyclic T' such that $Q^* \rightarrow T' \not\rightleftarrows T_i$.*



The digraph T_2

The digraph T_3

The digraph T_4

Figure 10: The digraphs T_2, T_3 and T_4 and some of its levels.

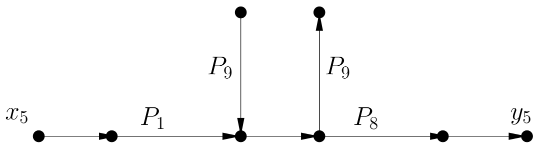


Figure 11: The digraph T_5.

Proof: We only prove the claim for $i = 1$. For $i = 2, 3, 4$ the proof is analogous. By contradiction, assume that there exists an acyclic digraph T' such that $Q^* \xrightarrow{h} T' \xrightarrow{g} T_1$ and $T_1 \not\to T'$. If $h(a_1) = h(a_7), h(a_2) = h(a_6)$ and $h(a_3) = h(a_5)$, using Claim 4.8, it follows that $T_1 \to T'$, which is a contradiction. Thus, we have that either $h(a_1) \neq h(a_7), h(a_2) \neq h(a_6)$ or $h(a_3) \neq h(a_5)$. Using Claim 8.3, we have that $g \circ h = h_1$. Notice by definition that the sets $\{h_1(a_8)\}$, $\{h_1(a_1), h_1(a_7)\}$, $\{h_1(a_2), h_1(a_6)\}$, $\{h_1(a_3), h_1(a_5)\}$ and $\{h_1(a_4)\}$ are mutually disjoint, and hence the sets $\{h(a_8)\}$, $\{h(a_1), h(a_7)\}$, $\{h(a_2), h(a_6)\}$, $\{h(a_3), h(a_5)\}$ and $\{h(a_4)\}$ must be mutually disjoint as well. Since $h(a_1) \neq h(a_7), h(a_2) \neq h(a_6)$ or $h(a_3) \neq h(a_5)$, necessarily T' has an oriented cycle, which is a contradiction. □

We define an acyclic directed graph T_5 as follows: Consider the disjoint union of P_1 and P_8. Add two new nodes x_5 and y_5 and three new edges: from x_5 to the initial node of P_1, from the terminal node of P_1 to the initial node of P_8, and from the terminal node of P_8 to y_5. Finally, add two disjoint copies of P_9 and identify the terminal node of one copy with the terminal node of P_1, and the initial node of the other copy with the initial node of P_8. The resulting graph T_5 is depicted in Figure 11.

Since T_5 is acyclic, it is also balanced. Using Lemma 4.5 and the incomparability of the P_i's, it easily follows that T_5 is incomparable with T_1, T_2, T_3, T_4 and Q^*.

Claim 8.5 *For each $(i, j) \in \{(1, 5), (2, 5), (3, 5), (1, 2), (1, 3), (2, 3)\}$, there exists an acyclic digraph T_{ij} such that $T_{ij} \to T_i$, $T_{ij} \to T_j$, and $T_{ij} \not\to T_k$, for $1 \leq k \leq 5$ with $k \neq i$ and $k \neq j$.*

Proof: Consider the oriented path P constructed as follows: Take the disjoint union of P_1 and P_8. Add two new nodes p_1 and p_2, and three new edges: from p_1 to the initial node of P_1, from the terminal node of P_1 to the initial node of P_8, and from the terminal node of P_8 to p_2. Each T_{ij} is obtained from P by adding a disjoint copy of a specific path X_{ij} and identifying the terminal node of X_{ij} with the terminal node of P_1, as depicted in Figure 12. The X_{ij}'s are: $X_{15} = P_{79}, X_{25} = P_{59}, X_{35} = P_{39}, X_{12} = P_{57}, X_{13} = P_{37}$ and $X_{23} = P_{35}$. Using Lemma 4.5 and Claim 8.1 it can be proved that these T_{ij}'s satisfy the required conditions. □

Claim 8.6 *For each $(i, j, k) \in \{(1, 2, 5), (2, 4, 5), (3, 4, 5)\}$, there exists an acyclic digraph T_{ijk} such that $T_{ijk} \to T_i$, $T_{ijk} \to T_j$, $T_{ijk} \to T_k$, and $T_{ijk} \not\to T_\ell$ for $1 \leq \ell \leq 5$, $\ell \neq i$, $\ell \neq j$ and $\ell \neq k$.*

Proof: Consider the oriented path P as in the proof of Claim 8.5. The digraph T_{125} is obtained from P by adding a disjoint copy of P_{579} and identifying the terminal node of P_{579} with the terminal node of P_1. The others T_{ijk}'s are obtained from P by adding a disjoint copy of a path X_{ijk}

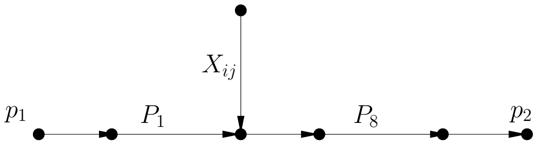


Figure 12: The structure of the T_{ij}'s.

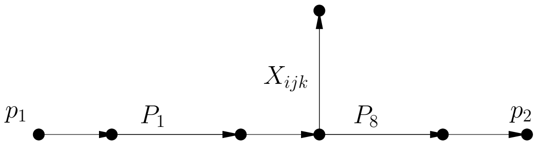


Figure 13: The structure of T_{245} and T_{345}.

and identifying the initial node of X_{ijk} with the initial node of P_8. The X_{ijk}'s are: $X_{245} = P_{269}$ and $X_{345} = P_{249}$; see Figure 13 . From Lemma 4.5 and Claim 8.2, it follows that these T_{ijk}'s satisfy the required conditions. □

We introduce some notation. We work with digraphs that have an initial and a terminal node, which are simply two distinct distinguished elements in the digraph. Consider a digraph G with two distinguished nodes i_1 and t_1, which are its initial and terminal node, respectively. Similarly, consider a digraph H with nodes i_2 and t_2 distinguished as initial and terminal node, respectively. We define the concatenation of G and H, denoted $G \cdot H$, to be the digraph obtained from the disjoint union of G and H identifying t_1 with i_2. The initial and terminal node of $G \cdot H$ is i_1 and t_2, respectively. Finally, we define G^{-1} to be the digraph obtained from G by switching the roles of the initial and terminal nodes, that is, the initial node of G is t_1 and the terminal one is i_1.

Let us assume now that the initial node of Q^* is x and its terminal node is y. For each $1 \leq i \leq 5$, we also assume that x_i and y_i are the initial and terminal nodes of T_i, respectively. Similarly, for each T_{ij} and T_{ijk} defined as in Claims 8.5 and 8.6, respectively, we assume its initial and terminal nodes to be the only nodes in the respective graphs with level 0 and 25, respectively. In all figures, an edge uv labeled with a digraph G with initial node i and terminal node t, represents the digraph constructed as follows: delete the arc uv, add a disjoint copy of G and identify i with u and t with v.

Let T be the acyclic digraph constructed as follows: Consider the disjoint union of $T_1 \cdot T_5^{-1}$, $T_2 \cdot T_5^{-1}$, $T_3 \cdot T_5^{-1}$ and $T_4 \cdot T_5^{-1}$, and identify all of their initial nodes into a single new node v. Observe that $hg(T) = 25$, the only nodes of T with level 0 are v, u_1, u_2, u_3 and u_4, and the only nodes of T with level 25 are t_1, t_2, t_3 and t_4. This is shown in Figure 14.

Now we recall the notion of (i,j)-*chooser* from [26].

Definition 8.7 *Let $X = \{1, 2, 3\}$ and $i, j \in X$ be indices. An (i,j)-chooser is a digraph T^* with two distinguished nodes a and b such that:*

- *For every homomorphism $h : T^* \to T$, we have $h(a) = t_1$ and $h(b) \neq t_i$, or $h(a) = t_2$ and $h(b) \neq t_j$.*

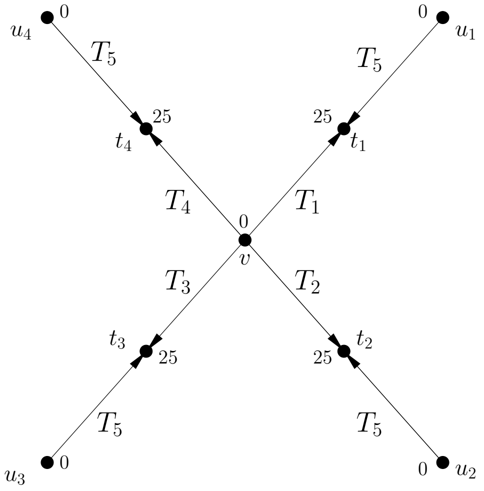


Figure 14: The digraph T and some of its levels.

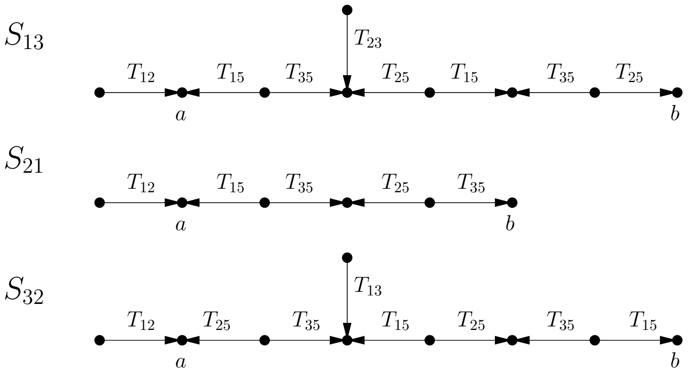


Figure 15: The choosers S_{13}, S_{21} and S_{32}.

- *For every $k \in X$ with $k \neq i$, there is a homomorphism $h : T^* \to T$ such that $h(a) = t_1$ and $h(b) = t_k$.*
- *For every $k \in X$ with $k \neq j$, there is a homomorphism $h : T^* \to T$ such that $h(a) = t_2$ and $h(b) = t_k$.*

An *extended (i, j)-chooser* is defined exactly as in Definition 8.7, but this time we consider $X = \{1, 2, 3, 4\}$.

Claim 8.8 *There exists a $(1, 3)$-chooser S_{13}, a $(2, 1)$-chooser S_{21}, and a $(3, 2)$-chooser S_{32}.*

Proof: The digraphs S_{13}, S_{21} and S_{32} are shown in Figure 15. Proving that they are indeed the corresponding choosers it is not hard (and one can in fact mimic the proof of Lemma 4 in [26]). □

We rename the nodes a and b in S_{13}, S_{21} and S_{32} to a_1, b_1; a_2, b_2 and a_3, b_3, respectively.

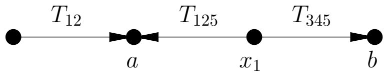


Figure 16: The extended $(2,1)$-chooser $\tilde{S}_{21}$.

Claim 8.9 *There exists an extended* $(2,1)$*-chooser* $\tilde{S}_{21}$ *and an extended* $(3,4)$*-chooser* $\tilde{S}_{34}$*.*

Proof: We define $\tilde{S}_{21} = T_{12} \cdot T_{125}^{-1} \cdot T_{345}$ and $\tilde{S}_{34} = T_{12} \cdot T_{25}^{-1} \cdot T_{35} \cdot T_{15}^{-1} \cdot T_{245} \cdot T_{35}^{-1} \cdot T_{15}$. For $\tilde{S}_{21}$ and $\tilde{S}_{34}$, we denote by a the terminal node of their respective copies of T_{12}, and by b their respective terminal nodes, as illustrated in Figures 16 and 17.

Consider first a homomorphism $h : \tilde{S}_{21} \to T$. It is not hard to see that either $h(a) = t_1$ or $h(a) = t_2$. Assume first that $h(a) = t_1$. Then either $h(x_1) = u_1$ or $h(x_1) = v$. If $h(x_1) = u_1$ then $h(b) = t_1$. If $h(x_1) = v$ then either $h(b) = t_3$ or $h(b) = t_4$. Thus, if $h(a) = t_1$, then $h(b) \neq t_2$, and all the following combinations are possible:

$$\text{(i) } h(a) = t_1, h(b) = t_1; \quad \text{(ii) } h(a) = t_1, h(b) = t_3; \quad \text{and} \quad \text{(iii) } h(a) = t_1, h(b) = t_4.$$

Assume, on the other hand, that $h(a) = t_2$. Then either $h(x_1) = u_2$ or $h(x_1) = v$. If $h(x_1) = u_2$ then $h(b) = t_2$. If $h(x_1) = v$ then either $h(b) = t_3$ or $h(b) = t_4$. Thus, if $h(a) = t_2$ then $h(b) \neq t_1$, and all the following combinations are possible:

$$\text{(i) } h(a) = t_2, h(b) = t_2; \quad \text{(ii) } h(a) = t_2, h(b) = t_3; \quad \text{and} \quad \text{(iii) } h(a) = t_2, h(b) = t_4.$$

Therefore, $\tilde{S}_{21}$ is an extended $(2,1)$-chooser.

Consider now a homomorphism $h : \tilde{S}_{34} \to T$. Again, either $h(a) = t_1$ or $h(a) = t_2$. Assume first that $h(a) = t_1$. Then $h(x_1) = u_1$ and $h(x_2) = t_1$, and we have either $h(x_3) = u_1$ or $h(x_3) = v$. If $h(x_3) = u_1$ then $h(x_4) = t_1$, $h(x_5) = u_1$ and $h(b) = t_1$. If $h(x_3) = v$ then $h(x_4) = t_2$ or $h(x_4) = t_4$. If $h(x_4) = t_2$, then $h(x_5) = u_2$ and $h(b) = t_2$. If $h(x_4) = t_4$ then $h(x_5) = u_4$ and $h(b) = t_4$. Thus, if $h(a) = t_1$, then $h(b) \neq t_3$, and all the following combinations are possible:

$$\text{(i) } h(a) = t_1, h(b) = t_1; \quad \text{(ii) } h(a) = t_1, h(b) = t_2; \quad \text{and} \quad \text{(iii) } h(a) = t_1, h(b) = t_4.$$

Assume, on the other hand, that $h(a) = t_2$. Then we have either $h(x_1) = u_2$ or $h(x_1) = v$. If $h(x_1) = u_2$ then $h(x_2) = t_2$, $h(x_3) = u_2$, $h(x_4) = t_2$, $h(x_5) = u_2$ and $h(b) = t_2$. If $h(x_1) = v$ then $h(x_2) = t_3$, $h(x_3) = u_3$ and $h(x_4) = t_3$, and we have either $h(x_5) = u_3$ or $h(x_5) = v$. If $h(x_5) = u_3$ then $h(b) = t_3$. If $h(x_5) = v$ then $h(b) = t_1$. Thus, if $h(a) = t_2$, then $h(b) \neq t_4$, and all the following combinations are possible:

$$\text{(i) } h(a) = t_2, h(b) = t_1; \quad \text{(ii) } h(a) = t_2, h(b) = t_2; \quad \text{and} \quad \text{(iii) } h(a) = t_2, h(b) = t_3.$$

We conclude that $\tilde{S}_{34}$ is an extended $(3,4)$-chooser. □

Next we borrow techniques from [26], and define an acyclic graph T' as follows: We take the disjoint union of S_{13}, S_{21} and S_{32}, and identify their respective terminal nodes b_1, b_2 and b_3 into a new node b (see Figure 18). The following claim will be useful.

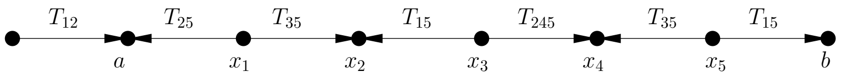


Figure 17: The extended (3, 4)-chooser $\tilde{S}_{34}$.

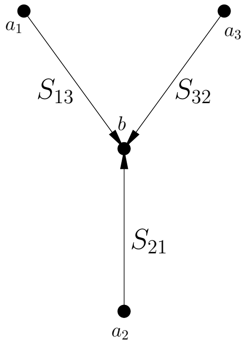


Figure 18: The digraph T'.

Claim 8.10 *There is no homomorphism $h : T' \to T$ such that $h(a_1) = h(a_2) = h(a_3)$. Furthermore, for any triple $(t_i, t_j, t_k) \in \{t_1, t_2\}^3 \setminus \{(t_1, t_1, t_1), (t_2, t_2, t_2)\}$ there exists a homomorphism $h : T' \to T$ such that $h(a_1) = t_i$, $h(a_2) = t_j$ and $h(a_3) = t_k$.*

Proof: Exactly as in the proof of Lemma 5 in [26]. □

Next, we construct our main gadget $\tilde{T}$: Let p and q be two fresh nodes. We add two disjoint copies of T', namely, T'_1 and T'_2. We rename the nodes a_1, a_2, a_3 and b in T'_1 to a^1_1, a^1_2, a^1_3 and b^1, respectively, and the nodes a_1, a_2, a_3 and b in T'_2 to a^2_1, a^2_2, a^2_3 and b^2, respectively. We then add three disjoint copies of $\tilde{S}_{21}$, namely, $\tilde{S}^1_{21}, \tilde{S}^2_{21}$ and $\tilde{S}^3_{21}$, and for each $1 \leq i \leq 3$ we rename the nodes a and b in $\tilde{S}^i_{21}$ to a^i_{21} and b^i_{21}, respectively. Afterwards, we identify the nodes b^1_{21} and b^2_{21} with p, and b^3_{21} with q. We also identify, for each $1 \leq i \leq 3$, the node a^i_{21} with a^1_i. Analogously, we add three disjoint copies of $\tilde{S}_{34}$, namely, $\tilde{S}^1_{34}, \tilde{S}^2_{34}$ and $\tilde{S}^3_{34}$, and for each $1 \leq i \leq 3$ we rename the nodes a and b in $\tilde{S}^i_{34}$ to a^i_{34} and b^i_{34}, respectively. We then identify the vertices b^1_{34} and b^2_{34} with p, and b^3_{34} with q. Finally, for each $1 \leq i \leq 3$, we identify the node a^i_{34} with a^2_i. The resulting digraph T' is shown in Figure 19.

Claim 8.11 *There is no homomorphism $h : \tilde{T} \to T$ such that $h(p) = h(q)$. Furthermore, for any pair $(t_i, t_j) \in \{t_1, t_2, t_3, t_4\}^2 \setminus \{(t_1, t_1), (t_2, t_2), (t_3, t_3), (t_4, t_4)\}$ there exists a homomorphism $h : \tilde{T} \to T$ such that $h(p) = t_i$ and $h(q) = t_j$.*

Proof: Assume, for the sake of contradiction, that there exists $h : \tilde{T} \to T$ such that $h(p) = h(q)$. Since $\tilde{S}_{21}$ and $\tilde{S}_{34}$ are extended choosers, either $h(a^1_1) = h(a^1_2) = h(a^1_3)$ or $h(a^2_1) = h(a^2_2) = h(a^2_3)$, which contradicts Claim 8.10. We only prove the second part of the claim for the pair (t_1, t_2), all other cases being similar. We define $h : \tilde{T} \to T$ in such a way that $h(p) = t_1$ and $h(q) = t_2$. We start by defining $h(a^1_1) = t_1, h(a^1_2) = t_1$ and $h(a^1_3) = t_2$. We then extend h to the disjoint

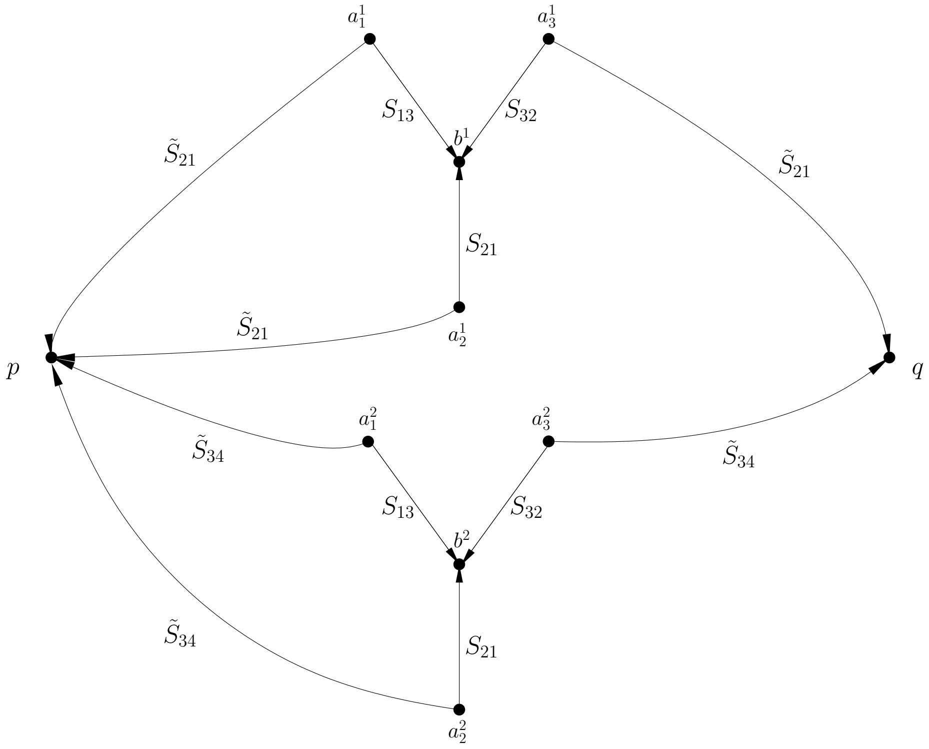


Figure 19: The gadget $\tilde{T}$.

copies of $\tilde{S}_{21}$ in $\tilde{T}$ using the definition of extended chooser. Also, we extend h to T_1' using Claim 8.10. If we choose $(h(a_1^2), h(a_2^2), h(a_3^2))$ to be any triple in $\{t_1, t_2\}^3 \setminus \{(t_1, t_1, t_1), (t_2, t_2, t_2)\}$, then it is easy to extend h to the disjoint copies of $\tilde{S}_{34}$ in $\tilde{T}$. Finally, using Claim 8.10 we extend h to T_2'. □

Note that the digraph T contains as a subgraph a copy of $T_i \cdot T_5^{-1}$, for each $1 \leq i \leq 4$. Abusing notation, we will say that $T_i \cdot T_5^{-1}$ is a subgraph of T, or write $T_i \cdot T_5^{-1} \subseteq T$, for $1 \leq i \leq 4$. Let Z be the subgraph of T, defined by the union of $T_1 \cdot T_5^{-1}$, $T_2 \cdot T_5^{-1}$ and $T_3 \cdot T_5^{-1}$. From the proof of Claim 8.11 and Claim 8.9, we obtain the next corollary.

Corollary 8.12 *For each pair $(t_i, t_j) \in \{t_1, t_2, t_3\}^2 \setminus \{(t_1, t_1), (t_2, t_2), (t_3, t_3)\}$ there exists a homomorphism $h : \tilde{T} \to T$ such that $h(p) = t_i$, $h(q) = t_j$, and the image of h is contained in Z.*

We can now define the reduction from Exact Four Colorability to Graph Acyclic Approximation: Given an undirected graph $G = \langle V, E\rangle$, the output of our reduction is the instance $(\varphi(G), T)$, where $\varphi(G)$ is a digraph and T is the directed acyclic graph we defined before. The digraph $\varphi(G)$ is constructed as follows: The node set of $\varphi(G)$ is V. For each edge $\{u, u'\} \in E$, we add a new disjoint copy of $\tilde{T}$ and identify the node p in $\tilde{T}$ with u and the node q in $\tilde{T}$ with u'. We then add a new node v_0. For each node $u \in V$ we add new disjoint copies of Q^* and T_5 (as defined before), and identify the initial node of Q^* with v_0, the terminal node of Q^* with u, and the terminal node of the copy of T_5 with u. Figure 20 shows the graph $\varphi(G)$ for a particular graph G. Clearly, the reduction can be computed in polynomial time in the size of G. We prove next that G is 4-colorable but not 3-colorable if and only if $\varphi(G) \to T$ but there is no proper subgraph S of T such that $\varphi(G) \to S$.

First, suppose that $G = \langle V, E\rangle$ is 4-colorable but not 3-colorable. Since G is 4-colorable, there exists a 4-coloring $c : V \to \{1, 2, 3, 4\}$ of G. We shall define a homomorphism $h : \varphi(G) \to T$. For each $u \in V$ let $h(u) = t_{c(u)}$. Then for each $\{u, u'\} \in E$ it holds that $h(u) \neq h(u')$ (because c is a coloring). Using Claim 8.11 we can extend h to all disjoint copies of $\tilde{T}$ in $\varphi(G)$. Finally, we define $h(v_0) = v$. Notice that the images of the copies of $Q^*$and T_5 in $\varphi(G)$ are completely determined. For example, if $u \in V$ satisfies $h(u) = t_1$, then the copy of Q^* associated with u has to be mapped to T_1 in T and the copy of T_5 associated with u has to be mapped to the copy of T_5 in T whose initial node is u_1. We conclude that $\varphi(G) \to T$.

Assume, for the sake of contradiction, that there exists a proper subgraph S of T such that $\varphi(G) \xrightarrow{g} S$. Then there exists $i^* \in \{1, 2, 3, 4\}$ such that $g(u) \neq t_{i^*}$ for all $u \in V$. Assume this is not the case, i.e. for all $i \in \{1, 2, 3, 4\}$ there exists $u \in V$ such that $g(u) = t_i$. Consider $i = 1$ and take $u \in V$ with $g(u) = t_1$. Since $hg(\varphi(G)) = hg(S) = 25$ we have that g preserves levels, implying that $g(v_0)$ is either v, u_1, u_2, u_3 or u_4. Because Q^* and T_5 are incomparable, it follows that $g(v_0) = v$. This implies that the copy of Q^* associated to u is mapped via g to the copy of T_1 in T. Using Lemma 8.3, Q^* is mapped via g in a surjective manner. Also, using again the incomparability between Q^* and T_5, it follows that the copy of T_5 associated with u is mapped via g to the copy of T_5 in T whose initial node is u_1. Furthermore, since T_5 is a core, this mapping is surjective as well. Then we conclude that $T_1 \cdot T_5^{-1}$ is contained in the homomorphic image of g. The same argument can be mimicked for each $i \in \{2, 3, 4\}$, and, thus, g is surjective, which implies that $S = T$, which is a contradiction. Thus, effectively there exists $i^* \in \{1, 2, 3, 4\}$, such that $g(u) \neq t_{i^*}$ for all $u \in V$. Using Claim 8.11 we have that $g(u) \neq g(u')$ for all $\{u, u'\} \in E$. This implies that G is 3-colorable, which is a contradiction.

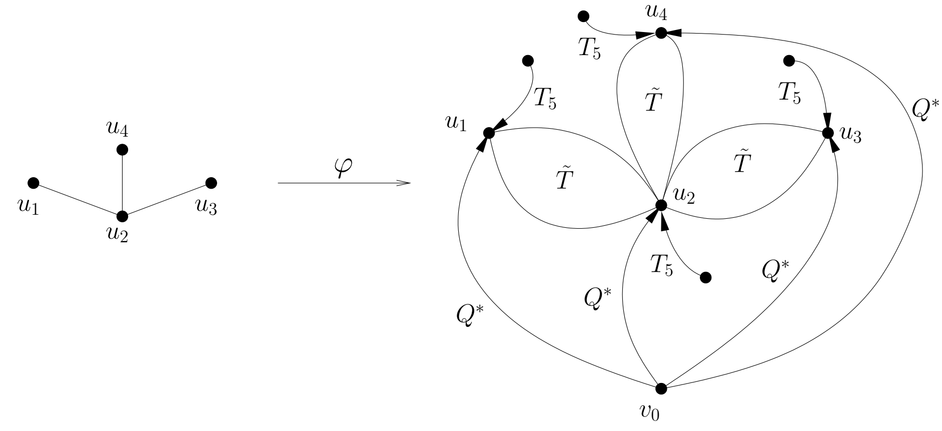


Figure 20: The digraph $\varphi(G)$ for the graph $G = (\{u_1, u_2, u_3, u_4\}, \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_2, u_4\}\})$.

Assume now that $\varphi(G) \to T$ but there is no proper subgraph S of T such that $\varphi(G) \to S$. Since $\varphi(G) \xrightarrow{h} T$, using Claim 8.11 we can deduce that $h(u) \neq h(u')$ for all $\{u, u'\} \in E$. Furthermore, we can define a 4-coloring for G. In fact, we can just take $c : V \to \{1, 2, 3, 4\}$ such that $c(u) = i$ if and only if $h(u) = t_i$, for each $u \in V$.

Assume, for the sake of contradiction, that there exists a 3-coloring $c : V \to \{1, 2, 3\}$ of G. We then define a homomorphism $g : \varphi(G) \to T$ as follows: First, for each $u \in V$ we define $g(u) = t_{c(u)}$. Since c is a coloring we have that $g(u) \neq g(u')$, for all $\{u, u'\} \in E$. Using Corollary 8.12, we can extend g to all $\varphi(G)$ in a way that the homomorphic image of g is contained in Z. But this is a contradiction since Z is a proper subgraph of T.

Note that as a corollary, we obtain that the following problem is also DP-complete:

PROBLEM:	EXACT ACYCLIC HOMOMORPHISM
INPUT:	a digraph G, an acyclic digraph T.
QUESTION:	Is $G \to T$ and $G \not\to S$ for every proper subgraph S of T?

Before proceeding with the proof, we recall the following lemma from [25]:

Lemma 8.13 *If G and H are two balanced digraphs such that $G \to H$, then $hg(G) \leq hg(H)$.*

We conclude the DP-hardness result proving the following proposition, that tell us that $(\varphi(G), T)$ is actually a reduction from EXACT FOUR COLORABILITY to GRAPH ACYCLIC APPROXIMATION.

Proposition 8.14 *Let $G = \langle V, E \rangle$ be a graph. Then $\varphi(G) \to T$ but there is no proper subgraph S of T such that $\varphi(G) \to S$ if and only if $\varphi(G) \to T$ but there is no acyclic digraph S such that $\varphi(G) \to S \not\rightleftarrows T$.*

Proof: The backward direction is trivial using the easily verifiable fact that T is a core. For the forward direction, assume by contradiction that there exists an acyclic digraph A such that

$\varphi(G) \xrightarrow{h} A \xrightarrow{g} T$ and $T \not\to A$. Suppose first that there exists $i^* \in \{1,2,3,4\}$, such that $g \circ h(u) \neq t_{i^*}$, for all $u \in V$. Using similar arguments as before, we have that G is 3-colorable and then there exists a proper subgraph S of T such that $\varphi(G) \to S$ (in fact, we can take $S = Z$), which is a contradiction. Thus, necessarily for each $i \in \{1,2,3,4\}$ there exists $u \in V$ such that $g \circ h(u) = t_i$.

Consider $i = 1$ and take $u \in V$ with $g \circ h(u) = t_1$. Let Q_1^* be the copy of Q^* associated with u, and let A' be the homomorphic image of the restriction of h to Q_1^* . Using the incomparability of Q^* and T_5, and the fact that $g \circ h$ preserves levels, it follows that the homomorphic image of the restriction of $g \circ h$ to Q_1^* is exactly T_1. This implies that the homomorphic image of A' via g is T_1. Thus $Q_1^* \to A' \to T_1$. Using the fact that A' is acyclic (it is a subgraph of A) and Claim 8.4, we have that there exists a homomorphism $g' : T_1 \to A'$. Let h_5 be the restriction of h to the copy of T_5 associated with u. We define a homomorphism r_1 from $T_1 \cdot T_5^{-1} \subseteq T$ to A. For each z in the copy of T_1 we define $r_1(z) = g'(z)$, and for each z in the copy of T_5 we define $r_1(z) = h_5(z)$. We prove that r_1 is well defined, i.e., $g'(t_1) = h_5(t_1)$. Using Lemma 8.13, we know that $hg(A) = 25$, so h preserves levels. Thus, $h(u)$ and $h(v_0)$ are the only nodes in A' with level 25 and 0, respectively. Again, since $hg(A') = 25$ we have that g' preserves levels, implying that $g'(t_1) = h(u) = h_5(u) = h_5(t_1)$. Notice also that $g'(v) = h(v_0)$. Thus, r_1 is well defined. Moreover, it is a homomorphism and $r_1(v) = g'(v) = h(v_0)$. We can apply the same argument for each $i \in \{2,3,4\}$, obtaining for each $1 \leq i \leq 4$ a homomorphism r_i from $T_i \cdot T_5^{-1} \subseteq T$ to A, such that $r_i(v) = h(v_0)$. Finally, we can define a homomorphism $r : T \to A$ as follows: First, if u is in $T_i \cdot T_5^{-1}$, for $1 \leq i \leq 4$, then $r(u) = r_i(u)$. Since v is the only common node of the subgraphs $T_i \cdot T_5^{-1}$, with $i \in \{1,2,3,4\}$, and $r_1(v) = r_2(v) = r_3(v) = r_4(v) = h(v_0)$, r is well defined and it is a homomorphism, which is a contradiction. □

Finally, we show that Graph Acyclic Approximation is DP-hard even when G and T are cores and T is fixed. We define a new function $\tilde{\varphi}$ from φ, such that for each undirected graph G it is the case that $\tilde{\varphi}(G)$ is a core and G is 4-colorable but not 3-colorable if and only if $\tilde{\varphi}(G) \to T$ and there is no acyclic S such that $\tilde{\varphi}(G) \to S \rightleftarrows\!\!\!\!/\; T$. Since T is already a fixed core, this is enough to prove the result.

First, note that $\tilde{T}$ is not a core, since for each a_j^i there are two distinct copies of T_{12} whose terminal nodes are identified with a_j^i (due to the structure of the choosers S_{13}, S_{21} and S_{32}). We modify $\tilde{T}$ and leave only one copy of T_{12} for each a_j^i. Observe that Claim 8.11 is still valid.

Next we introduce some notation. For a set of indices $X \subseteq \{1,2,3,4,5\}$, we denote by T_X the corresponding digraph from Claims 8.5 and 8.6. For example, for $X = \{3,5\}$ and $X = \{1,2,5\}$, T_X denotes T_{35} and T_{125} from Claims 8.5 and 8.6, respectively. If $X = \{k\}$, then $T_X = T_k$. Notice that $T_X \not\to T_Y$, for each $X, Y \subseteq \{1,2,3,4,5\}$ such that $Y \not\subseteq X$.

Claim 8.15 *If h is a homomorphism from $\tilde{T}$ to $\tilde{T}$ such that $h(p) = p$ and $h(q) = q$, then h is the identity mapping.*

Proof: Using Lemma 4.5, Claim 8.5, Claim 8.6 and the fact that $T_X \not\to T_Y$, for each $X, Y \subseteq \{1,2,3,4,5\}$ such that $Y \not\subseteq X$, the claim follows by a straightforward case analysis. □

For each $n \geq 1$, consider the oriented path $W_n = 000(10)^n 0$, illustrated in Figure 21. Notice that $hg(W_n) = 4$. For each $1 \leq k \leq n$ we define W_n^k to be W_n plus an edge from a new element z_k to x_k (an example is depicted in Figure 22). Notice that $hg(W_n^k) = 4$ as well.

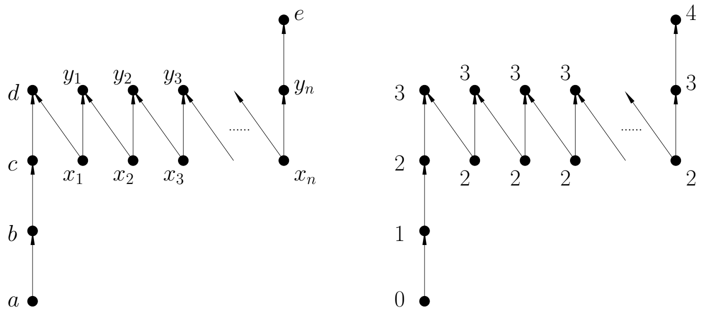


Figure 21: The digraph W_n and its levels.

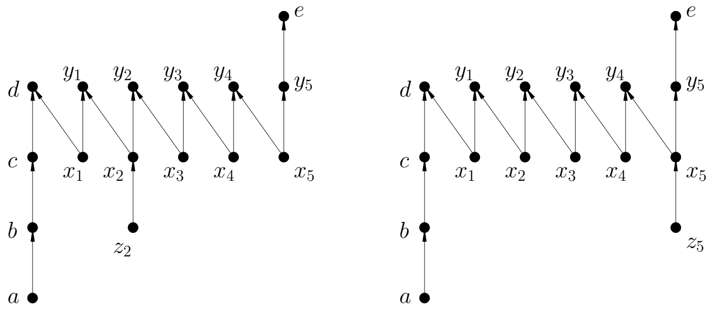


Figure 22: The digraphs W_5^2 and W_5^5.

Claim 8.16 *For each $n \geq 1$ the digraphs W_n^k ($k \in \{1, ..., n\}$) are incomparable cores.*

Proof: Suppose that W_n^k is not a core, for some $1 \leq k \leq n$. Then there exists $W_n^k \xrightarrow{h} W_n^k$, where h is not surjective. Necessarily, $h(a) = a$, $h(b) = b$, $h(c) = c$, $h(d) = d$ and $h(e) = e$ (since h preserves levels). This implies that $h(x_i) = x_i$ and $h(y_i) = y_i$, for each $1 \leq i \leq n$ (see Figure 22). Since $h(x_k) = x_k$, it must be the case that $h(z_k) = z_k$. This implies that h is surjective, which is a contradiction.

Now, suppose that $W_n^k \xrightarrow{h} W_n^{k'}$ for $k \neq k'$. Since h preserves levels, we have that h maps a, b, c, d and e in W_n^k to a, b, c, d and e in $W_n^{k'}$, respectively. This implies that h maps, for each $1 \leq i \leq n$, x_i and y_i in W_n^k to x_i and y_i in $W_n^{k'}$, respectively. Since $k \neq k'$, we have that z_k cannot be mapped in $W_n^{k'}$ via h, which is a contradiction. □

Consider the digraph S as defined in Figure 23. Recall that $\vec{P}_k = 0^k$, $P_6 = 0^7 10^5$ and $P_8 = 0^9 10^3$. The oriented path P_{135} is from Claim 8.2.

For each $n \geq 1$ and $1 \leq k \leq n$, we define the digraph S_n^k as follows: Take S and replace the directed path of length 4 in S that starts at z' and ends at z by a copy of W_n^k, identifying a with z' and renaming e to z (see Figure 24). Observe that $W_n^k \to \vec{P}_4$, thus $S_n^k \to S$.

Claim 8.17 *For each $n \geq 1$ the digraphs S_n^k ($k \in \{1, ..., n\}$) are incomparable cores.*

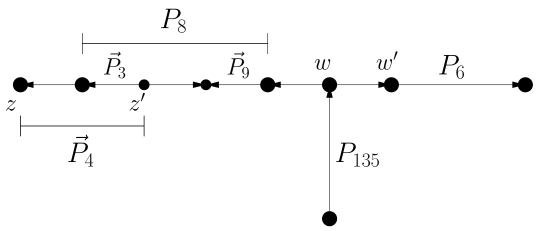


Figure 23: The digraph S.

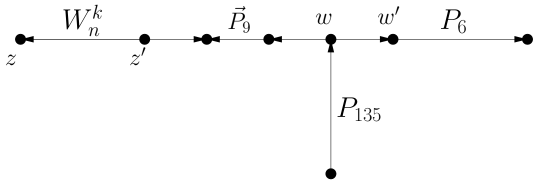


Figure 24: The digraph S_n^k.

Proof: Follows easily from Lemma 4.5, Claim 8.16 and the fact that P_6 and P_8 are incomparable. □

Now we define our main construction. Consider an undirected graph G and let $u_1, u_2, ..., u_n$ be an arbitrary enumeration of its nodes. We define $\tilde{\varphi}(G)$ to be $\varphi(G)$, where, in addition, for each $1 \leq k \leq n$ we add a disjoint copy of S_n^k and identify z in S_n^k with u_k. Clearly, $\tilde{\varphi}(G)$ can be computed in polynomial time in the size of G. To conclude, we prove the following proposition:

Proposition 8.18 *For every undirected graph G, the digraph $\tilde{\varphi}(G)$ is a core. Furthermore, G is 4-colorable but not 3-colorable if and only if $\tilde{\varphi}(G) \to T$ and there is no acyclic S such that $\tilde{\varphi}(G) \to S \not\rightleftarrows T$.*

Proof: Let G be an undirected graph and h a homomorphism from $\tilde{\varphi}(G)$ to $\tilde{\varphi}(G)$. We shall prove that h is surjective, implying that $\tilde{\varphi}(G)$ is a core. Recall that $u_1, ..., u_n$ is an enumeration of the nodes of G, which are by definition contained in $\tilde{\varphi}(G)$. We have that $h(u_k) = u_k$, for each $1 \leq k \leq n$. Indeed, since h preserves levels, we know that h maps u_k to u_l, for some $1 \leq l \leq n$, or maps u_k to some node in a copy of $\tilde{T}$ with level 25. Using the facts that $Q^* \not\to T_X$, for some X that contains element 5 (since $Q^* \not\to T_5$) and $T_X \not\to T_Y$, for $Y \nsubseteq X$, we can easily show that the second case is not possible, since we have copies of T_{345} (from $\tilde{S}_{21}$), T_{15} (from $\tilde{S}_{34}$) and Q^* whose terminal nodes are identified with u_k, as shown in Figure 25. For example, h cannot map u_k to a_1^1, otherwise $T_{345} \to T_{12}$, $T_{345} \to T_{15}$ or $T_{345} \to T_{125}$. Similarly, h cannot map u_k to b^1, otherwise $Q^* \to T_{15}$, $Q^* \to T_{25}$ or $Q^* \to T_{35}$ (see Figure 25). For the other nodes in $\tilde{T}$ with level 25, we have similar contradictions.

Thus, h maps u_k to u_l, for some $1 \leq l \leq n$. We shall show that $l = k$. Indeed, suppose that $l \neq k$. Since h preserves levels and the only node with level 25 in the copy of S_n^k whose terminal node is u_k, is precisely u_k, we have that h maps this copy of S_n^k either to a copy of Q^*, T_{345}, T_{15}, T_5 or S_n^l, whose terminal node is u_l. Suppose that h maps the copy of S_n^k to the copy of Q^*. Necessarily, h maps w in S_n^k to a_7 in Q^* (see Figures 24 and 7). This implies that $P_{135} \to P_7$, which

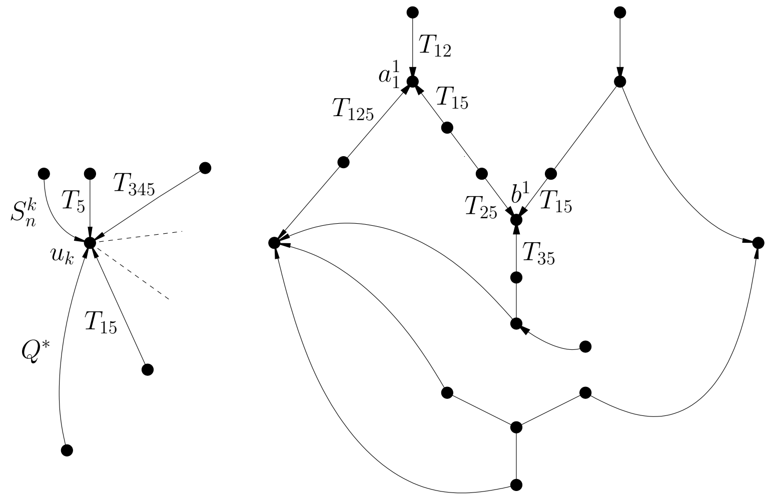


Figure 25: The node u_k and a copy of $\tilde{T}$.

is a contradiction with Claim 8.2. Now, suppose that h maps the copy of S_n^k to the copy of T_5. It follows that h maps w' in S_n^k to the initial node of the copy of P_8 in T_5, implying that $P_6 \to P_8$ or $P_6 \to P_9$ (Figures 24 and 11), which is a contradiction. The cases when the copy of S_n^k is mapped to T_{345} or T_{15} lead a contradiction as well, since $T_{345} \to T_5$ and $T_{15} \to T_5$. Finally, suppose that h maps the copy of S_n^k to the copy of S_n^l whose terminal node is u_l. It follows that $S_n^k \to S_n^l$, which contradicts Claim 8.17. In any case we have a contradiction, thus $l = k$ and $h(u_k) = u_k$, for each $1 \leq k \leq n$.

Now, observe that for each $1 \leq k \leq n$, the copies of T_5 and S_n^k whose terminal node is u_k have to be mapped via h to themselves, in a surjective manner. Moreover, $h(v_0) = v_0$, otherwise $Q^* \to T_5$, $Q^* \to T_{15}$ or $Q^* \to T_{345}$, which is a contradiction. This implies that h maps all disjoint copies of Q^* to themselves, in a surjective manner as well. Finally, observe that h maps all the copies of $\tilde{T}$ to themselves too. Indeed, using the facts that h preserves levels, $T_X \not\to T_Y$, for $Y \not\subseteq X$, and $h(u_k) = u_k$, for each $1 \leq k \leq n$, we can easily show that h maps each node in a copy of $\tilde{T}$ to a node inside the same copy of $\tilde{T}$. Thus, we can use Claim 8.15, to conclude that h maps all copies of $\tilde{T}$ to themselves in a surjective manner, implying that h is actually surjective. This proves that $\tilde{\varphi}(G)$ is a core.

Finally, observe that for each $1 \leq i \leq 4$ and $1 \leq k \leq n$, there is a homomorphism g_i from S_n^k to T_i such that g_i maps the node z in S_n^k to the terminal node of T_i. Thus, if we are constructing a homomorphism from $\tilde{\varphi}(G)$ to T, for any values of the images of the u_k's, we can always define images for the copies of the S_n^k's. Therefore, we can use exactly the same arguments as in the proofs of the correctness of φ and Proposition 8.14. □

This finishes the proof of Theorem 4.12.