SIAM J. COMPUT.
Vol. 45, No. 4, pp. 1339–1376

SEMANTIC ACYCLICITY ON GRAPH DATABASES*

PABLO BARCELÓ†, MIGUEL ROMERO†, AND MOSHE Y. VARDI‡

Abstract. It is known that unions of acyclic conjunctive queries (CQs) can be evaluated in linear time, as opposed to arbitrary CQs, for which the evaluation problem is NP-complete. It follows from techniques in the area of constraint-satisfaction problems that *semantically acyclic* unions of CQs—i.e., unions of CQs that are equivalent to a union of acyclic ones—can be evaluated in polynomial time, though testing membership in the class of semantically acyclic CQs is NP-complete. We study here the fundamental notion of semantic acyclicity in the context of graph databases and unions of conjunctive regular path queries with inverse (UC2RPQs). It is known that unions of acyclic C2RPQs can be evaluated efficiently, but it is by no means obvious whether similarly good evaluation properties hold for the class of UC2RPQs that are semantically acyclic. We prove that checking whether a UC2RPQ is semantically acyclic is EXPSPACE-complete and obtain as a corollary that evaluation of semantically acyclic UC2RPQs is fixed-parameter tractable. In addition, our tools yield a strong theory of approximations for UC2RPQs when no equivalent acyclic UC2RPQ exists.

Key words. graph databases, conjunctive regular path queries, conjunctive queries, acyclicity, query evaluation, query approximation, constraint satisfaction problems

AMS subject classification. 68P15

DOI. 10.1137/15M1034714

1. Introduction.

Conjunctive queries (CQs) are the most fundamental class of database queries and also the most intensively studied in the database theory community. The evaluation problem for CQs is as follows: Given a CQ θ, a database $\mathcal{D}$, and a tuple $\bar{a}$ of elements in $\mathcal{D}$, does $\bar{a}$ belong to the result $\theta(\mathcal{D})$ of applying θ to $\mathcal{D}$? Notice that the cost of evaluation is thus measured in terms of the size $|\mathcal{D}|$ of the database $\mathcal{D}$ and $|\theta|$ of the query θ, which in database terminology corresponds to the *combined complexity* of the problem [41].

The evaluation problem for CQs is NP-complete [12]; this motivated a flurry of activity for finding tractable cases of the problem. One of the oldest and most important such restrictions is *acyclicity*. Acyclic CQs can in fact be evaluated in linear time in both data and query size—$O(|\mathcal{D}| \cdot |\theta|)$ [44]. This good behavior extends to unions of CQs (UCQs), each one of which is acyclic (the so-called acyclic UCQs).

Acyclicity is a syntactic property of queries that is by now well-understood [25]. On the other hand, the *space* of UCQs that is defined by the notion of *semantic acyclicity*—that is, the UCQs that are *equivalent* to an acyclic one—has not received much attention. We call this the space of *semantically acyclic* UCQs. Two questions naturally arise in this context:

1. Is the evaluation problem for semantically acyclic UCQs still tractable?
2. What is the cost of verifying whether a UCQ is semantically acyclic?

Answers to these questions can be found by applying known techniques in the area of *constraint satisfaction problems* (CSP). This is based on the fact that CQ evalua-

*Received by the editors August 12, 2015; accepted for publication (in revised form) May 4, 2016; published electronically August 2, 2016. This paper is the full version of the conference article [7].
http://www.siam.org/journals/sicomp/45-4/M103471.html

†Center for Semantic Web Research & Department of Computer Science, University of Chile, Santiago, Chile (pbarcelo@dcc.uchile.cl, mromero@dcc.uchile.cl). These authors were funded by the Millennium Nucleus Center for Semantic Web Research under grant NC120004. The second author was also funded by a CONICYT Ph.D. Scholarship.

‡Department of Computer Science, Rice University, Houston, TX 77251 (vardi@cs.rice.edu).

tion and CSP have a common root—they are both equivalent to the *homomorphism problem* [33]. CSP techniques establish the following:

1. Semantically acyclic UCQs can be evaluated in polynomial time.
2. Verifying whether a UCQ is semantically acyclic is NP-complete [14, 17].

In this paper we extend the concept of semantic acyclicity from the classical setting of relational databases to the newer setting of graph databases [2], which has been the focus of much research in the last few years [4, 19, 20, 5, 22]. In fact, acyclicity has been identified as a fundamental tool for obtaining tractable—and even linear time—query evaluation in such a context [28, 4, 5, 35]. It is thus of theoretical importance to understand what is the space of queries defined by the notion of acyclicity over graph databases.

Graph databases are typically modeled as edge-labeled directed graphs. In this context, query languages are *navigational*, in the sense that they allow one to recursively traverse the edges of the graph while checking for some regular condition [16, 1, 10, 43, 3]. Navigation is often performed by traversing edges in both directions, which allows one to express important properties about the inverse of the relations defined by the labels of those edges [9, 10]. When this is combined with the expressive power of CQs, it yields a powerful class of queries—the so-called *conjunctive regular path queries with inverse*, or C2RPQs—that lies at the core of many query languages for graph databases (see, e.g., [16, 15, 9, 4]).

Evaluation of unions of C2RPQs (UC2RPQs) is not more expensive than evaluation of CQs, i.e., NP-complete. Recent works have studied the class of acyclic UC2RPQs—where acyclicity is defined in terms of the underlying CQ of each element of the union—and proved that queries in this class preserve the good properties of acyclic UCQs for evaluation, i.e., they can be evaluated in polynomial time, and even linearly for suitable restrictions [4, 5].

In this work we study the notion of *semantic acyclicity* for UC2RPQs, that is, we study the class of UC2RPQs that are equivalent to an acyclic one, and try to answer the same questions that we posed before for the class of semantically acyclic UCQs:

1. What is the cost of evaluating queries in this class?
2. How hard is it to recognize whether a UC2RPQ Γ is semantically acyclic, and, if so, what is the cost of computing an equivalent acyclic UC2RPQ for Γ?

The first question is important since we want to understand whether semantic acyclicity leads to larger classes of UC2RPQs with good evaluation properties. The second question is relevant for static optimization, as a positive answer would allow us to construct an equivalent query in a well-behaved fragment for each semantically acyclic UC2RPQ. We present answers to both questions in the paper, in a way that our answer to the first question crucially depends on our answer to the second one.

As mentioned before, the evaluation problem for semantically acyclic UCQs is tractable, and this is proved by applying known techniques from CSP. Those techniques are specifically tailored for checking the existence of a homomorphism from a relational structure into another one, which fits the semantics of CQs well. On the other hand, the semantics of C2RPQs is based on a richer notion of homomorphism, which maps the atoms of a query into pairs of nodes in a graph database linked by a path satisfying some regular condition. Such a notion of homomorphism does not fit well in the current landscape of CSP techniques, and, in fact, CSP theory does not yield answers to our questions about semantically acyclic UC2RPQs.

Example 1. To illustrate the usefulness of semantic acyclicity, consider an alpha-

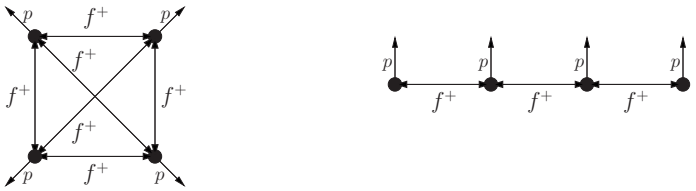

FIG. 1.1. *The query from Example* 1 *and its equivalent acyclic query.*

bet $\{p, f\}$ modeling social networks. Labels p and f stand for "person" and "friend", respectively. An object in a graph database is a person if it has an outgoing p-labeled edge (e.g., to its id). Two persons are friends if there is an f-labeled edge between them. Suppose we want to know whether there is a group of persons of size $n \geq 2$ in the graph database $\mathcal{G}$ that forms a *clique*. This means that there is an f-labeled edge between each pair of persons in the group. Evaluating this query basically corresponds to the well-known *clique problem*, which is NP-complete. Typical algorithms for this problem take $|\mathcal{G}|^{O(n)}$ time.

On the other hand, in the context of graph databases it is sometimes more interesting knowing whether two persons are linked by a path labeled in the friendship relationship than in checking whether they are actual friends. This means that there is a sequence of mutual friends that connects these two persons, or, more formally, that there is a path between them labeled by the regular expression f^+. Our clique query can thus also be relaxed by using this condition. The left-hand side figure in Figure 1.1 depicts the result for $n = 4$ (dots represent variables and arrows represent labeled atoms). Notice that this query is nonacyclic, and by applying standard algorithms we can still evaluate it in $|\mathcal{G}|^{O(n)}$ time. However, it is not hard to prove that the query is semantically acyclic. Indeed, it is equivalent to a *path* of n persons that are consecutively linked by paths of mutual friends. This query is depicted in the right-hand side of Figure 1.1 (for $n = 4$ again). The existence of such a path can be checked in time $O(n \cdot |\mathcal{G}|^2)$. Therefore, evaluating a semantically acyclic query via its equivalent acyclic query can significantly improve the running time. ∎

To attack our questions about evaluation of semantically acyclic UC2RPQs, we consider first the problem of UC2RPQ *approximations*, which is motivated by recent work on approximations of UCQs [6]. Typical algorithms require $|\mathcal{D}|^{O(|\theta|)}$ time to evaluate a CQ θ on a database $\mathcal{D}$, which might be prohibitively expensive for a large dataset $\mathcal{D}$ even if θ is small. This led the idea of finding *approximations* of (U)CQs in tractable classes [6], in particular, in the class of acyclic (U)CQs. Intuitively, an acyclic UCQ Θ' is an approximation of a UCQ Θ if Θ' is contained in Θ and it is "as close as possible" to Θ in the class of acyclic UCQs. The latter means that Θ' is a maximal acyclic UCQ that is contained in Θ. It follows from techniques in [6] that UCQs have good properties in terms of acyclic approximations: Each UCQ Θ has a unique acyclic approximation (up to equivalence), and this approximation can be computed in single-exponential time. These good properties imply that computing and running the acyclic approximation of a UCQ Θ on a database $\mathcal{D}$ takes time $O(|\mathcal{D}| \cdot 2^{p(|\Theta|)})$, for some polynomial $p : \mathbb{N} \to \mathbb{N}$. This is much faster than $|\mathcal{D}|^{O(|\Theta|)}$ on large databases. Thus, if the quality of the approximation is good, we may prefer to run this faster query instead of Θ.

We show here that the good properties of UCQs in terms of acyclic approximations extend to UC2RPQs. In particular, we show that each UC2RPQ Γ has a unique

acyclic approximation (up to equivalence) and that an approximation of exponential size can be computed in EXPSPACE. The data complexity of evaluating this approximation is then quadratic in the size of the data, just as the data complexity of 2RPQs. This shows that acyclic approximations might be useful when evaluating whether the original query is infeasible, though the cost of computing the approximation is quite high. We also show that UC2RPQs behave provably worse than UCQs in terms of approximations: Verifying whether an acyclic UCQ Θ' is the approximation of the UCQ Θ is in the second-level of the polynomial hierarchy, but it becomes EXPSPACE-complete in the case of UC2RPQs. This is not surprising, as it is known that testing containment of UC2RPQs is EXPSPACE-complete [9].

Finally, we apply the machinery of acyclic approximation of UC2RPQs to address semantic acyclicity of this class of queries. As noted above, we can construct in EXPSPACE an exponential-sized acyclic approximation Γ' of a given UC2RPQ Γ. By construction, Γ' is contained in Γ. To check whether Γ is semantically acyclic we just have to check whether Γ is contained in Γ'. Because Γ' is exponentially large, we trivially get a 2EXPSPACE upper bound for the complexity of this step. Surprisingly, we show that this can actually be improved to EXPSPACE. In order to prove this, we show that checking containment of a UC2RPQ Γ_1 in an acyclic UC2RPQ Γ_2 can be solved using exponential space but in such a way that the exponent only depends on the *width* of Γ_2, which is the maximum number of atoms in some disjunct of Γ_2 that mention the same pair of variables. As it turns out, this parameter is polynomial in $|\Gamma|$ for the acyclic approximation Γ'. Thus, applying this refined algorithm for containment we can check whether $\Gamma \subseteq \Gamma'$ in EXPSPACE. We also provide a matching lower bound for this problem.

This machinery allows us to get answers to the two questions we posed above:

1. First, query evaluation for semantically acyclic UC2RPQ is *fixed-parameter tractable*, when we consider the size of the query as the parameter. Recall that this means that there exist a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant $k \geq 1$ such that evaluating a UC2RPQ Θ over a database $\mathcal{D}$ can be done in time $O(|\mathcal{D}|^k \cdot f(|\Theta|))$. This shows that the class of semantically acyclic UC2RPQs is well-behaved in terms of evaluation, as in general evaluation of UC2RPQs (and even CQs) is not fixed-parameter tractable [37] (under widely held complexity theoretical assumptions).
2. Second, testing semantic acyclicity for UC2RPQs is EXPSPACE-complete.

The question of whether semantically acyclic UC2RPQs can be evaluated in polynomial time is left as an interesting open problem.

Organization. The rest of the paper is organized as follows. In section 2, we study semantic acyclicity for UCQs and show that the answers to the most basic questions follow from known CSP techniques. In section 3, we introduce graph databases and UC2RPQs. Section 4 is devoted to presenting the main techniques and results related to containment of UC2RPQs. In section 5, we study acyclic approximations of UC2RPQs and show some of their good properties. Then in section 6, we study semantic acyclicity of UC2RPQs. We provide upper and lower bounds for the problem of verifying whether a UC2RPQ is semantically acyclic and show that this implies that evaluation of semantically acyclic UC2RPQs is fixed-parameter tractable. In section 7 we provide concluding remarks and a list of open problems. Finally, the appendix contains the proofs of some technical results in the paper.

2. Interlude on unions of CQs.

We start by considering semantic acyclicity in the context of traditional relational databases and unions of conjunctive queries.

Although the results in this section follow from known techniques, we state them for the sake of completeness and because they will help us develop the necessary intuitions for the more complicated case of graph databases and unions of conjunctive regular path queries.

2.1. Basic concepts. We first provide the necessary terminology. A *schema* is a set σ of relation names $R_1, \ldots, R_\ell$, each relation R_i having an arity $n_i > 0$. A *database* of schema σ is a function $\mathcal{D}$ that maps each relation symbol R_i in σ into a finite n_i-ary relation $R_i^{\mathcal{D}}$ over a countably infinite domain dom (i.e., $R_i^{\mathcal{D}} \subseteq \mathsf{dom}^{n_i}$).

A conjunctive query (CQ) over σ is a logical formula in the $\exists, \wedge$-fragment of first-order logic, i.e., a formula of the form

$$\theta(\bar{x}) \;=\; \exists \bar{y} \bigwedge_{i=1}^{m} P_i(\bar{u}_i), \tag{2.1}$$

where each P_i is a symbol from σ and $\bar{u}_i$ is a tuple of variables among $\bar{x}, \bar{y}$ whose length is the arity of P_i. As usual, we assume that $\bar{x}$ are the *free* variables of θ, i.e., the variables mentioned in θ that are not existentially quantified. Each $P_i(\bar{u}_i)$ is an *atom* of $\theta(\bar{x})$.

As customary, we define the semantics of CQs in terms of *homomorphisms*. Let $\theta(\bar{x})$ be a CQ of the form (2.1) and let $\mathcal{D}$ be a database over schema σ. A homomorphism h from $\theta(\bar{x})$ to $\mathcal{D}$ is a mapping from the variables that appear in the atoms of $\theta(\bar{x})$ to the elements of $\mathcal{D}$ such that $P_i(h(\bar{u}_i)) \in \mathcal{D}$ for each $1 \leq i \leq m$. The *evaluation of* θ *over* $\mathcal{D}$, denoted by $\theta(\mathcal{D})$, is the set of all tuples of the form $h(\bar{x})$, for h a homomorphism from $\theta(\bar{x})$ to $\mathcal{D}$. If θ is a Boolean query (i.e., $\bar{x}$ is the empty tuple), the answer `true` is, as usual, modeled by the set containing the empty tuple, and the answer `false` by the empty set.

A *union* of conjunctive queries (UCQ) is a first-order formula of the form

$$\Theta(\bar{x}) \;=\; \bigvee_{1 \leq i \leq m} \theta_i(\bar{x}),$$

where $\theta_i(\bar{x})$ is a CQ for each $1 \leq i \leq m$. Note that every disjunct in $\Theta(\bar{x})$ has the same tuple of free variables, namely, $\bar{x}$. We define $\Theta(\mathcal{D})$ to be $\bigcup_{1 \leq i \leq m} \theta_i(\mathcal{D})$.

As it is common in database theory [3, 25, 37, 41], we formalize query evaluation as a decision problem. The *evaluation problem for UCQs* is as follows: Given a database $\mathcal{D}$, a UCQ $\Theta(\bar{x})$, and a tuple $\bar{a}$ in $\mathcal{D}$, is $\bar{a} \in \Theta(\mathcal{D})$?

It is well known that the evaluation of CQs is NP-complete [12]. On the other hand, tractability of (U)CQ evaluation can be obtained by restricting the syntactic shape of CQs. The oldest and most common of such restrictions is *α-acyclicity* (or, simply, acyclicity) [18], that can be defined in terms of the existence of a well-behaved *tree decomposition* of the *hypergraph* of a CQ [26]. We review such notions below.

Recall that a hypergraph is a tuple $\mathcal{H} = (V, E)$, where V is its finite set of vertices and $E \subseteq 2^V$ is its set of *hyperedges*. With each CQ θ we associate its hypergraph $\mathcal{H}(\theta) = (V, E)$ such that V is the set of variables of θ and E consists of all sets of variables that appear in the same atom of θ. Consider, for instance, the CQ

$$\theta(x) = \exists y \exists z \exists u \exists v \big(R(x, y, z) \wedge T(y, u, u) \wedge S(y, v)\big).$$

Then $\mathcal{H}(\theta) = (V, E)$, where $V = \{x, y, z, u, v\}$ and E consists of the hyperedges $\{x, y, z\}$, $\{y, u\}$, and $\{y, v\}$.

A tree decomposition of a hypergraph $\mathcal{H} = (V, E)$ is a pair (T, λ), where T is a tree and λ is a mapping from the nodes of T to 2^V, that satisfies the following:

- For each $v \in V$ the set $\{t \in T \mid v \in \lambda(t)\}$ is a connected subset of T.
- Each hyperedge in E is contained in one of the sets $\lambda(t)$, for $t \in T$.

Then $\mathcal{H}$ is *acyclic* if there is a tree decomposition (T, λ) of it such that $\lambda(t)$ is a hyperedge in E, for each $t \in T$.

A CQ θ is acyclic if its hypergraph $\mathcal{H}(\theta)$ is acyclic. A UCQ $\bigvee_{1 \leq i \leq m} \theta_i(\bar{x})$ is acyclic if each $\theta_i(\bar{x})$ is acyclic $(1 \leq i \leq m)$. For instance, the CQ

$$\theta(x) \;=\; \exists y \exists z \exists u \exists v \, R(x, y, z) \wedge T(y, u, u) \wedge S(y, v)$$

presented above is acyclic, as witnessed by the following tree decomposition (T, λ) of $\mathcal{H}(\theta)$: T consists of vertices $\{1, 2, 3\}$ and edges $\{(1, 2), (1, 3)\}$, and $\lambda(1) = \{x, y, z\}$, $\lambda(2) = \{y, u\}$, and $\lambda(3) = \{y, v\}$.

It follows from the seminal work of Yannakakis that acyclic UCQs have good properties in terms of evaluation.

PROPOSITION 2.1 (see [44]). *The evaluation problem for acyclic UCQs can be solved in linear time* $O(|\mathcal{D}| \cdot |\Theta|)$.

2.2. Semantically acyclic UCQs. Acyclicity is a syntactic property of UCQs. On the other hand, a nonacyclic UCQ can still be equivalent to an acyclic one. Formally, a UCQ $\Theta(\bar{x})$ is *semantically acyclic* if there exists an acyclic UCQ $\Theta'(\bar{x})$ such that $\Theta(\mathcal{D}) = \Theta'(\mathcal{D})$ for each database $\mathcal{D}$. Recall that we are interested in two questions regarding semantic acyclicity:

1. Is the evaluation problem for semantically acyclic UCQs tractable?
2. What is the cost of verifying semantic acyclicity for UCQs?

As pointed out in [33], there is a close connection between CQ evaluation and constraint satisfaction: Both can be recast as the problem of determining whether there is a homomorphism from one relational structure into another one. This tight connection allows us to export tools from CSP [33, 14] and prove that semantically acyclic UCQs can be evaluated in polynomial time.

THEOREM 2.2. *The evaluation problem for semantically acyclic UCQs can be solved in polynomial time.*

The CSP techniques that imply Theorem 2.2 first establish a sophisticated equivalence between the problems of query evaluation for semantically acyclic CQs and the existence of winning strategies for the duplicator in some refined version of the existential pebble game, and then prove that the required condition on games can be checked efficiently. Since the proof of Theorem 2.2 is not essential to our main argumentation line we relegate it to the appendix.

Notice that the class of acyclic UCQs is remarkably well behaved for evaluation: Queries in the class are not only tractable, but also verifying whether a given UCQ belongs to the class can be done in polynomial (in fact, linear) time [40]. On the other hand, using techniques similar to those in [17], one can prove that this good behavior does not extend to the class of semantically acyclic UCQs, as the problem of verifying whether a query is semantically acyclic is computationally hard (again, we relegate the proof of this fact to the appendix).

PROPOSITION 2.3. *The problem of verifying whether a UCQ* $\Theta(\bar{x})$ *is semantically acyclic is* NP*-complete. It remains* NP*-hard even when the input is restricted to Boolean CQs whose schema consists of a single binary relation (i.e., directed graphs).*

3. Graph databases and conjunctive regular path queries.

3.1. Graph databases and C2RPQs. A *graph database* [2, 10, 16] is just a finite edge-labeled graph. Let Σ be a finite alphabet, and let $\mathcal{N}$ be a countably infinite set of node ids. Then a graph database over Σ is a pair $\mathcal{G} = (N, E)$, where N is the set of nodes (a finite subset of $\mathcal{N}$), and E is the set of edges, i.e., $E \subseteq N \times \Sigma \times N$. That is, we view each edge as a triple (n, a, n'), whose interpretation, of course, is an a-labeled edge from n to n'. When Σ is clear from the context, we shall simply speak of a graph database.

As mentioned before, in this work we consider navigational queries that traverse edges in both directions. The building block of these languages is the class of *regular path queries with inverse*, or 2RPQs [9, 10]. A 2RPQ over finite alphabet Σ is a nondeterministic finite automaton (NFA) over the alphabet $\Sigma^{\pm}$ that extends Σ with the symbol a^- for each $a \in \Sigma$. Intuitively, a^- represents the inverse of a.

Intuitively, the evaluation of a 2RPQ $\mathcal{A}$ *without inverses* over a graph database $\mathcal{G}$ consists of all those pairs (n, n') of nodes in $\mathcal{G}$ such that n' can be reached from n in $\mathcal{G}$ by following a path whose label is accepted by $\mathcal{A}$. In order to accommodate inverses one interprets $\mathcal{A}$ not over $\mathcal{G}$ but over the *completion* $\mathcal{G}^{\pm}$ of $\mathcal{G}$, which is the graph database over $\Sigma^{\pm}$ that is obtained from $\mathcal{G}$ by adding the edge (n', a^-, n), for each edge $(n, a, n') \in E$. We formalize this below.

Let $\mathcal{G} = (N, E)$ be a graph database. A *path* ρ from node n to n' in $\mathcal{G}$ is a sequence

$$\rho \;=\; n_0 a_1 n_1 a_2 n_2 \dots n_{k-1} a_k n_k,$$

where $k \geq 0$, $n_0 = n$, $n_k = n'$, and for each $1 \leq i \leq k$ it is the case that (n_{i-1}, a_i, n_i) is an edge in E. Notice that when $k = 0$ this path consists of the single node n_0. The *label* of ρ, denoted by $\mathsf{label}(\rho)$, is the word $a_1 a_2 \dots a_k \in \Sigma^*$. The evaluation $\mathcal{A}(\mathcal{G})$ of the 2RPQ $\mathcal{A}$ over the graph database $\mathcal{G}$ is the set of all pairs $(n, n') \in N \times N$ such that there is a path ρ in $\mathcal{G}^{\pm}$ from n to n' for which it is the case that the word $\mathsf{label}(\rho)$ is accepted by $\mathcal{A}$.

A folklore result (see, e.g., [3]) establishes that the problem of computing $\mathcal{A}(\mathcal{G})$, for a given graph database $\mathcal{G}$ and 2RPQ $\mathcal{A}$, can be solved in polynomial time $O(|\mathcal{G}|^2 \cdot |\mathcal{A}|)$.

When the expressive power of 2RPQs is combined with the ability of CQs to express arbitrary joins and existential quantification, it yields a powerful class of queries—namely, the *conjunctive regular path queries with inverses*, or C2RPQs [9]—that we define next.

Formally, a C2RPQ over a finite alphabet Σ is an expression of the form

$$\gamma(\bar{x}) \;=\; \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, \mathcal{A}_i, v_i), \tag{3.1}$$

such that each u_i and v_i is a variable among $\bar{x}, \bar{y}$ and each $\mathcal{A}_i$ is a 2RPQ over Σ, for $1 \leq i \leq m$. A CRPQ is a C2RPQ without inverses, i.e., a C2RPQ of the form (3.1) in which each $\mathcal{A}_i$ $(1 \leq i \leq m)$ is an NFA over Σ. As usual, we assume that $\bar{x}$ are the *free* variables of γ, i.e., the variables mentioned in γ that are not existentially quantified. Each $(u_i, \mathcal{A}_i, v_i)$ is an *atom* of γ.

We formally define the semantics of C2RPQs by using a notion of homomorphism that maps atoms of γ into pairs that satisfy the corresponding 2RPQs. Given $\gamma(\bar{x})$ of the form (3.1) and a graph database $\mathcal{G} = (N, E)$, a homomorphism from $\gamma(\bar{x})$ to $\mathcal{G}$ is a map $h : \bigcup_{1 \leq i \leq m} \{u_i, v_i\} \to N$ such that $(h(u_i), h(v_i)) \in \mathcal{A}_i(\mathcal{G})$ for every $1 \leq i \leq m$. We then define the evaluation $\gamma(\mathcal{G})$ of γ over $\mathcal{G}$ to be the set of all tuples $h(\bar{x})$ such that h is a homomorphism from $\gamma(\bar{x})$ to $\mathcal{G}$.

A union of C2RPQs (UC2RPQ) is a formula of the form

$$\Gamma(\bar{x}) \;=\; \bigvee_{1\leq i\leq m} \gamma_i(\bar{x}),$$

where each $\gamma_i(\bar{x})$ is a C2RPQ ($1 \leq i \leq m$). We define $\Gamma(\mathcal{G})$ as $\bigcup_{1\leq i\leq m} \gamma_i(\mathcal{G})$, for each graph database $\mathcal{G}$. As before, the UC2RPQ Γ is Boolean if $\bar{x}$ is the empty tuple. In such a case the evaluation of Γ over $\mathcal{G}$ corresponds to either `true` or `false`.

Remark. While we use NFAs in order to specify regular languages in UC2RPQs, it is sometimes more convenient (especially in practice) to specify such languages using regular expressions. In fact, this is the approach often followed in the graph database literature (see, e.g., [3, 43]). Each regular expression can be easily translated into an NFA in polynomial time, and, therefore, all the evaluation upper bounds obtained in the paper continue to hold in the case when UC2RPQs are specified using regular expressions. The reason why we use NFA instead is that they allow one to express regular languages more succinctly than the latter (which is crucial for some of our results). For the sake of readability though, all the examples we present in the paper are specified using regular expressions.

Example 2. Consider a graph database $\mathcal{G} = (N, E)$ of researchers, papers, conferences, and journals over the alphabet $\Sigma = \{\mathsf{creator}, \mathsf{inJournal}, \mathsf{inConf}\}$. The set of edges E consists of the following:

- All tuples $(r, \mathsf{creator}, p)$ such that r is a researcher that (co-)authors paper p.
- Each tuple $(p, \mathsf{inJournal}, j)$ such that paper p appeared in journal j.
- All tuples (p, inConf, c) such that paper p was published in conference c.

Consider the C2RPQ $\gamma(x, y)$ defined as

$$\exists z \exists w \big((x, \mathsf{creator}, z) \wedge (z, \mathsf{inConf}, w) \wedge (z, \mathsf{creator}^-, y)\big).$$

Its evaluation $\gamma(\mathcal{G})$ over the graph database $\mathcal{G}$ consists of the set of pairs (r, r') such that researchers r and r' have a joint conference paper.

The evaluation over $\mathcal{G}$ of the C2RPQ $\gamma'(x, y)$ defined as

$$\exists z \exists w \big((x, (\mathsf{creator} \cdot \mathsf{creator}^-)^*, y) \wedge (y, \mathsf{creator}, z) \wedge (z, \mathsf{inJournal}, w)\big)$$

consists of the set of pairs (r, r') of researchers that are linked by a co-authorhsip sequence and such that r' has at least one journal paper. ∎

If Γ and Γ' are UC2RPQs, then Γ is *contained* in Γ', denoted by $\Gamma \subseteq \Gamma'$, if $\Gamma(\mathcal{G}) \subseteq \Gamma'(\mathcal{G})$ for each graph database $\mathcal{G}$. In addition, Γ and Γ' are equivalent, denoted by $\Gamma \equiv \Gamma'$, if $\Gamma \subseteq \Gamma'$ and $\Gamma' \subseteq \Gamma$, or, equivalently, $\Gamma(\mathcal{G}) = \Gamma'(\mathcal{G})$ for each graph database $\mathcal{G}$.

The *evaluation problem for UC2RPQs* is defined in the same way as for UCQs; i.e., this is the problem of checking whether $\bar{n}$ belong to $\Gamma(\mathcal{G})$, given a UC2RPQ $\Gamma(\bar{x})$, a graph database $\mathcal{G}$, and a tuple $\bar{n}$ of nodes in $\mathcal{G}$ such that $|\bar{n}| = |\bar{x}|$. It is folklore that evaluating UC2RPQs is not more expensive than evaluating CQs, i.e., NP-complete (see, e.g., [4, 3]).

Acyclic UC2RPQs. Acyclicity of C2RPQs has been studied in several recent papers that define it in terms of the acyclicity of its *underlying conjunctive query* [4, 5]. Let $\gamma(\bar{x}) = \exists \bar{y} \bigwedge_{1\leq i\leq m}(u_i, \mathcal{A}_i, v_i)$ be a C2RPQ. Its underlying CQ is the query over the schema of binary relation symbols $T_1, \ldots, T_m$ defined as $\exists \bar{y} \bigwedge_{1\leq i\leq m} T_i(u_i, v_i)$.

Intuitively, this underlying conjunctive query represents the structure of γ when the regular languages that label the atoms of γ are turned into relation symbols.

A C2RPQ is *acyclic* if its underlying CQ is acyclic. A UC2RPQ is acyclic if each one of its C2RPQs is acyclic. By combining techniques for UC2RPQ evaluation and polynomial time evaluation of acyclic CQs, it is possible to prove that the evaluation problem for acyclic UC2RPQs can be solved in polynomial time [44, 4]. We present the simple proof of this fact for the sake of completeness.

THEOREM 3.1. *The evaluation problem for acyclic UC2RPQs can be solved in time* $O(|\mathcal{G}|^2 \cdot |\Gamma|^2)$.

Proof. Let $\mathcal{G}$ be a graph database, and let Γ be a UC2RPQ. Let us denote by Θ the UCQ that is obtained from Γ by replacing each C2RPQ γ in Γ by its underlying CQ. Let $T_1(u_1, v_1), \ldots, T_n(u_n, v_n)$ be an enumeration of all the relational atoms in Θ, and assume that for each i with $1 \leq i \leq n$ the atom $T_i(u_i, v_i)$ is introduced in Θ as a replacement for the 2RPQ $(u_i, \mathcal{A}_i, v_i)$. Further, let $\mathcal{D}$ be a relational database over schema $\{T_1, \ldots, T_n\}$ such that the interpretation of the binary relation symbol T_i over $\mathcal{D}$, for each $1 \leq i \leq n$, is precisely $\mathcal{A}_i(\mathcal{G})$. Notice that $\mathcal{D}$ can be constructed in time $O(|\mathcal{G}|^2 \cdot |\Gamma|)$ from $\mathcal{G}$ and Γ, using the aforementioned fact that each $\mathcal{A}_i(\mathcal{G})$ can be computed in time $O(|\mathcal{G}|^2 \cdot |\mathcal{A}_i|)$ and $|\mathcal{A}_1| + \cdots + |\mathcal{A}_n|$ is $O(|\Gamma|)$.

It is clear then that checking whether a tuple $\bar{n}$ of nodes in $\mathcal{G}$ belongs to $\Gamma(\mathcal{G})$ reduces to checking whether $\bar{n} \in \Theta(\mathcal{D})$. Since Θ is acyclic, it follows from Proposition 2.1 that the latter can be done in time $O(|\mathcal{D}| \cdot |\Theta|)$, that is, in time $O(|\mathcal{G}|^2 \cdot |\Gamma|^2)$. ∎

Recall that a CQ θ is acyclic if its hypergraph $\mathcal{H}(\theta)$ admits a tree decomposition (T, λ) such that each set of the form $\lambda(t)$, for $t \in T$, is a hyperdge of $\mathcal{H}(\theta)$. The fact that acyclicity of C2RPQs is defined in terms of the acyclicity of its underlying CQ—and the latter is specified in a schema of binary arity—allows us to provide a simple characterization of the class of acyclic C2RPQs that will be useful in our proofs. We explain this below.

The *simple undirected underlying graph* of C2RPQ $\gamma(\bar{x}) = \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, \mathcal{A}_i, v_i)$, which is denoted by $\mathcal{U}(\gamma)$, is the graph whose vertices are the variables of γ and its set of edges is $\{\{u_i, v_i\} \mid 1 \leq i \leq m \text{ and } u_i \neq v_i\}$. Notice that $\mathcal{U}(\gamma)$ is indeed simple (it contains neither loops nor multiedges) and undirected. The following self-evident proposition provides a simple reformulation of the notion of acyclicity of C2RPQs in terms of the acyclicity of their simple undirected underlying graph.

PROPOSITION 3.2. *A C2RPQ* γ *is acyclic if and only if* $\mathcal{U}(\gamma)$ *is acyclic.*

Example 3. Let us consider again the C2RPQ:

$$\gamma(x, y) \;=\; \exists z \exists w \big((x, \mathsf{creator}, z) \wedge (z, \mathsf{inConf}, w) \wedge (z, \mathsf{creator}^-, y) \big)$$

in Example 2. The graph $\mathcal{U}(\gamma)$ consists of nodes x, z, w, y and edges $\{x, z\}$, $\{z, w\}$, and $\{z, y\}$. Clearly, $\mathcal{U}(\gamma)$ is acyclic, and, therefore, $\gamma(x, y)$ is acyclic. Similarly, it can be proved that the C2RPQ $\gamma'(x, y)$ in Example 2 is acyclic. ∎

Notice that this definition of acyclicity allows for the existence of loops and multiedges in the structure of a C2RPQ, that is, in its underlying CQ, as shown in the following example.

Example 4. Let L_1, L_2, and L_3 be arbitrary regular expressions over Σ. The CRPQs $\gamma = \exists x (x, L_1, x)$ and $\gamma' = \exists x \exists y \big((x, L_1, y) \wedge (y, L_2, x) \big)$ are acyclic. Notice that the underlying CQ of γ contains a loop, while the underlying CQ of γ' contains edges

from x to y and from y to x. On the other hand, the CRPQ $\gamma'' = \exists x \exists y \exists z \big((x, L_1, y) \wedge (y, L_2, z) \wedge (z, L_3, x)\big)$ is not acyclic. ∎

Our goal is to study the notion of semantic acyclicity for UC2RPQs. To attack the problem of evaluation for UC2RPQs that are semantically acyclic we make a necessary detour in the next section to study the containment problem for UC2RPQs, and then in section 5 to study approximations of UC2RPQs.

4. Containment of UC2RPQs. To prove our results, it is essential having a good understanding of the machinery used to study the containment problem for UC2RPQs. We develop such machinery in this section.

Calvanese et al. proved that the containment problem for C2RPQs is in EXPSPACE [9]. More specifically, it follows from [9] that checking whether $\gamma \subseteq \gamma'$, when γ and γ' are C2RPQs, can be solved using exponential space in such a way that the exponent only depends on $|\gamma'|$ and the number of variables in γ. For a UC2RPQ Γ, we define $\mathsf{maxvar}(\Gamma)$ to be the maximum number of variables over all disjuncts of Γ. A straightforward extension of the techniques in [9] shows that containment of Γ in Γ', when Γ and Γ' are UC2RPQs, can also be solved in exponential space, but now the exponent depends only on $|\Gamma'|$ and $\mathsf{maxvar}(\Gamma)$. It is also shown in [9] that the containment problem for C2RPQs is EXPSPACE-hard even when both γ and γ' are acyclic CRPQs [9]. In summary, we have the following proposition.

PROPOSITION 4.1. *The following hold:*

1. *Checking whether $\Gamma \subseteq \Gamma'$, for UC2RPQs Γ and Γ', can be solved using exponential space where the exponent only depends on $|\Gamma'|$ and $\mathsf{maxvar}(\Gamma)$.*
2. *The problem of checking whether $\gamma \subseteq \gamma'$, for C2RPQs γ and γ', is* EXPSPACE*-hard. It remains hard even if both γ and γ' are acyclic CRPQs.*

When Γ' is an acyclic UC2RPQ it is possible to show that the containment of Γ in Γ' can be checked in exponential space, but where the exponent only depends on Γ', specifically, on the *width* of Γ' (defined below). This result will be crucial to later prove in section 6 that the problem of checking whether a UC2RPQ is semantically acyclic (i.e., equivalent to an acyclic UC2RPQ) is in EXPSPACE.

Let γ be a C2RPQ over Σ. For two distinct variables x, y in γ we define $\mathtt{Atoms}(x, y)$ to be the set of atoms in γ that are of the form $(x, \mathcal{A}, y)$ or $(y, \mathcal{A}, x)$, where $\mathcal{A}$ is an NFA over $\Sigma^{\pm}$. The *width* $w(\gamma)$ of γ is defined as follows:

$$w(\gamma) \;=\; \max\{\, |\mathtt{Atoms}(x, y)| \;\mid\; x, y \text{ are distinct variables appearing in } \gamma\}.$$

If $\Gamma = \gamma_1 \vee \cdots \vee \gamma_m$ is a UC2RPQ, then the *width* of Γ is $w(\Gamma) = \max\{w(\gamma_i) \mid 1 \leq i \leq m\}$. We devote the rest of this section to proving the following theorem.

THEOREM 4.2. *Containment of a UC2RPQ Γ in an acyclic UC2RPQ Γ' can be solved in deterministic $O((|\Gamma| + |\Gamma'|)^{C \cdot w(\Gamma')})$ space, for some constant $C \geq 1$.*

The proof exploits automata techniques as in [9, 11, 38]. It follows from [38] that checking containment of two UC2RPQs can be reduced to checking containment of a Boolean *single-atom* C2RPQ (i.e., a C2RPQ of the form $\exists x \exists y (x, \mathcal{A}, y)$, for $\mathcal{A}$ an NFA) in a Boolean UC2RPQ (Boolean single-atoms C2RPQs are called *Boolean* 2RPQs in [38]). In particular, it is shown that there are integers $c, c' \geq 1$ and a polynomial time algorithm that, given UC2RPQs Γ and Γ', constructs a Boolean single-atom C2RPQ $\tilde{E}$ and a Boolean UC2RPQ $\tilde{\Gamma}$ such that the following hold:

1. $\Gamma \subseteq \Gamma'$ if and only if $\tilde{E} \subseteq \tilde{\Gamma}$,
2. $|\tilde{E}| = O(|\Gamma|^c)$ and $|\tilde{\Gamma}| = O((|\Gamma| + |\Gamma'|)^{c'})$,

3. $w(\Gamma') = w(\tilde{\Gamma})$, and
4. if Γ' is acyclic, then $\tilde{\Gamma}$ also is.

Our proof is based on the following lemma.

LEMMA 4.3. *There are integers $d, d' \geq 1$ such that the problem of checking containment of a Boolean single-atom C2RPQ Γ in an acyclic UC2RPQ Γ' can be solved in deterministic $O(|\Gamma'|^{d \cdot w(\Gamma')} \cdot |\Gamma|^{d'})$ space.*

Lemma 4.3 directly implies Theorem 4.2. Indeed, in order to check containment of a UC2RPQ Γ in an acyclic UC2RPQ Γ' we first construct the single-atom C2RPQ $\tilde{E}$ and the acyclic UC2RPQ $\tilde{\Gamma}$, and then apply the algorithm from Lemma 4.3. The space used is

$$O(|\tilde{\Gamma}|^{d \cdot w(\tilde{\Gamma})} |\tilde{E}|^{d'}) \;=\; O((|\Gamma| + |\Gamma'|)^{c' d \cdot w(\Gamma')} |\Gamma|^{cd'}),$$

which is $O((|\Gamma| + |\Gamma'|)^{Cw(\Gamma')})$, for $C \geq c'd + cd'$.

Before proving Lemma 4.3 we make a necessary detour through the notions of *canonical* graph database and *foldings*, which are important components of several containment algorithms for UC2RPQs and its extensions [23, 9, 38].

4.1. Canonical databases and foldings. A *semipath* is a graph database whose underlying graph is a path. Formally, the graph database $\mathcal{G} = (N, E)$ over Σ is a semipath if $N = \{n_0, \ldots, n_k\}$, $E = \{e_1, \ldots, e_k\}$, and for each $1 \leq i \leq k$ it is the case that e_i is of the form (n_{i-1}, a_i, n_i) or (n_i, a_i, n_{i-1}) (for $a_i \in \Sigma$). The *internal nodes* of $\mathcal{G}$ are $\{n_1, \ldots, n_{k-1}\}$. If $k = 0$, we call $\mathcal{G}$ the *empty semipath*.

Notice that the completion $\mathcal{G}^{\pm}$ of the semipath $\mathcal{G}$ contains a single path ρ from n_0 to n_k. The label of ρ is $b_1 b_2 \ldots b_k$, where for each $1 \leq i \leq k$ it is the case that $b_i = a_i$, if e_i is of the form (n_{i-1}, a_i, n_i), and $b_i = a_i^-$, otherwise. We slightly abuse notation and write $\mathsf{label}(\mathcal{G})$ in order to denote $\mathsf{label}(\rho)$ whenever $\mathcal{G}$ is a semipath. Notice that if $\mathcal{G}$ is an empty semipath, then $\mathsf{label}(\mathcal{G}) = \varepsilon$.

Containment of UC2RPQs can be recast in terms of the notion of *canonical* database [23, 9], which we describe below. Let $\gamma(\bar{x})$ be a C2RPQ of the form $\gamma(\bar{x}) = \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, \mathcal{A}_i, v_i)$. A graph database $\mathcal{G}$ is *canonical* for γ if there exists a mapping ν from the variables of γ to the nodes of $\mathcal{G}$ such that the following hold:

1. $\mathcal{G}$ consists of m semipaths, one for each conjunct of γ. Formally, for each $1 \leq i \leq m$ there is a semipath κ_i in $\mathcal{G}$ from $\nu(u_i)$ to $\nu(v_i)$. These semipaths are node and edge disjoint, save for the start and end nodes that can be shared between different κ_i's. For example, if $u_i = v_j$, for $1 \leq i, j \leq m$, then $\nu(u_i) = \nu(v_j)$ and, thus, the start node of κ_i coincides with the end node of κ_j.
2. For each $1 \leq i \leq m$ it is the case that $\mathsf{label}(\kappa_i)$ is accepted by $\mathcal{A}_i$.
3. For every two distinct variables z and z' in γ it is the case that $\nu(z) = \nu(z')$ if and only if there is a sequence $z_0, z_1, \ldots, z_\ell$ of variables in γ such that $z_0 = z$, $z_\ell = z'$ and for each $1 \leq j \leq \ell$ there is an $1 \leq i \leq m$ such that (i) the atom $(u_i, \mathcal{A}_i, v_i)$ in γ corresponds to either $(z_{j-1}, \mathcal{A}_i, z_j)$ or $(z_j, \mathcal{A}_i, z_{j-1})$, and (ii) κ_i is an empty semipath.

We say that mapping ν is *associated* with the canonical database $\mathcal{G}$.

The intuition is that a canonical database $\mathcal{G}$ is obtained from $\gamma(\bar{x})$ as follows. We view each variable in γ as a node in $\mathcal{G}$. For each atom $(u_i, \mathcal{A}_i, v_i)$ in γ, we add to $\mathcal{G}$ a semipath κ_i from u_i to v_i with fresh internal nodes that is accepted by $\mathcal{A}_i$. We then identify nodes u_i and v_i each time κ_i is chosen as an empty semipath. Condition 3. in the definition of canonical database ensures that the identification of nodes only occurs due to those empty paths.

A *canonical database* for a UC2RPQ $\Gamma(\bar{x})$ is a canonical database for some disjunct of $\Gamma(\bar{x})$. A slight extension of the techniques used in [9] yields the following proposition, which states that if $\Gamma \not\subseteq \Gamma'$, for UC2RPQs Γ and Γ', then a counterexample to $\Gamma \subseteq \Gamma'$ can be found in the set of canonical databases for Γ.

PROPOSITION 4.4. *Let $\Gamma(\bar{x})$ and $\Gamma'(\bar{x})$ be two UC2RPQs. Then $\Gamma \subseteq \Gamma'$ if and only if for each canonical database $\mathcal{G}$ for Γ with associated mapping ν, it is the case that $\nu(\bar{x}) \in \Gamma'(\mathcal{G})$.*[1]

We now define *foldings*. Let Σ be a finite alphabet. Recall that we denote by $\Sigma^{\pm}$ the alphabet $\Sigma \cup \{a^- \mid a \in \Sigma\}$. If $p \in \Sigma^{\pm}$ and $p = a$ for some $a \in \Sigma$, then p^- denotes a^-. On the other hand, if $p = a^-$ for $a \in \Sigma$, then p^- denotes a. Let $s = s_1 \dots s_k$ and $t = t_1 \dots t_\ell$ be words over $\Sigma^{\pm}$. We say that t *folds* into s [10] from j_1 to j_2, for $j_1, j_2 \in \{0, \dots, k\}$, if there is a sequence $i_0, \dots, i_\ell$ of positions in the set $\{0, \dots, k\}$ such that the following hold:

- $i_0 = j_1$ and $i_\ell = j_2$.
- For each $1 \leq j \leq \ell$, it is the case that $i_j = i_{j-1}+1$ and $t_j = s_{i_j}$, or $i_j = i_{j-1}-1$ and $t_j = s^-_{i_{j-1}}$.

Intuitively, t folds into s if t can be read in s by a *two-way automaton* that outputs symbol p, each time p is read from left-to-right, and symbol p^-, each time p is read from right-to-left. For instance, the word $abb^-a^-abb^-c$ folds into abb^-c from 0 to 5.

4.2. Proof of Lemma 4.3. We need to introduce some concepts from automata theory. Recall that *two-way alternating automata* generalize NFAs with the ability to move on the input word in both directions, and with the possibility to perform universal or existential moves (actually a combination of both). We define them following [11]. Given a set X, let $\mathcal{B}(X)$ be the set of positive Boolean formulae over X, built inductively by applying $\wedge$ and $\vee$ starting from `true`, `false`, and elements of X. For a set $Y \subseteq X$ and a formula $\varphi \in \mathcal{B}(X)$, we say that Y *satisfies* φ if and only if assigning `true` to the elements in Y and `false` to those in $X \setminus Y$ makes φ true. A two-way alternating finite automaton (2AFA) is a tuple $\mathcal{A} = (\Sigma, Q, \delta, Q_0, F)$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to \mathcal{B}(\{-1, 0, 1\} \times Q)$ is the transition function. Intuitively, a transition $\delta(q, a)$ spawns several copies of $\mathcal{A}$, each one starting in a certain state and with the head on the symbol to the left of a (-1), to the right of a (1), or on a itself (0), and specifies by means of a positive Boolean formula how to combine acceptance or nonacceptance of the spawned copies.

A *run* of $\mathcal{A}$ on a word $w = a_0 \dots a_{\ell-1}$, for $\ell \geq 0$, is a pair (T, λ), where T is a finite rooted tree and λ is a labeling of the nodes of T by elements in $Q \times \{0, \dots, \ell\}$ such that the following hold:

1. $\lambda(r) = (q_0, 0)$, for some $q_0 \in Q_0$, where r is the root of T.
2. For each node u in T with $\lambda(u) = (q, i)$ and $\delta(q, a_i) = \varphi$, where $q \in Q$, $0 \leq i \leq \ell - 1$, and $\varphi \in \mathcal{B}(\{-1, 0, 1\} \times Q)$, there is a (possible empty) set $X = \{(d_1, q_1), \dots, (d_n, q_n)\} \subseteq \{-1, 0, 1\} \times Q$ such that (i) X satisfies φ, and (ii) for all $j \in \{1, \dots, n\}$ there is a child of u in T labeled $(q_i, i + d_j)$.

The run is *accepting* if for each leaf u of T it is the case that $\lambda(u)$ is of the form (q, ℓ), for $q \in F$. The 2AFA $\mathcal{A}$ accepts w if it has an accepting run on it. It is known

[1] The original definition of canonical database in [9] only considers conditions 1. and 2. Condition 3. is innocuous in terms of the characterization of containment we provide here (i.e., Proposition 4.4 continues to hold if we remove such condition). As we shall see, on the other hand, condition 3. is important for our proof.

that 2AFAs define regular languages [34]. The following fact was shown in [11].

LEMMA 4.5. *Given a 2AFA $\mathcal{A}$ with n states, one can construct an NFA with $2^{O(n)}$ states that accepts precisely those words which are not accepted by $\mathcal{A}$.*

We now start with the proof of the lemma. Given a Boolean single-atom C2RPQ Γ and an acyclic UC2RPQ Γ' over Σ, we define an NFA $\mathcal{A}_{\Gamma,\Gamma'}$ (over an extended alphabet) such that $\Gamma \not\subseteq \Gamma'$ if and only if the language accepted by $\mathcal{A}_{\Gamma,\Gamma'}$ is not empty. Assume that Γ is of the form $\exists x \exists y(x, \mathcal{A}_\Gamma, y)$, where $\mathcal{A}_\Gamma$ is an NFA over $\Sigma^\pm$. The definition of $\mathcal{A}_{\Gamma,\Gamma'}$ is based on NFAs $\mathcal{A}_{cd}$ and $\mathcal{A}_{loops}$, and on a 2AFA $\mathcal{A}_{\Gamma'}$, all of which are introduced below.

All these automata are defined over a suitable alphabet Δ. Let $\texttt{Loops}$ be $2^{\mathcal{T}_{\Gamma'}}$, where $\mathcal{T}_{\Gamma'}$ is the set

$$\{(\mathcal{A}, q, q') \mid \mathcal{A} \text{ is an NFA mentioned in some atom of } \Gamma' \text{ and } q, q' \text{ are states of } \mathcal{A}\}.$$

Then $\Delta := \Sigma^\pm \cup \texttt{Loops}$. Note that the size of Δ is bounded by $O(2^{|\Gamma'|^2})$.

Let us start by defining the NFA $\mathcal{A}_{cd}$ as the product of the following two NFAs:

1. $\mathcal{A}_{\text{format}}$, which accepts all words over Δ satisfying the regular expression $\texttt{Loops} \cdot (\Sigma^\pm \cdot \texttt{Loops})^*$. (Here we are slightly abusing notation by writing $\texttt{Loops}$ and $\Sigma^\pm$ for the regular expression that represents the union of all symbols in $\texttt{Loops}$ and $\Sigma^\pm$, respectively).
2. $\mathcal{A}_{\text{acc}}$, which accepts all words $a_1 \dots a_m$ over Δ such that $a_2 a_4 \dots a_{2\lfloor m/2 \rfloor}$ is a word in $\Sigma^\pm$ accepted by $\mathcal{A}_\Gamma$.

It is easy to see that the number of states of $\mathcal{A}_{cd}$ can be bounded by $O(|\Gamma|)$.

Since Γ is a single-atom C2RPQ, it is the case that if G is a canonical database of Γ with associated mapping ν, then $\mathcal{G}$ is simply a semipath from $\nu(x)$ to $\nu(y)$ such that $\mathsf{label}(\mathcal{G})$ is accepted by $\mathcal{A}_\Gamma$. Thus we can naturally associate the canonical databases of Γ with the words accepted by $\mathcal{A}_\Gamma$. The NFA $\mathcal{A}_{cd}$ then accepts all the (labels of) canonical databases of Γ "adorned" with some additional information from $\texttt{Loops}$. More formally, $\mathcal{A}_{cd}$ accepts words of the form $L_0 a_1 L_1 \dots L_{m-1} a_m L_m$, where $L_i \in \texttt{Loops}$ and $a_i \in \Sigma^\pm$ for each $0 \leq i \leq m$, such that (i) $a_1 a_2 \dots a_m$ is the label of some canonical database of Γ, and (ii) $L_0 L_1 \dots L_m$ represents an "adornment" of $a_1 a_2 \dots a_m$ over $\texttt{Loops}$.

Let us now define the NFA $\mathcal{A}_{loops}$ over alphabet Δ. This NFA will help us restrict the possible adornments of the words $L_0 a_1 L_1 \dots L_{m-1} a_m L_m$ accepted by $\mathcal{A}_{cd}$ in such a way that the following holds for each $0 \leq i \leq m$:

(†) $(\mathcal{A}, q, q')$ belongs to L_i if and only if there is a word w over $\Sigma^\pm$ and a run of $\mathcal{A}$ over w from state q to q' such that w folds into $a_1 \dots a_m$ from i to i.

If the word $L_0 a_1 L_1 \dots L_{m-1} a_m L_m$ satisfies (†) for each $0 \leq i \leq m$, then we say that it is *correct*. The NFA $\mathcal{A}_{loops}$ then satisfies that the intersection of $\mathcal{A}_{cd}$ and $\mathcal{A}_{loops}$ accepts precisely those words accepted by $\mathcal{A}_{cd}$ which are correct. We explain next how $\mathcal{A}_{loops}$ is defined.

We start by defining a function $\xi : \texttt{Loops} \times \texttt{Loops} \to \texttt{Loops}$ as follows. Given $(L_1, L_2) \in \texttt{Loops} \times \texttt{Loops}$, the set $\xi(L_1, L_2)$ contains exactly all those triples $(\mathcal{A}, q, q') \in \mathcal{T}_{\Gamma'}$ such that either (i) $q = q'$, or (ii) there is a sequence

$$(\mathcal{A}, q_0, q_1), (\mathcal{A}, q_1, q_2), \dots, (\mathcal{A}, q_{k-1}, q_k)$$

for which (a) $q_0 = q$, $q_k = q'$ and the states $q_0, \dots, q_k$ are all distinct, and (b) $(\mathcal{A}, q_j, q_{j+1}) \in L_1 \cup L_2$, for each $0 \leq j \leq k-1$.

We also define two functions ξ^ℓ and ξ^r from $\texttt{Loops} \times \Sigma^\pm$ to $\texttt{Loops}$. Given $L \in \texttt{Loops}$ and $a \in \Sigma^\pm$, the set $\xi^\ell(L, a)$ contains exactly all those triples $(\mathcal{A}, q, q') \in \mathcal{T}_{\Gamma'}$

such that either (i) $q = q'$, or (ii) there is a sequence

$$(\mathcal{A}, q_0, q_1), (\mathcal{A}, q_2, q_3), \ldots, (\mathcal{A}, q_{2k}, q_{2k+1}),$$

where all the q_i's are distinct and the following hold:

1. There is an a^--labeled transition in $\mathcal{A}$ from q to q_0,
2. there is an a-labeled transition from q_{2k+1} to q' in $\mathcal{A}$, and
3. for each $0 \leq j \leq k$ the tuple $(\mathcal{A}, q_{2j}, q_{2j+1})$ belongs to L, and for each $0 \leq j \leq k-1$ there is a run of $\mathcal{A}$ over aa^- from q_{2j+1} to q_{2j+2}.

We define $\xi^r(L, a)$ analogously, but switching a for a^-, and vice versa.

Consider a word $a_1 \ldots a_m$, where $a_i \in \Sigma^{\pm}$ for each $1 \leq i \leq m$. For each $0 \leq i \leq m$ we define sets $T_i^\ell, T_i^r \in \texttt{Loops}$ as follows:

- The set T_i^ℓ contains precisely those triples $(\mathcal{A}, q, q')$ such that there is a word w over $\Sigma^{\pm}$ and a run of $\mathcal{A}$ over w from state q to q' such that w folds into $a_1 \ldots a_i$ from i to i. If $i = 0$, then T_i^ℓ is just the set of all triples of the form $(\mathcal{A}, q, q)$.
- The set T_i^r contains precisely those triples $(\mathcal{A}, q, q')$ such that there is a word w over $\Sigma^{\pm}$ and a run of $\mathcal{A}$ over w from state q to q' such that w folds into $a_{i+1} \ldots a_m$ from 0 to 0. If $i = m$, then T_i^r is just the set of all triples of the form $(\mathcal{A}, q, q)$.

As it turns out, the fact that L_i satisfies (†) can be stated in terms of T_i^ℓ and T_i^r, for each $0 \leq i \leq m$. Indeed, it is not hard to see that L_i satisfies (†) if and only if $L_i = \xi(T_i^\ell, T_i^r)$. Furthermore, consecutive T_i^ℓ's (resp., T_i^r's) can be expressed in terms of the function ξ^ℓ (resp., ξ^r). In fact, it is easy to see that for each $0 \leq i \leq m$ it is the case that $T_{i+1}^\ell = \xi^\ell(T_i^\ell, a_{i+1})$ (resp., $T_i^r = \xi^r(T_{i+1}^r, a_{i+1})$).

Now we construct the NFA $\mathcal{A}_{loops}$. The set of states is $\texttt{Loops} \times \texttt{Loops}$. The initial states are the pairs of the form (L_1, L_2), where L_1 is the set of triples in $\mathcal{T}_{\Gamma'}$ of the form $(\mathcal{A}, q, q)$. Similarly, the final states are the pairs of the form (L_1, L_2), where now L_2 is the set of all triples in $\mathcal{T}_{\Gamma'}$ of the form $(\mathcal{A}, q, q)$. The transition function δ is defined as follows:

1. Let L_1, L_2, L be elements in $\texttt{Loops}$. Then $(L_1, L_2) \in \delta((L_1, L_2), L)$ whenever $L = \xi(L_1, L_2)$.
2. Assume that L_1, L_2, L_1', L_2' are elements in $\texttt{Loops}$ and a is a symbol in $\Sigma^{\pm}$. Then $(L_1', L_2') \in \delta((L_1, L_2), a)$ whenever $L_1' = \xi^\ell(L_1, a)$ and $L_2 = \xi^r(L_2', a)$.

Notice that the size of $\mathcal{A}_{loops}$ is bounded by $2^{O(|\Gamma'|^2)}$.

Therefore, an accepting run of $\mathcal{A}_{loops}$ over $L_0 a_1 L_1 \ldots L_{m-1} a_m L_m$ is a sequence of the form

$$(L_1^0, L_2^0), (L_1^0, L_2^0), (L_1^1, L_2^1), (L_1^1, L_2^1), \ldots, (L_1^m, L_2^m), (L_1^m, L_2^m),$$

where the following hold by the definition of the transition function δ:

1. $(L_1^{i+1}, L_2^i) = (\xi^\ell(L_1^i, a_{i+1}), \xi^r(L_2^{i+1}, a_{i+1}))$, for each $0 \leq i \leq m-1$, and
2. $L_i = \xi(L_1^i, L_2^i)$, for each $0 \leq i \leq m$.

Further, (L_1^0, L_2^0) and (L_1^m, L_2^m) are an initial and final state of $\mathcal{A}_{loops}$, respectively.

We prove now that $\mathcal{A}_{loops}$ has an accepting run over $L_0 a_1 L_1 \ldots L_{m-1} a_m L_m$ if and only if this word is correct. From our previous remarks, it is sufficient to prove that $L_i = \xi(T_i^\ell, T_i^r)$ for each $0 \leq i \leq m$. But since we know that $L_i = \xi(L_1^i, L_2^i)$, we need only prove that $L_1^i = T_i^\ell$ and $L_2^i = T_i^r$ for each $0 \leq i \leq m$. We prove the former first by induction on $0 \leq i \leq m$. For the base case $i = 0$, notice that L_1^0 is the set of all triples in $\mathcal{T}_{\Gamma'}$ of the form $(\mathcal{A}, q, q)$ (because (L_1^0, L_2^0) is an initial state of $\mathcal{A}_{loops}$). But this set is precisely T_0^ℓ by definition. Consider now the inductive case

$i+1$, for $0 \leq i < m$. We know from our previous remarks that $T^{\ell}_{i+1} = \xi^{\ell}(T^{\ell}_i, a_{i+1})$. By inductive hypothesis, $T^{\ell}_i = L^i_1$, and, therefore, $T^{\ell}_{i+1} = \xi^{\ell}(L^i_1, a_{i+1})$. But the latter is precisely L^{i+1}_1. Proving that $L^i_2 = T^r_i$ for each $0 \leq i \leq m$ is completely analogous, but this time we do it by induction on i from m to 0 (more formally, we prove by induction on i that $L^{m-i}_2 = T^r_{m-i}$ for each $0 \leq i \leq m$).

Given a canonical database $\mathcal{G}$ of Γ, we denote by $w_{\mathcal{G}}$ the unique word $L_0 a_1 L_1 \dots a_m L_m$ over Δ such that $\mathsf{label}(\mathcal{G}) = a_1 \cdots a_m$ and the L_i's satisfy (†). In particular, $L_0 a_1 L_1 \dots a_m L_m$ is correct. We now define the 2AFA $\mathcal{A}_{\Gamma'}$. This 2AFA satisfies the following for each canonical database $\mathcal{G}$ of Γ: The word $w_{\mathcal{G}}$ is accepted by $\mathcal{A}_{\Gamma'}$ if and only if Γ' evaluates to `true` over $\mathcal{G}$. In order to do this, we define for each disjunct γ' of Γ' a 2AFA $\mathcal{A}_{\gamma'}$ such that the following holds for each canonical database $\mathcal{G}$ of Γ: The word $w_{\mathcal{G}}$ is accepted by $\mathcal{A}_{\gamma'}$ if and only if $\gamma'(\mathcal{G}) = \mathtt{true}$. Then $\mathcal{A}_{\Gamma'}$ is obtained by simply taking the union of all the $\mathcal{A}_{\gamma'}$'s.

Let γ' be a disjunct of Γ'. A *connected component* of γ' is a maximal subquery γ'' of γ' such that its underlying undirected graph $\mathcal{U}(\gamma'')$ is connected. Since γ' is acyclic, we can naturally interpret (the underlying undirected graph of) each connected component of γ' as a rooted tree. This allows us to talk about the *parent* or a *child* of a variable in γ', with the obvious meaning. Given a variable x in γ', we define $\mathtt{Loops}(x)$ to be the set of atoms of γ' of the form $(x, \mathcal{A}, x)$, for $\mathcal{A}$ an NFA over $\Sigma^{\pm}$. Recall that if x and y are variables in γ', then $\mathtt{Atoms}(x, y)$ denotes the set of atoms in γ' that mention both x and y. Without loss of generality, we assume that each atom in $\mathtt{Atoms}(x, y)$ is of the form $(x, \mathcal{A}, y)$ (otherwise, we simply "reverse" $\mathcal{A}$).

Let x and y be variables in γ' such that x is the parent of y in γ'. For the rest of the proof, we fix an enumeration

$$(x, \mathcal{A}_1, y), \dots, (x, \mathcal{A}_p, y)$$

of the elements in $\mathtt{Atoms}(x, y)$. We then define $\mathtt{Cuts}(x, y)$ to be the set of all tuples of the form $(q_1, \dots, q_p)$, where q_i is a state of $\mathcal{A}_i$ for each $1 \leq i \leq p$. A tuple $(q_1, \dots, q_p) \in \mathtt{Cuts}(x, y)$ is an *initial cut* (respectively, a *final cut*) if q_i is an initial state (respectively, a final state) of $\mathcal{A}_i$ for each $1 \leq i \leq p$. Given cuts $C = (q_1, \dots, q_p)$ and $C' = (q'_1, \dots, q'_p)$ in $\mathtt{Cuts}(x, y)$, and a symbol $a \in \Sigma^{\pm}$, we say that C' is *a-reachable* from C, if there is an a-transition in $\mathcal{A}_i$ from q_i to q'_i for each $1 \leq i \leq p$. For a symbol $L \in \mathtt{Loops}$ we say that C' is *L-reachable* from C, if $(\mathcal{A}_i, q_i, q'_i) \in L$ for each $1 \leq i \leq p$.

Intuitively, while reading word $w_{\mathcal{G}} = L_0 a_1 L_1 \dots a_m L_m$, for $\mathcal{G}$ a canonical database of Γ, the 2AFA $\mathcal{A}_{\gamma'}$ looks for a homomorphism from γ' to $\mathcal{G}$. The positions of the L_i's represent the nodes of $\mathcal{G}$ and the a_i's represent the edges. At any particular moment, $\mathcal{A}_{\gamma'}$ is trying to map a particular variable x in γ'. Once $\mathcal{A}_{\gamma'}$ maps x, a universal transition takes place to ensure that *all* the children of x can be mapped too. Suppose x is mapped to the node i, for $0 \leq i \leq m$. Then, while reading the symbol L_i, each copy of the 2AFA $\mathcal{A}_{\gamma'}$ guesses a mapping position $0 \leq j \leq m$ for a particular child y of x. This copy then moves from position i to j while it verifies that all the atoms in $\mathtt{Atoms}(x, y)$ can be correctly mapped to $w_{\mathcal{G}}$.

Consider an atom of the form $(x, \mathcal{A}, y)$ in $\mathtt{Atoms}(x, y)$. The 2AFA $\mathcal{A}_{\gamma'}$ then has to verify that there is a a word w over $\Sigma^{\pm}$ that can be read in $\mathcal{A}$ from an initial state q_I to a final state q_F, such that w folds into $a_1 \dots a_m$ from i to j. Let us assume without loss of generality that $i < j$ (any other case is analogous). Observe that the word w can always be decomposed in the form

$$w \;=\; w_i a_{i+1} w_{i+1} \dots a_j w_j,$$

where w_h is a word over $\Sigma^{\pm}$ that can be folded into $a_1 \dots a_m$ from h to h for each $i \leq h \leq j$. Then $\mathcal{A}_{\gamma'}$ guesses the run on $\mathcal{A}$ over w from q_I to q_F, while it reads $L_i a_{i+1} L_{i+1} \dots a_j L_j$ from left-to-right. If $\mathcal{A}_{\gamma'}$ reads a_h, with $h \in \{i+1, \dots, j\}$, and the current guessed state is q, then it guesses q' such that there is an a_h-transition in $\mathcal{A}$ from q to q'. If $\mathcal{A}_{\gamma'}$ reads L_h, with $h \in \{i, \dots, j\}$, and the current guessed state is q, then it guesses q' such that $(\mathcal{A}, q, q') \in L_h$. However, there is a slight complication as the automaton has to do this simultaneously for all atoms in $\texttt{Atoms}(x, y)$. Thus $\mathcal{A}_{\gamma'}$ actually guesses a sequence of cuts $C_0, \dots, C_{2(j-i)+1}$ from $\texttt{Cuts}(x, y)$, where C_0 is an initial cut, $C_{2(j-i)+1}$ is a final cut, and C_h is α_h-reachable from C_{h-1}, for each $h \in \{1, \dots, 2(j-1)+1\}$, where α_h is the hth symbol in $L_i a_{i+1} L_{i+1} \dots a_j L_j$. The 2AFA $\mathcal{A}_{\gamma'}$ is formally defined below.

Let us assume that γ' has K connected components with roots $r_1, \dots, r_K$, respectively. Then the set of states of $\mathcal{A}_{\gamma'}$ is

$$Q \;=\; \texttt{Cuts} \times \{-1, 1\} \cup \{r_1, \dots, r_K, \mathit{start}, \mathit{accept}\},$$

where $\texttt{Cuts}$ is the set $\bigcup\{\texttt{Cuts}(x, y) \mid x \text{ is the parent of } y \text{ in } \gamma'\}$. The initial state is *start* and the final state is *accept*. Next we define the transition function δ' of $\mathcal{A}_{\gamma'}$. For readability, whenever $\delta'(q, a) = \varphi_1 \vee \dots \vee \varphi_n$, we write $\varphi_i \in \delta'(q, a)$ for each $i \in \{1, \dots, n\}$. We also assume that whenever $\mathcal{A}_{\gamma'}$ reaches the final state *accept*, it stays in the same state and moves its head all the way to the right. The function δ' is defined as follows:

1. There is an initial transition $(0, r_1) \wedge \dots \wedge (0, r_K) \in \delta'(\mathit{start}, L)$, for each $L \in \texttt{Loops}$.
2. There is a transition $(1, r_i) \in \delta'(r_i, \alpha)$, for each $i \in \{1, \dots, K\}$ and $\alpha \in \Sigma^{\pm} \cup \texttt{Loops}$. These transitions look for the position where the root r_i is mapped.
3. For each $i \in \{1, \dots, K\}$ and $L \in \texttt{Loops}$, we have a "root mapping" transition:

   $$[(0, (I_1, -1)) \vee (0, (I_1, 1))] \wedge \dots \wedge [(0, (I_t, -1)) \vee (0, (I_t, 1))] \in \delta'(r_i, L),$$

   where (a) $u_1, \dots, u_t$ are the children of r_i in γ', (b) for each $1 \leq j \leq t$ it is the case that I_j is an initial cut from $\texttt{Cuts}(r_i, u_j)$, and (c) $\texttt{Loops}(r_i)$ is *compatible* with L, i.e., for each atom $(r_i, \mathcal{A}, r_i) \in \texttt{Loops}(r_i)$ there are initial and final states q_I and q_F of $\mathcal{A}$ such that $(\mathcal{A}, q_I, q_F) \in L$.
   Intuitively, a "root mapping" transition maps r_i to the current position h, and for each child u_j it guesses an initial cut and the fact whether u_j will be mapped to the right of h (indicated by the symbol 1) or to the left of h (indicated by the symbol -1).
4. For each $C \in \texttt{Cuts}(x, y)$, with x the parent of y, each $D \in \{-1, 1\}$, and each $a \in \Sigma^{\pm}$, we have an "atom mapping" transition:

   $$(D, (C', D)) \in \delta'((C, D), a),$$

   where $C' \in \texttt{Cuts}(x, y)$ is a-reachable from C if $D = 1$ or a^--reachable from C if $D = -1$.
5. For each $C \in \texttt{Cuts}(x, y)$, with x the parent of y, each $D \in \{-1, 1\}$, and each $L \in \texttt{Loops}$, we have a "loop guessing" transition:

   $$(D, (C', D)) \in \delta'((C, D), L),$$

   where $C' \in \texttt{Cuts}(x, y)$ is L-reachable from C.

We also have a "variable mapping" transition:

$$[(0,(I_1,-1)) \vee (0,(I_1,1))] \wedge \cdots \wedge [(0,(I_t,-1)) \vee (0,(I_t,1))] \in \delta'((C,D),L),$$

where (a) $u_1,\ldots,u_t$ are the children of y in γ', (b) for each $1 \leq j \leq t$ we have that I_j is an initial cut from $\texttt{Cuts}(y,u_j)$, (c) $\texttt{Loops}(y)$ is compatible with L, and (d) there is a final cut $C' \in \texttt{Cuts}(x,y)$ which is L-reachable from C.

Finally, we have a "leaf mapping" transition:

$$(0,accept) \in \delta'((C,D),L),$$

whenever (a) y is a leaf in γ', (b) $\texttt{Loops}(y)$ is compatible with L, and (c) there is a final cut $C' \in \texttt{Cuts}(x,y)$ which is L-reachable from C.

It is not hard to see that $\mathcal{A}_{\gamma'}$ is sound and complete, i.e., that for each canonical database $\mathcal{G}$ of Γ the 2AFA $\mathcal{A}_{\gamma'}$ accepts $w_{\mathcal{G}}$ if and only if $\gamma'(\mathcal{G}) = \texttt{true}$.

Recall that $\mathcal{A}_{\Gamma'}$ is the union of the $\mathcal{A}_{\gamma'}$s, for γ' a disjunct of Γ'. We now analyze the size of $\mathcal{A}_{\Gamma'}$. Let γ' be a disjunct of Γ'. Recall that the width $w(\gamma')$ of γ' is the maximum value of $|\texttt{Atoms}(x,y)|$, for x and y distinct variables in γ'. The number of states of $\mathcal{A}_{\gamma'}$ is $2|\texttt{Cuts}|+K+2$ (where K is the number of connected components of γ'). Recall that $\texttt{Cuts}$ is the set $\bigcup\{\texttt{Cuts}(x,y) \mid x \text{ is the parent of } y \text{ in } \gamma'\}$. Notice for each distinct variable x and y in γ' it is the case that $|\texttt{Cuts}(x,y)|$ is bounded by $O(|\gamma'|^{w(\gamma')})$, and, therefore, $|\texttt{Cuts}|$ is bounded by $O(|\gamma'|\cdot|\gamma'|^{w(\gamma')}) = O(|\gamma'|^{w(\gamma')+1})$. Further, clearly K is bounded by $|\gamma'|$. Thus, the number of states of $\mathcal{A}_{\gamma'}$ is $O(|\gamma'|^{w(\gamma')+1})$. Since the width $w(\Gamma')$ of Γ' is defined as the maximum value of $w(\gamma')$, for γ' a disjunct of Γ', it follows that the number of states of $\mathcal{A}_{\Gamma'}$ is at most $O(|\Gamma'|\cdot|\Gamma'|^{w(\Gamma')+1}) = O(|\Gamma'|^{w(\Gamma')+2})$.

Now we are ready to define the NFA $\mathcal{A}_{\Gamma,\Gamma'}$. This NFA is the product of $\mathcal{A}_{cd}$, $\mathcal{A}_{loops}$, and $\overline{\mathcal{A}_{\Gamma'}}$, where $\overline{\mathcal{A}_{\Gamma'}}$ is the NFA from Lemma 4.5 that accepts the complement of the language accepted by $\mathcal{A}_{\Gamma'}$. From our previous remarks, $\Gamma \not\subseteq \Gamma'$ if and only if the language accepted by $\mathcal{A}_{\Gamma,\Gamma'}$ is not empty. Furthermore, notice that the number of states of $\mathcal{A}_{\Gamma,\Gamma'}$ is bounded by $2^{O(\log|\Gamma|+|\Gamma'|^{w(\Gamma')+2})}$.

It is well known that nonemptiness of an NFA with n states can be checked in nondetermistic $O(\log n)$ space. Following this approach, our algorithm simply checks that the language accepted by $\mathcal{A}_{\Gamma,\Gamma'}$ is nonemtpy using nondeterministic $O(\log|\Gamma| + |\Gamma'|^{w(\Gamma')+2})$ space. Of course we cannot construct explicitly $\mathcal{A}_{\Gamma,\Gamma'}$, as it might be of double exponential size. Instead, we use a standard "on the fly" implementation: We generate the states of $\mathcal{A}_{\Gamma,\Gamma'}$ on the fly and whenever is needed, we check whether there is a transition from one state to another. It is straightforward to check that, given two states, we can decide if there is a transition in $\mathcal{A}_{\Gamma,\Gamma'}$ from one state to the other using only polynomial space in $|\Gamma|$ and $|\Gamma'|$. Thus the on the fly implementation actually uses only $O(\log|\Gamma| + |\Gamma'|^{w(\Gamma')+2})$ space. By Savitch's theorem, we conclude that containment of Γ in Γ' can be solved in deterministic $O((\log|\Gamma|+|\Gamma'|^{w(\Gamma')+2})^2) = O(|\Gamma'|^{2w(\Gamma')+4}\cdot\log^2|\Gamma|)$ space, which is $O(|\Gamma'|^{d\cdot w(\Gamma')}\cdot|\Gamma|^{d'})$ space, for suitable constants $d,d' \geq 1$. This concludes the proof of the lemma. ∎

5. Approximations of UC2RPQs. Acyclic UC2RPQs form a good class in terms of complexity of evaluation: They are tractable as opposed to arbitrary C2RPQs (and even CQs) for which the evaluation problem is NP-complete and even hard in parameterized complexity [37]. This motivates our study of approximations of UC2RPQs in the class of acyclic UC2RPQs, which is inspired by recent research on approximations of UCQs. We explain this below.

Evaluating an arbitrary CQ on a big database might be prohibitively expensive. This has led to the recent study of (U)CQ *approximations* in tractable classes [6],

in particular, in the class of acyclic (U)CQs. Intuitively, an acyclic UCQ Θ' is an approximation of a UCQ Θ if the following holds: (1) Θ' is contained in Θ (i.e., Θ' returns no false positives with respect to Θ) and (2) Θ' is "as close as possible" to Θ among all acyclic UCQs.

It follows from results and techniques in [6] that approximations of UCQs have good properties.

1. First of all, they always exist, that is, each UCQ has at least one acyclic approximation, and, in addition, such an approximation is unique (up to equivalence) and of at most exponential size.
2. Second, for each UCQ Θ, its acyclic approximation Θ' can be computed in single-exponential time.
3. Third, verifying whether an acyclic UCQ Θ' is an approximation of a UCQ Θ is decidable in the second-level of the polynomial hierarchy.

These good properties imply that computing and running the acyclic approximation of a UCQ Θ on a database $\mathcal{D}$ takes time $O(2^{p(|\theta|)} + |\mathcal{D}| \cdot 2^{r(|\Theta|)})$, for polynomials $p, r : \mathbb{N} \to \mathbb{N}$, which is $O(|\mathcal{D}| \cdot 2^{s(|\Theta|)})$, for a polynomial $s : \mathbb{N} \to \mathbb{N}$. On large databases, this is much better than the general $|\mathcal{D}|^{O(|\Theta|)}$ cost of evaluating Θ over $\mathcal{D}$. Thus, if evaluation of Θ is infeasible or too slow and the quality of its acyclic approximation is good, we may prefer to run this faster approximation instead of Θ.

Here we study acyclic approximations for UC2RPQs, and show that several of the good properties mentioned above for acyclic approximations of UCQs extend to the class of UC2RPQs.

5.1. Approximations: Existence and computation. Suppose we want to approximate a UC2RPQ Γ in the class AC of acyclic UC2RPQs. As explained earlier, we are interested in approximations that are guaranteed to return correct results only. Thus, we are looking for an acyclic UC2RPQ that is *maximally contained* in Γ.

DEFINITION 5.1 (approximations). *Let Γ and Γ' be UC2RPQs such that $\Gamma' \in \mathsf{AC}$ and $\Gamma' \subseteq \Gamma$. Then Γ' is an* approximation *of Γ if for every query $\Gamma'' \in \mathsf{AC}$ with $\Gamma'' \subseteq \Gamma$ we have that $\Gamma'' \subseteq \Gamma'$.*

It is worth noticing that the definition of approximations in [6] is different, but equivalent to this one.

An important property of UCQs is that each query in the class has an acyclic approximation, and that such an approximation is unique. We can prove that this is also true for the class of UC2RPQs.

THEOREM 5.2. *Each UC2RPQ has a unique acyclic approximation (up to equivalence).*

As a corollary to the proof of Theorem 5.2 we get the following important result about the computation and size of approximations.

COROLLARY 5.3. *There exists an* EXPSPACE *algorithm that takes as input a UC2RPQ Γ and computes the approximation Γ' of Γ. This approximation is of at most exponential size.*

It follows from Corollary 5.3 that approximations of UC2RPQs are meaningful. In fact, computing and running the acyclic approximation of a UC2RPQ Γ on a graph database $\mathcal{G}$ takes time

$$O\left(2^{2^{p(|\Gamma|)}} + |\mathcal{G}|^2 \cdot 2^{r(|\Gamma|)}\right),$$

for polynomials $p, r : \mathbb{N} \to \mathbb{N}$, which is $O\big(|\mathcal{G}|^2 \cdot 2^{2^{p(|\Gamma|)}}\big)$. In terms of data complexity

this is only $O(|\mathcal{G}|^2)$, such like the data complexity of 2RPQs. This is much faster than $|\mathcal{G}|^{O(|\Gamma|)}$—the order of the evaluation problem for Γ on $\mathcal{G}$—on large datasets.

We finish by proving that there is an important aspect of approximations that is harder for UC2RPQs than for UCQs: the identification problem, i.e., verifying whether a query is an approximation of another. We mentioned above that checking whether an acyclic UCQ Θ' is an approximation of a UCQ Θ can be solved in the second-level of the polynomial hierarchy [6]; more precisely, it is complete for the class DP, that consists of all those languages that are the intersection of an NP and a coNP problem [36]. This problem is considerably harder for UC2RPQs.

PROPOSITION 5.4. *Let Γ and Γ' be UC2RPQs such that $\Gamma' \in \mathsf{AC}$. The problem of verifying whether Γ' is an acyclic approximation of Γ is* EXPSPACE*-complete.*

We prove Theorem 5.2, Corollary 5.3, and Proposition 5.4 in the following section.

5.2. Proofs of results. All results in section 5.1 follow from an important lemma that states that there exists an EXPSPACE algorithm that, on input a UC2RPQ Γ, computes an acyclic UC2RPQ Γ_{app}—of at most exponential size—which is a maximum for the class of acyclic UC2RPQs that are contained in Γ. This lemma will also be crucial for proving decidability of the notion of semantic acyclicity for UC2RPQs in section 6.

LEMMA 5.5. *There exists an* EXPSPACE *algorithm that given a UC2RPQ Γ computes an acyclic UC2RPQ Γ_{app} such that the following hold:*

1. $\Gamma_{\text{app}} \subseteq \Gamma$.
2. *For every acyclic UC2RPQ Γ' such that $\Gamma' \subseteq \Gamma$ it is the case that $\Gamma' \subseteq \Gamma_{\text{app}}$.*
3. *The number of atoms and variables in each disjunct of Γ_{app} is at most polynomial in $|\Gamma|$.*
4. *The number of disjuncts and the size of each NFA appearing in Γ_{app} is at most exponential in $|\Gamma|$.*

In particular, the size of Γ_{app} is at most exponential in $|\Gamma|$.

Before proving Lemma 5.5 we show how the results in section 5.1 follow from it.

Proofs of Theorem 5.2*, Corollary* 5.3*, and Proposition* 5.4. The algorithm in Lemma 5.5 computes for each UC2RPQ Γ a query Γ_{app} which is the maximum for the class of acyclic UC2RPQs that are contained in Γ. In other words, Γ_{app} is an approximation of Γ. This approximation must be unique (up to equivalence) by definition.

In order to prove Corollary 5.3, we use the algorithm in Lemma 5.5 to compute the approximation Γ_{app} of a UC2RPQ Γ. The algorithm runs in EXPSPACE, and its output Γ_{app} is of at most exponential size in $|\Gamma|$.

Finally, we prove Proposition 5.4. Checking whether the acyclic UC2RPQ Γ' is an approximation of the UC2RPQ Γ is equivalent to checking whether (†) $\Gamma' \subseteq \Gamma$ and (††) $\Gamma_{\text{app}} \subseteq \Gamma'$. Indeed, if this is the case, then $\Gamma' \equiv \Gamma_{\text{app}}$, and, therefore, Γ' is the approximation of Γ. We prove that (†) and (††) can be checked in EXPSPACE.

It directly follows from the first part of Proposition 4.1 that (†) can be verified in EXPSPACE. Furthermore, checking (††) requires computing Γ_{app} and then checking whether $\Gamma_{\text{app}} \subseteq \Gamma'$. The first step can be done in EXPSPACE from Lemma 5.5, while the second one can be carried out in exponential space, where the exponent depends only on $|\Gamma'|$ and $\mathsf{maxvar}(\Gamma_{\text{app}})$ (again from the first part of Proposition 4.1). Part 3. of Lemma 5.5 implies that $\mathsf{maxvar}(\Gamma_{\text{app}})$ is at most polynomial in the size of $|\Gamma|$. Therefore, the second step can also be done in EXPSPACE. We conclude that checking

whether Γ' is an approximation of Γ can be carried out in EXPSPACE.

For the lower bound, observe that $\Gamma' \subseteq \Gamma$ if and only if Γ' is an acyclic approximation of $\Gamma \wedge \Gamma'$. The result now follows since Proposition 4.1 states that containment of CRPQs is EXPSPACE-hard even when Γ' is acyclic. ∎

We devote the rest of this section to proving Lemma 5.5. We start by presenting some definitions and a technical lemma which is crucial for our proof. We conclude by explaining the construction of the acyclic approximation Γ_{app} for a UC2RPQ Γ.

5.2.1. A technical lemma. The lemma requires some terminology which we define next.

Pseudoacyclic graph databases. A graph database $\mathcal{G}$ is *connected* if for each pair of nodes n, n' in $\mathcal{G}$ there is a path from n to n' in $\mathcal{G}^{\pm}$ (that is, if the underlying undirected graph of $\mathcal{G}$ is connected). A *subgraph database* of $\mathcal{G} = (N, E)$ is a graph database $\mathcal{G}' = (N', E')$ such that $N' \subseteq N$ and $E' \subseteq E$. A *connected component* of $\mathcal{G}$ is a maximal connected subgraph database of $\mathcal{G}$. We call $\mathcal{G}$ *pseudoacyclic* if each connected component of $\mathcal{G}$ can be obtained from a tree T as follows:

- Each edge $\{n, n'\}$ in T is replaced by a finite number of semipaths from n to n' whose internal nodes are "fresh"; i.e., they are disjoint from any other nodes in $\mathcal{G}$.
- For some (maybe none) nodes n in T we add a finite number of semipaths from n to n, again with fresh internal nodes.

The fresh internal nodes of the paths in the previous definition are the *internal* nodes of the pseudoacyclic database $\mathcal{G}$. The rest of the nodes (that is, the nodes that are inherited from T) are called *external* nodes.

We will slightly abuse notation and talk about the *parent* (respectively, *ancestor*) of an external node n in the pseudoacyclic graph database $\mathcal{G}$. By this we refer to the external node in $\mathcal{G}$ that corresponds to the parent (respectively, an ancestor) of n in the tree T. Similarly, we talk about the *least common ancestor* of two external nodes in $\mathcal{G}$, which refers to the external node in $\mathcal{G}$ that corresponds to their least common ancestor in T.

The following proposition is straightforward.

PROPOSITION 5.6. *Let Γ be an acyclic UC2RPQ. Then each canonical database of Γ is pseudoacyclic.*

We also define pseudoacyclicity for graphs in the obvious way: instead of adding semipaths to T we add undirected paths. For pseudoacyclic graphs we define the concepts of external/internal node and parent of a node as before.

Types for UC2RPQs. Let Γ be UC2RPQ over Σ. Recall that $\mathcal{T}_\Gamma$ denotes the set of all triples of the form $(\mathcal{A}, q, q')$, where $\mathcal{A}$ is an NFA mentioned in some atom of Γ and q and q' are states of $\mathcal{A}$. A Γ-*type* is a 4-tuple $\pi = (\tau_1, \tau_2, \tau_3, \tau_4)$, where τ_i is a subset of $\mathcal{T}_\Gamma$, for each $1 \leq i \leq 4$. Let w be a word over $\Sigma^{\pm}$ of length $\ell \geq 0$. The Γ-*type of* w is the Γ-type $\pi_w = (\tau_{it}, \tau_{ti}, \tau_{ii}, \tau_{tt})$ such that $(\mathcal{A}, q, q')$ belongs to τ^* if and only if there is a word u over $\Sigma^{\pm}$ and a run of $\mathcal{A}$ over u from state q to q', and u can be folded into w from (i) 0 to ℓ, if $* = it$, (ii) from ℓ to 0, when $* = ti$, (iii) from 0 to 0, if $* = ii$, and (iv) from ℓ to ℓ, when $* = tt$.

Given a Γ-type $\pi = (\tau_1, \tau_2, \tau_3, \tau_4)$, we define $\mathcal{L}^{\subseteq}(\pi)$ to be the language of words w over $\Sigma^{\pm}$ such that π is coordinatewise contained in the Γ-type π_w of w. The proof of the following lemma is given in the appendix.

LEMMA 5.7. *Let Γ be a UC2RPQ and π a Γ-type. Then the language $\mathcal{L}^{\subseteq}(\pi)$ is regular and can be defined by an NFA $\mathcal{A}_\pi$ over $\Sigma^{\pm}$ of at most exponential size in $|\Gamma|$.*

The technical lemma. We are now ready to state the main lemma of this section.

LEMMA 5.8. *Let $\mathcal{G}$ be a pseudoacyclic graph database, and let $\bar{n}$ be a tuple of external nodes in $\mathcal{G}$. Let $\Gamma(\bar{x})$ be a UC2RPQ with $|\bar{n}| = |\bar{x}|$ such that $\bar{n} \in \Gamma(\mathcal{G})$. There exists an acyclic C2RPQ $\alpha(\bar{x})$ and a finite set $\mathcal{R}_\Gamma$ of NFAs over $\Sigma^{\pm}$ such that the following hold:*

1. *$\mathcal{R}_\Gamma$ can be constructed from Γ in single exponential time.*
2. *$\bar{n} \in \alpha(\mathcal{G})$ and $\alpha \subseteq \Gamma$.*
3. *The number of atoms in α is at most polynomial in $|\Gamma|$.*
4. *Each NFA mentioned in α belongs to $\mathcal{R}_\Gamma$.*

Proof. Since $\bar{n} \in \Gamma(\mathcal{G})$, there is a disjunct γ of Γ such that $\bar{n} \in \gamma(\mathcal{G})$. Assume that $\gamma(\bar{x}) = \exists \bar{y} \bigwedge_{i=1}^{m} (u_i, \mathcal{A}_i, v_i)$. Since $\bar{n} \in \gamma(\mathcal{G})$, there is a mapping h from the variables of γ to the nodes of $\mathcal{G}$ that satisfies $h(\bar{x}) = \bar{n}$ and a path ρ_i from $h(u_i)$ to $h(v_i)$ in $\mathcal{G}^{\pm}$ such that $\mathsf{label}(\rho_i)$ is accepted by $\mathcal{A}_i$, for each $1 \leq i \leq m$. We assume without loss of generality that for each $1 \leq i \leq m$ it is the case that ρ_i is of minimal length among the set of paths from $h(u_i)$ to $h(v_i)$ in $\mathcal{G}^{\pm}$ whose label is accepted by $\mathcal{A}_i$.

We define a subset $\mathcal{V}$ of the nodes of $\mathcal{G}$ that will help us define the variable set of the C2RPQ $\alpha(\bar{x})$. Let $\mathcal{I}$ be the set $\{h(z) \mid z \text{ is a variable in } \gamma\}$. We define $\mathcal{I}'$ to be the set of nodes obtained from $\mathcal{I}$ by adding, for each node u in $\mathcal{I}$ that is an internal node of $\mathcal{G}$, its associated external nodes in $\mathcal{G}$ (that is, the external nodes that are at the beginning and at the end of the semipath where u belongs). We then define $\mathcal{V}$ as the set of nodes which is obtained from $\mathcal{I}'$ by adding the least common ancestor in $\mathcal{G}$ of each pair n, n' of external nodes from $\mathcal{I}'$ in the same connected component of $\mathcal{G}$. Observe that each node in $\bar{n}$ belongs to $\mathcal{I}$ (since $h(\bar{x}) = \bar{n}$), and thus to $\mathcal{V}$. Also notice that $|\mathcal{V}|$ is $O(|\gamma|^2)$. Indeed, $|\mathcal{I}| \leq var$, where var is the number of variables in γ. Moreover, we have that $|\mathcal{I}'| \leq 3|\mathcal{I}|$ and $|\mathcal{V}| \leq |\mathcal{I}'| + |\mathcal{I}'|^2$, as the number of least common ancestors we add to $\mathcal{V}$ is bounded by the number of pairs in $|\mathcal{I}'|$.

For each $1 \leq i \leq m$, we decompose the path ρ_i according to $\mathcal{V}$ in the natural way. That is, the path ρ_i is decomposed as the concatenation $\rho_i^1 \cdots \rho_i^{k_i}$ of paths $\rho_i^1, \ldots, \rho_i^{k_i}$ such that each path ρ_i^j starts and ends at a node in $\mathcal{V}$ and each internal node of ρ_i^j is not in $\mathcal{V}$. As ρ_i itself starts and ends at $h(u_i)$ and $h(v_i)$, respectively, which are nodes in $\mathcal{V}$, this decomposition is always possible. The decomposition is also unique (assuming that no ρ_i^j is the empty path, except in the case when ρ_i itself is empty).

Let $\rho_i^1 \cdots \rho_i^{k_i}$ be the decomposition of the path ρ_i, for $1 \leq i \leq m$. By the minimality of ρ_i, it follows that $k_i \leq |Q_i| \cdot |\mathcal{V}| - 1$, where Q_i is the set of states of the NFA $\mathcal{A}_i$. Assume for the sake of contradiction that $k_i \geq |Q_i| \cdot |\mathcal{V}|$. We fix an accepting run $q_0 \cdots q_\ell$ of $\mathcal{A}_i$ over $\mathsf{label}(\rho_i)$ (i.e., each q_j is a state in Q_i, q_0 and q_ℓ are an initial and final state of $\mathcal{A}_i$, respectively, and q_{j+1} is obtained from q_j according to the transition function of $\mathcal{A}_i$). For each $1 \leq j \leq k_i$, let n^j be the last node of ρ_i^j (which belongs to $\mathcal{V}$), and let q^j be the state of the run $q_0 \cdots q_\ell$ which is associated with the prefix $\mathsf{label}(\rho_i^1 \ldots \rho_i^j)$ of $\mathsf{label}(\rho_i^1 \ldots \rho_i^{k_i})$. In particular, $n^{k_i} = h(v_i)$ and $q^{k_i} = q_\ell$. We also define $n^0 = h(u_i)$ and $q^0 = q_0$. Since $k_i + 1 > |\mathcal{V}| \cdot |Q_i|$, there are two positions $0 \leq \ell_1 < \ell_2 \leq k_i$ such that $n^{\ell_1} = n^{\ell_2}$ and $q^{\ell_1} = q^{\ell_2}$. It is clear then that we can ignore the subpath $\rho_i^{\ell_1+1} \cdots \rho_i^{\ell_2}$. This contradicts the minimality of ρ_i. We conclude that the number of paths involved in all the decompositions of the ρ_i's is at most $(|Q_1| + \cdots + |Q_m|) \cdot |\mathcal{V}|$. Clearly, $|Q_1| + \cdots + |Q_m|$ is polynomially bounded by $|\gamma|$.

Further, we know that $|\mathcal{V}|$ is $O(|\gamma|^2)$ and $|\gamma| \leq |\Gamma|$. We then conclude that the number of paths involved in all the decompositions of the ρ_i's is polynomial in $|\Gamma|$.

Let ρ_i^j be a path in the decomposition of ρ_i, for $1 \leq i \leq m$ and $1 \leq j \leq k_i$. We denote by $\ell(i,j)$ the length of the word $\mathsf{label}(\rho_i^j)$, and define (i) $c(j) := \sum_{t=1}^{j-1} \ell(i,t)$, and (ii) $d(j) := \sum_{t=1}^{j} \ell(i,t)$ (we set $c(1)$ to be 0). Let $q_0 \cdots q_\ell$ be an accepting run of $\mathcal{A}_i$ over $\mathsf{label}(\rho_i)$. Then E_i^j is an NFA obtained from $\mathcal{A}_i$ by setting $q_{c(j)}$ and $q_{d(j)}$ as the unique initial and final state, respectively. Notice that $\mathsf{label}(\rho_i^j)$ is accepted by E_i^j. Furthermore, E_i^j is of at most polynomial size in $|\mathcal{A}_i|$, and, therefore, in $|\Gamma|$.

We now define a C2RPQ $\alpha'(\bar{x})$, which will be the basis for constructing the C2RPQ $\alpha(\bar{x})$ from the statement of the lemma. The variable set of α' is $\{z_n \mid n \in \mathcal{V}\}$. The free variables are $\bar{x} = (z_{p_1}, \ldots, z_{p_r})$, assuming that $\bar{n} = (p_1, \ldots, p_r)$. Finally, for each $1 \leq i \leq m$ and $1 \leq j \leq k_i$, the atom $(z_n, E_i^j, z_{n'})$ is in α', where n and n' are the initial and terminal nodes of ρ_i^j, respectively. The C2RPQ $\alpha'(\bar{x})$ is not necessarily acyclic, but it can be turned into an acyclic C2RPQ $\alpha(\bar{x})$ satisfying conditions 2. and 3. in the statement of the lemma as we shall see later. But before explaining how to turn α' into α, it is important to show that $\alpha'(\bar{x})$ satisfies conditions 2. and 3. in the statement of the lemma (recall that 2. states that $\bar{n} \in \alpha'(\mathcal{G})$ and $\alpha' \subseteq \Gamma$, while 3. states that the number of atoms in α' is at most polynomial in $|\Gamma|$).

First, α' satisfies condition 3. of the lemma. In fact, the number of atoms in α' is at most the number of paths ρ_i^j's involved in the decompositions of the ρ_i's, which is at most polynomial in $|\Gamma|$. Now we prove that α' satisfies condition 2. First, consider the mapping g from the variables of α' to $\mathcal{G}$ that maps each z_n to n. Since $\mathsf{label}(\rho_i^j)$ satisfies E_i^j, this mapping is actually a homomorphism. It follows that $\bar{n} \in \alpha'(\mathcal{G})$. In order to show that $\alpha \subseteq \Gamma$, we use Proposition 4.4. Let $\mathcal{G}'$ be a canonical database for α' with associated mapping ν. Consider the mapping f from the variables of γ to $\mathcal{G}'$ that maps each variable y to $\nu(z_{h(y)})$. The mapping f is well-defined since $h(y) \in \mathcal{I} \subseteq \mathcal{V}$, and thus $z_{h(y)}$ is a variable of α'. Now we show that for each $1 \leq i \leq m$, there is a path from $\nu(z_{h(u_i)})$ to $\nu(z_{h(v_i)})$ in $(\mathcal{G}')^{\pm}$ whose label is accepted by $\mathcal{A}_i$. We can simulate the path $\rho_i = \rho_i^1 \cdots \rho_i^{k_i}$ in $\mathcal{G}^{\pm}$ by the path $\chi_i^1 \cdots \chi_i^{k_i}$ in $(\mathcal{G}')^{\pm}$, where χ_i^j is the semipath in the canonical database $\mathcal{G}'$ that is associated with the atom of α' which is labeled E_i^j. By construction, there is a run of $\mathcal{A}_i$ over χ_i^j from $q_{c(j)}$ to $q_{d(j)}$, and thus $\mathsf{label}(\chi_i^1 \ldots \chi_i^{k_i})$ is accepted by $\mathcal{A}_i$ since it can be read in $\mathcal{A}_i$ from q_0 to q_ℓ. Therefore, f is a homomorphism. Moreover, $f(\bar{x}) = \nu(\bar{x})$. It follows that $\nu(\bar{x}) \in \gamma(\mathcal{G}') \subseteq \Gamma(\mathcal{G}')$. By Proposition 4.4, we conclude that $\alpha \subseteq \Gamma$.

As mentioned before, the problem with α' is that it is not necessarily acyclic. In fact, since $\mathcal{G}$ is pseudoacyclic, the underlying graph $\mathcal{U}(\alpha')$ of α' could be pseudoacyclic as opposed to acyclic. (Here we assume that the rooted tree used to construct the pseudoacyclic graph $\mathcal{U}(\alpha')$ is the natural one, that is, the sets of external and internal nodes of $\mathcal{U}(\alpha')$ are precisely $\{z_t \mid t \text{ is an external node in } \mathcal{V}\}$ and $\{z_t \mid t \text{ is an internal node in } \mathcal{V}\}$, respectively). For this to happen, the homomorphism h from γ to $\mathcal{G}$ must map two distinct variables in γ to internal nodes u and u' in two different semipaths π and π' in $\mathcal{G}$, such that π and π' connect exactly the same pair (n, n') of external nodes of $\mathcal{G}$. In such case, $\mathcal{U}(\alpha')$ might contain a cycle $n \to u \to n' \to u' \to n$.

Therefore, in order for $\mathcal{U}(\alpha')$ to be pseudoacyclic but not acyclic, it must contain *bad* paths, which are simple paths of the form $uu_1 \ldots u_k u'$, where u and u' are distinct external nodes of $\mathcal{U}(\alpha')$ and $u_1, \ldots, u_k$ are internal ones (notice that $k \geq 1$, i.e., at least one internal node belongs to a bad path). In other words, if $\mathcal{U}(\alpha')$ does not

contain bad paths, then it is actually acyclic. Thus, to obtain our desired acyclic query α we can modify α' in such a way that we eliminate all the bad paths from $\mathcal{U}(\alpha')$. This is what we do next.

Let us consider a bad path b in $\mathcal{U}(\alpha')$ of the form $z_{n_0} \cdots z_{n_{p+1}}$, where (i) $p \geq 1$, (ii) $z_{n_1}, \ldots, z_{n_p}$ are internal nodes of $\mathcal{U}(\alpha')$, and (iii) z_{n_0} and $z_{n_{p+1}}$ are distinct external nodes of $\mathcal{U}(\alpha')$. We assume without loss of generality that z_{n_0} is the parent of $z_{n_{p+1}}$ in $\mathcal{U}(\alpha')$. Therefore, we have that n_0 and n_{p+1} are external nodes in $\mathcal{G}$ such that n_0 is the parent of n_{p+1}, and there is a semipath ζ from n_0 to n_{p+1} such that $n_1, \ldots, n_p$ are precisely the internal nodes of ζ that belong to $\mathcal{V}$ (since ζ is a semipath, it does not repeat nodes by definition). For $k \in \{0, \ldots, p\}$, let ζ_k be the subpath of ζ that starts at n_k and ends at n_{k+1}. Further, for $k, k' \in \{0, \ldots, p+1\}$, let $\mathsf{Paths}_{k,k'}$ be the set of paths in some decomposition of some ρ_i that starts at n_k and end at $n_{k'}$. Notice that each path in $\mathsf{Paths}_{k,k}$, with $k \in \{1, \ldots, p\}$, satisfies that all its internal nodes are contained either in ζ_{k-1} or in ζ_k. For each $k \in \{1, \ldots, p\}$, we define $\mathsf{Paths}^{\uparrow}_{k,k}$ to be the set of paths in $\mathsf{Paths}_{k,k}$ whose internal nodes are in ζ_{k-1}. Analogously, $\mathsf{Paths}^{\downarrow}_{k,k} = \mathsf{Paths}_{k,k} \setminus \mathsf{Paths}^{\uparrow}_{k,k}$, i.e., $\mathsf{Paths}^{\downarrow}_{k,k}$ contains all the paths in $\mathsf{Paths}_{k,k}$ with internal nodes in ζ_k. For convenience, we define $\mathsf{Paths}^{\downarrow}_{0,0} = \mathsf{Paths}^{\uparrow}_{p+1,p+1} = \emptyset$.

In addition, for each $k \in \{0, \ldots, p\}$ we define a Γ-type $\pi_k = (\tau_{it}, \tau_{ti}, \tau_{ii}, \tau_{tt})$ as follows. The set τ_{it} contains the triple $(\mathcal{A}, q, q')$ if and only if there is a path $\rho_i^j \in \mathsf{Paths}_{k,k+1}$ such that $\mathcal{A} = \mathcal{A}_i$ and there is a run of $\mathcal{A}_i$ over $\mathsf{label}(\rho_i^j)$ from state q to q'. We define τ_{ti}, τ_{ii}, and τ_{tt} analogously, but this time replacing $\mathsf{Paths}_{k,k+1}$ with $\mathsf{Paths}_{k+1,k}$, $\mathsf{Paths}^{\downarrow}_{k,k}$, and $\mathsf{Paths}^{\uparrow}_{k+1,k+1}$, respectively. Observe that, since $\mathsf{Paths}^{\downarrow}_{0,0} = \mathsf{Paths}^{\uparrow}_{p+1p+1} = \emptyset$, we have that $\tau_{ii} = \emptyset$ for π_0 and $\tau_{tt} = \emptyset$ for π_p.

We then define $\mathcal{A}_{\pi_k}$ to be the NFA from Lemma 5.7; i.e., $\mathcal{A}_{\pi_k}$ defines the language $\mathcal{L}^{\subseteq}(\pi_k)$. Notice that $\mathsf{label}(\zeta_k)$ is accepted by $\mathcal{A}_{\pi_k}$. Furthermore, π_k, and, therefore, $\mathcal{A}_{\pi_k}$, is completely determined by ζ_k. For the sake of presentation we thus denote $\mathcal{A}_{\pi_k}$ by $\mathcal{A}_{\zeta_k}$ from now on. Finally, we define $\mathcal{I}_b$ to be the set of atoms

$$\{(z_{n_0}, \mathcal{A}_{\zeta_0}, z_{n_1}), (z_{n_1}, \mathcal{A}_{\zeta_1}, z_{n_2}), \ldots, (z_{n_p}, \mathcal{A}_{\zeta_p}, z_{n_{p+1}})\}$$

and $\mathcal{O}_b$ to be the set of atoms in α' of the form $(z_t, E_i^j, z_{t'})$, where (t, t') is a pair in $\{n_0, \ldots, n_{p+1}\} \times \{n_0, \ldots, n_{p+1}\} \setminus \{(n_0, n_0), (n_{p+1}, n_{p+1})\}$.

We define an auxiliary query α'' as the C2RPQ which is obtained from α' by simultaneously replacing all atoms in $\mathcal{O}_b$ with those in $\mathcal{I}_b$, for each bad path b in $\mathcal{U}(\alpha')$. Since $\mathcal{O}_b$ and $\mathcal{O}_{b'}$ are disjoint for different bad paths b and b', the query α'' is well defined. We claim that α'' satisfies conditions 2. and 3. of the lemma. For condition 3., note that for each bad path $b = z_{n_0} \cdots z_{n_{p+1}}$ in $\mathcal{U}(\alpha')$ the number of internal nodes p is bounded by the number of variables of γ, and in particular, by $|\Gamma|$. It follows that $|\mathcal{I}_b| \leq |\Gamma| + 1$. Since the number of bad paths is also polynomial in $|\Gamma|$, we conclude that the number of atoms in α'' is still polynomial in $|\Gamma|$.

Now we show that α'' satisfies condition 2. As noticed before, for an atom of the form $(\cdot, \mathcal{A}_{\zeta_i}, \cdot)$ in α'', we have that $\mathsf{label}(\zeta_i)$ is accepted by $\mathcal{A}_{\zeta_i}$. Putting this together with (i) the fact that the function that maps the variable z_n in α' to the node n in $\mathcal{G}$ is a homomorphism, and (ii) the way in which α'' is constructed from α', we conclude that $\bar{n} \in \alpha''(\mathcal{G})$. We prove that $\alpha'' \subseteq \Gamma$ using Proposition 4.4. Let $\mathcal{G}'$ be a canonical database for α'' with associated mapping ν, and assume that f is the mapping that sends each variable y in γ to $\nu(z_{h(y)})$ (as before, f is well-defined). Clearly, $f(\bar{x}) = \nu(\bar{x})$. We prove next that $(f(u_i), f(v_i))$ belongs to the evaluation $\mathcal{A}_i(\mathcal{G}')$ of $\mathcal{A}_i$ over $\mathcal{G}'$, for each $1 \leq i \leq m$. This implies that $\nu(\bar{x}) \in \gamma(\mathcal{G}')$, and, therefore, that $\alpha'' \subseteq \gamma \subseteq \Gamma$.

As before, one would like to simulate the path $\rho_i = \rho_i^1 \cdots \rho_i^{k_i}$ in $\mathcal{G}^{\pm}$ by the path $\chi_i^1 \cdots \chi_i^{k_i}$ in $(\mathcal{G}')^{\pm}$, where χ_i^j is the semipath in the canonical database $\mathcal{G}'$ that is associated with the atom of α'' which is labeled E_i^j. The only problem is that some of such atoms may have disappeared from α' and been replaced by atoms labeled with an NFA $\mathcal{A}_{\zeta_l}$ which accepts the semipath ζ_l in $\mathcal{G}$. Nevertheless, this could have only happened if one of the endpoints of a path of the form ρ_i^j is an internal node of $\mathcal{V}$. Let n and n' be external nodes in $\mathcal{V}$ such that n is the parent of n', and let ζ be a semipath in $\mathcal{G}$ from n to n'. Let t and t' be internal nodes of ζ (t is closer to n than t') and suppose that ρ_i^j goes from t to t'. (All other cases are analogous, i.e., when ρ_i^j goes from t' to t, when $t = t'$, and when t or t' is an external node). Further, let η be the subpath of ζ that goes from t to t'. We now explain how to simulate ρ_i^j by a path from $\nu(z_t)$ to $\nu(z_{t'})$ in $(\mathcal{G}')^{\pm}$.

By construction, we have that α'' contains the atom $(z_t, \mathcal{A}_\eta, z_{t'})$. Let $\pi = (\tau_{it}, \tau_{ti}, \tau_{ii}, \tau_{tt})$ be the Γ-type such that $\mathcal{A}_\eta = \mathcal{A}_\pi$. Then we have that $(\mathcal{A}_i, q_{c(j)}, q_{d(j)})$ is in τ_{it}. Let κ be the semipath from $\nu(z_t)$ to $\nu(z_{t'})$ in $\mathcal{G}'$ that is naturally associated with the atom $(z_t, \mathcal{A}_\eta, z_{t'})$. By definition, the first coordinate of the Γ-type of $\mathsf{label}(\kappa)$ contains $(\mathcal{A}_i, q_{c(j)}, q_{d(j)})$. Therefore, there is a word u such that $\mathcal{A}_i$ has a run over u from $q_{c(j)}$ to $q_{d(j)}$ and u can be folded into $\mathsf{label}(\kappa)$ from the initial to the final position. It follows that there is a path ϱ_i^j in $(\mathcal{G}')^{\pm}$ from $\nu(z_t)$ to $\nu(z_{t'})$ that can be read in $\mathcal{A}_i$ from $q_{c(j)}$ to $q_{d(j)}$. This is the path that simulates ρ_i^j. From this reasoning it is easy to conclude that there must be a path in $(\mathcal{G}')^{\pm}$ from $\nu(z_{h(u_i)})$ to $\nu(z_{h(v_i)})$ whose label is accepted by $\mathcal{A}_i$, for each $1 \leq i \leq m$. Therefore, $(f(u_i), f(v_i)) \in \mathcal{A}_i(\mathcal{G}')$ for each $1 \leq i \leq m$, which finishes the proof that α'' satisfies condition 2. in the statement of the lemma.

Now we define α. The query α is obtained from α'' by simultaneously replacing all atoms in $\mathcal{I}_b$ with the atom $(z_{n_0}, \mathcal{A}_{\zeta_0} \cdot \mathcal{A}_{\zeta_1} \cdots \mathcal{A}_{\zeta_p}, z_{n_{p+1}})$, for all bad paths $b = z_{n_0} \cdots z_{n_{p+1}}$ in $\mathcal{U}(\alpha'')$. Notice that this does not remove any free variables from α'', as they are external nodes of $\mathcal{U}(\alpha'')$. By construction, $\mathcal{U}(\alpha)$ does not contain bad paths, and thus it is acyclic. Further, α contains less atoms than α'' which implies that condition 3. still holds. Clearly, α is equivalent to α'', and, therefore, α also satisfies condition 2.

It remains to define the set $\mathcal{R}_\Gamma$. For an NFA $\mathcal{A}$ and states q, q' in $\mathcal{A}$, let $\mathcal{A}(q, q')$ be the NFA obtained from $\mathcal{A}$ by setting q as the initial state and q' as the only final state. Let $\mathcal{B}_\Gamma = \{\mathcal{A}(q, q') \mid \mathcal{A} \text{ appear in } \Gamma \text{ and } q, q' \text{ are states of } \mathcal{A}\}$. Further, let $\mathcal{C}_\Gamma$ be the set of all NFAs of the form $\mathcal{A}_\pi$, where π is a Γ-type (according to Lemma 5.7), and let $\mathcal{D}_\Gamma$ be the set of all NFAs that accept the concatenation of the languages accepted by at most p NFAs in $\mathcal{C}_\Gamma$, where $p \leq |\Gamma| + 1$. Then we define $\mathcal{R}_\Gamma = \mathcal{B}_\Gamma \cup \mathcal{D}_\Gamma$. It is easy to see that $\mathcal{R}_\Gamma$ can be constructed from Γ in single exponential time. Moreover, it is clear that all NFAs mentioned in α belong to $\mathcal{R}_\Gamma$. Therefore, conditions 1. and 4. are satisfied, which concludes the proof of the lemma. ∎

5.2.2. Construction of Γ_{app}. We now present the proof of Lemma 5.5. That is, we provide an EXPSPACE algorithm that, given a UC2RPQ Γ, constructs an acyclic UC2RPQ Γ_{app} such that the following hold:

1. $\Gamma_{\text{app}} \subseteq \Gamma$.
2. For every acyclic UC2RPQ Γ' such that $\Gamma' \subseteq \Gamma$ it is the case that $\Gamma' \subseteq \Gamma_{\text{app}}$.
3. The number of atoms and variables in each disjunct of Γ_{app} is at most polynomial in $|\Gamma|$.
4. The number of disjuncts and the size of each NFA appearing in Γ_{app} is at

most exponential in $|\Gamma|$.

Recall that Lemma 5.8 tells us that if $\mathcal{G}$ is a pseudoacyclic graph database, $\bar{n}$ is a tuple of external nodes in $\mathcal{G}$, and $\Gamma(\bar{x})$ is a UC2RPQ with $|\bar{n}| = |\bar{x}|$ such that $\bar{n} \in \Gamma(\mathcal{G})$, then there is an acyclic C2RPQ $\alpha(\bar{x})$ and a finite set $\mathcal{R}_\Gamma$ of NFAs over $\Sigma^\pm$ such that the following hold:

1. $\mathcal{R}_\Gamma$ can be constructed from Γ in time $2^{|\Gamma|^d}$, for $d \geq 1$.
2. $\bar{n} \in \alpha(\mathcal{G})$ and $\alpha \subseteq \Gamma$.
3. The number of atoms in α is at most $|\Gamma|^c$, for $c \geq 1$.
4. Each NFA mentioned in α belongs to $\mathcal{R}_\Gamma$.

Given a UC2RPQ $\Gamma(\bar{x})$ over Σ our algorithm proceed as follows:

a. We construct the finite set $\mathcal{R}_\Gamma$ of NFAs over $\Sigma^\pm$ from Lemma 5.8.
b. We iterate through every acyclic C2RPQ $\alpha'(\bar{x})$ with at most $|\Gamma|^c$ atoms, all of them labeled with NFAs in $\mathcal{R}_\Gamma$. If the C2RPQ $\alpha'(\bar{x})$ is contained in $\Gamma(\bar{x})$, then we add it as a disjunct to the output Γ_{app}.

First, observe that our algorithm is actually in EXPSPACE. In fact, step a. can be carried out in exponential time. Furthermore, the size of each NFA in $\mathcal{R}_\Gamma$ is at most $2^{|\Gamma|^d}$. This implies that the size of every acyclic C2RPQ α' that we need to consider in the loop of step b. is bounded by $|\Gamma|^c \cdot 2^{|\Gamma|^d}$, i.e., it is exponentially bounded by $|\Gamma|$. We can thus iterate over all such queries using no more than exponential space. Further, from item 1. of Proposition 4.1 it follows that checking whether $\alpha' \subseteq \Gamma$ can be done using space which is at most exponential in $|\Gamma|$ and $\mathsf{maxvar}(\alpha')$. But $\mathsf{maxvar}(\alpha')$ is at most $2|\Gamma|^c$, and thus checking $\alpha' \subseteq \Gamma$ can be carried out in exponential space in $|\Gamma|$. Therefore, the whole procedure can be performed in EXPSPACE.

Notice that Γ_{app} is nonempty. Indeed, assume that $x_1, \ldots, x_n$ are the free variables of Γ, i.e., those in $\bar{x}$. We assume without loss of generality that $\mathcal{R}_\Gamma$ contains the NFAs that define the empty word ε and the symbol a, for each $a \in \Sigma$. (If not, we simply extend $\mathcal{R}_\Gamma$ with those NFAs. The resulting set continues to satisfy all the desired conditions for $\mathcal{R}_\Gamma$). We also assume that the codification of Γ has size at least $n + |\Sigma|$. In particular, we have that $n + |\Sigma| \leq |\Gamma|^c$. Consider now the C2RPQ $\alpha^*(x_1, \ldots, x_n) := \bigwedge_{1 \leq i \leq n-1}(x_i, \varepsilon, x_{i+1}) \wedge \bigwedge_{a \in \Sigma}(x_1, a, x_1)$. Clearly, α^* is acyclic and its number of atoms is at most $n + |\Sigma| \leq |\Gamma|^c$. By the previous observations, it follows that α^* is one of the C2RPQs visited in step b. of the procedure. Moreover, it is easy to verify that $\alpha^* \subseteq \Gamma$. In fact, the only canonical database $\mathcal{G}$ of α^* consists of a single node u (which represents all the free variables $x_1, \ldots, x_n$) and a self-loop on u labeled a, for each $a \in \Sigma$. It is easy to see that $\bar{x} \in \Gamma(\mathcal{G})$. This is because if $\gamma(\bar{x})$ is an arbitrary C2RPQ in Γ, then the mapping h that sends every variable y of γ to u is a homomorphism which satisfies $h(\bar{x}) = \bar{x}$. We conclude that α^* is a disjunct of Γ_{app} and, therefore, that Γ_{app} is not empty.

We now prove that Γ_{app} satisfies conditions 1., 2., 3., and 4. in the statement of Lemma 5.5. Conditions 3. and 4. are trivially satisfied by construction. For condition 1., we have by definition that each disjunct α' of Γ_{app} is contained in Γ, from which we conclude that $\Gamma_{\text{app}} \subseteq \Gamma$.

For condition 2., let $\Gamma'(\bar{x})$ be an acyclic UC2RPQ such that $\Gamma' \subseteq \Gamma$. We need to show that $\Gamma' \subseteq \Gamma_{\text{app}}$. Let $\mathcal{G}$ be a canonical database of Γ' with associated mapping ν. By Proposition 4.4, it suffices to show that $\nu(\bar{x}) \in \Gamma_{\text{app}}(\mathcal{G})$. Since Γ' is acyclic, we have from Proposition 5.6 that $\mathcal{G}$ is pseudoacyclic. We also have that the tuple $\nu(\bar{x})$ only contains external nodes of $\mathcal{G}$. Moreover, since $\Gamma' \subseteq \Gamma$ it is the case that $\nu(\bar{x}) \in \Gamma(\mathcal{G})$. Therefore, we can apply Lemma 5.8 to UC2RPQ Γ, graph database $\mathcal{G}$, and tuple of nodes $\nu(\bar{x})$. This ensures the existence of an acyclic C2RPQ $\alpha(\bar{x})$

satisfying conditions 1.–4. in the statement of Lemma 5.8. Conditions 3. and 4. ensure that α is visited by our algorithm in step b. On the other hand, the second statement in condition 2. ensures that $\alpha \subseteq \Gamma$, from which we have that α is a disjunct of Γ_{app}. Finally, the first statement of condition 2. tells us that $\nu(\bar{x}) \in \alpha(\mathcal{G})$, and, therefore, $\nu(\bar{x}) \in \Gamma_{\text{app}}(\mathcal{G})$. This completes the proof of Lemma 5.5.

6. Semantic acyclicity of UC2RPQs. We finish the paper by studying the notion of semantic acyclicity in the context of graph databases and conjunctive regular path queries. As opposed to the case of CQs, the results in this section do not follow from known results in the literature and require new techniques. We start by defining the terminology and providing some basic insights about the nature of semantic acyclicity for UC2RPQs.

6.1. Basic terminology and insights. A UC2RPQ Γ is semantically acyclic if there exists an acyclic UC2RPQ Γ' such that $\Gamma \equiv \Gamma'$. As we mentioned before, we want to answer two basic questions about semantically acyclic UC2RPQs: (1) What is the cost of evaluating queries in this class? (2) What is the cost of checking whether a UC2RPQ is semantically acyclic? We will see that an answer to the second question will provide us with an answer for the first one.

Since acyclicity of C2RPQs is defined in terms of the acyclicity of its underlying CQ, one may be tempted to think that the two notions coincide. Clearly, if the underlying CQ of a C2RPQ γ is semantically acyclic, then γ is also semantically acyclic. The following example shows that the opposite does not hold.

Example 5. Consider again the nonacyclic CRPQ

$$\gamma'' = \exists x \exists y \exists z \big((x, L_1, y) \wedge (y, L_2, z) \wedge (z, L_3, x) \big)$$

in Example 4. It is not hard to prove that γ'' is equivalent to the acyclic CRPQ $\exists x (x, L_1 \cdot L_2 \cdot L_3, x)$, and, thus, it is semantically acyclic. On the other hand, the underlying CQ of γ'' is $\exists x \exists y \exists z \, \big(T_1(x, y), T_2(y, z), T_3(z, x) \big)$, which is not semantically acyclic.

Intuitively, the query γ'' is semantically acyclic because it can be "simplified" by concatenating the regular languages that label its atoms. A more interesting example is given by the Boolean CRPQ γ_{sa} over alphabet $\Sigma = \{a, \$_1, \$_2, \$_3\}$ shown in Figure 6.1. (Dots represent variables and arrows represent labeled atoms.)

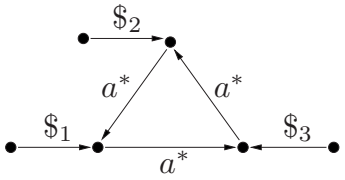

FIG. 6.1. *The CRPQ* γ_{sa}.

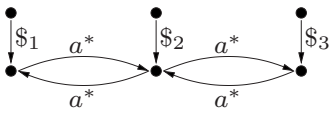

FIG. 6.2. *The acyclic CRPQ that is equivalent to* γ_{sa}.

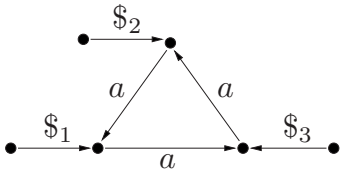

FIG. 6.3. *The CRPQ* γ_{na} *from Example* 6.

It is easy to see that the underlying CQ of γ_{sa} is not semantically acyclic. On the other hand, it can be proved that γ_{sa} is equivalent to the acyclic CRPQ shown in Figure 6.2. In this case, semantic acyclicity is obtained by the way in which the regular languages that label the atoms of γ_{sa} interact with each other. ∎

The previous example shows that the notion of semantic acyclicity of C2RPQs is richer than the notion of semantic acyclicity of its underlying CQs, as many queries fall in the former category but not in the latter. Not only that, the first notion is also theoretically more challenging: While the same techniques used in section 2 can be applied to prove that the evaluation problem is tractable for UC2RPQs whose underlying CQ is semantically acyclic, it is by no means clear whether the same is true for the class of semantically acyclic UC2RPQs (and even for semantically acyclic CRPQs). We delve into this issue below.

As is mentioned in the introduction, the CSP techniques used in section 2 to prove that the evaluation of semantically acyclic UCQs is tractable do not yield answers to our questions about semantically acyclic UC2RPQs. The results in section 6.2 help us prove, on the other hand, that the problem is fixed-parameter tractable (Theorem 6.3), which was not known to date. We leave as an open question whether the class of semantically acyclic UC2RPQs can be evaluated in polynomial time.

Before finishing the section we explore the limits of the notion of semantic acyclicity. The next example shows a simple CQ over graph databases that is not equivalent to any acyclic UC2RPQ.

Example 6. Let $\Sigma = \{a, \$_1, \$_2, \$_3\}$ be a finite alphabet and consider the Boolean CRPQ γ_{na} over Σ that is graphically depicted in Figure 6.3. Notice that the underlying CQ of γ_{na} coincides with that of the semantically acyclic CRPQ γ_{sa} from Figure 6.1. However, a simple case-by-case analysis shows that γ_{na} is not semantically acyclic. The reason is that γ_{na} forbids the interaction between the different RPQs that label its atoms by replacing each RPQ of the form a^* in γ_{sa} with a. ∎

6.2. Verification of semantic acyclicity. We start by considering our second question above: Is the notion of semantic acyclicity for UC2RPQs decidable? In this section we show that this is indeed the case and provide matching upper and lower bounds for its computational cost.

We start by proving that the notion of semantic acyclicity for UC2RPQs is decidable, and provide an EXPSPACE upper bound for the problem. The algorithm also yields an equivalent UC2RPQ Γ' of exponential size for a semantically acyclic UC2RPQ Γ.

THEOREM 6.1. *There exists an* EXPSPACE *algorithm that on input a UC2RPQ* Γ *does the following:*

1. *It checks whether* Γ *is semantically acyclic.*
2. *If the latter holds, it outputs an acyclic UC2RPQ* Γ' *of single-exponential size*

such that $\Gamma \equiv \Gamma'$.

Proof. The algorithm in Lemma 5.5 computes on input Γ an acyclic UC2RPQ Γ_{app} such that Γ_{app} is the maximum among all acyclic UC2RPQs that are contained in Γ. It follows that Γ is semantically acyclic if and only if $\Gamma \subseteq \Gamma_{\text{app}}$. Thus, in order to check semantic acyclicity of Γ we can compute Γ_{app}, which can be done in EXPSPACE, and then check whether $\Gamma \subseteq \Gamma_{\text{app}}$. It follows from item (1) of Proposition 4.1 that the latter can be done using space exponential in $\mathsf{maxvar}(\Gamma)$ and $|\Gamma_{\text{app}}|$, and hence double-exponential in $|\Gamma|$ (since Γ_{app} can be of exponential size in $|\Gamma|$). This provides us with an easy 2EXPSPACE procedure for checking semantic acyclicity of UC2RPQs.

To obtain an EXPSPACE procedure we exploit Theorem 4.2. Observe first that the width $w(\Gamma_{\text{app}})$ of Γ_{app} is at most the maximum number of atoms over its disjuncts, which by Lemma 5.5 item 3. is at most polynomial in $|\Gamma|$. Since Γ_{app} is acyclic, Theorem 4.2 tells us that the problem of checking $\Gamma \subseteq \Gamma_{\text{app}}$ can be decided using $O((|\Gamma| + |\Gamma_{\text{app}}|)^{C \cdot w(\Gamma_{\text{app}})})$ space, for some constant $C \geq 1$. Hence, from the fact that $w(\Gamma_{\text{app}})$ is polynomially bounded by $|\Gamma|$ we obtain that checking $\Gamma \subseteq \Gamma_{\text{app}}$ can be solved in EXPSPACE. This gives an EXPSPACE algorithm for checking whether Γ is semantically acyclic. For the second part of the theorem, we output the query Γ_{app} if $\Gamma \subseteq \Gamma_{\text{app}}$; otherwise, we reject the input. ∎

We now provide a lower bound for the problem that shows that checking semantic acyclicity of (U)C(2)RPQs is considerably harder than for UCQs.

PROPOSITION 6.2. *It is* EXPSPACE*-hard to check whether a UC2RPQ* Γ *is semantically acyclic. The problem remains* EXPSPACE*-hard even if the input is restricted to Boolean CRPQs.*

Proof. First, we claim that checking containment of γ_1 in γ_2, where γ_1 and γ_2 are UC2RPQs, is EXPSPACE-hard even when γ_1 is a Boolean acyclic CRPQ, γ_2 is a Boolean CRPQ with $\mathcal{U}(\gamma_2)$ connected, and γ_2 is not semantically acyclic.

Indeed, it follows from [9] (also stated in item 2. of Proposition 4.1) that containment is EXPSPACE-hard, even for γ_1 and γ_2 acyclic CRPQs. Moreover, the reduction in [9] yields CRPQs γ_1 and γ_2 of the following form:

$$\begin{aligned}
\gamma_1(x_1, x_2) &= (x_1, E, x_2),\\
\gamma_2(x_1, x_2) &= \exists y_1 \exists y_2 \left((x_1, E_1, y_1) \wedge \left(\bigwedge_{0 \leq i \leq n} (y_1, F_i, y_2) \right) \wedge (y_2, E_1, x_2) \right),
\end{aligned}$$

where $E, E_1, F_0, \ldots, F_n$ are RPQs over an alphabet Δ. We then construct Boolean CRPQs γ_1' and γ_2' from γ_1, γ_2 as follows:

$$\begin{aligned}
\gamma_1'() &= \exists x_1 \exists x_2 \left((x_1, E, x_2) \wedge (x_1, \#_1, x_1) \wedge (x_2, \#_2, x_2) \wedge \left(\bigwedge_{b \in \Sigma} (x_2, b, x_2) \right) \right),\\
\gamma_2'() &= \exists x_1 \exists x_2 \exists y_1 \exists y_2 \exists z_2 \exists z_3 \exists w_1 \exists w_2 \exists w_3\\
&\quad \left((x_1, \#_1, x_1) \wedge (x_2, \#_2, x_2) \wedge (x_1, E_1, y_1) \wedge \left(\bigwedge_{0 \leq i \leq n} (y_1, F_i, y_2) \right) \wedge (y_2, E_1, x_2) \right.\\
&\quad \left. \wedge (x_2, a, z_2) \wedge (z_2, a, z_3) \wedge (z_3, a, x_2) \wedge (w_1, \$_1, x_2) \wedge (w_2, \$_2, z_2) \wedge (w_3, \$_3, z_3) \right),
\end{aligned}$$

where $\Sigma = \{a, \$_1, \$_2, \$_3\}$ is disjoint from Δ and $\#_1, \#_2$ are fresh symbols not in $\Delta \cup \Sigma$. Intuitively, γ_1' is the Boolean version of γ_1, where we have marked the free variables

x_1 and x_2 of γ_1 with special symbols $\#_1$ and $\#_2$, respectively, and appended a loop labeled b to x_2, for each $b \in \Sigma$. Similarly, γ_2' is the Boolean version of γ_2, where we have again marked the free variables x_1 and x_2 of γ_2 with special symbols $\#_1$ and $\#_2$, respectively, and appended to x_2 a copy of the (nonsemantically acyclic) query γ_{na} from Example 6.

It is straightforward to show that $\gamma_1 \subseteq \gamma_2$ if and only if $\gamma_1' \subseteq \gamma_2'$. Note also that γ_1' is a Boolean acyclic CRPQ and γ_2' is a Boolean CRPQ such that $\mathcal{U}(\gamma_2')$ is connected. Moreover, since γ_{na} is not semantically acyclic, it is easy to show that γ_2' is not semantically acyclic either. Thus containment is EXPSPACE-hard even for these kinds of queries. Next, we reduce this EXPSPACE-hard restriction of the containment problem to our problem of checking whether a query is semantically acyclic.

Let γ_1 and γ_2 be Boolean CRPQs such that γ_1 is acyclic, $\mathcal{U}(\gamma_2)$ is connected, and γ_2 is not semantically acyclic. We claim that γ_1 is contained in γ_2 if and only if the CRPQ $\gamma_1 \wedge \gamma_2$ is semantically acyclic (we rename the variables of γ_1 and γ_2 so they are disjoint in $\gamma_1 \wedge \gamma_2$). Assume first that γ_1 is contained in γ_2. Then $\gamma_1 \wedge \gamma_2$ is equivalent to γ_1, which is acyclic, and thus $\gamma_1 \wedge \gamma_2$ is semantically acyclic.

On the other hand, assume that $\gamma_1 \wedge \gamma_2 \equiv \alpha$, where $\alpha = \bigvee_{1 \leq i \leq m} \alpha_i$ and each α_i is an acyclic C2RPQ. Since $\gamma_1 \wedge \gamma_2 \subseteq \gamma_2$, it follows that $\alpha \subseteq \gamma_2$. Also, since $\alpha_i \subseteq \alpha$ it follows that $\alpha_i \subseteq \gamma_2$, for each $1 \leq i \leq m$. Let $\alpha_i^1, \ldots, \alpha_i^{k_i}$ be the C2RPQs associated with each connected component of $\mathcal{U}(\alpha_i)$. Thus $\alpha_i \equiv \alpha_i^1 \wedge \cdots \wedge \alpha_i^{k_i}$. We show that for each $1 \leq i \leq m$, there exists $1 \leq j_i \leq k_i$ such that $\alpha_i^{j_i} \subseteq \gamma_2$. Assume to the contrary. Then, by Proposition 4.4, there exist for each $1 \leq j \leq k_i$ a canonical graph database $\mathcal{G}_j$ for α_i^j such that $\gamma_2(\mathcal{G}_j) = \mathsf{false}$. Note that since $\mathcal{U}(\alpha_i^j)$ is connected, then $\mathcal{G}_j$ also is. Consider the disjoint union $\mathcal{G}$ of $\mathcal{G}_1, \ldots, \mathcal{G}_{k_i}$. Clearly this is a canonical database for α_i. Since $\alpha_i \subseteq \gamma_2$, it follows that $\gamma_2(\mathcal{G}) = \mathsf{true}$. But $\mathcal{U}(\gamma_2)$ is connected, which implies that $\gamma_2(\mathcal{G}_j) = \mathsf{true}$, for some $1 \leq j \leq k_i$. This is a contradiction.

Consider the acyclic UC2RPQ $\alpha' = \bigvee_{1 \leq i \leq m} \alpha_i^{j_i}$. Notice that $\alpha \subseteq \alpha'$. By definition, we have that $\alpha' \subseteq \gamma_2$. Note also that since γ_2 is not semantically acyclic, it must be the case that $\gamma_2 \not\subseteq \alpha'$. Hence there is a canonical database $\mathcal{G}^*$ for γ_2 such that $\alpha'(\mathcal{G}^*) = \mathsf{false}$. We now prove that $\gamma_1 \subseteq \alpha'$, which implies $\gamma_1 \subseteq \gamma_2$. Let $\mathcal{G}$ be a canonical database for γ_1. Consider the disjoint union $\mathcal{G}'$ of $\mathcal{G}$ and $\mathcal{G}^*$. Clearly $\mathcal{G}'$ is a canonical database for $\gamma_1 \wedge \gamma_2$. Since $\gamma_1 \wedge \gamma_2 \equiv \alpha \subseteq \alpha'$, it follows that $\alpha'(\mathcal{G}') = \mathsf{true}$. Then $\alpha_i^{j_i}(\mathcal{G}') = \mathsf{true}$, for some $1 \leq i \leq m$. Since $\mathcal{U}(\alpha_i^{j_i})$ is connected, it follows that either $\alpha_i^{j_i}(\mathcal{G}) = \mathsf{true}$ or $\alpha_i^{j_i}(\mathcal{G}^*) = \mathsf{true}$. But $\alpha'(\mathcal{G}^*) = \mathsf{false}$, and thus $\alpha_i^{j_i}(\mathcal{G}^*) = \mathsf{false}$. We conclude that $\alpha_i^{j_i}(\mathcal{G}) = \mathsf{true}$, and thus $\alpha'(\mathcal{G}) = \mathsf{true}$. By Proposition 4.4, we conclude that $\gamma_1 \subseteq \alpha'$. This proves the proposition. ∎

6.3. Evaluation of semantically acyclic UC2RPQs. With the help of Theorem 6.1 we can provide an answer to our first question regarding semantically acyclic UC2RPQs: Its evaluation is fixed-parameter tractable.

THEOREM 6.3. *The problem of checking whether $\bar{n} \in \Gamma(\mathcal{G})$, for a given graph database $\mathcal{G}$, semantically acyclic UC2RPQ Γ, and tuple $\bar{n}$ of node ids in $\mathcal{G}$, is fixed-parameter tractable.*

Proof. Using the algorithm in Theorem 6.1 it is possible to compute in EXPSPACE (i.e., in double-exponential time) an acyclic UC2RPQ Γ' such that $\Gamma \equiv \Gamma'$. Clearly, $\bar{n} \in \Gamma(\mathcal{G})$ if and only if $\bar{n} \in \Gamma'(\mathcal{G})$. From Theorem 3.1 the latter can be checked in time $O(|\mathcal{G}|^2 \cdot |\Gamma'|^2)$, and hence in time $O(|\mathcal{G}|^2 \cdot 2^{p(|\Gamma|)})$, where $p : \mathbb{N} \to \mathbb{N}$ is a polynomial. Thus, semantically acyclic UC2RPQs can be evaluated in time $O(f(|\Gamma|) + |\mathcal{G}|^2 \cdot 2^{p(|\Gamma|)})$, where $p : \mathbb{N} \to \mathbb{N}$ is a polynomial and $f : \mathbb{N} \to \mathbb{N}$ is a double-exponential function. ∎

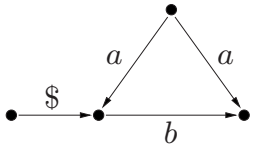

FIG. 6.4. *The query γ from Proposition* 6.4.

6.4. Features of the language: Inverses. The algorithm in Theorem 6.1 introduces inverses in the construction of an equivalent acyclic query, even if we start from a semantically acyclic UCRPQ (i.e., a UC2RPQ without inverses). A natural question is whether this is necessary, that is, whether there are semantically acyclic UCRPQs that find an equivalent query in the class of acyclic UC2RPQs, but not in the class of acyclic UCRPQs. We prove that this is the case.

PROPOSITION 6.4. *There is a semantically acyclic CRPQ that is not equivalent to any acyclic UCRPQ.*

Proof. The Boolean query $\gamma = \exists w, x, y, z((w, \$, x), (x, b, y), (z, a, y), (z, a, x))$ that is graphically depicted in Figure 6.4 is semantically acyclic. In fact, it is equivalent to the acyclic C2RPQ $\exists x \exists y((x, \$, y) \wedge (y, ba^{-}a, y))$. For the sake of contradiction, suppose there is an acyclic UCRPQ γ' equivalent to γ. It follows from Proposition 4.4 that there exists a canonical database $\mathcal{G}_{\gamma'}$, and homomorphisms h and g from $\mathcal{G}_\gamma$ to $\mathcal{G}_{\gamma'}$, and from $\mathcal{G}_{\gamma'}$ to $\mathcal{G}_\gamma$, respectively, where $\mathcal{G}_\gamma$ is the canonical database of γ. Observe that $\mathcal{G}_{\gamma'}$ is pseudoacyclic (recall the definitions from section 5.2.1), and since γ' is a UCRPQ, each internal node in $\mathcal{G}_{\gamma'}$ has in-degree and out-degree 1. Suppose first that $h(x), h(y), h(z)$ are distinct elements in $\mathcal{G}_{\gamma'}$. It follows that the in-degree of $h(x)$ and $h(y)$ is 2, and the out-degree of $h(z)$ is 2. By the observation above, we have that $h(x), h(y), h(z)$ must be external nodes in $\mathcal{G}_{\gamma'}$, which contradicts the acyclicity of γ'. Then, $h(x), h(y), h(z)$ are not distinct in $\mathcal{G}_{\gamma'}$. This is also a contradiction, as it implies that there is an a-labeled or b-labeled loop in $\mathcal{G}_{\gamma'}$, and thus g cannot be an homomorphism. ∎

6.5. Features of the language: Unions. The algorithm that constructs an equivalent acyclic UC2RPQ Γ' for a semantically acyclic UC2RPQ Γ (Theorem 6.1) outputs a union of C2RPQs Γ' even if Γ is a C2RPQ. But is this necessary? That is, is there a C2RPQ Γ that is semantically acyclic, but yet it is not equivalent to a *single* acyclic C2RPQ Γ'? We currently do not have an answer to this question. But as the following example shows, finding such an answer might not be as simple as one thinks beforehand. This is due to the interplay of regular expressions in C2RPQs, which allows for intricate ways of expressing unions of C2RPQs as single C2RPQs.

Example 7. Let γ be the CRPQ over $\Sigma = \{a, b\}$ shown in Figure 6.5, where a_ε and b_ε are abbreviations for $(a + \varepsilon)$ and $(b + \varepsilon)$, respectively. This query has four nonequivalent canonical databases, which are depicted in Figure 6.6. Notice that each one of these canonical databases is acyclic, and, therefore, γ is semantically acyclic (as a C2RPQ is always equivalent to the union of CQs represented by its canonical databases).

Somewhat surprisingly, γ is also equivalent to the single CRPQ γ' shown in Figure 6.7. In fact, it is tedious but not difficult to verify that each one of the canonical databases of γ shown in Figure 6.6 satisfies γ' and, conversely, that each canonical database of γ' satisfies γ. From this we conclude that $\gamma \equiv \gamma'$. ∎

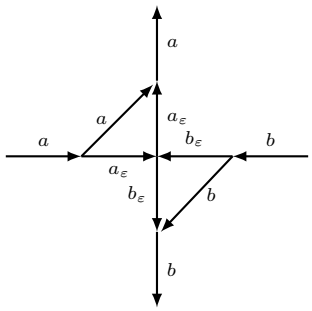

FIG. 6.5. *The CRPQ γ from Example* 7.

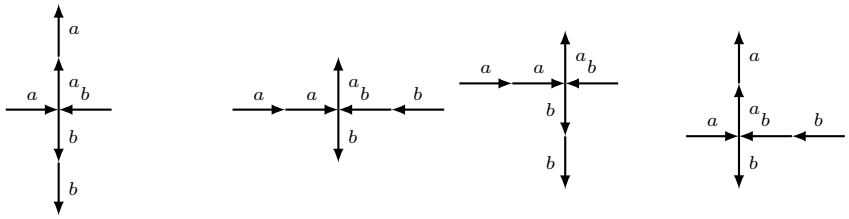

FIG. 6.6. *The four canonical databases of γ.*

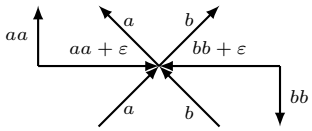

FIG. 6.7. *The acyclic CRPQ γ' which is equivalent to γ.*

7. Conclusions and open problems. We have studied the space of UCQs and UC2RPQs defined by the notion of acyclicity. This is relevant since acyclicity is a robust explanation for the tractability of several query languages for relational and graph databases. Furthermore, some notions of acyclicity explain the linear-time behavior of various querying mechanisms for graph databases (e.g., XPath [35], PDL [28], nested regular expressions [5], etc.).

While the results about semantic acyclicity of UCQs follow from techniques in CSP, studying the notion of semantic acyclicity of UC2RPQs requires new tools and insights. We have shown that it is EXPSPACE-complete to check whether a UC2RPQ is semantically acyclic, and that this shows that evaluation of queries in the class is fixed-parameter tractable. The techniques used to prove decidability also yield a strong theory of approximations of UC2RPQs.

As far as the notion of semantic acyclicity of UC2RPQs is concerned, in this work we have only uncovered the tip of the iceberg. Many questions remain open and we list some of them below.

Complexity. We have proven that evaluation of semantically acyclic UC2RPQs is fixed-parameter tractable. But is it also polynomial? Tractability of semantically acyclic UCQs follows from a sophisticated characterization of the problem in terms of winning strategies in the existential pebble game, but we do not know whether those techniques can be extended to deal with UC2RPQs.

Size of equivalent acyclic queries. The algorithm presented in Theorem 6.1 computes an equivalent acyclic query of single-exponential size for a semantically acyclic UC2RPQ. Is this optimal, i.e., is there a family $(\Gamma_n)_{n\geq 1}$ of semantically acyclic UC2RPQs such that (1) $|\Gamma_n|$ is polynomially bounded by n, and (2) the smallest acyclic UC2RPQ that is equivalent to Γ_n is of size $\Omega(2^n)$, for each $n \geq 1$?

Beyond acyclic queries. Acyclicity is a simple syntactic criterion that ensures efficient evaluation of CQs, but it is not the only one. In the last decades several criteria have been identified that extend acyclicity in different ways while retaining polynomial time evaluation for the CQs that satisfy them. Most of these criteria are based on the idea of restricting evaluation to CQs θ of bounded *treewidth* [13, 17], which are also defined in terms of the existence of a tree decomposition of θ with desirable properties.

It is known that the results in section 2 (Theorem 2.2 and Proposition 2.3) also apply to UCQs that are equivalent to unions of CQs of bounded treewidth [17, 14]. That is, such classes of UCQs can be evaluated in polynomial time, and it is NP-complete to check whether a CQ is equivalent to a CQ of treewidth at most k, for each $k \geq 1$.

On the other hand, our results for UC2RPQs are specifically designed for semantic acyclicity, and we do not know at this point how to extend them to verify whether a UC2RPQ is equivalent to a UC2RPQ of bounded treewidth. In the same way, one might be interested in extending results on approximations of UC2RPQs to classes of bounded treewidth, as it has been done for UCQs [6].

Beyond C2RPQs. Instead of working with C2RPQs one could also consider the class of conjunctions of nested regular expressions (CNREs), that properly extends the former [5]. Acyclicity of CNREs also leads to tractability, and thus it makes sense to study semantic acyclicity in this extended setting. The problem is relevant since several linear-time query languages for graph databases are contained in the class of CNREs but not in the class of UC2RPQs [28, 5].

Enumeration. In the evaluation problem, we are given a query θ, a database $\mathcal{D}$, and a tuple $\bar{t}$, and we have to decide whether $\bar{t}$ belongs to the evaluation of θ over $\mathcal{D}$. However, in many practical applications it is also useful to *enumerate* the tuples that belong to the evaluation of θ over $\mathcal{D}$. The complexity of this enumeration problem has been studied for several query languages, e.g., (U)CQs [8, 27]. It follows from [27] that the tuples in the evaluation of a semantically acyclic UCQs can be enumerated with *polynomial delay*. This means that there is an algorithm that, given a semantically acyclic UCQ Θ and a database $\mathcal{D}$, enumerates $\Theta(\mathcal{D})$ (without repetitions) in such a way that (i) the first tuple is produced in polynomial time, and (ii) each subsequent tuple is produced within polynomial time from the previously produced tuple. Notice that this tractability result actually extends Theorem 2.2. An important open question is whether we can obtain a similar tractability result for enumerating the output of semantically acyclic UC2RPQs.

CSP for C2RPQs. Our work can also be viewed as opening a new line of research in constraint satisfaction. As noted above, there is an intimate connection between CQ evaluation and constraint satisfaction. In general this problem is NP-complete, but there is an extensive body of research studying tractable cases, either by fixing the database and focusing on expression complexity, or by studying the combined complexity of restricted classes of queries [21, 32]. The same approach, fixing the database or restricting the class of queries, can also be applied to the

evaluation of C2RPQs. In particular, as noted above, it is an open question whether the class of semantically acyclic C2RPQs is an "island of tractability" in the sense of [32], that is, whether its evaluation problem is tractable.

8. Appendix.

Proof of Theorem 2.2. The *existential k-cover game* was introduced in [14], in order to define tractable restrictions of the *constraint satisfaction problem*. This game is a variant of the *existential k-pebble game* defined by Kolaitis and Vardi [31]. In the k-cover game, instead of imposing the Spoiler to use at most k pebbles—as in the existential k-pebble game—he is allowed to use any number of pebbles, but only as long as the set of elements where the pebbles are placed can be covered by at most k tuples in the database where the Spoiler is playing. We skip here the formal definition of the k-cover game but recall some important properties of the game that were proved in [14].

PROPOSITION 8.1. *The following hold:*

1. *There is a polynomial time algorithm that, given databases $\mathcal{D}$ and $\mathcal{D}'$ over the same schema, decides whether the Duplicator has a winning strategy in the 1-cover game on $\mathcal{D}$ and $\mathcal{D}'$.*
2. *Assume that the Duplicator has a winning strategy in the 1-cover game on $\mathcal{D}$ and $\mathcal{D}'$. Then for each acyclic Boolean CQ θ it is the case that $\theta(\mathcal{D}) = \texttt{true}$ implies $\theta(\mathcal{D}') = \texttt{true}$.*

Let $\theta(\bar{x}) = \exists \bar{y} \bigwedge_{i=1}^{m} P_i(\bar{u}_i)$ be a CQ over schema σ. As usual, we define the *canonical database* $\mathcal{D}_\theta$ of θ to be the database over σ whose facts are precisely the $P_i(\bar{u}_i)$'s, for $1 \leq i \leq m$. The proof of the theorem makes use of the following well-known characterization of containment of UCQs in terms of evaluation over canonical databases, which is due to Sagiv and Yannakakis.

PROPOSITION 8.2 (see [39]). *Let $\Theta(\bar{x}) = \bigvee_{1 \leq i \leq m} \theta_i(\bar{x})$ and $\Theta'(\bar{x}) = \bigvee_{1 \leq j \leq n} \theta'_j(\bar{x})$ be UCQs. Then $\Theta(\bar{x}) \subseteq \Theta'(\bar{x})$ if and only if for each $1 \leq i \leq m$ there exists $1 \leq j \leq n$ such that $\bar{x} \in \theta'_j(\mathcal{D}_{\theta_i})$.*

Let $\mathcal{D}$ be a database over schema σ and $\bar{a} = (a_1, \ldots, a_r)$ a tuple of elements over $\mathcal{D}$. We denote by $(\mathcal{D}, \bar{a})$ the database over $\sigma \cup \{P_1, \ldots, P_r\}$, where $P_1, \ldots, P_r$ are fresh symbols, obtained from $\mathcal{D}$ by adding the new facts $P_1(a_1), \ldots, P_r(a_r)$.

We now present an algorithm for evaluating semantically acyclic UCQs. Given a semantically acyclic UCQ $\Theta(\bar{x}) = \bigvee_{1 \leq i \leq m} \theta_i(\bar{x})$, a database $\mathcal{D}$, and a tuple $\bar{a}$ of elements in $\mathcal{D}$, the algorithm checks whether there is an index $i \in \{1, \ldots, m\}$ such that the Duplicator has a winning strategy in the 1-cover game on $(\mathcal{D}_{\theta_i}, \bar{x})$ and $(\mathcal{D}, \bar{a})$. If this is the case, it accepts (i.e., it declares that $\bar{a} \in \Theta(\mathcal{D})$); otherwise it rejects. From the first part of Proposition 8.1 it follows that this algorithm runs in polynomial time. We show next that the algorithm is sound and complete.

Assume first that $\bar{a} \in \Theta(\mathcal{D})$. Then, $\bar{a} \in \theta_i(\mathcal{D})$, for some $1 \leq i \leq m$. By definition, there is a homomorphism h from θ_i (or, equivalently, from $\mathcal{D}_{\theta_i}$) to $\mathcal{D}$ such that $h(\bar{x}) = \bar{a}$. In particular, h is also an homomorphism from $(\mathcal{D}_{\theta_i}, \bar{x})$ to $(\mathcal{D}, \bar{a})$. This trivially implies the existence of a winning strategy for the Duplicator in the 1-cover game on $(\mathcal{D}_{\theta_i}, \bar{x})$ and $(\mathcal{D}, \bar{a})$. Therefore, the algorithm accepts. (Notice that in this direction the assumption that Θ is semantically acyclic is not used.) Conversely, suppose that the algorithm accepts. Then there exists a winning strategy for the Duplicator in the 1-cover game on $(\mathcal{D}_{\theta_i}, \bar{x})$ and $(\mathcal{D}, \bar{a})$, for some $1 \leq i \leq m$. Let $\Theta'(\bar{x}) = \bigvee_{1 \leq j \leq n} \theta'_j(\bar{x})$ be an acyclic UCQ which is equivalent to Θ. By Proposition

8.2, there exists $1 \leq j \leq n$ such that $\bar{x} \in \theta'_j(\mathcal{D}_{\theta_i})$. Let θ''_j be the acyclic Boolean CQ obtained from $\theta'_j(\bar{x})$ by adding the atoms $P_1(x_1), \ldots, P_r(x_r)$, where $\bar{x} = (x_1, \ldots, x_r)$. Then, $\theta''_j((\mathcal{D}_{\theta_i}, \bar{x})) = \texttt{true}$. Using the second part of Proposition 8.1 with $\theta = \theta''_j$, it follows that $\theta''_j((\mathcal{D}, \bar{a})) = \texttt{true}$, and, thus, $\bar{a} \in \theta'_j(\mathcal{D})$. This implies that $\bar{a} \in \Theta'(\mathcal{D})$. Since $\Theta' \equiv \Theta$, we conclude that $\bar{a} \in \Theta(\mathcal{D})$.

Proof of Proposition 2.3. For the membership in NP we show that if $\Theta(\bar{x})$ is semantically acyclic, then it is equivalent to an acyclic UCQ $\Theta'(\bar{x})$ whose size is bounded by the size of Θ. Then, given a UCQ $\Theta(\bar{x})$, the NP algorithm guesses an acyclic UCQ $\Theta'(\bar{x})$ with $|\Theta'| \leq |\Theta|$, and then checks whether $\Theta(\bar{x}) \equiv \Theta'(\bar{x})$. The latter can be done in NP [12], and hence the whole procedure is in NP.

Let $\theta(\bar{x})$ be a CQ. We say that a CQ $\theta'(\bar{x})$ is a *core* of θ [29, 12] if (a) $\theta(\bar{x}) \equiv \theta'(\bar{x})$, and (b) no CQ with strictly fewer atoms than θ' is equivalent to θ. It is known (see, e.g., [29]) that each CQ $\theta(\bar{x})$ has a unique (up to isomorphism) core, and thus we can talk about *the* core of $\theta(\bar{x})$. We make use of the following proposition in our proof.

PROPOSITION 8.3. *If θ is an acyclic CQ, then the core of θ is also acyclic.*

Proof. We show the claim in two steps. First, we show that the class of acyclic CQs is closed under taking *strong induced subqueries*. Then we prove that the core of a CQ θ is always a strong induced subquery.

For each atom P of θ we denote by V_P the set of variables that are mentioned in P. A CQ θ' is a strong induced subquery of a CQ θ if the following hold:

1. The set $V_{\theta'}$ of variables mentioned in θ' is contained in the set V_θ of variables mentioned in θ.
2. The atoms of θ' are exactly the atoms of θ induced by the variables in $V_{\theta'}$.
3. The free variables of θ' are exactly the free variables of θ.
4. If P is an atom in θ but not in θ', then there exists an atom P' in θ' that contains all the variables in $V_P \cap V_{\theta'}$.

Suppose θ is an acyclic CQ and let θ' be a strong induced subquery of θ. Let (T, λ) be a tree decomposition of $\mathcal{H}(\theta)$ witnessing the acyclicity of θ. Consider the tree decomposition (T, λ') such that, for each $t \in T$, $\lambda'(t) = \lambda(t) \cap V_{\theta'}$. Clearly, (T, λ') is a tree decomposition of $\mathcal{H}(\theta')$. We now show that we can transform the decomposition (T, λ') into a tree decomposition $(\tilde{T}, \tilde{\lambda})$ of $\mathcal{H}(\theta')$ such that $\tilde{\lambda}(t)$ is a hyperedge of $\mathcal{H}(\theta')$, for each $t \in \tilde{T}$. This would imply that θ' is acyclic, as required.

Let $t \in T$ be a node such that $\lambda'(t)$ is not a hyperedge of $\mathcal{H}(\theta')$. By definition, $\lambda(t)$ is a hyperedge of $\mathcal{H}(\theta)$. We can then pick an atom P in θ such that the set of variables V_P mentioned in P is precisely $\lambda(t)$. Then, by definition, $\lambda'(t) = V_P \cap V_{\theta'}$. By condition 4. in the definition of strong induced subquery, there is an atom P' in θ' such that $\lambda'(t) = V_P \cap V_{\theta'} \subseteq V_{P'}$. Moreover, by definition of tree decomposition, there is a node $t^* \in T$ such that $V_{P'} \subseteq \lambda'(t^*)$. It follows that $\lambda'(t) \subseteq \lambda'(t^*)$. Applying standard techniques (see, e.g., [24]), we can remove the node t from T while preserving the tree decomposition properties. Iteratively applying this modification, we end up with a tree decomposition $(\tilde{T}, \tilde{\lambda})$ that witnesses the acyclicity of θ'.

Finally, we show that the core $\theta'(\bar{x})$ of a CQ $\theta(\bar{x})$ is a strong induced subquery. Indeed, it follows from well-known core properties [29] that conditions 1., 2., and 3. in the definition of strong induced subquery hold. We prove next that condition 4. also holds. Again by well-known properties of cores [29], we can assume without loss of generality that there is a *retract* h from θ to θ', i.e., a homomorphism from $\mathcal{D}_\theta$ to $\mathcal{D}'_\theta$ that is the identity over the variables $V_{\theta'}$ of θ'. Let $P = R(u_1, \ldots, u_m)$ be an atom in θ but not in θ'. Then it is the case that $R(h(u_1), \ldots, h(u_m))$ is an atom of

θ'. Since h is the identity in $V_P \cap V_{\theta'}$, we conclude that condition 4. holds. ∎

We also make use of the following proposition in our proof.

PROPOSITION 8.4. *Let $\theta(\bar{x})$ be a CQ. Then $\theta(\bar{x})$ is semantically acyclic if and only if its core is acyclic.*

Proof. The *if* direction is trivial since θ and its core are equivalent. For the *only if* direction, assume that θ is semantically acyclic. Then there is an acyclic UCQ $\Theta'(\bar{x}) = \bigvee_{1 \leq i \leq m} \theta'_i(\bar{x})$ such that $\theta(\bar{x}) \equiv \Theta'(\bar{x})$. It then follows from Proposition 8.2 that there is some $i \in \{1, \ldots, m\}$ for which it is the case that $\theta \equiv \theta'_i$. Since θ'_i is acyclic, we have from Proposition 8.3 that its core is also acyclic. On the other hand, it is known (see, e.g., [29]) that equivalent CQs have the same core (up to renaming of variables). It follows that the core of θ coincides with the core of θ'_i (up to renaming of variables), and, therefore, it is acyclic. ∎

Now we turn to our initial claim: If $\Theta(\bar{x})$ is a semantically acyclic UCQ, then it is equivalent to an acyclic UCQ $\Theta'(\bar{x})$ with $|\Theta'| \leq |\Theta|$. Assume that $\Theta(\bar{x})$ is semantically acyclic and let $\Theta'(\bar{x})$ be an equivalent acyclic UCQ. Further, let Θ_{min} be a subset of the disjuncts in Θ such that (1) Θ_{min} is equivalent to Θ, and (2) no proper subset of Θ_{min} is equivalent to Θ. Analogously, we define Θ'_{min} to be "minimally" equivalent to Θ'. By minimality, there are no distinct disjuncts θ_1 and θ_2 in Θ_{min} such that $\theta_2 \subseteq \theta_1$. From Proposition 8.2 and the fact that $\Theta_{min} \equiv \Theta'_{min}$, it follows that, for each disjunct θ of Θ_{min}, there is an equivalent disjunct θ' in Θ'_{min}. Thus, each disjunct of Θ_{min} is semantically acyclic. Let Θ^* be the UCQ obtained from Θ_{min}, by replacing each disjunct by its core. From Proposition 8.4, we have that Θ^* is acyclic. Moreover, $\Theta^* \equiv \Theta$ and $|\Theta^*| \leq |\Theta|$. This completes the proof of the NP upper bound.

The lower bound follows from [17], which shows that the problem of checking whether a Boolean CQ over a single binary relation is equivalent to an acyclic one is NP-hard.

Proof of Lemma 5.7. We start introducing some terminology. A *two-way nondeterminisitic finite automaton* (2NFA) [30, 42] is a tuple $\mathcal{B} = (\Sigma, Q, Q_0, \delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq S$ is the set of initial states, $\delta : Q \times \Sigma \to 2^{Q \times \{-1,0,1\}}$ is the transition function, and $F \subseteq Q$ is the set of final states. Intuitively, a transition indicates both the new state of the automaton and whether the head should move left (-1), move right (1), or stay in place (0). A *configuration* of $\mathcal{B}$ is a pair (q, j) consisting of a state $q \in Q$ and a position represented as an integer $j \geq 0$. The sequence $(q_0, j_0), \ldots, (q_m, j_m)$ of configurations of $\mathcal{B}$ is an *accepting run* of $\mathcal{B}$ on a word $w = a_0 \ldots a_{\ell-1} \in \Sigma^*$, for $\ell \geq 0$, if (i) $q_0 \in Q_0$, (ii) $j_0 = 0$, (iii) $q_m \in F$, (iv) $j_m = \ell$, and (v) for each $i \in \{0, \ldots, m-1\}$ we have that $0 \leq j_i < \ell$ and there is some $(q', d) \in \delta(q_i, a_{j_i})$ such that $q_{i+1} = q'$ and $j_{i+1} = j_i + d$. The 2NFA $\mathcal{B}$ *accepts* w if it has an accepting run on w. The following fact is well known [42].

PROPOSITION 8.5. *Given a 2NFA with n states, one can construct an equivalent NFA with $O(2^{n \log n})$ states.*

Assume that $\mathcal{A}_1, \ldots, \mathcal{A}_m$ is an enumeration of all the NFAs over $\Sigma^{\pm}$ mentioned in Γ. For the Γ-type π and $1 \leq i \leq 4$, we denote by $\pi(i)$ the ith coordinate of π. Let $1 \leq j \leq m$ and assume that $\mathcal{A}_j = (\Sigma^{\pm}, Q, Q_0, \delta, F)$, where δ is a transition function of the form $Q \times \Sigma \to 2^Q$, and that q and q' are states of $\mathcal{A}_j$. We then define a 2NFA $\mathcal{B}_{it}(j, q, q')$ that accepts the following language:

$$\{z \in (\Sigma^{\pm})^* \mid (\mathcal{A}_j, q, q') \in \pi_z(1), \text{ where } \pi_z \text{ is the } \Gamma\text{-type of } z\}.$$

Intuitively, $\mathcal{B}_{it}(j,q,q')$ guesses a folding from the first to the last position of z and verifies that such folding can be read from state q to q' in $\mathcal{A}_j$ (as this implies, by definition, that $(\mathcal{A}_j,q,q')$ belongs to $\pi_z(1)$). It is not hard to see that $\mathcal{B}_{it}(j,q,q')$ can be constructed in such a way that its size is bounded by $O(|\mathcal{A}_j|)$, and thus by $O(|\Gamma|)$.

Similarly, we define a 2NFA $\mathcal{B}_{ti}(j,q,q')$ that accepts the following language:

$$\{z \in (\Sigma^{\pm})^* \mid (\mathcal{A}_j,q,q') \in \pi_z(2), \text{ where } \pi_z \text{ is the } \Gamma\text{-type of } z\}.$$

This NFA also guesses a folding from the first to the last position of z—which represents the "inverse" of a folding from the last position of z to the first one—and then verifies that such folding can be read "backwards" from state q' to q in $\mathcal{A}_j$ (as this implies, by definition, that $(\mathcal{A}_j,q,q')$ belongs to $\pi_2(z)$). As before, the 2NFA $\mathcal{B}_{ti}(j,q,q')$ can be easily constructed in such a way that its size is bounded by $O(|\mathcal{A}_j|)$, and thus by $O(|\Gamma|)$.

Let $\mathcal{A}_{it}(j,q,q')$ and $\mathcal{A}_{ti}(j,q,q')$ be the NFAs over $\Sigma^{\pm}$ which are equivalent to the 2NFAs $\mathcal{B}_{it}(j,q,q')$ and $\mathcal{B}_{ti}(j,q,q')$, respectively, according to Proposition 8.5. Thus, the number of states in $\mathcal{A}_{it}(j,q,q')$ and $\mathcal{A}_{ti}(j,q,q')$ is bounded by $O(2^{s(|\Gamma|)})$, for a polynomial $s : \mathbb{N} \to \mathbb{N}$.

We now explain how to construct an NFA $\mathcal{A}_{ii}(j,q,q')$ which accepts the following language:

$$\{z \in (\Sigma^{\pm})^* \mid (\mathcal{A}_j,q,q') \in \pi_z(3), \text{ where } \pi_z \text{ is the } \Gamma\text{-type of } z\}.$$

Notice that if $z = a_0 \dots a_{\ell-1}$, where each a_k is a symbol in $\Sigma^{\pm}$, then the triple $(\mathcal{A}_j,q,q')$ belongs to $\pi_z(3)$ if and only if there is a state p of $\mathcal{A}_j$ and a position $0 \leq k \leq \ell-1$ such that $(\mathcal{A}_j,q,p) \in \pi_{z_k}(1)$ and $(\mathcal{A}_j,p,q') \in \pi_{z_k}(2)$, where for each $0 \leq k \leq \ell-1$ we have that $z_k = a_0 \dots a_k$. Then for each state p of $\mathcal{A}_j$ we define an NFA $\mathcal{A}_{ii}(j,q,p,q')$ which accepts the following language:

$$\{z \in (\Sigma^{\pm})^* \mid (\mathcal{A}_j,q,p) \in \pi_{z'}(1) \text{ and } (\mathcal{A}_j,p,q') \in \pi_{z'}(2), \text{ for some prefix } z' \text{ of } z\}.$$

It is easy to define the NFA $\mathcal{A}_{ii}(j,q,p,q')$ by (i) taking the cross product between $\mathcal{A}_{it}(j,q,p)$ and $\mathcal{A}_{ti}(j,p,q')$, (ii) adding an ε-transition from each final state of this cross product to a fresh state f, (iii) adding a transition a from f to f, for each $a \in \Sigma^{\pm}$, and (iv) letting f be the unique final state of the resulting NFA. The NFA $\mathcal{A}_{ii}(j,q,q')$ is then defined as the union of all the NFAs in the set

$$\{\mathcal{A}_{ii}(j,q,p,q') \mid p \text{ is a state in } \mathcal{A}_j\}.$$

The number of states of $\mathcal{A}_{ii}(j,q,q')$ is $O(2^{s'(|\Gamma|)})$ for a polynomial $s' : \mathbb{N} \to \mathbb{N}$.

Using the previous idea it is then straightforward to construct an NFA $\mathcal{A}_{tt}(j,q,q')$ which accepts the following language:

$$\{z \in (\Sigma^{\pm})^* \mid (\mathcal{A}_j,q,q') \in \pi_z(4), \text{ where } \pi_z \text{ is the } \Gamma\text{-type of } z\}.$$

Again, the number of states of $\mathcal{A}_{tt}(i,q,q')$ is $O(2^{s'(|\Gamma|)})$.

In order to construct the NFA $\mathcal{A}_\pi$ which accepts the language $\mathcal{L}^{\subseteq}(\pi)$, we proceed as follows. We first define NFAs $\mathcal{A}_{it}, \mathcal{A}_{ti}, \mathcal{A}_{ii}$, and $\mathcal{A}_{tt}$ as the product of the NFAs in the sets

$$\{\mathcal{A}_{it}(j,q,q') \mid (\mathcal{A}_j,q,q') \in \pi(1),\ 1 \leq j \leq m\},$$
$$\{\mathcal{A}_{ti}(i,q,q') \mid (\mathcal{A}_j,q,q') \in \pi(2),\ 1 \leq j \leq m\},$$
$$\{\mathcal{A}_{ii}(j,q,q') \mid (\mathcal{A}_j,q,q') \in \pi(3),\ 1 \leq j \leq m\},$$
$$\{\mathcal{A}_{tt}(j,q,q') \mid (\mathcal{A}_j,q,q') \in \pi(4),\ 1 \leq j \leq m\},$$

respectively. Then we define $\mathcal{A}_\pi$ as the product of $\mathcal{A}_{it}, \mathcal{A}_{ti}, \mathcal{A}_{ii}$, and $\mathcal{A}_{tt}$. It is straightforward to check that the language accepted by $\mathcal{A}_\pi$ is precisely $\mathcal{L}^{\subseteq}(\pi)$. Further, the number of states of $\mathcal{A}_\pi$ is $O(2^{s''(|\Gamma|)})$, for a polynomial $s'' : \mathbb{N} \to \mathbb{N}$. Thus the size of $\mathcal{A}_\pi$ is at most exponential in $|\Gamma|$, as required.

Acknowledgment. We are very grateful to Gaelle Fontaine for helpful discussions about the role of unions in approximations of UC2RPQs, and for providing us with Example 7.

REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kauffman, Burlington, MA, 1999.

[2] R. Angles and C. Gutiérrez, *Survey of graph database models*, ACM Comput. Surv., 40 (2008), 1.

[3] P. Barceló, *Querying graph databases*, in Proceedings of the 32nd Symposium on Principles of Database Systems, PODS'13, ACM, New York, 2013, pp. 175–188.

[4] P. Barceló, L. Libkin, A. W. Lin, and P. Wood, *Expressive languages for path queries over graph-structured data*, ACM T. Database Syst., 37 (2012), 31.

[5] P. Barceló, J. Perez, and J. Reutter, *Relative expressiveness of nested regular expressions*, in Proceedings of the 6th Alberto Mendelzon International Workshop on Foundations of Data Management, AMW'12, 2012, pp. 180–195.

[6] P. Barceló, L. Libkin, and M. Romero, *Efficient approximations of conjunctive queries*, SIAM J. Comput. 43 (2014), pp. 1085–1130, doi:10.1137/130911731.

[7] P. Barceló, M. Romero, and M. Y. Vardi, *Semantic acyclicity on graph databases*, in Proceedings of the 32nd ACM Symposium on Principles of Database Systems, PODS'13, ACM, New York, 2013, pp. 237–248.

[8] A. Bulatov, V. Dalmau, M. Grohe, and D. Marx, *Enumerating homomorphisms*, J. Comput. System Sci., 78 (2012), pp. 638–650.

[9] D. Calvanese, G. de Giacomo, M. Lenzerini, and M. Y. Vardi, *Containment of conjunctive regular path queries with inverse*, in Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning, KR'00, 2000, pp. 176–185.

[10] D. Calvanese, G. de Giacomo, M. Lenzerini, and M. Y. Vardi, *Rewriting of regular expressions and regular path queries*, J. Comput. System Sci., 64 (2002), pp. 443–465.

[11] D. Calvanese, G. de Giacomo, M. Lenzerini, and M. Y. Vardi, *View-based query answering and query containment over semistructured data*, in Proceedings of the 8th International Workshop on Database Programming Languages, DBPL'02, Lecture Notes in Comput. Sci. 2397, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 40–61.

[12] A. K. Chandra and P. M. Merlin, *Optimal implementation of conjunctive queries in relational data bases*, in Conference Record of the 9th Annual ACM Symposium on Theory of Computing, STOC'77, ACM, New York, 1977, pp. 77–90.

[13] C. Chekuri and A. Rajaraman, *Conjunctive query containment revisited*, Theoret. Comput. Sci., 239 (2000), pp. 211–229.

[14] H. Chen and V. Dalmau, *Beyond hypertree width: Decomposition methods without decompositions*, in Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP'05, Lecture Notes in Comput. Sci. 3709, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 167–181.

[15] M. P. Consens and A. O. Mendelzon, *GraphLog: A visual formalism for real life recursion*, in Proceedings of the Ninth ACM Symposium on Principles of Database Systems, PODS'90, ACM New York, 1990, pp. 404–416.

[16] I. Cruz, A. O. Mendelzon, and P. T. Wood, *A graphical query language supporting recursion*, in Proceedings of the ACM SIGMOD Conference, SIGMOD'87, ACM, New York, 1987, pp. 323–330.

[17] V. Dalmau, Ph. G. Kolaitis, and M. Y. Vardi, *Constraint satisfaction, bounded treewidth, and finite-variable logics*, in Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, CP'02, Lecture Notes in Comput. Sci. 2470, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 310–326.

[18] R. Fagin, *Degrees of acyclicity for hypergraphs and relational database schemes*, J. ACM, 30 (1983), pp. 514–550.

[19] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu, *Graph pattern matching: From intractable to polynomial time*, Proceedings of the VLDB Endowment, 3 (2010), pp. 264–275, doi:10.14778/1920841.1920878.
[20] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu, *Adding regular expressions to graph reachability and pattern queries*, Front. Comput. Sci., 6 (2012), pp. 313–338.
[21] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104, doi:10.1137/S0097539794266766.
[22] G. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu, *Relative expressive power of navigational querying on graphs*, Inform. Sci., 298 (2015), pp. 390–406.
[23] D. Florescu, A. Levy, and D. Suciu, *Query containment for conjunctive queries with regular expressions*, in Proceedings of the 17th ACM Symposium on Principles of Database Systems, PODS'98, ACM, New York, 1998, pp. 139–148.
[24] J. Flum, M. Frick, and M. Grohe, *Query evaluation via tree-decompositions*, J. ACM, 49 (2002), pp. 716–752.
[25] G. Gottlob, N. Leone, and F. Scarcello, *The complexity of acyclic conjunctive queries*, J. ACM, 48 (2001), pp. 431–498.
[26] G. Gottlob, N. Leone, and F. Scarcello, *Hypertree decompositions and tractable queries*, J. Comput. System Sci., 64 (2002), pp. 579–627.
[27] G. Greco and F. Scarcello, *Structural tractability of enumerating CSP solutions*, Constraints, 18 (2013), pp. 38–74.
[28] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*, MIT Press, Cambridge, MA, 2000.
[29] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford University Press, Oxford, UK, 2004.
[30] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.
[31] Ph. G. Kolaitis and M. Y. Vardi, *On the expressive power of Datalog: Tools and a case study*, J. Comput. System Sci., 51 (1995), pp. 110–134.
[32] Ph. G. Kolaitis and M. Y. Vardi, *A logical approach to constraint satisfaction*, in The Book Complexity of Constraints: An Overview of Current Research Themes, N. Creignou, P. G. Kolaitis, and H. Vollmer, eds., Lecture Notes in Comput. Sci. 5250, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 125–155.
[33] Ph. G. Kolaitis and M. Y. Vardi, *Conjunctive query-containment and constraint satisfaction*, J. Comput. System Sci., 61 (2002), pp. 302–332.
[34] R. E. Ladner, R. J. Lipton, and L. J. Stockmeyer, *Alternating pushdown and stack automata*, SIAM J. Comput., 13 (1984), pp. 135–155, doi:10.1137/0213010.
[35] L. Libkin, W. Martens, and D. Vrgoc, *Querying graph databases with XPath*, in Proceedings of the 16th International Conference on Database Theory, ICDT'13, ACM, New York, 2013, pp. 129–140.
[36] Ch. H. Papadimitriou and M. Yannakakis, *The complexity of facets (and some facets of complexity)*, J. Comput. System Sci., 28 (1986), pp. 244–259.
[37] Ch. H. Papadimitriou and M. Yannakakis, *On the complexity of database queries*, in Proceedings of the 16th ACM Symposium on Principles of Database Systems, PODS'97, ACM, New York, 1997, pp. 12–19.
[38] J. Reutter, M. Romero, and M. Y. Vardi, *Regular queries on graph databases*, in Proceedings of the 18th International Conference on Database Theory, ICDT'15, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015, pp. 177–194.
[39] Y. Sagiv and M. Yannakakis, *Equivalences among relational expressions with the union and difference operator*, J. ACM, 27 (1980), pp. 633–655.
[40] R. Tarjan and M. Yannakakis, *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM J. Comput., 13 (1984), pp. 566–579, doi:10.1137/0213035.
[41] M. Y. Vardi, *The complexity of relational query languages (extended abstract)*, in STOC 1982, ACM, New York, 1982, pp. 137–146.
[42] M. Y. Vardi, *A note on the reduction of two-way automata to one-way automata*, Inform. Process. Lett., 30 (1989), pp. 261–264.
[43] P. T. Wood, *Query languages for graph databases*, SIGMOD Record, 41 (2012), pp. 50–60.
[44] M. Yannakakis, *Algorithms for acyclic database schemes*, in Proceedings of the 7th International Conference on Very Large Data Bases, 1981, pp. 82–94.