# Semantic Acyclicity on Graph Databases

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

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

Moshe Vardi
Department of Computer Science, Rice University
vardi@cs.rice.edu

## 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 the same holds for the class of UC2RPQs that are semantically acyclic. We prove that checking whether a UC2RPQ is semantically acyclic is decidable in 2EXPSPACE, and that it is EXPSPACE-hard even in the absence of inverses. Furthermore, we show 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.

## Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query Languages*

## Keywords

Graph databases, conjunctive regular path queries, acyclicity, query evaluation, query approximation.

## 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 $Q$, a database $\mathcal{D}$, and a tuple $\bar{a}$ of elements in $\mathcal{D}$, does $\bar{a}$ belong to the result $Q(\mathcal{D})$ of applying $Q$ 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 $|Q|$ of the query $Q$, which in database terminology corresponds to the *combined complexity* of the problem.

The evaluation problem for CQs is NP-complete [8]; this motivated a flurry of activity for finding tractable cases of the problem. One of the oldest and most important such restrictions is *acyclicity*. Yannakakis proved that acyclic CQs can in fact be evaluated in linear time in both data and query size – $O(|\mathcal{D}| \cdot |Q|)$ [31]. 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 [19]. 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?

The answers to these questions follow easily from known techniques in the area of *constraint satisfaction problems* (CSP), as CQ evaluation and CSP are known to have a common root – they are both equivalent to the *homomorphism problem* [24]. 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 [9, 12].

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 [3, 14, 15, 4, 17]. In fact, acyclicity has been identified as a fundamental tool for obtaining tractable – and even linear time – query evaluation in such context [23, 3, 4, 26]. 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 to recursively traverse the edges of the graph while checking for some regular condition [11, 1, 7]. Navigation is often performed by traversing edges in both directions, which allows to express important properties about the inverse of the relations defined by the labels of those edges [6, 7]. 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*,


*PODS'13,* June 22–27, 2013, New York, New York, USA.

or C2RPQs – that lies at the core of many query languages for graph databases (see, e.g., [11, 10, 6, 3]).

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 [3, 4].

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 acylic 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 to recognize if a UC2RPQ $Q$ is semantically acyclic, and, if so, what is the cost of computing an equivalent acyclic UC2RPQ for $Q$?

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 of UC2RPQs, 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 question in the paper, in a way that our answer to the first question crucially depends on our answer to the second one.

As noted above, 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 well the semantics of CQs. 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 notion of homomorphism does not fit well in the current landscape of CSP techniques, which means that CSP theory does not yield answers to our questions about semantically acyclic UC2RPQs.

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 [5]. In general, the evaluation of a CQ $Q$ on a database $\mathcal{D}$ is of the order $|\mathcal{D}|^{O(|Q|)}$, which might be prohibitively expensive for a large dataset $\mathcal{D}$ even if $Q$ is small. This led the idea of finding *approximations* of (U)CQs in tractable classes [5], in particular, in the class of acyclic (U)CQs. Intuitively, an acyclic UCQ $Q'$ is an approximation of a UCQ $Q$ if $Q'$ is contained in $Q$ and it is "as close as possible" to $Q$ in the class of acyclic UCQs. The latter means that $Q'$ is a maximal acyclic UCQ that is contained in $Q$. It follows from techniques in [5] that UCQs have good properties in terms of acyclic approximations: Each UCQ $Q$ 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 $Q$ on a database $\mathcal{D}$ takes time $O(|\mathcal{D}| \cdot 2^{p(|Q|)})$, for some polynomial $p : \mathbb{N} \to \mathbb{N}$. This is much faster than $|\mathcal{D}|^{O(|Q|)}$ on large databases. Thus, if the quality of the approximation is good, we may prefer to run this faster query instead of $Q$.

We show here that the good properties of UCQs in terms of acyclic approximations extend to UC2RPQs. In particular, we show that each UC2RPQ $Q$ 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, such like the data complexity of 2RPQs. This shows that acyclic approximations might be useful when evaluating 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 $Q'$ is the approximation of the UCQ $Q$ is in the second-level of the polynomial hierarchy, but it becomes Expspace-complete if $Q$ and $Q'$ are UC2RPQs. This is not surprising, as it is known that testing containment of UC2RPQs is Expspace-complete [6].

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 $Q'$ of a given UC2RPQ $Q$. By construction, $Q'$ is contained in $Q$. To check whether $Q$ is semantically acyclic we just have to check if $Q$ is contained in $Q'$. Because $Q'$ is exponentially large, we get a 2Expspace upper bound for the complexity of the last step. We also prove an Expspace lower bound for the problem of checking semantic acyclicity. The precise complexity remains an open question.

Thus, we get answers to the two questions we posed above: (1) Query evaluation for semantically acyclic UC2RPQ is *fixed-parameter tractable.*[1] (2) Testing semantic acyclicity for UC2RPQs is in 2Expspace and Expspace-hard. The question whether semantically acyclic UC2RPQs can be evaluated in polynomial time is left as an 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 answer to the most basic questions follow from known CSP techniques. In Section 3, we introduce graph databases and UC2RPQs. In Section 4, we study acyclic approximations of UC2RPQs and show some of their good properties. Finally, in Section 5, we study semantic acyclicity of UC2RPQs. We provide upper and lower bounds for the problem of verfiying whether a UC2RPQ is semantically acyclic and show that this implies that evaluation of semantically acyclic UC2RPQs is fixed-parameter tractable. Finally, in Section 6 we provide concluding remarks and a list of open problems.

## 2. INTERLUDE ON UNIONS OF CONJUNCTIVE QUERIES

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 developing the necesssary intuitions for the more complicated case of graph databases and unions of conjunctive regular path queries.

[1]Recall that the evaluation problem for a class $\mathcal{C}$ of queries is fixed-parameter tractable if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant $k \geq 1$ such that evaluating a query $Q \in \mathcal{C}$ over a database $\mathcal{D}$ can be done in time $O(|\mathcal{D}|^k \cdot f(|Q|))$.

## 2.1 Basic concepts

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

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

$$Q(\bar{x}) \;=\; \exists \bar{y} \bigwedge_{j=1}^{m} R_{i_j}(\bar{x}_{i_j}),$$

where each $R_{i_j}$ is a symbol from $\sigma$, and $\bar{x}_{i_j}$ is a tuple of variables among $\bar{x}, \bar{y}$ whose length is the arity of $R_{i_j}$. Each $R_{i_j}(\bar{x}_{i_j})$ is an *atom* of $Q(\bar{x})$.

A *union* of conjunctive queries (UCQ) is a formula of the form $Q(\bar{x}) = \bigvee_{1 \leq i \leq m} Q_i(\bar{x})$, where $Q_i(\bar{x})$ is a CQ for each $1 \leq i \leq m$. We assume familiarity with the semantics (evaluation) of (U)CQs. The set of tuples that belong to the evaluation of a UCQ $Q$ over database $\mathcal{D}$ is denoted by $Q(\mathcal{D})$. If $Q$ 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.

The evaluation problem for UCQs is as follows: Given a database $\mathcal{D}$, a UCQ $Q(\bar{x})$ and a tuple $\bar{a}$ in $\mathcal{D}$, is $\bar{a} \in Q(\mathcal{D})$? It is well-known that the evaluation of CQs is NP-complete [8]. 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 *$\alpha$-acyclicity* (or, simply, acyclicity) [13], that can be defined in terms of the existence of a well-behaved *tree decomposition* of the *hypergraph* of a CQ [20]. 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 we associate its hypergraph $\mathcal{H}(Q) = (V, E)$ such that $V$ is the set of variables of $Q$ and $E$ consists of all sets of variables that appear in the same atom of $Q$. Consider for instance the CQ

$$Q(x) = \exists y \exists z \exists u \exists v \left(R(x, y, z) \wedge T(y, u, u) \wedge S(y, v)\right).$$

Then $\mathcal{H}(Q) = (V, E)$, 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 $(T, \lambda)$, where $T$ is a tree and $\lambda : T \to 2^V$, 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 $(T, \lambda)$ of it such that $\lambda(t)$ is a hyperedge in $E$, for each $t \in T$.

A CQ $Q$ is acyclic if its hypergraph $\mathcal{H}(Q)$ is acyclic. A UCQ $\bigvee_{1 \leq i \leq m} Q_i(\bar{x})$ is acyclic if each $Q_i(\bar{x})$ is acyclic ($1 \leq i \leq m$). For instance, the CQ $Q(x) = \exists y \exists z \exists u \exists v\, R(x, y, z) \wedge T(y, u, u) \wedge S(y, v)$ presented above is acyclic, as witnessed by the following tree decomposition $(T, \lambda)$ of $\mathcal{H}(Q)$: $T$ 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 the evaluation problem for acyclic UCQs can be solved in linear time $O(|\mathcal{D}| \cdot |Q|)$ [31].

## 2.2 Semantically acyclic UCQs

Acyclicity is a syntactic property of UCQs. On the other hand, a non-acyclic UCQ can still be equivalent to an acyclic one. Formally, a UCQ $Q(\bar{x})$ is *semantically acyclic* if there exists an acyclic UCQ $Q'(\bar{x})$ such that $Q(\mathcal{D}) = Q'(\mathcal{D})$ 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 whether semantic acyclicity for UCQs?

As first pointed out in [24], there is a close connection between conjunctive query evaluation and constraint satisfaction: Both can be recasted 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 [24, 9] and prove that semantically acyclic UCQs can be evaluated in polynomial time.

THEOREM 1. *The evaluation problem for semantically acyclic UCQs can be solved in polynomial time.*

The CSP techniques that imply Theorem 1 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.

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 [30]. On the other hand, using techniques similar to those in [12], 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:

PROPOSITION 1. *The problem of verifying whether a UCQ $Q(\bar{x})$ is semantically acyclic is* NP*-complete. It remains* NP*-hard even if the input is restricted to Boolean CQs whose schema consists of a single binary relation (i.e. the schema of directed graphs).*

# 3. GRAPH DATABASES & CONJUNCTIVE REGULAR PATH QUERIES

**Graph databases and C2RPQs** A *graph database* [2, 7, 11] is just a finite edge-labeled graph. Let $\Sigma$ be a finite alphabet, and $\mathcal{N}$ a countably infinite set of node ids. Then a graph database over $\Sigma$ 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 $\Sigma$ 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. This defines the notion of *regular path queries with inverse* (2RPQs) [6, 7]: A 2RPQ over finite alphabet $\Sigma$ is an expression $R$ defined by the grammar:

$$R \;:=\; \varepsilon \;\mid\; a\,(a \in \Sigma) \;\mid\; a^-\;(a \in \Sigma) \;\mid\; R + R \;\mid\; R \cdot R \;\mid\; R^*$$

That is, a 2RPQ is nothing else than a regular expression over the alphabet that extends $\Sigma$ with the symbol $a^-$ for each $a \in \Sigma$. Intuitively, $a^-$ represents the inverse of $a$.

The evaluation of a 2RPQ $R$ over a graph database $\mathcal{G} = (N, E)$ is a binary relation $[\![R]\!]_\mathcal{G}$ defined as follows, where $a$ is a symbol in $\Sigma$ and $R$, $R_1$ and $R_2$ are arbitrary 2RPQs over $\Sigma$:

$$\begin{aligned}
[\![\varepsilon]\!]_\mathcal{G} &= \{(u,u) \mid u \in N\} \\
[\![a]\!]_\mathcal{G} &= \{(u,v) \mid (u,a,v) \in E\} \\
[\![a^-]\!]_\mathcal{G} &= \{(v,u) \mid (u,a,v) \in E\} \\
[\![R_1 + R_2]\!]_\mathcal{G} &= [\![R_1]\!]_\mathcal{G} \cup [\![R_2]\!]_\mathcal{G} \\
[\![R_1 \cdot R_2]\!]_\mathcal{G} &= [\![R_1]\!]_\mathcal{G} \circ [\![R_2]\!]_\mathcal{G} \\
[\![R^*]\!]_\mathcal{G} &= [\![\varepsilon]\!]_\mathcal{G} \cup [\![R]\!]_\mathcal{G} \cup [\![R \cdot R]\!]_\mathcal{G} \cup \cdots
\end{aligned}$$

Here, the symbol $\circ$ denotes the usual composition of binary relations, that is, $[\![R_1]\!]_\mathcal{G} \circ [\![R_2]\!]_\mathcal{G} = \{(u,v) \mid$ there exists $w$ s.t. $(u,w) \in [\![R_1]\!]_\mathcal{G}$ and $(w,v) \in [\![R_2]\!]_\mathcal{G}\}$. As expected, the expression $a^-$ ($a \in \Sigma$) defines the inverse of the expression $a$ on each graph database. Intuitively, each matching of $a^-$ in $\mathcal{G}$ represents a backward traversal of an $a$-labeled edge.

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 – that we define next.

Formally, a C2RPQ over a finite alphabet $\Sigma$ is an expression of the form:

$$Q(\bar{x}) = \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, R_i, v_i), \tag{1}$$

such that each $u_i$ and $v_i$ is a variable among $\bar{x}, \bar{y}$ and each $R_i$ is a 2RPQ over $\Sigma$, for $1 \leq i \leq m$. A CRPQ is a C2RPQ without inverses, i.e. a C2RPQ of the form (1) in which no 2RPQ $R_i$ ($1 \leq i \leq m$) mentions the inverse $a^-$ of a symbol $a \in \Sigma$. As usual, we assume that $\bar{x}$ are the *free* variables of $Q$, i.e. the variables mentioned in $Q$ that are not existentially quantified.

Intuitively, a C2RPQ $Q(\bar{x})$ of the form (1) selects tuples $\bar{x}$ for which there exist values of the remaining node variables from $\bar{y}$ such that each pair $(u_i, v_i)$ belongs to the evaluation of the 2RPQ $R_i$. We formally define the semantics of C2RPQs by using a notion of homomorphism that maps atoms of $Q$ into pairs that satisfy the corresponding 2RPQs.

Given $Q(\bar{x})$ of the form (1) and a graph database $\mathcal{G} = (N, E)$, a homomorphism from $Q(\bar{x})$ to $\mathcal{G}$ is a map $h : \bigcup_{1 \leq i \leq m} \{u_i, v_i\} \to N$ such that $(h(u_i), h(v_i)) \in [\![R_i]\!]_\mathcal{G}$ for every $1 \leq i \leq m$. Then $Q(\mathcal{G})$ is the set of all tuples $h(\bar{x})$ such that $h$ is a homomorphism from $Q(\bar{x})$ to $\mathcal{G}$.

Example 1. Consider a graph database $\mathcal{G} = (N, E)$ of researchers, papers, conferences and journals over the alphabet $\Sigma = \{\texttt{creator}, \texttt{inJournal}, \texttt{inConf}\}$. The set of edges $E$ consists of the following:

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

Consider the C2RPQ $Q(x, y)$ defined as

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

Its evaluation over $\mathcal{G}$ 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 $Q'(x, y)$ defined as

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

consists of the set of pairs $(r, r')$ of researchers that are linked by a coauthorhsip sequence and such that $r'$ has at least one journal paper. □

A union of C2RPQs (UC2RPQ) is a formula of the form $\bigvee_{1 \leq i \leq m} Q_i(\bar{x})$, where each $Q_i(\bar{x})$ is a C2RPQ ($1 \leq i \leq m$). We define $Q(\mathcal{G})$ as $\bigcup_{1 \leq i \leq m} Q_i(\mathcal{G})$, for a graph database $\mathcal{G}$. If $Q$ and $Q'$ are UC2RPQs, then $Q$ is *contained* in $Q'$, denoted $Q \subseteq Q'$, if $Q(\mathcal{G}) \subseteq Q'(\mathcal{G})$ for each graph database $\mathcal{G}$. In addition, $Q$ and $Q'$ are equivalent, denoted $Q \equiv Q'$, if $Q \subseteq Q'$ and $Q' \subseteq Q$, or, equivalently, $Q(\mathcal{G}) = Q'(\mathcal{G})$ for each graph database $\mathcal{G}$.

The evaluation problem for UC2RPQs is defined in the same way as UCQs. That is, given a UC2RPQ $Q(\bar{x})$, a graph database $\mathcal{G}$, and a tuple $\bar{a}$ of node ids in $\mathcal{G}$ such that $|\bar{a}| = |\bar{x}|$, does $\bar{a}$ belong to $Q(\mathcal{G})$? It is known that evaluating UC2RPQs is not more expensive than evaluating CQs, i.e. NP-complete [3].

**Acyclic C2RPQs** Acyclicity of C2RPQs has been studied in several recent papers that define it in terms of the acyclicity of its *underlying conjunctive query* [3, 4]. Let $Q(\bar{x}) = \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, R_i, v_i)$ 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 $Q$ when the regular languages that label the atoms of $Q$ 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 [31, 3].

Theorem 2. *The problem of checking whether $\bar{a} \in Q(\mathcal{G})$, for a given graph database $\mathcal{G}$, acyclic UC2RPQ $Q$, and a tuple $\bar{a}$ of node ids in $\mathcal{G}$, can be solved in time $O(|\mathcal{G}|^2 \cdot |Q|^2)$.*

Recall that a CQ $Q$ is acyclic if its hypergraph $\mathcal{H}(Q)$ admits a tree decomposition $(T, \lambda)$ such that each set of the form $\lambda(t)$, for $t \in T$, is a hyperdge of $\mathcal{H}(Q)$. 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 a C2RPQ $Q(\bar{x}) = \exists \bar{y} \bigwedge_{1 \leq i \leq m} (u_i, R_i, v_i)$, denoted $\mathcal{U}(Q)$, is the graph whose vertices are the variables of $Q$ and its set of edges is $\{\{u_i, v_i\} \mid 1 \leq i \leq m$ and $u_i \neq v_i\}$. Notice that $\mathcal{U}(Q)$ is indeed simple (it contains neither loops nor multiedges) and undirected. Then:

PROPOSITION 2. *A C2RPQ $Q$ is acyclic if and only if $\mathcal{U}(Q)$ is acyclic.*

EXAMPLE 2. Both C2RPQs $Q(x,y)$ and $Q'(x,y)$ in Example 1 are acyclic. Recall that $Q(x,y)$ is defined as

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

and $Q'(x,y)$ is defined as

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

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

EXAMPLE 3. Let $L_1$, $L_2$ and $L_3$ be arbitrary regular expressions over $\Sigma$. The CRPQs $Q = \exists x\,(x, L_1, x)$ and $Q' = \exists x \exists y\,\big((x, L_1, y) \wedge (y, L_2, x)\big)$ are acyclic. Notice that the underlying CQ of $Q$ contains a loop, while the underlying CQ of $Q'$ contains edges from $x$ to $y$ and from $y$ to $x$. The CRPQ $Q'' = \exists x \exists y \exists z\,\big((x, L_1, y) \wedge (y, L_2, z) \wedge (z, L_3, x)\big)$ is not acyclic. $\square$

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 problem of UC2RPQ approximation.

# 4. 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 [28]. 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 [5], in particular, in the class of acyclic (U)CQs. Intuitively, an acyclic UCQ $T$ is an approximation of a UCQ $Q$ if the following holds: (1) $T$ is contained in $Q$ (i.e. $T$ returns no false positives with respect to $Q$) and (2) $T$ is "as close as possible" to $Q$ among all acyclic UCQs.

It follows from results and techniques in [5] 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 approximation is unique (up to equivalence) and of at most exponential size.

2. Second, for each UCQ $Q$, its acyclic approximation $T$ can be computed in single-exponential time.

3. Third, verifying whether an acyclic UCQ $T$ is an approximation of a UCQ $Q$ is decidable in the second-level of the polynomial hierarchy.

These good properties imply that computing and running the acyclic approximation of a UCQ $Q$ on a database $\mathcal{D}$ takes time $O(2^{p(|Q|)} + |\mathcal{D}| \cdot 2^{r(|Q|)})$, for polynomials $p, r : \mathbb{N} \to \mathbb{N}$, which is $O(|\mathcal{D}| \cdot 2^{s(|Q|)})$, for a polynomial $s : \mathbb{N} \to \mathbb{N}$. This is much faster than $|\mathcal{D}|^{O(|Q|)}$ on large databases. Thus, if the quality of the approximation is good, we may prefer to run this faster query instead of $Q$.

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

## 4.1 Approximations: Existence and computation

Suppose we want to approximate a UC2RPQ $Q$ in the class $\mathsf{AC}$ 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 $Q$:

DEFINITION 1. **(Approximations)** *Let $Q$ and $Q'$ be UC2RPQs such that $Q' \in \mathsf{AC}$ and $Q' \subseteq Q$. Then $Q'$ is an* approximation *of $Q$ if there is no query $Q'' \in \mathsf{AC}$ with $Q'' \subseteq Q$ such that $Q' \subsetneq Q''$.*

It is worth noticing that the definition of approximations in [5] 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 approximation is unique. We can prove that this is also true for the class of UC2RPQs.

THEOREM 3. *Each UC2RPQ has a unique acyclic approximation (up to equivalence).*

As a corollary to the proof of Theorem 3 we get the following important result about the computation and size of approximations.

COROLLARY 1. *There exists an* EXPSPACE *algorithm that takes as input a UC2RPQ $Q$ and computes the approximation $Q'$ of $Q$. This approximation is of at most exponential size.*

Corollary 1 implies that approximations of UC2RPQs are meaningful. In fact, computing and running the acyclic approximation of a UC2RPQ $Q$ on a graph database $\mathcal{G}$ takes time

$$O\Big(2^{2^{p(|Q|)}} + |\mathcal{G}|^2 \cdot 2^{r(|Q|)}\Big),$$

for polynomials $p, r : \mathbb{N} \to \mathbb{N}$, which is

$$O\Big(|\mathcal{G}|^2 \cdot 2^{2^{p(|Q|)}}\Big).$$

In terms of data complexity this is only $O(|\mathcal{G}|^2)$, such like the data complexity of 2RPQs. This is much faster than $|\mathcal{G}|^{O(|Q|)}$ – the order of the evaluation problem for $Q$ 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 if a query is an approximation of another. We mentioned above that checking whether an acyclic UCQ $T$ is an approximation of a UCQ $Q$ can be solved in the second-level of the polynomial hierarchy [5]; 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 [27]. This problem is considerably harder for UC2RPQs:

PROPOSITION 3. *Let $Q$ and $T$ be UC2RPQs such that $T \in \mathsf{AC}$. The problem of verifying whether $T$ is an acyclic approximation of $Q$ is* EXPSPACE*-complete.*

We prove Theorem 3, Corollary 1 and Proposition 3 in the following section.

## 4.2 Proofs of results

All results in Section 4.1 follow from an important lemma that states that there exists an EXPSPACE algorithm that, on input a UC2RPQ $Q$, computes an acyclic UC2RPQ $T_Q$ – of at most exponential size – that is a maximum of the class of acyclic UC2RPQs that are contained in $Q$. This lemma will also be crucial for proving decidability of the notion of semantic acyclicity for UC2RPQs in Section 5:

LEMMA 1. *There exists an* EXPSPACE *algorithm that given a UC2RPQ $Q$ computes an acyclic UC2RPQ $T_Q$ such that:*

1. *$T_Q \subseteq Q$.*
2. *For every acyclic UC2RPQ $T$ such that $T \subseteq Q$ it is the case that $T \subseteq T_Q$.*
3. *The size of $T_Q$ is at most exponential in $|Q|$.*

Before proving Lemma 1 we show how the results in Section 4.1 easily follow from it.

*Proofs of Theorem 3, Corollary 1 and Proposition 3:* The algorithm in Lemma 1 computes for each UC2RPQ $Q$ a query $T_Q$ that is the maximum of the class of acyclic UC2RPQs that are contained in $Q$. We conclude that $T_Q$ is an approximation of $Q$. This approximation must be unique (up to equivalence) by definition.

In order to prove Corollary 1, we use the algorithm in Lemma 1 to compute the approximation $T_Q$ of a UC2RPQ $Q$. The algorithm runs in EXPSPACE and its output $T_Q$ is of at most exponential size in $|Q|$.

Finally, we prove Proposition 3. Given a UC2RPQ $Q$ and $T \in \mathsf{AC}$, the following algorithm verifies whether $T$ is an approximation of $Q$: Check whether $T \subseteq Q$ and $T_Q \subseteq T$. If this is the case $T \equiv T_Q$ and, thus, $T$ is an approximation of $Q$. Using techniques in [6] both containments can be performed in EXPSPACE. The lower bound follows by an easy reduction from the containment problem of an acyclic CRPQ $T$ in a CRPQ $Q$, which is EXPSPACE-complete [6]. □

We prove Lemma 1 now. We only sketch the main ideas since the complete proof is rather technical.

*Proof (Sketch) of Lemma 1:* The proof is based on the following claim:

CLAIM 1. *There exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every UC2RPQ $Q$ there exists a set $\mathcal{R}_Q$ of 2RPQs for which the following holds:*

1. *$\mathcal{R}_Q$ can be constructed in time $O(2^{p(|Q|)})$.*
2. *There is an acyclic UC2RPQ $T \subseteq Q$ such that each disjunct in $T$ is of the form $\exists \bar{y} \bigwedge_{1 \leq i \leq m}(u_i, R_i, v_i)$, for $m \leq p(|Q|)$ and $R_i \in \mathcal{R}_Q$ $(1 \leq i \leq m)$. Moreover, the union of all such $T$'s is $T_Q$, i.e., the maximum of the class of acyclic UC2RPQs that are contained in $Q$.*

We show first that Claim 1 implies Lemma 1. In fact, in order to build $T_Q$ from $Q$ we have to do the following: Construct $\mathcal{R}_Q$ in exponential time, and then iterate through every acyclic C2RPQ $T$ with at most $p(|Q|)$ atoms, all of them labeled with 2RPQs in $\mathcal{R}_Q$, adding as a disjunct to $T_Q$ each such $T$ that satisfies $T \subseteq Q$. The second item of Claim 1 implies that $T_Q$ constructed in this way is nonempty (i.e. it contains at least one disjunct). Since each $T$ with at most $p(|Q|)$ atoms – all of them labeled with 2RPQs in $\mathcal{R}_Q$ – is of exponential size, and checking whether $T \subseteq Q$ can be done in EXPSPACE using techniques in [6], the whole procedure can be performed in EXPSPACE. Furthermore, the size of $T_Q$ is at most exponential in $|Q|$.

We continue with the proof of Claim 1. Let $Q$ be a UC2RPQ. We start by explaining how the set $\mathcal{R}_Q$ of 2RPQs that label the atoms of $T_Q$ is defined. The intuition is that we do not need more information in those labels than the *types* they describe with respect to the 2RPQs that label the atoms of $Q$. Formalizing such intuition requires the introduction of several concepts.

To start with, we need to define the notion of *folding*, which is very useful when dealing with 2RPQs [7]. Let $w = p_1 p_2 \cdots p_k$ be a word over alphabet $\Sigma' = \Sigma \cup \{a^- \mid a \in \Sigma\}$. For $p \in \Sigma'$ we define $p^- = a^-$ if $p = a$ and $a \in \Sigma$, and $p^- = a$ if $p = a^-$ and $a \in \Sigma$. Then the word $u = q_1 q_2 \cdots q_\ell$ over $\Sigma'$ *folds* into $w$ from $j_1$ to $j_2$, for $j_1, j_2 \in \{0, \ldots, k\}$, if there is a sequence $i_0, i_1, \ldots, i_\ell$ of positions in the set $\{0, \ldots, k\}$ such that:

- $i_0 = j_1$ and $i_\ell = j_2$, and
- for each $1 \leq j \leq \ell$ it is the case that $i_j = i_{j-1} + 1$ and $q_j = p_{i_j}$, or $i_j = i_{j-1} - 1$ and $q_j = p_{i_{j-1}}^-$.

Intuitively, $u$ folds into $w$ if $u$ can be read in $w$ by a two-way automaton that outputs symbol $p$, for $p \in \Sigma'$, each time that it is read from left-to-right, and symbol $p^-$, for $p \in \Sigma'$, each time that it is read from right-to-left. For instance, the word $abb^-a^-abb^-c$ folds into $abb^-c$ from 0 to 5.

Assume that $\{R_1, \ldots, R_p\}$ is the set of 2RPQs that label the atoms of $Q$. With each $R_i$ we associate an NFA $\mathcal{A}_i$ over $\Sigma'$ that accepts the language defined by $R_i$. The *$Q$-type of a word $w$* over $\Sigma'$ is the tuple

$$\Gamma \;=\; (\tau_{it}, \tau_{ti}, \tau_{ii}, \tau_{tt}),$$

such that $\tau_{it}$ corresponds to the set of triples of the form $(\mathcal{A}_i, s, t)$, for $1 \leq i \leq p$, for which the following holds: (i) $s$ and $t$ are states of $\mathcal{A}_i$, and (ii) there is a word $u$ that folds into $w$ from position 0 to $|w|$ and a run of $\mathcal{A}_i$ from state $s$ to $t$ over $u$. Correspondingly, in $\tau_{ti}$ we ask $u$ to fold in $w$ from $|w|$ to 0, in $\tau_{ii}$ from 0 to 0, and in $\tau_{tt}$ from $|w|$ to $|w|$.

A *$Q$-type* is the $Q$-type of some word $w$. Given two $Q$-types $\Gamma_1$ and $\Gamma_2$, we say that $\Gamma_1$ is contained in $\Gamma_2$ if each coordinate of $\Gamma_1$ is contained in the corresponding coordinate of $\Gamma_2$. For a $Q$-type $\Gamma$, we denote by $\mathcal{W}(\Gamma)$ the set of words $w$ over $\Sigma'$ such that $\Gamma$ is contained in the $Q$-type of $w$. The next result is crucial for our proof:

LEMMA 2. *For each $Q$-type $\Gamma$, the language $\mathcal{W}(\Gamma)$ is regular and can be defined by a 2RPQ $\mathcal{R}_\Gamma$ of at most exponential size in $|Q|$.*

Let $\mathcal{R}^+$ be the set of all 2RPQs of the form $R_\Gamma$, for $\Gamma$ a $Q$-type. Since the class of 2RPQs is closed under *inverse*

(i.e. for each 2RPQ $R$ there is a 2RPQ $R^-$ such that $[\![R]\!]_{\mathcal{G}} = ([\![R^-]\!]_{\mathcal{G}})^{-1}$, for every graph database $\mathcal{G}$), we can also define the set of 2RPQs $\mathcal{R}^-$ that contains each inverse of a 2RPQ in $\mathcal{R}^+$. Finally, we define a set $\mathcal{R}^\circ$ that consists of the concatenation of at most $k$ elements in $\mathcal{R}^+ \cup \mathcal{R}^-$, where $k$ is polynomially bounded by $|Q|$ (precise bounds are not given in order not to complicate the presentation).

The definition of $\mathcal{R}_Q$ is rather technical, but what is important for us is that it contains the whole set of 2RPQs in $\mathcal{R}^\circ$ and can be computed in exponential time:

LEMMA 3. *A set $\mathcal{R}_Q$ that contains each 2RPQ in $\mathcal{R}^\circ$ can be constructed in exponential time in $|Q|$.*

In order to prove Claim 1 we use the following important lemma:

LEMMA 4. *For every UC2RPQ $Q$ and acyclic UC2RPQ $T$ that is contained in $Q$, there is an acyclic UC2RPQ $T'$ with the following properties:*

1. *$T \subseteq T' \subseteq Q$,*
2. *each 2RPQ that labels an atom of a disjunct of $T'$ belongs to $\mathcal{R}_Q$, and*
3. *the number of atoms in each disjunct of $T'$ is polynomially bounded by $|Q|$.*

*In particular, $|T'|$ is at most exponential in $|Q|$.*

We show first that Lemma 4 implies Claim 1. The set $\mathcal{R}_Q$ can be constructed in exponential time, so we have to prove the following: (†) There is an acyclic UC2RPQ $T \subseteq Q$ such that each disjunct in $T$ is of the form $\exists \bar{y} \bigwedge_{1 \leq i \leq m}(u_i, R_i, v_i)$, where $m \leq p(|Q|)$ and $R_i \in \mathcal{R}_Q$, for each $1 \leq i \leq m$. (††) The UC2RPQ $T^*$ defined by the union of all such $T$'s is precisely $T_Q$, i.e. $T^*$ is the maximum acyclic UC2RPQ contained in $Q$.

Assume that $\bar{x} = (x_1, \ldots, x_r)$ is the tuple of free variables of the UC2RPQ $Q$. It is easy to see that the CRPQ $\bigwedge_{1 \leq j \leq r-1}(x_j, \varepsilon, x_{j+1}) \wedge \bigwedge_{a \in \Sigma}(x_1, a, x_1)$ is contained in $Q$, and, thus, from Lemma 4 we have (†), i.e. there is an acyclic UC2RPQ $T$ of the form $\exists \bar{y} \bigwedge_{1 \leq i \leq m}(u_i, R_i, v_i)$ such that (i) $m$ is polynomially bounded by $|Q|$, (ii) for each $1 \leq i \leq m$ it is the case that $R_i \in \mathcal{R}_Q$, and (iii) $T \subseteq Q$. We prove (††) next, i.e. that $T^* \equiv T_Q$. Clearly, $T^* \subseteq Q$ since each disjunct of $T^*$ is contained in $Q$. Let $T$ be an arbitrary acyclic UC2RPQ that is contained in $Q$. Then Lemma 4 tells us that $T$ is contained in some union of C2RPQs that is contained in $T^*$, and hence $T \subseteq T^*$. Thus, $T^*$ is the maximum of the class of acyclic UC2RPQs contained in $Q$, and hence $T^*$ is equivalent to $T_Q$.

Next we prove Lemma 4. Let $T$ be an acyclic UC2RPQ such that $T \subseteq Q$. We make use of the following lemma to prove Lemma 4:

LEMMA 5. *Let $\mathcal{G}$ be an arbitrary graph database and $\bar{a}$ a tuple of node ids in $T(\mathcal{G})$. Then there exists an acyclic C2RPQ $T_{(\mathcal{G},\bar{a})}$ such that:*

1. *$T_{(\mathcal{G},\bar{a})} \subseteq Q$ and $\bar{a} \in T_{(\mathcal{G},\bar{a})}(\mathcal{G})$,*
2. *each 2RPQ that labels an atom of $T_{(\mathcal{G},\bar{a})}$ belongs to $\mathcal{R}_Q$, and*
3. *the number of atoms in $T_{(\mathcal{G},\bar{a})}$ is polynomially bounded by $|Q|$.*

We start by proving that Lemma 5 implies Lemma 4. In fact, we can simply define $T'$ to be the UC2RPQ

$$\bigcup \{T_{(\mathcal{G},\bar{a})} \mid \mathcal{G} \text{ is a graph database and } \bar{a} \text{ a tuple in } T(\mathcal{G})\}.$$

From Lemma 5 each disjunct in $T'$ is contained in $Q$, and, thus, $T' \subseteq Q$. Furthermore, the same lemma implies that for every graph database $\mathcal{G}$ and tuple $\bar{a} \in T(\mathcal{G})$ it is the case that $\bar{a} \in T'(\mathcal{G})$. Thus, $T \subseteq T'$. Finally, each disjunct of $T'$ has a polynomial number of atoms and each one of its atoms is labeled by a 2RPQ in $\mathcal{R}_Q$.

The last step in the construction is proving Lemma 5, which is our technically more involved result. By known techniques we do not have to consider each graph database $\mathcal{G}$, but only those that are *canonical* for some disjunct $T_1$ of $T$ [18, 6]. Intuitively, these are the graph databases $\mathcal{G}'$ that can be constructed as follows. Each variable in $T_1$ is added as a node id to $\mathcal{G}'$, and then for each atom of the form $(u, R, v)$ in $T_1$ we build a single *semipath* (i.e. a path that traverses edges in both directions) of fresh node ids from $u$ to $v$ whose label satisfies $R$. Since a 2RPQ may accept an infinite number of strings, the space of canonical graph databases for $T$ is potentially infinite.

Let $\mathcal{G}$ be an arbitrary canonical graph database for $T$ and $\bar{a}$ a tuple in $T(\mathcal{G})$. Assume that $Q = \bigvee_{1 \leq i \leq l} Q_i$, where each $Q_i$ is a C2RPQ. Since $T \subseteq Q$ we have that $\bar{a} \in Q(\mathcal{G})$, and, thus, that $\bar{a} \in Q_i(\mathcal{G})$ for some $i \leq \ell$. This implies that there is a homomorphism $h$ from $Q_i(\bar{x})$ to $\mathcal{G}$ such that $h(\bar{x}) = \bar{a}$. Assume that $Q_i = \exists \bar{y} \bigwedge_{1 \leq i \leq m}(u_i, R_i, v_i)$. Then $h$ satisfies that for each $1 \leq i \leq m$ the pair $(h(u_i), h(v_i)) \in [\![R_i]\!]_{\mathcal{G}}$, or, equivalently, that there is a semipath $\rho_i$ in $\mathcal{G}$ from $h(u_i)$ to $h(v_i)$ whose label satisfies the 2RPQ $R_i$. We choose each $\rho_i$ to be of minimal length.

Let $\mathcal{I}$ be the set of node ids in $\mathcal{G}$ that are of the form $h(u)$, for some variable $u$ of $Q$, or the *least common ancestor* in $\mathcal{G}$ of two elements in $\mathcal{I}$ that are in the same connected component of $\mathcal{G}$. (This least common ancestor is well-defined since $\mathcal{G}$ "resembles" $T$, which is acyclic). In the search for an acyclic C2RPQ $T_{(\mathcal{G},\bar{a})}$, as defined in the statement of Lemma 5, we construct from $\mathcal{G}$ a graph database $\mathcal{G}'$ that contains as node id each element in $\mathcal{I}$. In addition, we connect $h(u_i)$ to $h(v_i)$ in $\mathcal{G}'$ by the semipath $\rho_i$, for each $1 \leq i \leq m$. However, since we construct $\mathcal{G}'$ in the search for the acyclic C2RPQ $T_{(\mathcal{G},\bar{a})}$, this last step has to be carried out carefully in order to avoid undesirable cycles. We explain the process below.

The idea is to avoid cycles among elements in $\mathcal{I}$. In order to do that, we first have to define a notion of adjacency between elements of $\mathcal{I}$ that defines an acylic graph, and then add semipath $\rho_i$ from $h(u_i)$ to $h(v_i)$ in $\mathcal{G}'$, $1 \leq i \leq m$, by joining contiguous elements of $\mathcal{I}$ with respect to this adjacency relationship.

Finally, from $\mathcal{G}'$ we construct the C2RPQ $T_{(\mathcal{G},\bar{a})}$ as follows: For each pair of contiguous elements in $\mathcal{I}$ that are linked in $\mathcal{G}'$ by a semipath labeled by word $w \in \Sigma'$, we replace such semipath by the 2RPQ in $\mathcal{R}^\circ$ that completely describes $w$ with respect to $Q$. Clearly, each atom in $T_{(\mathcal{G},\bar{a})}$ is labeled by a 2RPQ in $\mathcal{R}_Q$. It can also be proved that $T_{(\mathcal{G},\bar{a})}$ is acyclic, and, further, that $\bar{a} \in T_{(\mathcal{G},\bar{a})}(\mathcal{G})$. The latter is not hard to see since the identity mapping is a homomorphism from $T_{(\mathcal{G},\bar{a})}$ in $\mathcal{G}$. Furthermore, the number of atoms in $T_{(\mathcal{G},\bar{a})}$ is polynomially bounded by $|Q|$. This follows from the fact

that $|\mathcal{I}|$ is polynomially bounded and the minimality of the semipaths $\rho_i$ that have been chosen to populate $\mathcal{G}'$ (as each path $\rho_i$ is decomposed in a polynomial number of contiguous segments in $\mathcal{G}'$).

It just rests to show that $T_{(\mathcal{G},\bar{a})}$ is contained in $Q$. But this can be intuitively explained using the facts that $\bar{a} \in Q(\mathcal{G})$, and the canonical databases of $T_{(\mathcal{G},\bar{a})}$ are indistinguishable from $\mathcal{G}$ by $Q$, i.e. for each canonical database $\mathcal{G}_1$ of $T_{(\mathcal{G},\bar{a})}$ it is the case that $\bar{a} \in Q(\mathcal{G}) \Leftrightarrow \bar{a} \in Q(\mathcal{G}_1)$. The latter holds because $T_{(\mathcal{G},\bar{a})}$ is constructed from $\mathcal{G}$ by replacing semipaths with 2RPQs that respect the $Q$-type of its label.

# 5. 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.

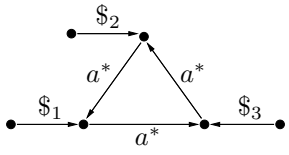
## 5.1 Basic terminology and insights

A UC2RPQ $Q$ is semantically acyclic if there exists an acyclic UC2RPQ $Q'$ such that $Q \equiv Q'$. 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 if 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.

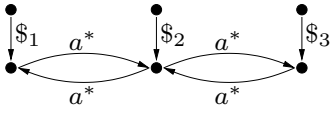
Since acyclicity of C2RPQs is defined in terms of the acyclicity of its underlying CQ, one may be tempted to think that the two notions coincide. Clearly, if the underlying CQ of a C2RPQ $Q$ is semantically acyclic then $Q$ is also semantically acyclic. The following example shows that the opposite does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

EXAMPLE 4. Consider again the non-acyclic CRPQ $Q'' = \exists x \exists y \exists z \big((x, L_1, y) \wedge (y, L_2, z) \wedge (z, L_3, x)\big)$ in Example 3. It is not hard to prove that $Q''$ is equivalent to the acyclic CRPQ $\exists x\, (x, L_1 L_2 L_3, x)$, and, thus, it is semantically acyclic. On the other hand, the underlying CQ of $Q''$ 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 $Q''$ is semantically acyclic because it can be "simplified" by concatenating the regular languages that label its atoms. A more interesting example is the following Boolean CRPQ $Q_{\text{sa}}$ over $\Sigma = \{a, \$_1, \$_2, \$_3\}$




(Dots represent variables and arrows represent labeled atoms). It is easy to see that the underlying CQ of $Q_{\text{sa}}$ is not semantically acyclic. On the other hand, it can be proved that $Q_{\text{sa}}$ is equivalent to the acyclic CRPQ



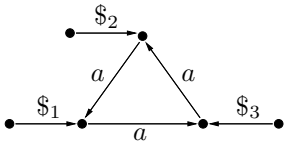

In this case, semantic acyclicity is obtained by the way in which the regular languages that label the atoms of $Q_{\text{sa}}$ interact with each other. □

The previous example shows that the notion of semantic acyclicity of C2RPQs is richer than the notion of 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 5.2 help us proving, on the other hand, that the problem is fixed-parameter tractable (Theorem 5), 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 5. Let $\Sigma = \{a, \$_1, \$_2, \$_3\}$ be a finite alphabet and consider the Boolean CRPQ $Q_{\text{na}}$ over $\Sigma$ that is graphically defined as:




Notice that the underlying CQ of $Q_{\text{na}}$ coincides with that of the semantically acyclic CRPQ $Q_{\text{sa}}$ from Example 4. However, a simple case-by-case analysis shows that $Q_{\text{na}}$ is not semantically acyclic. The reason is that $Q_{\text{na}}$ forbids the interaction between the different RPQs that label its atoms by replacing each RPQ of the form $a^*$ in $Q_{\text{sa}}$ with $a$. □

## 5.2 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 prove both 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 elementary upper bound for the problem. The algorithm also yields an equivalent UC2RPQ $Q'$ of exponential size for a semantically acyclic UC2RPQ $Q$.

THEOREM 4. *There exists a* 2EXPSPACE *algorithm that on input a UC2RPQ $Q$ does the following:*

1. *It checks whether $Q$ is semantically acyclic.*
2. *If the latter holds, it outputs an acyclic UC2RPQ $Q'$ of single-exponential size such that $Q \equiv Q'$.*

*Proof:* The algorithm in Lemma 1 computes on input $Q$ an acyclic UC2RPQ $T_Q$ such that $T_Q$ is the maximum among all acyclic UC2RPQs that are contained in $Q$. It follows that $Q$ is semantically acyclic iff $Q \subseteq T_Q$. Thus, in order to check semantic acyclicity of $Q$ we only have to compute $T_Q$, which can be done in EXPSPACE, and then check whether $Q \subseteq T_Q$, which can be done in exponential space in $(|Q| + |T_Q|)$ [6], and hence in double-exponential space in $|Q|$ (because $T_Q$ is of at most exponential size in $|Q|$). The whole procedure can be done in 2EXPSPACE. □

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

*Proof:* Since the proof is rather tedious we only sketch it here. Checking whether a Boolean CRPQ $Q_1$ is contained in a Boolean CRPQ $Q_2$ is EXPSPACE-complete [6]. From the proof it follows that this problem remains hard even if $Q_1$ is acyclic and the underlying undirected graph $\mathcal{U}(Q_2)$ of $Q_2$ is connected, which is crucial for our construction.

The construction in [6] also yields an acyclic $Q_2$, but for our proof to work we require $Q_2$ not to be semantically acyclic. Nevertheless, this can be easily fixed: In fact, given $Q_1$ and $Q_2$ as above, we can do the following: (1) Construct in polynomial time CRPQ $Q_2'$ by "appending" to a particular variable of $Q_2$ a fresh copy of the Boolean CRPQ $Q_{\text{na}}$ in Example 5 (which is not semantically acyclic), over an alphabet $\Sigma$ that is disjoint from that of $Q_1$ and $Q_2$. (2) Construct in polynomial time CRPQ $Q_1'$ from $Q_1$ by "appending" to a suitable variable in $Q_1$ a fresh copy of the acyclic query $Q = \exists x \bigwedge_{a\in\Sigma}(x, a, x)$. It can be proved that $Q_1$ is contained in $Q_2$ iff $Q_1'$ is contained in $Q_2'$. Moreover, $Q_1'$ is acyclic (because $Q$ is acyclic and "appending" it to $Q_1$ does not create cycles), $\mathcal{U}(Q_2')$ is connected, and $Q_2'$ is not semantically acyclic (because otherwise $Q_{\text{na}}$ would be semantically acyclic). We use this EXPSPACE-hard restriction of the containment problem to prove that our problem is also EXPSPACE-hard.

Let $Q_1$ and $Q_2$ be Boolean CRPQs such that $Q_1$ is acyclic, $\mathcal{U}(Q_2)$ is connected and $Q_2$ is not semantically acyclic. We claim that $Q_1$ is contained in $Q_2$ iff the CRPQ $Q_1 \wedge Q_2$ is semantically acyclic. Assume first that $Q_1$ is contained in $Q_2$. Then $Q_1 \wedge Q_2$ is equivalent to $Q_1$, which is acyclic. Assume, on the other hand, that $Q_1 \wedge Q_2 \equiv T$, where $T = \bigvee_{1\le i\le m} T_i$ and each $T_i$ is an acyclic C2RPQ. Since $Q_1 \wedge Q_2 \subseteq Q_2$, it follows that $T \subseteq Q_2$. Also, since $T_i \subseteq T$, it follows that $T_i \subseteq Q_2$, for each $1 \le i \le m$.

Let $T_i^1, \ldots, T_i^{k_i}$ be the C2RPQs associated with each connected component of $\mathcal{U}(T_i)$. Thus, $T_i \equiv T_i^1 \wedge \cdots \wedge T_i^{k_i}$. For each $i$ with $1 \le i \le m$, we shall prove that $T_i^j \subseteq Q_2$ for some $1 \le j \le k_i$. Assume to the contrary. Then there exist graph databases $\mathcal{G}_1, \ldots \mathcal{G}_{k_i}$ such that $T_i^j(\mathcal{G}_j) = \texttt{true}$ and $Q_2(\mathcal{G}_j) = \texttt{false}$, for each $1 \le j \le k_i$. Since $\mathcal{U}(T_i^j)$ is connected, we can choose $\mathcal{G}_j$ to be connected as well (in the sense that the underlying undirected graph of $\mathcal{G}_j$ is connected), for each $1 \le j \le k_i$. Consider the disjoint union $\mathcal{G}$ of $\mathcal{G}_1, \ldots \mathcal{G}_{k_i}$. Clearly, $T_i(\mathcal{G}) = \texttt{true}$, and since $T_i \subseteq Q_2$, it follows that $Q_2(\mathcal{G}) = \texttt{true}$. But since $\mathcal{U}(Q_2)$ is connected it must be the case that $Q_2(\mathcal{G}_j) = \texttt{true}$, for some $1 \le j \le k_i$, which is a contradiction.

Therefore, for each $i$ with $1 \le i \le m$ there exists $j_i$ with $1 \le j_i \le k_i$ such that $T_i^{j_i} \subseteq Q_2$. Consider the acyclic UC2RPQ $T' = \bigvee_{1\le i\le m} T_i^{j_i}$. Notice that $T \subseteq T'$. Moreover, $T' \subseteq Q_2$. We shall prove that $Q_1 \subseteq T'$, which implies our desired result that $Q_1 \subseteq Q_2$. Since $Q_2$ is not semantically acyclic, it must be the case that $Q_2 \nsubseteq T'$. Hence there exists a graph database $\mathcal{G}^*$ such that $Q_2(\mathcal{G}^*) = \texttt{true}$ and $T'(\mathcal{G}^*) = \texttt{false}$. Consider an arbitrary database $\mathcal{G}$ such that $Q_1(\mathcal{G}) = \texttt{true}$. We prove next that $T'(\mathcal{G}) = \texttt{true}$, which implies that $Q_1 \subseteq T'$. Consider the disjoint union $\mathcal{G}'$ of $\mathcal{G}$ and $\mathcal{G}^*$. Clearly, $Q_1 \wedge Q_2(\mathcal{G}') = \texttt{true}$, and since $Q_1 \wedge Q_2 \equiv T \subseteq T'$, it follows that $T'(\mathcal{G}') = \texttt{true}$. Then $T_i^{j_i}(\mathcal{G}') = \texttt{true}$, for some $1 \le i \le m$. Since $\mathcal{U}(T_i^{j_i})$ is connected, it follows that either $T_i^{j_i}(\mathcal{G}) = \texttt{true}$ or $T_i^{j_i}(\mathcal{G}^*) = \texttt{true}$. But $T'(\mathcal{G}^*) = \texttt{false}$, and, thus, $T_i^{j_i}(\mathcal{G}^*) = \texttt{false}$. We conclude that $T_i^{j_i}(\mathcal{G}) = \texttt{true}$, and, thus, $T'(\mathcal{G}) = \texttt{true}$. □

**Evaluation of semantically acyclic UC2RPQs** With the help of Theorem 4 we can provide an answer to our first question regarding semantically acyclic UC2RPQs: Its evaluation is fixed-parameter tractable.

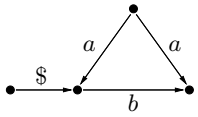
THEOREM 5. *The problem of checking whether $\bar{a} \in Q(\mathcal{G})$, for a given graph database $\mathcal{G}$, semantically acyclic UC2RPQ $Q$, and tuple $\bar{a}$ of node ids in $\mathcal{G}$, is fixed-parameter tractable.*

In particular, semantically acyclic UC2RPQs can be evaluated in time $O(f(|Q|) + |\mathcal{G}|^2 \cdot 2^{p(|Q|)})$, where $p : \mathbb{N} \to \mathbb{N}$ is a polynomial and $f : \mathbb{N} \to \mathbb{N}$ is a triple-exponential function.

**Features of the language: Inverses** The algorithm in Theorem 4 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 next that this is the case:

PROPOSITION 5. *There is a semantically acyclic CRPQ that is no equivalent to any acyclic UCRPQ.*

PROOF. The Boolean query $Q$ that is graphically depicted below




is semantically acyclic. In fact, it is equivalent to the acyclic C2RPQ $\exists x \exists y((x, \$, y) \wedge (y, ba^- a, y))$. A tedious case-by-case analysis proves that it is not equivalent to any acyclic UCRPQ. □

# 6. 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 [26], PDL [23], nested regular expressions [4], 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 decidable in 2EXPSPACE 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.

Also, we have left a gap in the complexity problem of identifying whether a UC2RPQ is semantically acyclic: Our current upper and lower bounds are 2EXPSPACE and EXPSPACE, respectively. The reason why we do not know how to lower the upper bound at this point, is that we need to check containment of a UC2RPQ $Q$ into $T_Q$, which might be exponentially bigger than $Q$. The main problem with this is not that there might be an exponential number of disjuncts in $T_Q$, but that some of the 2RPQs that label the atoms of $T_Q$ might be of exponential size. Proving that this containment can be checked in EXPSPACE may require of more sophisticated techniques.

**Size of equivalent acyclic queries** The algorithm presented in Theorem 4 computes an equivalent acyclic query of single-exponential size for a semantically acyclic UC2RPQ. Is this optimal, i.e. is there a family $(Q_n)_{n\geq 1}$ of semantically acyclic UC2RPQs such that (1) $|Q_n|$ is polynomially bounded by $n$, and (2) the smallest acyclic UC2RPQ that is equivalent to $Q_n$ is of size $\Omega(2^n)$, for each $n \geq 1$?

**Features of the language** The algorithm that constructs an equivalent acyclic UC2RPQ $T$ for a semantically acyclic UC2RPQ $Q$ (Theorem 4) outputs a union of C2RPQs $T$ even if $Q$ is a C2RPQ. But is this necessary? That is, is there a C2RPQ $Q$ that is semantically acyclic, but yet it is not equivalent to a *single* acyclic C2RPQ $T$?

**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 years 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 $Q$ of bounded *(hyper)-treewidth* [21], which are also defined in terms of the existence of a tree decomposition of $Q$ with desirable properties.

It is known that the results in Section 2 (Theorem 1 and Proposition 1) also apply to UCQs that are equivalent to unions of CQs of bounded treewidth [12, 9]. 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 hyper-treewidth. In the same way, one might be interested in extending results on approximations of UC2RPQs to classes of bounded hyper-treewidth, as it has been done for UCQs [5].

**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 [4]. 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 [23, 4].

**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 conjunctive-query 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 [16, 25]. 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 [25], that is, whether its evaluation problem is tractable.

**Acknowledgments** Barceló is funded by Fondecyt grant 1130104. Romero is funded by CONICYT Ph.D. Scholarship. We would like to thank Gaelle Fontaine for useful conversations about the nature of semantic acyclicity.

# 7. REFERENCES

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

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

[3] P. Barceló, L. Libkin, A.W. Lin, P. Wood. Expressive languages for path queries over graph-structured data. *ACM TODS* 38(4), 2012.

[4] P. Barceló, J. Perez, J. Reutter. Relative expressiveness of nested regular expressions. In *AMW* 2012, pages 180–195.

[5] P. Barceló, L. Libkin, M. Romero. Efficient approximations of conjunctive queries. In *PODS* 2012, pages 249–260.

[6] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR'00*, pages 176–185.

[7] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Rewriting of regular expressions and regular path queries. *JCSS*, 64(3):443–465, 2002.

[8] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC* 1977, pages 77–90.

[9] H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *CP* 2005, pages 167–181.

[10] M. P. Consens, A. O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *PODS'90*, pages 404–416.

[11] I. Cruz, A. Mendelzon, P. Wood. A graphical query language supporting recursion. In *SIGMOD'87*, pages 323-330.

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

[13] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *JACM* 30(3), pages 514–550, 1983.

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

[15] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu. Adding regular expressions to graph reachability and pattern queries. In *ICDE 2011*, to appear.

[16] T. Feder, M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.* 28(1), pages 57–104, 1998.

[17] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, Y. Wu. Relative expressive power of navigational querying on graphs. In *ICDT 2011*, pages 197–207.

[18] D. Florescu, A. Levy, D. Suciu. Query containment for conjunctive queries with regular expressions. In *PODS'98*, pages 139–148.

[19] G. Gottlob, N. Leone, F. Scarcello. The complexity of acyclic conjunctive queries. J. ACM 48(3), 2001, pages 431–498.

[20] G. Gottlob, N. Leone, F. Scarcello. Hypertree decompositions and tractable queries.*J. Comput. Syst. Sci.* 64(3), pages 579–627, 2002.

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

[22] M. Grohe. The structure of tractable constraint satisfaction problems. In *MFCS* 2006, pages 58–72.

[23] D. Harel, D. Kozen, J. Tiuryn. *Dynamic logic*. MIT Press, 2000.

[24] Ph. Kolaitis, M. Y. Vardi. Conjunctive query-containment and constraint satisfaction. *JCSS* 61(2), pages 302–332, 2000.

[25] Ph. Kolaitis, M. Y. Vardi. A Logical Approach to Constraint Satisfaction. *Complexity of Constraints*, pages 125–155, 2008.

[26] L. Libkin, W. Martens, D. Vrgoc. Querying graph databases with XPath. Accepted for publication, *ICDT* 2013.

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

[28] Ch. Papadimitriou, M. Yannakakis. On the complexity of database queries. In *PODS* 1997, pages 12-19.

[29] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operator. *JACM* 27(4), 1980, pages 633–655.

[30] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test selectivity of hypergraphs and selectively reduce acyclic hypergraphs. *Siam J. of Comp.*, 13, 1984, pages 566–579.

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