When is the evaluation of Extended CRPQ tractable?

Diego Figueira, Varun Ramanathan

When is the evaluation of Extended CRPQ tractable?

Diego Figueira
Varun Ramanathan
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800
F-33400, Talence, France

ABSTRACT

We investigate the complexity of the evaluation problem for ECRPQ: Conjunctive Regular Path Queries (CRPQ), extended with synchronous relations (*a.k.a.* regular or automatic). We give a characterization for the evaluation and parameterized evaluation problems of ECRPQ in terms of the underlying structure of queries. As we show, complexity can range between PSpace, NP and polynomial time for the evaluation problem, and between XNL, W[1], and FPT for parameterized evaluation.

CCS CONCEPTS

• **Information systems** → **Query languages for non-relational engines**; • **Theory of computation** → *Parameterized complexity and exact algorithms*; *Database query processing and optimization (theory).*

KEYWORDS

extended CRPQ, graph databases, regular path queries (RPQ), synchronous / regular / automatic relations

1 INTRODUCTION

Graph databases are finite edge-labeled graphs which find applications in several new domains [1]. In graph databases, a fundamental querying mechanism is based on the existence of some paths in the database with certain properties. These properties include that the labels seen along a path must belong to a given language, or that the starting or terminal vertices of some paths must be equal. Path languages are typically specified by regular expressions (or restrictions thereof) over the alphabet of edge labels. This gives rise to the much studied class of Conjunctive Regular Path Queries (CRPQ).

Example 1.1. An example of a CRPQ is

$$q_1 = \exists y\, x \xrightarrow{\pi_1} y \wedge x \xrightarrow{\pi_2} y \wedge \mathit{label}(\pi_1) \in a^*b \wedge \mathit{label}(\pi_2) \in (a+b)^*c$$

which outputs all vertices v having one outgoing path with label in a^*b and one outgoing path with label in $(a+b)^*c$. Further these paths must end at the same vertex. One can think of such a query as having a "reachability subquery" on variables x, y (*i.e.*, $x \xrightarrow{\pi_1} y \wedge x \xrightarrow{\pi_2} y$) and a "path testing subquery" on variables π_1, π_2 (*i.e.*, $\mathit{label}(\pi_1) \in a^*b \wedge \mathit{label}(\pi_2) \in (a+b)^*c$). Since this last part involves only *unary* properties on path variables, such a query is usually written more succinctly as $\exists y\, x \xrightarrow{a^*b} y \wedge x \xrightarrow{(a+b)^*c} y$. ◁

CRPQs are often considered as the graph database equivalent to Conjunctive Queries (CQ) on relational databases. Indeed, it is the closure under conjunction and projection of "RPQ atoms" of the form $x \xrightarrow{rexp} y$, where *rexp* is a regular expression.

In the example above, the restrictions on the labels of paths are given by monadic properties (*i.e.*, languages). There is, hence, no way to relate the path labels.

It has often been argued that the class of CRPQ is not expressive enough for many natural querying tasks. One obvious shortcoming is that we cannot express any inter-path dependencies, *i.e.*, relations between the paths of the database that are matched by the edges of the graph pattern, except that they must start or end with the same node. Indeed, some scenarios require the capability to compare these path labels (see [3] for several examples from different domains) and the extension of CRPQ with non-monadic word relations gives rise to several expressive extensions which have been studied lately [2–4, 19]. Concretely, for a class of finite word relations $\mathcal{R}$ one can consider "CRPQ+$\mathcal{R}$", as the result of extending CRPQ with testing of $\mathcal{R}$ relations on path labels. Following our example, we would now have that

$$q_2 = \exists y\, x \xrightarrow{\pi_1} y \wedge x \xrightarrow{\pi_2} y \wedge (\mathit{label}(\pi_1), \mathit{label}(\pi_2)) \in R$$

is a formula of CRPQ+$\mathcal{R}$, assuming R is a binary relation from $\mathcal{R}$. Using this terminology, CRPQ is equivalent to CRPQ+Reg, where Reg is the class of regular languages (seen as unary relations). Among the most basic and studied classes of finite word relations we have the classes of *Recognizable*, *Synchronous*, and *Rational* relations, which form a strict hierarchy: Recognizable ⊊ Synchronous ⊊ Rational. It is well known that any CRPQ+Recognizable query is equivalent to a finite union of CRPQ (known as UCRPQ), and that the evaluation and satisfiability problems of CRPQ+Rational are undecidable, even for very simple Rational relations [2]. Hence, the first one has too little expressive power and the last one too much. On the contrary, CRPQ+Synchronous enjoys a good tradeoff between complexity and expressive power. Synchronous relations are relations recognizable by synchronous multi-tape finite automata, and they constitute a very robust and studied class, closed under Boolean operations and enjoying most of the decidabilty and algorithmic properties inherited from regular languages. This is the reason why CRPQ+Synchronous has been studied thoroughly in [3], and is commonly known as ECRPQ ('Extended' CRPQ). One may argue that ECRPQ considerably improves not only the expressive power of CRPQ, but also the succinctness, since they can encode CRPQ-expressible queries in a succinct way, even by a non-elementary factor [13].

The data complexity for the evaluation of ECRPQ queries is the same as for CRPQ and even RPQ (*i.e.*, NL-complete), but the combined complexity jumps from NP (of CRPQ and CQ) to PSpace. As we will show, even the parameterized complexity increases in comparison with that of CRPQ or CQ. One may however consider this as a reasonable compromise for having substantially more expressive power and succinctness. Indeed, even first-order logic

—arguably the kernel of the SQL language— is already PSpace-complete. But is this necessarily the case? For which sort of ECRPQ do we have a 'low' complexity for the evaluation problem? Can we characterize these classes? These are the questions we study here.

As we will show, the complexity of the evaluation problem for ECRPQ hinges on the interplay between the structure of the *reachability* and *path testing* subqueries (borrowing the jargon from Example 1.1). In a nutshell, we characterize the complexity of the evaluation problem and of its parameterized version in terms of the underlying skeletal structure of queries. For the evaluation problem, the complexity may vary between polynomial time, NP, and PSpace, while for the parameterized evaluation it ranges between FPT, W[1], and XNL. Concretely, for each class C of such underlying query structures, ECRPQ(C) consists of all ECRPQ having some element of C as underlying structure. Under some mild hypothesis, we characterize all the C under which evaluation of ECRPQ(C) is polynomial time, NP, or PSpace-complete. And similarly, those C under which parameterized evaluation is FPT, W[1]-complete, or XNL-complete. This can be regarded as the lifting to ECRPQ of the following (now classical) result on CQs: For any class $\mathcal{G}$ of graphs, evaluation of CQ($\mathcal{G}$) is in polynomial time iff parameterized evaluation of CQ($\mathcal{G}$) is FPT iff $\mathcal{G}$ has bounded treewidth [15].[1] While the underlying structure of a CQ is simply "a graph", for an ECRPQ the structure is slightly richer. For our characterization, instead of having *one* measure (*i.e.*, treewidth) we will need to handle *three* independent measures. Depending on which combinations of these measures are bounded we will obtain different complexity classes.

Related works. The most relevant related work is [3], where it is shown that the evaluation problem for ECRPQ is PSpace-complete and the containment problem is undecidable. [2] shows that the extension of ECRPQ with a single non-synchronous relation, such as the suffix, infix or scattered subword relations, makes the evaluation problem either undecidable or with non primitive recursive complexity. [13] shows that the undecidability for containment holds even for every simple queries, and that ECRPQ can be arbitrarily more succinct than CRPQ.

Outline. Before giving the characterization statement we will need to introduce more formally the query language and measures: we do this in the next preliminary Section 2. In Section 3 we state the characterization results for the parameterized and non-parameterized versions of the evaluation problem. The following three sections are devoted to proving these results, in particular Section 6 contains the proofs for the main theorems, using the bounds and reductions of Sections 4 and 5. We conclude with Section 7. The appendix contains some missing details.

2 PRELIMINARIES

For a set X, we use $\wp(X)$ to denote the set of all non-empty subsets of X, and $\wp_2(X)$ to the denote all the non-empty subsets of size at most 2. We write "c.e." as short for computably enumerable language (*a.k.a.* recursively enumerable). We write X^* to denote all finite words over the set X, and X^k to denote k-tuples of elements from X. We also write X^{k*} and X^{*k} as short for $(X^k)^*$ and $(X^*)^k$ respectively. By DFA and NFA we denote deterministic and non-deterministic finite automata respectively, defined in the usual way.

Graphs. A **multi-hypergraph** is a tuple $G = (V, H, \eta)$ where V is a finte set of vertices, H is a finite set of hyperedges, and $\eta : H \to \wp(V)$. If $\eta : H \to \wp_2(V)$, we say that G is a **multigraph**. If η is also injective we say that it is a **simple graph** (or just graph), and in this case we just write it as $G = (V, E)$, where $E = \{\eta(h) : h \in H\}$. A **tree** is a connected graph with no cycles. By the **subgraph induced by** $V' \subseteq V$ we denote $(V', E \cap \wp_2(V'))$.

Regular languages and synchronous relations. Let $\mathbb{A}$ be a finite alphabet and $\perp \notin \mathbb{A}$ a special 'blank' symbol. Given words $w_1, \ldots, w_\ell$ over $\mathbb{A}$, their **convolution** $w_1 \otimes \cdots \otimes w_\ell$ is the smallest word over $(\mathbb{A} \dot\cup \{\perp\})^\ell$ such that for every $i \leq \ell$, its projection onto the i-th component yields a word from $w_i \cdot \{\perp\}^*$. For example, $aab \otimes c \otimes bb = (a, c, b)(a, \perp, b)(b, \perp, \perp)$. A k-ary word relation $R \subseteq \mathbb{A}^{*k}$ is **synchronous** (*a.k.a.* regular, automatic) if $\{w_1 \otimes \cdots \otimes w_k : (w_1, \ldots, w_k) \in R\}$ is a regular language over $(\mathbb{A} \dot\cup \{\perp\})^\ell$. In light of this definition, we assume that k-ary synchronous relations are represented as a NFA[2] over an alphabet $(\mathbb{A} \dot\cup \{\perp\})^k$. Observe that, in particular, a unary synchronous relation is a regular language. The class of synchronous relations is known to be particularly well behaved: synchronous relations are effectively closed under all Boolean operators (also component-projection, alphabetic morphism, etc.), and they have decidable emptiness, universality and containment problems [5]. They also admit logical characterizations [12] on what is known as Automatic structures [6]. Some classical examples of synchronous relations include the prefix, equality, and equal-length binary relations, and some non-examples are the suffix, factor, (scattered) subword or alphabet projection.

Conjunctive queries. A **conjunctive query** (CQ) is a formula of the form $q(\bar{x}) = \exists \bar{y}\, R_1(\bar{z}_1) \wedge \cdots \wedge R_k(\bar{z}_k)$, where $\bar{x}$ and $\bar{y}$ are tuples of pairwise distinct variables and each tuple $\bar{z}_i$ ranges over the variables of $\bar{x}\bar{y}$. One standard abstraction of a CQ q as before is by means of its **Gaifman graph**, which is the simple graph having the variables as vertices and an edge between x and y whenever there is an atom containing both x and y. In this way, for a set of graphs C, we refer to CQ(C) as the class of CQ's whose abstraction is in C. Given a relational structure A, an assignment v from the variables to the domain of A is **satisfying** for q if for every i, the relation R_i contains the tuple $\bar{t}_i$ obtained by replacing every variable of $\bar{z}_i$ with its assignment according to v. A tuple $\bar{t}$ of elements of A is in the answer of q to A if there exists a satisfying assignment $\bar{x}$ to $\bar{t}$. If q is Boolean (*i.e.*, no free variables) then we write $A \models q$ if there exists some satisfying assignment.

ECRPQ on graph databases. A **graph database** is a finite edge-labelled graph, that is, $D = (V, E)$ where V is a finite set of vertices, $E \subseteq V \times \mathbb{A} \times V$ is the set of labeled edges, and $\mathbb{A}$ is a finite alphabet. A **path** p of D from v_0 to v_n of length $n \geq 0$ is a (possibly empty) sequence of edges from E of the form

$$(v_0, a_1, v_1), (v_1, a_2, v_2), \ldots, (v_{n-1}, a_n, v_n).$$

There is always an empty path from v to v for any $v \in V$. The **label** $label(p)$ of such a path is $a_1 \cdots a_n \in \mathbb{A}^*$, or ε if the path is empty.

[1] Under the assumption that W[1] ≠ FPT and $\mathcal{G}$ is computably enumerable.

[2] The choice of representation by NFA, DFA, or regular expressions is unessential for the complexity results we shall present.

We fix some infinite sets of **node variables** and of **path variables**. An ECRPQ is a query which can test for synchronous relations on paths between nodes of the graph. Let us first show an example and then give the formal definition.

Example 2.1. Here is an ECRPQ:

$$q(x, x') = \exists y\ x \xrightarrow{\pi_1} y \wedge x' \xrightarrow{\pi_2} y \wedge \text{eq-len}(\pi_1, \pi_2)$$

where 'eq-len' is the binary synchronous relation $\{(w, w') \in \mathbb{A}^* \times \mathbb{A}^* : |w| = |w'|\}$. The evaluation of q on a graph database D retrieves all the pairs of vertices (v, v') such that for some vertex u there is a path p_1 from v to u and a path p_2 from v' to u such that p_1 and p_2 have the same length. Instead of eq-length we could have chosen any other synchronous relation, such as, for example, "equality", or "edit-distance at most 14". In this case it would mean that the word of labels read by p_1 and p_2 are in the chosen relation.

An **extended conjunctive regular path query (ECRPQ)** is a pair $(q, \mathcal{R})$ where $\mathcal{R}$ is a finite set of synchronous relations, each $R \in \mathcal{R}$ having arity $arity(R) \geq 1$, and specified as an NFA over $(\mathbb{A} \dot\cup \{\bot\})^{arity(R)}$, for some fixed alphabet $\mathbb{A}$. On the other hand, q is a query, possibly having some free variables $\bar{x}$, of the form

$$q(\bar{x}) = \exists \bar{y}\, \exists \bar{\pi}\ \gamma(\bar{x}\bar{y}\bar{\pi}) \wedge \rho(\bar{\pi}). \tag{1}$$

Here $\bar{x}, \bar{y}$ are tuples which span over node variables and $\bar{\pi}$ over path variables. The idea is that γ tells us how node variables are connected through path variables, while ρ describes the properties and relations between path variables in terms of the regular languages and relations of $\mathcal{R}$. Concretely, the subformula $\gamma(\bar{x}\bar{y}\bar{\pi})$, called the **reachability subquery**, is a finite conjunction of **reachability atoms** of the form $z \xrightarrow{\pi} z'$, where z, z' are from $\bar{x}\bar{y}$ and π is from $\bar{\pi}$, with the restriction that no path variable π can appear in two distinct reachability atoms. That is, node variables may repeat in γ, but path variables may not. The subformula $\rho(\bar{\pi})$ is called the **relation subquery** and it is a finite conjunction of atoms of the form $R(\pi_1, \ldots, \pi_r)$, were $R \in \mathcal{R}$, $r = arity(R)$ and $\pi_1, \ldots, \pi_r$ are pairwise distinct path variables from $\bar{\pi}$.

Given a graph database $D = (V, E)$ over an alphabet $\mathbb{A}$, an assignment f_n from $\bar{x}\bar{y}$ to V and an assignment f_p from $\bar{\pi}$ to paths of D, we say that (f_n, f_p) is a **satisfying assignment** if (1) for every reachability atom $z \xrightarrow{\pi} z'$ of γ we have that $f_p(\pi)$ is a path from $f_n(z)$ to $f_n(z')$ in D; and (2) for every atom $R(\pi_1, \ldots, \pi_r)$ of ρ the tuple $(w_1, \ldots, w_r)$ is in the relation $R \in \mathcal{R}$, where $w_i = label(f_p(\pi_i))$, for every i. Assuming $\bar{x} = (x_1, \ldots, x_\ell)$, the answers $q(D)$ of the query q to the database D is the set of all $(f_n(x_1), \ldots, f_n(x_\ell)) \in V^\ell$ for every satisfying assignment (f_n, f_p). If q is Boolean, then we say that D **satisfies** q (and we write $D \models q$) if there exists some satisfying assignment.

A **Conjunctive Regular Path Query (CRPQ)** is an ECRPQ such that: (1) every relation is of arity one (*i.e.*, a regular language); and (2) no path variable appears in more than one atom of the relation subquery. That is, a CRPQ $q(\bar{x})$ is a query of the form

$$\exists \bar{y}\, \exists \bar{\pi}\ (z_1 \xrightarrow{\pi_1} z_1' \wedge \cdots \wedge z_n \xrightarrow{\pi_n} z_n') \wedge (L_1(\pi_1) \wedge \cdots \wedge L_n(\pi_n)),$$

which is usually more succinctly written as

$$q(\bar{x}) = \exists \bar{y}\ z_1 \xrightarrow{L_1} z_1' \wedge \cdots \wedge z_n \xrightarrow{L_n} z_n'.$$

In order to keep the technical developments down to the essential, we henceforward assume that all queries are Boolean. However, our results can be easily extended to UECRPQ, that is, finite unions of possibly non-Boolean ECRPQ queries.

Two-level graphs. Observe that ECRPQs have *two levels* of atoms: the reachability atoms stating a property of pairs of *vertices* $x \xrightarrow{\pi} y$, and the relational atoms $R(\pi_1, \ldots, \pi_r)$ stating properties of tuples of *paths*. In order to capture the structural information of these queries in the same way as done for CQs through its Gaifman graph, we introduce here the notion of a *two-level graph*. A two-level graph is a graph having a first sort of edges between vertices and a second sort of edges between edges of the first sort.

Formally, a **two-level multi-hypergraph** (or **2L graph** for short) is defined as a tuple $G = (V, E, H, \eta, \nu)$, where (V, E, η) is a multigraph and (E, H, ν) is a multi-hypergraph. In other words, V, E, H are finite sets, $\eta : E \to \wp_2(V)$ and $\nu : H \to \wp(E)$. Intuitively, E are "first-level" edges and H are "second-level" hyperedges. For simplicity we shall always assume $E \cap H = \emptyset$.

The **abstraction of an ECRPQ** query such as the one in (1) is a 2L-graph (V, E, H, η, ν), where (V, E, η) is the multi-hypergraph on the node variables and (E, H, ν) is the multi-hypergraph on the path variables. That is:

- V is the set of node variables $\bar{x}\bar{y}$,
- E is the set of path variables $\bar{\pi}$,
- H is the set of relation atoms of ρ,
- $\eta(\pi) = \{z, z'\}$ if $z \xrightarrow{\pi} z'$ is an atom of γ, and
- $\nu(A) = \{\pi_1, \ldots, \pi_r\}$ if $A = R(\pi_1, \ldots, \pi_r)$ is an atom of ρ.

Given a class of 2L-graphs C we consider ECRPQ(C) as the set of all ECRPQ whose abstractions are in C. On the other hand, the **abstraction of a CRPQ** is a graph (V, E) having V as set of node variables, and and edge $x, y \in E$ for every atom $x \xrightarrow{\pi} y$ in its reachability subquery. Similarly, for a class C of graphs, we denote by CRPQ(C) the set of CRPQ having an abstraction in C.

Treewidth A **tree decomposition** of a graph $G = (V, E)$ is a tree $T = (V', E')$ together with a mapping $\lambda : V' \to \wp(V)$ such that

(i) for every $\{u, v\} \in E$ there is $w \in V'$ such that $\{u, v\} \subseteq \lambda(w)$;
(ii) for every $v \in V$, the subgraph of T induced by $\{v' \in V' : v \in \lambda(v')\}$ is a tree.

For $v' \in V'$ we refer to $\lambda(v')$ as the **bag** of v'. The **width** of such tree decomposition is $\max_{v' \in V'} |\lambda(v')|$, that is, the maximum size of a bag therein. The **treewidth** of a graph G, denoted by $tw(G)$, is the minimum width among all its tree decompositions. For a set of graphs C, we define its treewidth as $tw(C) = \sup_{G \in C} tw(G)$, and we write $tw(C) = \infty$ when it is unbounded.

2.1 Parameterized decision problems

A **parameterized problem** is a set $P \subseteq \mathbb{A}^* \times \mathbb{N}$, where $\mathbb{A}$ is a finite alphabet. If $(x, k) \in \mathbb{A}^* \times \mathbb{N}$ is an instance of a parameterized problem, we refer to x and k as the *input* and *parameter* respectively.

Complexity classes. A parameterized problem is said to be **fixed-parameter tractable**, or FPT, if every instance $(x, k) \in \mathbb{A}^* \times \mathbb{N}$ can be solved in time $f(k) \cdot |x|^c$ for some computable function f and constant c. A (many-one) FPT-reduction between a parameterized problem P to another one P' (noted $P \leq^{\text{fpt}} P'$) is an algorithm that

computes for every instance (x, k) of P an instance (x', k') of P' such that *(i)* $(x, k) \in P$ iff $(x', k') \in P'$, *(ii)* $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$, and *(iii)* the algorithm runs in time $f(k) \cdot |x|^c$, for some computable $f : \mathbb{N} \to \mathbb{N}$ and constant c.

We will consider here two other parameterized complexity classes, namely W[1] and XNL. We will try to keep definitions at the minimum needed. For our purposes we will not need the definition of W[1], but just that it may be thought as the analog of NP in parameterized complexity. On the other hand, the class XNL, introduced in [11] and studied in [10] under the name of [Uniform-XNL]$^{\text{FPT}}$, is the closure under FPT-reductions of the class of parameterized problems P such that there is a computable function assigning to each parameter k an NL algorithm solving the decision problem $\{x : (x, k) \in P\}$. For these classes we have FPT $\subseteq$ W[1] $\subseteq$ XNL, and it is conjectured that these containments are strict.

Some complete parameterized problems. A complete problem for XNL is the parameterized intersection non-emptiness problem. This is the parameterized version of the classical PSpace-complete intersection non-emptiness problem for regular languages.

Problem	Parameterized Intersection Non-Emptiness (p-IE)
Input	A set S of DFA.
Parameter	The cardinality of S.
Question	Is $\bigcap_{\mathcal{A} \in S} L(\mathcal{A}) \neq \emptyset$?

It was shown in [20] that p-IE is complete for XNL under FPT-reductions. We denote by IE the non-parameterized version of this problem, which is PSpace-complete [16].

For a class of query languages $\mathcal{Q}$ over relational structures, we define the classical evaluation problem eval-$\mathcal{Q}$ and its parameterized version p-eval-$\mathcal{Q}$ problems.

Problem	$\mathcal{Q}$ evaluation (eval-$\mathcal{Q}$)
Input	A relational structure D; a Boolean query $q \in \mathcal{Q}$.
Question	Does $D \models q$?

Problem	Parameterized $\mathcal{Q}$ evaluation (p-eval-$\mathcal{Q}$)
Input	A relational structure D; a Boolean query $q \in \mathcal{Q}$.
Parameter	The size of q
Question	Does $D \models q$?

It is known that eval-CQ is NP-complete and p-eval-CQ is W[1]-complete, and it is folklore that this extends to CRPQ. Regarding ECRPQ, its evaluation problem is complete for PSpace.

Proposition 2.2 ([3]). *eval-*ECRPQ *is* PSpace*-complete.*

In fact, a characterization for the evaluation problem for CQ and CRPQ can be stated in terms of the underlying graph:

Proposition 2.3 ([15]). *For any c.e. class C of graphs,*

*(1) if tw$(C) < \infty$, eval-*CQ(C) *is in polynomial time;*
*(2) otherwise, p-eval-*CQ(C) *is* W[1]*-complete*

Corollary 2.4 (Folklore). *For any c.e. class C of graphs,*

*(1) if tw$(C) < \infty$, eval-*CRPQ(C) *is in polynomial time;*
*(2) otherwise, p-eval-*CRPQ(C) *is* W[1]*-complete*

Proof of Corollary 2.4. Item 2 follows from the fact that on binary signatures CQ$(C) \subseteq$ CRPQ(C), and item 1 from the existence of a polynomial-time reduction from eval-CRPQ(C) to eval-CQ(C). This relies on the fact that for every regular language L the relation R_L consisting of all pair of vertices (v, v') such that there is a path with label in L from v to v' is computable in polynomial time. □

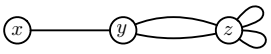
This implies that, assuming FPT $\neq$ W[1], CQ(C) [resp. CRPQ(C)] has a tractable evaluation problem iff it has a tractable parameterized evaluation problem iff tw$(C) < \infty$. This has been extended to arbitrary classes of CQs in the following sense:

Proposition 2.5 ([14]). *For any c.e. class $\mathcal{Q}$ of* CQ*s,*

(1) if for some k every $q \in \mathcal{Q}$ is equivalent to some $q' \in$ CQ of treewidth $\leq k$, eval-$\mathcal{Q}$ is in polynomial time;
(2) otherwise, p-eval-$\mathcal{Q}$ is W[1]*-complete.*

Item (1) above has been shown to hold also for CRPQ, where equivalence has to be replaced by a well-suited notion of homomorphisms between CRPQ [18]. Proposition 2.5 assumes that the arity of relations is bounded, and in fact a generalization of these results also holds in the absence of such a constraint [9, 17], where treewidth is replaced with a "more general" hypergraph measure.

For the purposes of the present work, we will sometimes focus on CQs over binary relations, which we denote by CQ$_{bin}$. For technical reasons that will become apparent, it will be convenient to use *multigraphs* as abstractions for CQ$_{bin}$ (as opposed to *graphs*): we define the multigraph of a CQ$_{bin}$ query q as having the set of variables as vertices, and for every two vertices x, x' it has k edges $\{x, x'\}$, where k is the number of atoms $R(z, z')$ of q such that $\{z, z'\} = \{x, x'\}$. For example, the multigraph corresponding to $R(x, y) \wedge S(z, y) \wedge S(y, z) \wedge S(z, z) \wedge R(z, z) \in$ CQ$_{bin}$ is:



Then, for a class C of multigraphs we define CQ$_{bin}(C)$ as the class of all CQ$_{bin}$ queries whose underlying multigraph is in C. It is easy to see that the previous results imply that the same characterization holds, where the treewidth of a multigraph is simply the treewidth of its underlying simple graph.

Lemma 2.6 (Corollary of [15]). *For any c.e. class C of multigraphs such that tw$(C) = \infty$, p-eval-*CQ$_{bin}(C)$ *is* W[1]*-complete.*

Proof. This follows directly from a trivial adaptation of the lower bound reduction of [15, Theorem 17]. □

3 CHARACTERIZATION RESULTS

We wish to obtain a characterization result as the one for CQ's or CRPQ's already mentioned, of the form

"The evaluation problem for ECRPQ(C) *is tractable if, and only if, C has bounded f-measure."*

for a suitable 2L graph measure f. For the case of CQ's and CRPQ's the right measure f is *treewidth*, as witnessed by Proposition 2.3 and Corollary 2.4. For ECRPQ's, the measure involves treewidth but also other measures, and the situation for complexity classes is more complex. In particular, remember that for CQ's and CRPQ's we have that (assuming W[1] $\neq$ FPT) eval-CQ(C) is tractable iff p-eval-CQ(C) is FPT. However, for ECRPQ's, there is a mismatch between "tractable evaluation" and "tractable parameterized evaluation". For instance, there are classes which are FPT for p-eval-ECRPQ(C) but PSpace-complete for eval-CQ(C). Further, the complexity for the

parameterized evaluation of ECRPQ may be FPT, W[1]-complete or XNL-complete while it can 'only' be either FPT or W[1] for CQ's and CRPQ's. In a similar vein, the non-parameterized version may vary between polytime, NP and PSpace-complete.

Going further: What measures on 2L graphs make sense for characterizing the parameterized and combined complexity of the evaluation for ECRPQ? Unsurprisingly, treewidth plays a fundamental role, but we also need to take into account two other measures. These two measures are the maximum size of connected components in the underlying multi-hypergraph of the relation subquery, either in terms of *number of hyperedges* it contains (denoted by cc_{hedge}) or in the *number of vertices* (cc_{vertex}). Note that neither a bound on the number of edges implies a bound on the number of vertices nor the other way round, since we work with multi-hypergraphs.

Before defining more formally these measures, we can already give the general characterization statements we obtain.

Theorem 3.1 (Characterization for p-eval-ECRPQ). *For any c.e. class C of 2L graphs,*

*(1) if $cc_{vertex}(C) = \infty$, then p-eval-*ECRPQ(C) *is* XNL*-complete;*
*(2) if $cc_{vertex}(C) < \infty$ and $tw(C) = \infty$, then p-eval-*ECRPQ(C) *is* W[1]*-complete;*
*(3) if $cc_{vertex}(C) < \infty$ and $tw(C) < \infty$, then p-eval-*ECRPQ(C) *is* FPT.

Under the complexity theoretic hypothesis that W[1] $\neq$ FPT and some some mild assumptions on the class (*cc*-tameness, whose definition follows), we also obtain a complete characterization for the evaluation problem.

Theorem 3.2 (Characterization for eval-ECRPQ). *For every cc-tame class C of 2L graphs, and assuming* W[1] $\neq$ FPT*,*

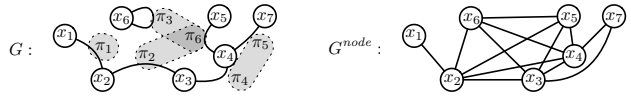
*(1) if $cc_{vertex}(C) = \infty$ or $cc_{hedge}(C) = \infty$, then eval-*ECRPQ(C) *is* PSpace*-complete;*
*(2) if $cc_{vertex}(C) < \infty$, $cc_{hedge}(C) < \infty$ and $tw(C) = \infty$, then eval-*ECRPQ(C) *is in* NP *and not in polynomial time;*
*(3) if $cc_{vertex}(C) < \infty$, $cc_{hedge}(C) < \infty$ and $tw(C) < \infty$, then eval-*ECRPQ(C) *is in polynomial time.*

In the statement above, we say that a class C of 2L graphs is ***cc*-tame**, if either $cc_{vertex}(C) + cc_{hedge}(C) < \infty$ or there exists a polynomial time function $f : \mathbb{N} \to C$ such that $cc_{vertex}(f(n)) + cc_{hedge}(f(n)) \geq n$ for all n. In other words, that there is some tractable algorithm to produce elements of C witnessing big connected components.[3]

For *data complexity*, remember that RPQ, CRPQ and ECRPQ are all NL-complete. In the next paragraphs we give the precise definition of treewidth, cc_{vertex} and cc_{hedge} for (classes of) 2L graphs.

2L graph measures

For a 2L graph $G = (V, E, H, \eta, \nu)$, let G^{rel} be the multigraph (E, H, ν), and let G^{node} be the graph (V, E'), where E' consists of all the sets $\{v, v'\}$ for which there are $h, h' \in H$ and $e, e' \in E$ such that (1) $v \in e$, $v' \in e'$, (2) $e \in h$, $e' \in h'$, and (3) h and h' belong to the same connected component in G^{rel}. Here is an illustration for a 2L graph $G = (V, E, H, \eta, \nu)$, where edges of η are depicted with full lines and hyperedges of ν as dashed blobs.



The intuition is that G^{node} is the graph resulting from replacing connected components of G^{rel} with cliques on their incident vertices. The treewidth $tw(G)$ of a 2L graph G is the treewidth of G^{node}, and for a class C we extend it as usual: $tw(C) = \sup_{G \in C} tw(G)$. We also define the measures cc_{vertex} and cc_{hedge} of 2L graphs which count the sizes of connected components in G^{rel}. We define $cc_{vertex}(G)$ as the maximum number of vertices of a connected component of G^{rel}, and we define $cc_{hedge}(G)$ as the maximum number of hyperedges contained in a connected component of G^{rel}. In the example above, $cc_{vertex}(G) = 3$ and $cc_{hedge}(G) = 2$, both of them witnessed by the connected component of $\{\pi_2, \pi_3, \pi_6\}$. For a class C of 2L-graphs, we let $C^{rel} = \{G^{rel} : G \in C\}$ and $C^{node} = \{G^{node} : G \in C\}$.

4 UPPER BOUNDS

For any 2L-graph G, let $\hat{G}$ be the result of merging all hyperedges of G^{rel} in the same connected component. Concretely, assuming $G = (V, E, H, \eta, \nu)$ let $\hat{G} = (V, E, \hat{H}, \eta, \hat{\nu})$, where

$$\hat{H} = \{h_C \subseteq H : C \text{ is a maximal connected component of } G^{rel}\}$$

and $\hat{\nu}(h_C) = \{e \in E : e \in \bigcup C\}$ for every $h_C \in \hat{H}$. Observe that $\hat{q}$ uses relations of arity bounded by $cc_{vertex}(G)$. In the proofs that follow we will use the following fact.

Lemma 4.1. *There is a* PSpace *procedure which produces, for any ECRPQ q with abstraction G, an equivalent ECRPQ $\hat{q}$ with abstraction $\hat{G}$. Further, if $cc_{vertex}(G)$ and $cc_{hedge}(G)$ are considered as being constants, the algorithm runs in polynomial time.*

Lemma 4.2. *p-eval-*ECRPQ *is in* XNL*.*

Proof. Consider the following procedure which produces, for every ECRPQ q, an NL algorithm for $\{D : D \models q\}$. Fix an arbitrary ECRPQ q with abstraction G, and suppose we are given a graph database D as input. To simplify the algorithm, as a first step we transform q into an equivalent ECRPQ q' such that every connected component of G^{rel} consists of only one hyperedge using Lemma 4.1. This transformation is in constant time since q is fixed.

We now guess a mapping v from the node variables of q' to the nodes of the input database D in NSPACE($|q'| \cdot \log(|D|)$), hence in NL since q is fixed. Next, we verify that v is a satisfying assignment. For each atom $R(\pi_1, \ldots, \pi_\ell)$ in the relation subquery such that $\{x_i \xrightarrow{\pi_i} y_i\}_{1 \leq i \leq \ell}$ are the corresponding atoms in the reachability subquery, we non-deterministically guess simultaneously a path p_i from $v(x_i)$ to $v(y_i)$ for each $i \leq \ell$ (using ℓ log-sized pointers to navigate the paths in D) and we verify that the labels of the guessed paths $p_1, \ldots, p_\ell$ are in the relation R. Since no two relations share a path variable this implies that there is a satisfying assignment

[3] This *cc*-tameness condition is solely used for the PSpace-hardness statement, and its purpose is simply to show that it takes a very weak extra hypothesis to obtain PSpace-hardness —and, in fact, the condition could be replaced with any hypothesis enforcing that elements of C with big connected components can be produced in polynomial time. Indeed, without the *cc*-tameness hypothesis we would obtain the exact same theorem statement, with the only difference that "PSpace-complete" is replaced with "in PSpace".

and thus that $D \models q$; and since ℓ is bounded by a constant, this last step is also in NL. □

Lemma 4.3. *Given a class C of 2L graphs such that $cc_{vertex}(C) < \infty$*

(1) there is an FPT *reduction from p-eval-*ECRPQ(C) *to p-eval-*CQ(C^{node})*;*

*(2) if further $cc_{hedge}(C) < \infty$, there is polynomial time reduction from eval-*ECRPQ(C) *to eval-*CQ(C^{node}).

Proof. Let $G \in C$ be the abstraction of an ECRPQ q and D be a graph database over an alphabet $\mathbb{A}$. We show how to produce a CQ q' (depending only on q) and a relational database D' (depending on q and D) such that

- $D \models q$ iff $D' \models q'$,
- the underlying Gaifman graph of q' is G^{node}, and
- all relations used in q' have arity $\leq 2 \cdot cc_{vertex}(G)$.

Further, if $cc_{hedge}(C) < \infty$, the construction runs in polynomial time. We first produce an equivalent query $\hat{q} \equiv q$ having $\hat{G}$ as abstraction using Lemma 4.1, which is in PSpace, or in polynomial time if $cc_{vertex}(G)$ and $cc_{hedge}(G)$ are constant.

We now show how to build q', which will have one atom with relation R' for each atom with relation R in the relation subquery of $\hat{q}$. Let $R(\pi_1, \ldots, \pi_n)$ be an atom of the relation subquery of $\hat{q}$, where for every i there is an atom $x_i \xrightarrow{\pi_i} y_i$ in the reachability subquery of q. We then produce the atom $R'(x_1, y_1, \ldots, x_n, y_n)$ in q'. The CQ q' is obtained as the conjunction of all such atoms R'. Observe that the Gaifman graph of q' is precisely G^{node}.

Finally, the relational database D' is built using these relations R', where we populate each relation R' of arity $2n$ as follows

$$R' = \{(u_1, v_1, \ldots, u_n, v_n) : \text{ for every } i \text{ there is a path from } u_i \text{ to } v_i \text{ in } D \text{ with label } w_i \in \mathbb{A}^* \text{ such that } (w_1, \ldots, w_n) \in R\}$$

We then obtain that $D \models q$ iff $D' \models q'$. Since $cc_{vertex}(G)$ is bounded by a constant, so is the arity of the relations in q' and $\hat{q}$. Thus, D' can be produced from $\hat{q}$ in $O(|D|^{2 \cdot cc_{vertex}(G)})$, that is, in polynomial time. That is, D' is built in time $f(|q|) \cdot |D|^{2 \cdot cc_{vertex}(G)}$, where f is the time needed to compute $\hat{q}$ from q. In other words, this is an FPT reduction or even a polynomial reduction if we assume that $cc_{hedge}(G)$ is bounded by a constant (and thus that f is polynomial). □

5 LOWER BOUNDS

5.1 Combined complexity

We begin with identifying the hardest, PSpace-complete, cases of eval-ECRPQ. As a consequence of Lemma 4.3(2) of the previous section, as soon as $cc_{vertex}(C)$ and $cc_{hedge}(C)$ are bounded, the evaluation problem becomes an NP problem. We now show that, under *cc*-tameness, the remaining case is PSpace-complete.

Lemma 5.1. *For every cc-tame class C of 2L graphs, if $cc_{vertex}(C) + cc_{hedge}(C) = \infty$ then eval-ECRPQ(C) is* PSpace*-complete.*

Proof. Since eval-ECRPQ is in PSpace by Proposition 2.2, the statement boils down to showing that eval-ECRPQ(C) is PSpace-hard. We reduce from the PSpace-complete problem of intersection non-emptiness (IE) for regular languages over a fixed alphabet $\mathbb{A}$. Let the IE instance be given as n regular languages $L_1, \ldots, L_n$. It follows that there exists a computable 2L graph $G \in C$ such that G^{rel} contains a 'big' connected component C, having either (1) at least n vertices, or (2) at least one vertex incident to n hyper-edges (see Lemma A.1 in appendix). In both cases we will construct, in polynomial time, an ECRPQ q whose abstraction is G and a graph database D such that the following holds.

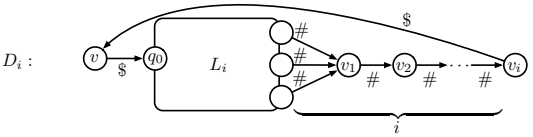
Claim 5.1. *$D \models q$ if, and only if, $L_1 \cap \cdots \cap L_n \neq \emptyset$.*

Case (1) If C has m vertices $\pi_1, \ldots, \pi_m$ with $m \geq n$, we define each relation $R(\pi_{i_1}, \ldots, \pi_{i_k})$ corresponding to a k-ary hyperedge $\{\pi_{i_1}, \ldots, \pi_{i_k}\}$ to be the set of all k-tuples

$$(\$u\underbrace{\#\cdots\#}_{i_1}\$, \ldots, \$u\underbrace{\#\cdots\#}_{i_k}\$) \in (\mathbb{A} \dot\cup \{\$, \#\})^{*k}$$

for every $u \in \mathbb{A}^*$. It is easy to see that each such R is a synchronous relation which can be built in polynomial time (see appendix for more details). All other relations, corresponding to hyperedges outside C, are simply 'universal'. That is, we define the relation $R'(\pi'_1, \ldots, \pi'_\ell)$ corresponding to a hyperedge outside the connected component C to be $(\mathbb{A} \dot\cup \{\$, \#\})^{*\ell}$. In this way we have produced, in polynomial time, an ECRPQ q whose abstraction is G.

Without loss of generality we assume that $n = m$, note that this can be guaranteed by extending the intersection problem with $m-n$ 'dummy' languages $\mathbb{A}^*$. For each language L_i let us define the graph database D_i as the transition graph of the NFA recognizing L_i, plus: (1) one distinguished vertex v, (2) i other vertices $v_1, \ldots, v_i$, (3) edges $v \xrightarrow{\$} q_0$ and $v_i \xrightarrow{\$} v$, and (4) a path $q_f \xrightarrow{\#} v_1 \xrightarrow{\#} \cdots \xrightarrow{\#} v_i$ of length i, for every final state q_f. Here's an example:



Finally, we define D as the union of $D_1, \ldots, D_n$, which is disjoint with the sole exception of the distinguished vertex v. We finish this first part of the proof by showing that Claim 5.1 holds.

If $D \models q$, consider the satisfying assignment of the m path variables $\pi_1, \ldots, \pi_m$ in the connected component C. These define m words $u_1, \ldots, u_m \in (\mathbb{A} \dot\cup \{\$, \#\})^*$ of the form $u_i = \$w\#^i\$$ for a given w, by definition of the relation R. Further, by the shape of D, the path π_i must go through the subdatabase D_i of D, for each i. Hence $w \in L_i$ for every i, witnessing that $\bigcap_i L_i \neq \emptyset$. Conversely, observe that for any $w \in \bigcap_i L_i$ the assignment sending

- every node variable to v,
- every path variable π_i from C to the path starting and ending in v, going through D_i along the path reading w from the initial state to a final state (it exists since $w \in L_i$), and
- every other path variable π' outside C to any arbitrary path starting and ending in v

is a satisfying assignment, and thus $D \models q$.

Case (2) If, on the other hand, C contains an element π incident to n hyperedges $h_1, \ldots, h_n$, we define a relation R_i for each hyperedge h_i as follows. If h_i is incident to $k \geq 0$ other vertices $\nu(h_i) = \{\pi, \pi_1, \ldots, \pi_k\}$, we produce an atom $R_i(\pi, \pi_1, \ldots, \pi_k)$ defined by

the synchronous relation $\{(u, u_1, \ldots, u_k) : u \in L_i \text{ and } u_1, \ldots, u_k \in \mathbb{A}^*\} \subseteq \mathbb{A}^{*(k+1)}$. In a similar way as before, we define all other relations corresponding to hyperedges outside C as being universal, and we thus obtain, in polynomial time, an ECRPQ q whose abstraction is G. Let D be the graph database having only one vertex, and a self-loop labeled a for each $a \in \mathbb{A}$. We then have that Claim 5.1 also holds in this case. Indeed, q is satisfiable iff $D \models q$ iff $\bigcap_i L_i \neq \emptyset$. □

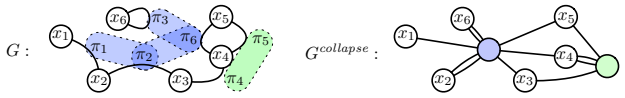
5.2 Parameterized complexity

The lower bounds for the parameterized complexity will be based on a reduction from parameterized evaluation problem for CQ's on graph databases. However, in order to present this reduction in a clear and modular way, we will need to introduce yet another multi-graph representation of a given 2L graph G, which roughly corresponds to collapsing connected components of G^{rel} into single vertices.

Concretely, given a 2L graph $G = (V, E, H, \eta, \nu)$ let $G^{collapse}$ be the multi-graph $(V \dot\cup C, \{(v, e) : e \in E, v \in \eta(e)\}, \mu)$, where

- $C \subseteq 2^H$ is the set of maximal connected components of G^{rel},
- $\mu = \{(v, e) \mapsto \{v, c\} : c \in C, h \in c, e \in h\}$.

For a class $\mathcal{C}$ of 2L-graphs, let $\mathcal{C}^{collapse} = \{G^{collapse} : G \in \mathcal{C}\}$. Here is an example (with similar visual encoding as the example of p. 5).



We will later exploit the following simple fact that, under the hypothesis that $\mathcal{C}$ has bounded cc_{vertex}, $\mathcal{C}$ has bounded treewidth if $\mathcal{C}^{collapse}$ has bounded treewidth.

Lemma 5.2. *For every class $\mathcal{C}$ of 2L graphs, if $cc_{vertex}(\mathcal{C}) < \infty$ and $tw(\mathcal{C}^{node}) = \infty$, then $tw(\mathcal{C}^{collapse}) = \infty$.*

Proof. Assume $cc_{vertex}(\mathcal{C}) = n < \infty$. We show the counterpositive $tw(\mathcal{C}^{collapse}) < \infty \Rightarrow tw(\mathcal{C}^{node}) < \infty$. Given a tree decomposition of $G^{collapse}$ of width k, consider the result of replacing, on every bag, each vertex c corresponding to a connected component of G^{rel} with the set of all the vertices incident to c (at most $2n$). Since each bag is of size $\leq k + 1$ we obtain bags of size $\leq (k + 1) \cdot 2n$. It follows then that it is a tree decomposition of width $\leq (k + 1) \cdot 2n - 1$. □

We now show a reduction from the parameterized evaluation of $\mathrm{CQ}_{bin}(\mathcal{C}^{collapse})$.

Lemma 5.3. *For any c.e. class $\mathcal{C}$ of 2L graphs, there exists an* FPT *reduction from p-eval-$\mathrm{CQ}_{bin}(\mathcal{C}^{collapse})$ to p-eval-ECRPQ($\mathcal{C}$).*

Proof. Given $q \in \mathrm{CQ}_{bin}(\mathcal{C}^{collapse})$ and a database D, we first find $G \in \mathcal{C}$ such that $G^{collapse}$ is the multigraph of q (computable since $\mathcal{C}$ is c.e.). We now produce an ECRPQ query q_G whose abstraction is G, and a graph database $\hat{D}$ such that $\hat{D} \models q_G$ iff $D \models q$.

Without loss of generality, we assume that for every relation R of D we have also its inverse R^{-1} (*i.e.*, that the relation R^{-1} is part of the database alphabet, and it is interpreted as $\{(v, v') : (v', v) \in R\}$), otherwise we add it to the database.

Let us recall the shape of a multigraph of the form $G^{collapse} = (V \dot\cup C, E', \mu)$. It has two sorts of vertices: 'node' vertices from V and 'component' vertices from C, corresponding to maximal connected components of G^{rel}. It is a partition, in the sense that edges go always between these two sorts of vertices, and they are the result of *splitting* each edge $\eta(e) = \{v, v'\}$ into two edges: $\{v, c_e\}$ and $\{c_e, v'\}$, where $c_e \in C$ is the connected component of e in G^{rel}. We can then assume that the query q is of the form

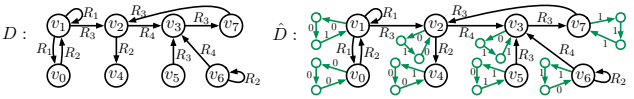
$$q = \exists \bar{x}\ R_1(x_1, y_1) \wedge R'_1(y_1, x'_1) \wedge \cdots \wedge R_\ell(x_\ell, y_\ell) \wedge R'_\ell(y_\ell, x'_\ell)$$

where the x_i's and x'_i's correspond to node variables and the y_i's to component variables. Observe that we can ensure these precise directions of q since we have the inverses: otherwise we could transform q into an equivalent query with this form, having the same underlying multi-graph.

Let q_G be any ECRPQ query whose abstraction is G. Let $\mathbb{A}$ be the alphabet of all the relation names of D, plus two new symbols: 0 and 1. We will now show how to implement the relations over $\mathbb{A}$. Remember that for every atom $R(\pi_1, \ldots, \pi_r)$ of the relation subquery of q_G (where $r = arity(R)$) and for every $1 \leq j \leq r$, there is (exactly) one atom of the form $x_{j_i} \xrightarrow{\pi_i} x'_{j'_i}$ in the reachability subquery of q_G. Thus, for each such $R(\pi_1, \ldots, \pi_r)$ we produce the following synchronous relation

$$R = \{(R_{j_1} w R'_{j'_1}, \ldots, R_{j_r} w R'_{j'_r}) : w \in \{0, 1\}^+\} \subseteq \mathbb{A}^{*r}.$$

Intuitively, R is implemented to ensure that, in the context of the evaluation of the database we construct, there are paths $x_{j_i} \xrightarrow{R_{j_i}} y_{j_i} \xrightarrow{R'_{j_i}} x'_{j'_i}$ for each i, where all the y_{j_i}'s are equal, and identified by a word $w \in \{0, 1\}^*$. We have now defined q_G, and we are left with defining $\hat{D}$, which we will define as an extension $\hat{D} \supseteq D$ over the expanded alphabet $\mathbb{A}$, consisting of adding simple cycles labeled with words over $\{0, 1\}$. Suppose D has (active) domain $\{v_1, \ldots, v_n\}$. $\hat{D}$ is the result of adding a simple cycle incident to v_i reading the $n' = \lceil \log(n) \rceil$-bit binary expansion of i, that is, a word over the alphabet $\{0, 1\}^{n'}$. This involves adding $(n' - 1) \cdot n$ new vertices to D, and can be done in polynomial time. For instance:



Observe that this is an FPT procedure since (a) $\hat{D}$ is generated in polynomial time in $|D|$ and does not depend on q, and (b) the procedure to generate q_G is effective since $\mathcal{C}$ is c.e. and does not depend on D. We finally show that this is a sound reduction.

Claim 5.2. *$\hat{D} \models q_G$ if, and only if, $D \models q$.*

⇐ Assume $D \models q$ through a satisfying mapping μ from the variables of q to the nodes of D. In other words, for every $1 \leq i \leq \ell$ we have that $R_i(\mu(x_i), \mu(y_i))$ and $R'_i(\mu(y_i), \mu(x'_i))$ are in D. Remember that G is the abstraction of q_G, and $G^{collapse}$ is obtained from G by splitting edges and adding component vertices. Let $R(\pi_1, \ldots, \pi_r)$ (where $r = arity(R)$) be an atom of the relational subquery of q_G, and let $1 \leq i \leq r$. Let the clause containing π_i in the reachability subquery of q_G be of the form $x_{j_k} \xrightarrow{\pi_i} x'_{j'_k}$. We then know that there

is a subquery of q of the form $R_{j_k}(x_{j_k}, y_{j_k}) \wedge R'_{j_k}(y_{j_k}, x'_{j'_k})$, where y_{j_k} is the variable corresponding to the connected component of π_i. Hence, by our construction of $\hat{D}$, there exists a path $\mu(x_{j_k}) \xrightarrow{R_{j_k}} \mu(y_{j_k}) \xrightarrow{w} \mu(y_{j_k}) \xrightarrow{R'_{j_k}} \mu(x'_{j'_k})$, where $w \in \{0,1\}^+$corresponds to reading the $\{0,1\}$-labeled simple cycle attached to the vertex $\mu(y_{j_k})$. Thus we build a satisfying assignment for q_G on $\hat{D}$.

$\Rightarrow$ Given satisfying assignments f_n, f_p for the node and path variables of q_G on $\hat{D}$, one can easily build a satisfying assignment for q on D. We send every node variable x to $f_n(x)$. Given a component variable y of q corresponding to a connected component c of G^{rel}, consider any path variable π of c such that $label(f_p(\pi)) = RwR'$ for some R, w, R', where w is the binary encoding of the number i (by construction, all such π in c will read the same w). Then, we can send the variable y to v_i. In this way we produce an assignment witnessing $D \models q$. □

Lemma 5.4. *For any c.e. class C of 2L graphs such that $cc_{vertex}(C) = \infty$, p-eval-*ECRPQ(C) *is* XNL*-hard.*

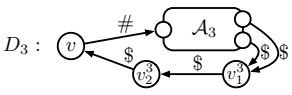
Proof. We show an FPT reduction from the p-IE problem (*cf.* § 2), which is XNL-complete under FPT reductions [20]. We divide the reduction into two cases: (a) the case where the size of hyperedges in $\{G^{rel} : G \in C\}$ is bounded and (b) the case in which for every $n \in \mathbb{N}$ there exists $G \in C$ containing a hyperedge of size $\geq n$.

(a) Bounded hyperedges Given k DFA $\mathcal{A}_1, \ldots, \mathcal{A}_k$ over an alphabet $\mathbb{A}$, we want to test whether $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k) \neq \emptyset$ where k is the parameter. We can first look for a 2L graph $G \in C$ such that G^{rel} contains a connected component having a "long path". That is, a 2L graph $G = (V, E, H, \eta, \nu)$ with a set of k hyperedges $h_1, \ldots, h_k \in H$ of size ≥ 2, and $k-1$ edges $u_1, \ldots, u_{k-1} \in E$ such that each u_i is in h_i, h_{i+1}, and nowhere else. In other words, $u_i \in (\nu(h_i) \cap \nu(h_{i+1})) \setminus \bigcup_{j \notin \{i, i+1\}} \nu(h_j)$ for every i. Observe that, since $cc_{vertex}(C) = \infty$ and the sizes of hyperedges are bounded, we must necessarily find such 2L graph G. Further, this is computable since C is c.e.

We now produce an ECRPQ q over an alphabet $\mathbb{B} = \mathbb{A} \dot{\cup} \{\#, \$\}$ with abstraction G. Let $h_{k+1}, \ldots, h_N$ be the remaining hyperedges of H, and let $q = \varphi \wedge \bigwedge_{1 \leq i \leq N} R_i(\bar{t}_i)$ be such that (1) q abstracts to G, mapping $R_i(\bar{t}_i)$ to h_i for every i, where φ is the reachability subquery, and (2) u_i is the last element of $\bar{t}_i$ and the first element of $\bar{t}_{i+1}$ for every $1 \leq i < k$. Observe that such q exists and that it can be trivially produced from G in polynomial time. In order to complete the definition of q, we need to define the synchronous relation corresponding to each R_i. For $i \leq k$, we define the relation $R_i \subseteq \mathbb{B}^{*r}$ of arity r as follows

$$R_i = \{(\# w \$^i, u_1, \ldots, u_{r-2}, \# w \$^{i+1}) : w \in \mathbb{A}^* u_1, \ldots, u_{r-2} \in \mathbb{B}^*\}$$

and for $i > k$, we let $R_i = \mathbb{B}^{*r}$, where $r = arity(R_i)$. Each R_i is synchronous and can be built in polynomial time. Now we define a graph database D such that $D \models q$ iff $\bigcap_i L_i \neq \emptyset$. For each $i \leq n$, we build a graph database D_i over the alphabet $\mathbb{B}$, which consists of the transition graph $\mathcal{A}_i$ plus i vertices $v, v^i_1, \ldots, v^i_{i-1}$, one edge from v to the initial state of $\mathcal{A}_i$ reading #, and for every final state vertex q_f of $\mathcal{A}_i$ a path from q_f to v going through $v^i_1, \ldots, v^i_{i-1}$ reading the word $\$^i$. Here is an example of D_3, assuming $\mathcal{A}_3$ has 2 final states.



We finally define $D = \bigcup_i D_i$, which is a disjoint union except for v.

Consider, for each u_i (remember u_i is a path variable of q), the source node variable x_i and the target node variable y_i such that $x_i \xrightarrow{u_i} y_i$ is an atom in the reachability subquery of q. Let $X = \{x_1, \ldots, x_k, y_1, \ldots, y_k\}$. For any mapping f_n from the node variables of q to the vertices of D such that $f_n(x) = v$ for every $x \in X$, observe that there is some f_p such that (f_n, f_p) is a satisfying assignment for q if, and only if, $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k) \neq \emptyset$. On the other hand, any assignment f_n in which some node variable $x \in X$ is not mapped to v cannot be part of any satisfying assignment (f_n, f_p), because the relations R_i ($i \leq k$) demand the first and last symbols of $label(f_p(u_i))$ to be # and \$ respectively, and this can only happen if $f_p(u_i)$ starts and ends at v —in other words, if $f_n(x_i) = f_n(y_i) = v$. We have then produced D and q such that D, q is a 'yes' instance of p-eval-ECRPQ(C) if, and only if, $\{\mathcal{A}_i\}_{i \leq k}$ is a 'yes' instance of the p-IE problem. Further, the size of q depends only on k and the class C (which is fixed) and can be bound via a computable function (since C is c.e.), and the database D can be produced in polynomial time (in fact, linear) with respect to the size of $\{\mathcal{A}_i\}_{i \leq k}$. All in all, this means that this is an FPT reduction.

(b) Unbounded hyperedges The FPT reduction just shown can be adapted to this case by simply finding a 2L graph G with a hyperedge h of size $r \geq k$, implementing the graph G as an ECRPQ q as before, this time with a relation R corresponding to h implemented as

$$R = \{\# w \$^1, \# w \$^2, \ldots, \# w \$^k : w \in \mathbb{A}^*\} \times \mathbb{B}^{*(r-k)} \subseteq (\mathbb{B}^*)^r$$

and defining all other relations R' as universal: $R' = \mathbb{B}^{*arity(R')}$. Note that R can be produced in polynomial time as a synchronous relation, and we still have $D \models q$ iff $\bigcap_i L(\mathcal{A}_i) \neq \emptyset$. □

6 PUTTING ALL TOGETHER

We finally show how all ingredients fit together to prove the results promised in Section 3.

Theorem 3.1 (Characterization for p-eval-ECRPQ). *For any c.e. class C of 2L graphs,*

*(1) if $cc_{vertex}(C) = \infty$, then p-eval-*ECRPQ(C) *is* XNL*-complete;*
*(2) if $cc_{vertex}(C) < \infty$ and $tw(C) = \infty$, then p-eval-*ECRPQ(C) *is* W[1]*-complete;*
*(3) if $cc_{vertex}(C) < \infty$ and $tw(C) < \infty$, then p-eval-*ECRPQ(C) *is* FPT.

Proof. (1) The upper bound follows from Lemma 4.2 and the lower bound from Lemma 5.4.

(2) For the lower bound, suppose $cc_{vertex}(C) = c$ and $tw(C) = \infty$. By Lemma 5.2, this implies $tw(C^{collapse}) = \infty$, and thus we have

that p-eval-CQ$_{bin}(C^{collapse})$ is W[1]-complete by Corollary 2.6. Applying the FPT reduction of Lemma 5.3, we obtain that p-eval-ECRPQ(C) is W[1]-hard. The upper bound follows from Lemma 4.3(1) combined with the upper bound of Proposition 2.3(2).

(3) Since $tw(C^{node}) < \infty$, observe that p-eval-CQ(C^{node}) is FPT (further, in polynomial time) by Proposition 2.3(1). Hence, the FPT reduction of Lemma 4.3(1) yields an FPT algorithm. □

Theorem 3.2 (Characterization for eval-ECRPQ). *For every cc-tame class C of 2L graphs, and assuming* W[1] ≠ FPT,

(1) *if $cc_{vertex}(C) = \infty$ or $cc_{hedge}(C) = \infty$, then eval-*ECRPQ(C) *is* PSpace*-complete;*
(2) *if $cc_{vertex}(C) < \infty$, $cc_{hedge}(C) < \infty$ and $tw(C) = \infty$, then eval-*ECRPQ(C) *is in* NP *and not in polynomial time;*
(3) *if $cc_{vertex}(C) < \infty$, $cc_{hedge}(C) < \infty$ and $tw(C) < \infty$, then eval-*ECRPQ(C) *is in polynomial time.*

Proof. (1) The lower bound is given by Lemma 5.1, and it is the only place where *cc*-tameness is used. The upper bound follows from the PSpace upper bound of eval-ECRPQ of Proposition 2.2.

(2) For the lower bound, observe that if eval-ECRPQ(C) was in polynomial time, then in particular p-eval-ECRPQ(C) would be FPT. The previous Theorem 3.1(2) has established that, however, p-eval-ECRPQ(C) is W[1]-complete, and thus this would imply FPT = W[1], contradicting the hypothesis. The upper bound follows from the polynomial time reduction to eval-CQ of Lemma 4.3(2), combined with the fact that eval-CQ is in NP [7].

(3) This follows again by the polynomial time reduction to eval-CQ(C^{node}) of Lemma 4.3(2), combined with the fact that, since $tw(C) = tw(C^{node}) < \infty$, eval-CQ($C^{node}$) is in polynomial time by Proposition 2.3(1) [8]. □

7 CONCLUSION

We have studied ECRPQ, a previously studied expressive extension of CRPQ with path relations. We have classified the complexity of the (parameterized) evaluation probem for fragments of ECRPQ, as defined by their underlying structure. Contrary to CQ and CRPQ, the different scenarios in this case are more elaborate, and in particular tractable evaluation does not coincide with tractable parameterized evaluation. However, they can be succinctly described by means of three measures: treewidth, and the sizes (either as #edges or #vertices) of connected components of the relations used.

The characterization results can be extended in a standard way to non-Boolean queries having free node variables (considering the corresponding decision problem of whether a tuple of vertices is in in the answerset), and to finite unions of ECRPQ (*a.k.a.* UECRPQ).

REFERENCES

[1] Pablo Barceló. 2013. Querying graph databases. In *ACM Symposium on Principles of Database Systems (PODS)*. ACM Press, 175–188. https://doi.org/10.1145/2463664.2465216

[2] Pablo Barceló, Diego Figueira, and Leonid Libkin. 2013. Graph Logics with Rational Relations. *Logical Methods in Computer Science (LMCS)* 9, 3 (2013). https://doi.org/10.2168/LMCS-9(3:1)2013

[3] Pablo Barceló, Leonid Libkin, Anthony Widjaja Lin, and Peter T. Wood. 2012. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Transactions on Database Systems (TODS)* 37, 4 (2012), 31:1–31:46. https://doi.org/10.1145/2389241.2389250

[4] Pablo Barceló and Pablo Muñoz. 2017. Graph Logics with Rational Relations: The Role of Word Combinatorics. *ACM Transactions on Computational Logic* 18, 2 (2017), 10:1–10:41. https://doi.org/10.1145/3070822

[5] Jean Berstel. 1979. *Transductions and context-free languages*. Teubner Studienbücher : Informatik, Vol. 38. Teubner. https://www.worldcat.org/oclc/06364613

[6] Achim Blumensath and Erich Grädel. 2000. Automatic Structures. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 51–62. https://doi.org/10.1109/LICS.2000.855755

[7] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Symposium on Theory of Computing (STOC)*. ACM Press, 77–90. https://doi.org/10.1145/800105.803397

[8] Chandra Chekuri and Anand Rajaraman. 2000. Conjunctive query containment revisited. *Theoretical Computer Science* 239, 2 (2000), 211–229. https://doi.org/10.1016/S0304-3975(99)00220-0

[9] Hubie Chen, Georg Gottlob, Matthias Lanzinger, and Reinhard Pichler. 2020. Semantic Width and the Fixed-Parameter Tractability of Constraint Satisfaction Problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 1726–1733. https://doi.org/10.24963/ijcai.2020/239

[10] Yijia Chen, Jörg Flum, and Martin Grohe. 2003. Bounded Nondeterminism and Alternation in Parameterized Complexity Theory. In *Annual IEEE Conference on Computational Complexity*. IEEE Computer Society Press, 13–29. https://doi.org/10.1109/CCC.2003.1214407

[11] Rodney G. Downey and Michael R. Fellows. 1999. *Parameterized Complexity*. Springer. https://doi.org/10.1007/978-1-4612-0515-9

[12] Samuel Eilenberg, Calvin C Elgot, and John C Shepherdson. 1969. Sets recognized by n-tape automata. *Journal of Algebra* 13, 4 (1969), 447–464.

[13] Dominik D. Freydenberger and Nicole Schweikardt. 2013. Expressiveness and static analysis of extended conjunctive regular path queries. *Journal of Computer and System Sciences (JCSS)* 79, 6 (2013), 892–909. https://doi.org/10.1016/j.jcss.2013.01.008

[14] Martin Grohe. 2007. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM* 54, 1 (2007), 1:1–1:24. https://doi.org/10.1145/1206035.1206036

[15] Martin Grohe, Thomas Schwentick, and Luc Segoufin. 2001. When is the evaluation of conjunctive queries tractable?. In *Symposium on Theory of Computing (STOC)*. ACM Press, 657–666. https://doi.org/10.1145/380752.380867

[16] Dexter Kozen. 1977. Lower Bounds for Natural Proof Systems. In *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 254–266. https://doi.org/10.1109/SFCS.1977.16

[17] Dániel Marx. 2013. Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. *J. ACM* 60, 6 (2013), 42:1–42:51. https://doi.org/10.1145/2535926

[18] Miguel Romero, Pablo Barceló, and Moshe Y. Vardi. 2017. The homomorphism problem for regular graph patterns. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1–12. https://doi.org/10.1109/LICS.2017.8005106

[19] Markus L. Schmid. 2020. Conjunctive Regular Path Queries with String Variables. In *ACM Symposium on Principles of Database Systems (PODS)*. ACM Press, 361–374. https://doi.org/10.1145/3375395.3387663

[20] Michael Wehar. 2016. *On the complexity of intersection non-emptiness problems*. Ph.D. Dissertation. Ph.D. thesis, University at Buffalo.

A MISSING DETAILS

Lemma A.1. *Let C be a cc-tame class of 2L graphs such that $cc_{vertex}(C) = \infty$ or $cc_{hedge}(C) = \infty$. Then, for every $n \in \mathbb{N}$, there exists a computable 2L-graph $G \in C$ such that G^{rel} contains some connected component c having either (i) n vertices, or (ii) a vertex incident to n hyperedges.*

Proof. Recall that C is cc-tame and $cc_{vertex}(C) + cc_{hedge}(C) = \infty$. So there exists a computable function f such that for all $k \in \mathbb{N}$, $cc_{vertex}(f(k)) + cc_{hedge}(f(k)) \geq n$. Let $G = f(n + (n-1)^2)$. Then G is computable (by the computability of f).

We argue that G^{rel} contains a connected component c which has either at least n vertices (Case (i)), or at least one vertex incident to n hyperedges (Case (ii)). If (i) is not true, then every connected component of G^{rel} contains at most $(n-1)$ vertices, so $cc_{vertex}(G) \leq n-1$. Further, if (ii) is not true, then every vertex in G^{rel} is incident to at most $(n-1)$ hyperedges, so $cc_{hedge}(G) \leq (n-1)^2$. Therefore, the largest possible value of $cc_{vertex}(G) + cc_{hedge}(G)$ is $(n-1) + (n-1)^2$.

However, $G = f(n + (n-1)^2)$ and by definition f satisfies

$$cc_{vertex}(f(n + (n-1)^2)) + cc_{hedge}(f(n + (n-1)^2)) \geq n + (n-1)^2$$

This is a contradiction. Therefore either (i) or (ii) is true. □

Proof of Lemma 4.1. Consider an edge $h_C \in \hat{H}$, where $C = \{h_1, \ldots, h_\ell\}$ is a maximal connected component of G^{rel}, and consider the atoms $R_1(\bar{\pi}_1), \ldots, R_\ell(\bar{\pi}_\ell)$ from the relation subquery of q corresponding to C. Suppose that $\bar{\pi}_1, \ldots, \bar{\pi}_\ell$ use the path variables $\pi_1, \ldots, \pi_r$, and that each $\bar{\pi}_i$ is of length r_i (*i.e.*, it denotes an r_i-ary relation). We build an r-ary relation $R_C(\bar{\pi})$ for $\pi = \pi_1, \ldots, \pi_r$ such that for every mapping $f : \bar{\pi} \to \mathbb{A}^*$ we have

$$f(\bar{\pi}_i) \in R_i \text{ for every } i \leq \ell \text{ if, and only if, } f(\bar{\pi}) \in R_C.$$

Consider the NFA $\mathcal{A}_i = (Q_i, q_0^i, \delta_i, F_i)$ over $(\mathbb{A} \dot\cup \{\bot\})^{r_i}$ corresponding to each R_i of arity r_i. Let $|\pi| = r$.

For every $i \leq \ell$, let $\gamma_i : \{1, \ldots, r_i\} \to \{1, \ldots, r\}$ be the mapping sending j to j' if the j-th variable of $\bar{\pi}_i$ is $\pi_{j'}$.

We define the relation R via the following NFA $\mathcal{A} = (Q, q_0, \delta, F)$ over $(\mathbb{A} \dot\cup \{\bot\})^r$, where

- $Q = Q_1 \times \cdots \times Q_\ell$
- $q_0 = (q_0^1, \ldots, q_0^\ell)$
- $F = \{(q_1, \ldots, q_\ell) : q_1 \in F_1, \ldots, q_\ell \in F_\ell\}$
- $(q_1, \ldots, q_\ell) \xrightarrow{(a_1, \ldots, a_r)} (q'_1, \ldots, q'_\ell) \in \delta$ if we have that $q_i \xrightarrow{(a_{\gamma_i(1)}, \ldots, a_{\gamma_i(r_i)})} q'_i \in \delta_i$ for every $i \leq \ell$.

The query $\hat{q}$ is then be built by replacing each $R_1(\bar{\pi}_1), \ldots, R_\ell(\bar{\pi}_\ell)$ with R_C, for every maximal connected component C.

The procedure is in polynomial space, since every element of Q, δ, F can be described in polynomial space and the membership to these three sets is in polynomial time. Observe that if $cc_{vertex}(G)$ and $cc_{hedge}(G)$ are constant, so is r and ℓ in the construction above. Hence, $\mathcal{A}$ can be built in polynomial time. □

Proof of Claim 5.2 (with more details). $\Leftarrow$ Assume $D \models q$ through a satisfying mapping μ from the variables of q to the nodes of D. In other words, for every $1 \leq i \leq \ell$ we have that $R_i(\mu(x_i), \mu(y_i))$ and $R'_i(\mu(y_i), \mu(x'_i))$ are in D. Remember that G is the abstraction of q_G, and $G^{collapse}$ is obtained from G by splitting edges and adding component vertices. Let $R(\pi_1, \ldots, \pi_r)$ (where $r = arity(R)$) be an atom of the relational subquery of q_G, and let $1 \leq i \leq r$. Let the clause containing π_i in the reachability subquery of q_G be of the form $x_{j_k} \xrightarrow{\pi_i} x'_{j'_k}$. We then know that there is a subquery of q of the form $R_{j_k}(x_{j_k}, y_{j_k}) \wedge R'_{j_k}(y_{j_k}, x'_{j'_k})$, where y_{j_k} is the variable corresponding to the connected component of π_i. Hence, by our construction of $\hat{D}$, there exists a path

$$\mu(x_{j_k}) \xrightarrow{R_{j_k}} \mu(y_{j_k}) \xrightarrow{w} \mu(y_{j_k}) \xrightarrow{R'_{j_k}} \mu(x'_{j'_k})$$

where $w \in \{0,1\}^+$corresponds to reading the $\{0,1\}$-labeled simple cycle attached to the vertex $\mu(y_{j_k})$. Denote this path by $f_p(\pi_i)$, and let $f_n = \mu$. Then f_n, f_p serves as a satisfying assignment for q_G on $\hat{D}$.

$\Rightarrow$ Conversely, given satisfying assignments f_n, f_p for the node and path variables of q_G on $\hat{D}$, one can easily build a satisfying assignment μ for q on D. Restricted to node variables, $\mu(x) = f_n(x)$. Given a component variable y of q corresponding to a connected component c of G^{rel}, consider any path variable π of c such that $label(f_p(\pi)) = RwR'$ for some R, w, R', where w is the binary encoding of the number i (by construction, all such π in c will read the same w). Then, we can send the variable y to v_i. For each $1 \leq i \leq \ell$, the existence of the $\hat{D}$ path

$$f_n(x) \xrightarrow{R_i} f_n(y_i) \xrightarrow{R'_i} f_n(x'_i)$$

shows that $R_i(\mu(x_i), \mu(y_i)) \wedge R'_i(\mu(x'_i), \mu(y_i))$ holds true in D. Thus μ is an assignment witnessing $D \models q$. □

Missing details from proof of Lemma 5.1. We show that the relation R from Case (1) is synchronous and can be built in polynomial time. Let $n = \max_{1 \leq \ell \leq k} i_\ell$. We define R via the NFA $\mathcal{A} = (Q, q_0, \delta, \{q_f\})$ over the alphabet $(\mathbb{A} \dot\cup \{\$, \#, \bot\})^k$ where

$$Q = \{q_0, q_1, \ldots, q_{n+1}, q_f\}$$

and δ has the following transitions:

- $q_0 \xrightarrow{(\$, \ldots, \$)} q_1$;
- $q_1 \xrightarrow{(a, \ldots, a)} q_1$ for every $a \in \mathbb{A}$;
- $q_i \xrightarrow{(z_1, \ldots, z_k)} q_{i+1}$ for every $1 \leq i \leq n$ where, for every j;
  (1) $z_j = \#$ if $i \leq i_j$;
  (2) $z_j = \$$ if $i = i_j + 1$ and
  (3) $z_j = \bot$ otherwise;
- $q_{n+1} \xrightarrow{(z_1, \ldots, z_k)} q_f$ where, for every j
  (1) $z_j = \$$ if $n + 1 = i_j + 1$ and
  (2) $z_j = \bot$ otherwise.

Observe that this NFA can be built in polynomial time and that it denotes the relation R of all k-tuples

$$(\$u\underbrace{\#\cdots\#}_{i_1}\$, \ldots, \$u\underbrace{\#\cdots\#}_{i_k}\$) \in (\mathbb{A} \dot\cup \{\$, \#\})^{*k}$$

for every $u \in \mathbb{A}^*$. □