Relative Expressiveness of Nested Regular Expressions

Pablo Barceló[1], Jorge Pérez[1], and Juan L. Reutter[2]

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

Abstract. Nested regular expressions (NREs) have been proposed as a powerful formalism for querying RDFS graphs, but not too much investigation on NREs has been pursued in a more general graph database context. In this paper we study the relative expressiveness of NREs by comparing it with the language of conjunctive regular path queries (CRPQs), which is one of the most widely studied query languages for graph databases. Among other results, we show that NREs and CRPQs are incomparable in terms of expressive power, but NREs properly extend the language of acyclic CRPQs. Even more, there is a natural fragment of NREs that coincide in expressive power with the class of acyclic CRPQs. Our results, plus previous results that show that NREs can be evaluated in linear time in combined complexity, put forward NREs as a query language for graph-structured data that deserves further attention.

1 Introduction

Graph-structured data has become ubiquitous in current data-centric applications. Social networks, bioinformatics, astronomic databases, digital libraries, Semantic Web, and linked government data, are only a few examples of applications in which structuring data as graphs is essential. This has naturally raised the need for data management tools that can cope with the challenges posed by today's graph data applications.

Traditional relational query languages do not appropriately cope with the querying problematics raised by graph-structured data. The reason for this is twofold. First, in the context of graph databases one is typically interested in *navigational* queries, i.e. queries that traverse the edges of the graph while checking for the existence of paths satisfying certain conditions. However, most relational query languages, such as SQL, are not designed to deal with this kind of recursive queries [1]. Second, current graph database applications tend to be massive in size (think, for instance, of social networks or astronomic databases, that may store terabytes of information). Thus, one can immediately dismiss any query language that cannot be evaluated in polynomial time (or even in linear time!). On the other hand, even the core of the usual relational query languages – *conjunctive* queries (CQs) – does not satisfy this property (under widely-held complexity theoretical assumptions) [16].

One of the languages for graph-structured data that has received much attention in the literature is the class of *conjunctive regular path* queries (CRPQs) [9, 12, 3, 2], together with their extension with *inverses* (that is known as the class of C2RPQs). This language can express complex graph patterns by means of navigational features and existential quantification over node ids. But as it is to be expected, the good expressiveness properties of CRPQs are not for free. In fact, CRPQs properly contain the class of CQs, and hence its applicability is limited, at least for querying very large graph databases.

To overcome this problem, while at the same time retain reasonable expressiveness, one can follow at least two strategies. The first one is to consider structural restrictions on the *joins* of C2RPQs that yield efficient query evaluation. A typical example is the class of *acyclic* C2RPQs [2], that is, C2RPQs whose *underlying undirected graph* is acyclic. A query Q from this class can be evaluated over a graph database G in time $O(|G| \cdot |Q|)$ (see [18]), that is, linearly in both the size of the graph database and the query, showing that acyclic C2RPQs are well-suited for practical applications. The same complexity bound applies when Q is a union of acyclic C2RPQs.

Another strategy is enhancing the navigational features of the language, while disallowing conjunctions altogether. A language that has been designed following this premise is the language of *nested regular expressions* (NREs) [17], that extends classical regular expressions with inverses and an existential branching operator *à la* XPath [15]. This class of expressions was proposed in [17] for querying Semantic Web data, and was shown to be useful for querying RDF graphs (the data model for the Semantic Web) in the presence of the RDFS vocabulary [14]. The language of NREs share the good query evaluation properties of the class of acyclic C2RPQs. In particular, as shown in [17], NREs can be evaluated in linear time both in the size of the data and the expression.

Even though the expressiveness of NREs has been studied in the RDF/S context [17], not much is known about the relative expressive power of NREs with respect to C2RPQs. This paper intends to fill this gap, providing a complete picture of the relative expressiveness of NREs and C2RPQs. We start by showing that NREs and C2RPQs are incomparable in terms of their expressive power. On the other hand, we show that NREs properly extend the class of unions of acyclic C2RPQs that satisfy a mild syntactic condition (namely, that the underlying undirected graph of each disjunct is connected). Finally, we prove that there is a natural fragment of NREs that coincides in expressiveness with the class of unions of acyclic C2RPQs.

Our results can be read in the following way. The class of acyclic C2RPQs is well-behaved in terms of query evaluation. Both NREs and C2RPQs extend this class in incomparable ways. But only one of these extensions, namely NREs, preserves the good evaluation properties of acyclic C2RPQs. As such, the results of this paper are also a way of putting forward NREs, as they show that NREs constitute a class of graph queries that exhibit a good balance between efficiency and expressiveness. Therefore, we believe that NREs deserve a place as a general query language for graphs, beyond the RDFs context for which it was originally proposed.

In terms of related work, the relative expressiveness of a myriad of navigational languages for graph databases has been recently studied in [11]. But none of our results can be directly obtained from the results in such paper. This is because in [11] the study was restricted to navigational query algebras that do not allow expressing complex joins, such as CRPQs. Moreover, no syntactical restriction that resembles acyclicity is considered in such work.

Organization of the paper. Section 2 introduces the notions that will be used along the paper. In Section 3 we prove that C2RPQs and NREs are incomparable in terms of expressive power. Then in Section 4 we show that, under a mild restriction on the class of queries allowed, the class of unions of acyclic C2RPQs coincides with a syntactic fragment of the class of NREs. Finally, in Section 5, we provide conclusions for the paper. Due to space limitations some proofs have been moved to the appendix.

2 Preliminaries

2.1 Graph databases

Let $\mathbf{V}$ be a countably infinite set of *node ids*, and Σ a finite alphabet. A *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids (that is, V is a finite subset of $\mathbf{V}$) and $E \subseteq V \times \Sigma \times V$. Therefore, G is essentially an *edge-labeled* directed graph, where the fact that (u, a, v) belongs to E means that there is an edge from node u into node v labeled a. For a graph database $G = (V, E)$, we write $(u, a, v) \in G$ whenever $(u, a, v) \in E$.

2.2 Nested regular expressions

Let Σ be a finite alphabet. *Nested regular expressions* (NREs) over Σ extend regular expressions with an *existential nesting test* operator $[(\cdot)]$ (or just nesting operator, for short), and an *inverse* operator a^-, over each $a \in \Sigma$ [17]. The syntax of NREs is given by the following grammar.

$$\begin{aligned} nexp \;:=\; & \varepsilon \;\mid\; a \;(a \in \Sigma) \;\mid\; a^- \;(a \in \Sigma) \;\mid\; nexp \cdot nexp \\ & nexp^* \;\mid\; nexp + nexp \;\mid\; [nexp] \end{aligned} \qquad (1)$$

As it is customary, we use n^+ as shortcut for $n \cdot n^*$.

Intuitively, NREs specify pairs of node ids in a graph database, subject to the existence of a path satisfying a certain regular condition among them. That is, each NRE *nexp* over Σ defines a binary relation $[\![nexp]\!]_G$ when evaluated over a graph database G over Σ. This binary relation is defined inductively as follows,

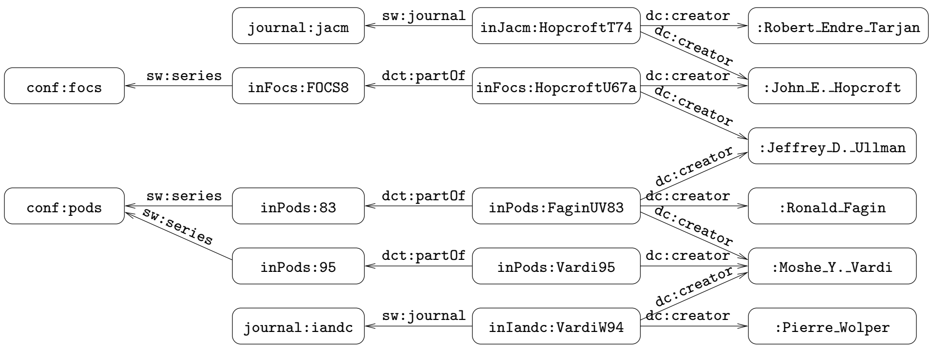


Fig. 1. A fragment of the RDF Linked Data representation of DBLP [10] available at `http://dblp.l3s.de/d2r/`.

where we assume that a is a symbol in Σ, and n, n_1 and n_2 are arbitrary NREs:

$$\begin{aligned}
[\![\varepsilon]\!]_G &= \{(u,u) \mid u \text{ is a node id in } G\}\\
[\![a]\!]_G &= \{(u,v) \mid (u,a,v) \in G\}\\
[\![a^-]\!]_G &= \{(u,v) \mid (v,a,u) \in G\}\\
[\![n_1 \cdot n_2]\!]_G &= [\![n_1]\!]_G \circ [\![n_2]\!]_G\\
[\![n_1 + n_2]\!]_G &= [\![n_1]\!]_G \cup [\![n_2]\!]_G\\
[\![n^*]\!]_G &= [\![\varepsilon]\!]_G \cup [\![n]\!]_G \cup [\![n \cdot n]\!]_G \cup [\![n \cdot n \cdot n]\!]_G \cup \cdots\\
[\![\,[n]\,]\!]_G &= \{(u,u) \mid \text{there exists } v \text{ s.t. } (u,v) \in [\![n]\!]_G\}.
\end{aligned}$$

Here, the symbol $\circ$ denotes the usual composition of binary relations, that is, $[\![n_1]\!]_G \circ [\![n_2]\!]_G = \{(u,v) \mid \text{there exists } w \text{ s.t. } (u,w) \in [\![n_1]\!]_G \text{ and } (w,v) \in [\![n_2]\!]_G\}$.

Example 1. Let G be the graph database in Figure 1. This graph contains a fragment of the *RDF Linked Data* representation of DBLP [10].[3] The following is a simple NRE that matches all pairs (x,y) such that x is an author that published a paper in conference y (for simplicity, we omit prefixes `dc:`, `dct:`, and `sw:` in edge labels):

$$n_1 \;=\; \texttt{creator}^- \cdot \texttt{partOf} \cdot \texttt{series}$$

For example, (`:Jeffrey_D._Ullman`, `conf:focs`) and (`:Ronald_Fagin`, `conf:pods`) are in $[\![n_1]\!]_G$.

Consider now the following expression that matches pairs (x,y) such that x and y are connected by a *coautorship sequence*:

$$n_2 \;=\; (\texttt{creator}^- \cdot \texttt{creator})^+$$

For example, the pair (`:John_E._Hopkroft`, `:Pierre_Wolper`) is in $[\![n_2]\!]_G$.

[3] For brevity we have not depicted all the RDF-URI prefixes in the figure.

Finally the following expression matches all pairs (x, y) such that x and y are connected by a coautorship sequence that only considers conference papers:

$$n_3 \;=\; (\texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator})^+$$

Let us give the intuition of the evaluation of this expression. Assume that we start at node u. The (inverse) edge $\texttt{creator}^-$ makes us to navigate from u to a paper v created by u. Then the existential test $[\,\texttt{partOf} \cdot \texttt{series}]$ is used to check that from v we can navigate to a conference (and thus, v is a conference paper). Finally, we follow edge $\texttt{creator}$ from v to an author w of v. The $(\cdot)^+$ over the expression allows us to repeat this sequence several times. For instance, $(\texttt{:John\_E.\_Hopkroft}, \texttt{:Moshe\_Y.\_Vardi})$ is in $[\![n_3]\!]_G$, but the pair $(\texttt{:John\_E.\_Hopkroft}, \texttt{:Pierre\_Wolper})$ is not in $[\![n_3]\!]_G$. $\square$

The following result, proved in [17], shows a remarkable property of NREs. It states that the query evaluation problem for NREs is not only polynomial in *combined* complexity (i.e. when both the database and the query are given as input), but also that it can be solved linearly in both the size of the database and the expression. Given a graph database G and an NRE *nexp*, we use $|G|$ to denote the size of G (in terms of the number of edges $(u, a, v) \in G$), and $|nexp|$ to denote the size of *nexp*.

Proposition 1 (from [17]). *Checking, given a graph database G, a pair of nodes (u, v), and an NRE nexp, whether $(u, v) \in [\![nexp]\!]_G$, can be done in time $O(|G| \cdot |nexp|)$.*

2.3 Conjunctive regular path queries

In Section 3 we compare NREs with one of the the most classical languages used for querying graph data: *conjunctive* regular-path queries (CRPQs) [5, 6]. Since NREs consider the inverse operator, we actually compare NREs with the language obtained by extending CRPQs with inverse. This extension is known as C2RPQs [6, 7]. We also consider the language of unions of C2RPQs (UC2RPQs).

The definition of C2RPQs is based on the notion of regular expression with *inverse*. This is an NRE that does not use the nesting operator $[\,\cdot\,]$. For instance, both expressions n_1 and n_2 in Example 1 are regular expressions with inverse. The evaluation of a regular expression with inverse r over a graph database G is inherited from the evaluation of NREs, i.e. the evaluation of r over a graph database G is the set of pairs $[\![r]\!]_G$.

Let $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_m)$ be (possibly empty) tuples of distinct variables. We denote by $\bar{x} \cup \bar{y}$ the set $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. A C2RPQ over alphabet Σ with free variables $\bar{x}$, is a formula $\varphi(\bar{x})$ of the form

$$\exists \bar{y}\Big((z_1, r_1, z_1') \wedge (z_2, r_2, z_2') \wedge \cdots \wedge (z_k, r_k, z_k') \Big), \tag{2}$$

where (1) r_i is a regular expression with inverse over Σ, for every $1 \leq i \leq k$, (2) $z_1, z_1', \ldots z_k, z_k'$ are not necessarily distinct variables, and (3) the set

$\{z_1, z_1', \dots, z_k, z_k'\}$ is precisely $\bar{x} \cup \bar{y}$. The arity of a C2RPQ is the number of free variables in the formula (n in this case).

Given a tuple $\bar{c} = (c_1, \dots, c_n)$ and a graph database G over Σ, we say that G satisfies $\varphi(\bar{c})$, denoted by $G \models \varphi(\bar{c})$, if there exists a mapping h from $\bar{x} \cup \bar{y}$ to the node ids in G such that $h(x_j) = c_j$ for every $1 \leq j \leq n$, and $(h(z_i), h(z_i')) \in [\![r_i]\!]_G$ for every $1 \leq i \leq k$. The evaluation of $\varphi(\bar{x})$ over G, denoted by $[\![\varphi(x)]\!]_G$, is the set of tuples $\{\bar{c} \mid G \models \varphi(\bar{c})\}$. CRPQs are obtained from C2RPQs by forbidding the use of the inverse operator. Finally, a UC2RPQ (resp. UCRPQ) is a formula $\psi(\bar{x})$ of the form $\varphi_1(\bar{x}) \vee \dots \vee \varphi_k(\bar{x})$ where $\varphi_i(\bar{x})$ is a C2RPQ (resp. CRPQ) for each $1 \leq i \leq k$, and we have that $\bar{c} \in [\![\psi(\bar{x})]\!]_G$ if and only if $\bar{c} \in [\![\varphi_i(\bar{x})]\!]_G$ for some $1 \leq i \leq k$.

Example 2. Let us consider again the expression n_1 used in Example 1. It is easy to see that n_1 can be expressed as the following C2RPQ:

$$Q_{n_1}(x, y) \;:=\; \exists u \exists v \big((x, \texttt{creator}^-, u) \wedge (u, \texttt{partOf}, v) \wedge (u, \texttt{series}, y) \big).$$

That is, for each graph database G it is the case that $[\![n_1]\!]_G = [\![Q_{n_1}]\!]_G$. In the same way, n_2 can be expressed as the C2RPQ $Q_{n_2}(x, y) := (x, (\texttt{creator}^- \cdot \texttt{creator})^+, y)$, that does not use existential quantification.

On the other hand, it can be proved that n_3 cannot be expressed as a UC2RPQ. The intuitive reason is that n_3 makes use of recursion (Kleene-$*$) over an expression that uses the nesting operator $[\,\cdot\,]$, which is a feature that cannot be codified by UC2RPQs in general. We will see a simpler example of this kind of behavior later when we prove Theorem 1. $\square$

As we already mentioned in Section 1, C2RPQs do not share the good query evaluation properties of NREs. In fact, the combined complexity of CRPQs is NP-complete [8, 2]. Furthermore, under widely-held complexity theoretical assumptions, the *parameterized* complexity of CRPQs is also intractable [16], which means that they cannot be evaluated in time $O(|G|^c \cdot f(|Q|))$, for a constant $c \geq 0$ and a computable function $f : \mathbb{N} \to \mathbb{N}$.

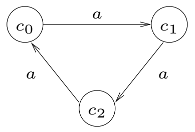
3 Expressiveness of NREs in terms of C2RPQs

In what follows we compare the expressive power of NREs with respect to C2RPQs. In particular, we say that an NRE *nexp* over Σ can be expressed as a (U)C2RPQ, if there exists a (U)C2RPQ $\varphi(x, y)$ over Σ such that for every graph database G over Σ it holds that $[\![nexp]\!]_G = [\![\varphi(x, y)]\!]_G$. Symmetrically, we say that a (U)C2RPQ $\varphi(x, y)$ over Σ can be expressed as an NRE, if there exists an NRE *nexp* over Σ such that for every graph database G over Σ it holds that $[\![\varphi(x, y)]\!]_G = [\![nexp]\!]_G$. Notice that it is only meaningful to compare NREs with binary (U)C2RPQs (that is, with UC2RPQs with exactly two free variables).

The main result of this section is that NREs and C2RPQs are incomparable in terms of expressive power. This follows from Proposition 2 and Theorem 1 below. In fact, we prove something stronger: there is a CRPQ that cannot be expressed as an NRE, and there is an NRE that cannot be expressed as a UC2RPQ.

Proposition 2. *There exists a binary CRPQ over the unary alphabet $\Sigma = \{a\}$ that cannot be expressed as an NRE.*

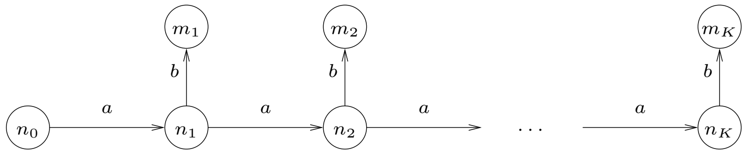
Proof. Consider the CRPQ $\varphi(x,y)$ given by the expression $(x,a,y) \wedge (y,a,x)$. We next show that $\varphi(x,y)$ cannot be expressed as an NRE. Consider the following graph G_1 over alphabet Σ:



Notice that $[\![\varphi(x,y)]\!]_{G_1} = \emptyset$, that is, there are no values u, v such that $G_1 \models \varphi(u,v)$. We show that every NRE *nexp* satisfies $[\![nexp]\!]_{G_1} \neq \emptyset$, which proves that φ cannot be expressed as an NRE. Details can be found in the appendix. □

Theorem 1. *There exists an NRE over binary alphabet $\Sigma = \{a,b\}$ that cannot be expressed as a UC2RPQ.*

Proof. Consider the NRE $nexp = (a \cdot [b])^+$. We next show that *nexp* cannot be expressed as a UC2RPQ $\psi(x,y)$. Assume for the sake of contradiction that this is not the case and let N be the maximum number of conjuncts in a disjunct of $\psi(x,y)$. Now consider the following graph G



where $K > N+1$. Now, since we are assuming that $\psi(x,y)$ is equivalent to *nexp* and given that $(n_0, n_K) \in [\![nexp]\!]_G$, we have that there exists a disjunct $\varphi(x,y)$ of $\psi(x,y)$, such that $(n_0, n_K) \in [\![\varphi(x,y)]\!]_G$. Assume without loss of generality that the conjunctive part of $\varphi(x,y)$ is of the form $(z_1, r_1, z_1') \wedge \cdots \wedge (z_N, r_N, z_N')$. Then by the semantics of C2RPQs, we know that there exists a mapping h from $\{z_1, z_1', \ldots, z_N, z_N'\}$ to $\{n_0, n_1, m_1, \ldots, n_K, m_K\}$ such that $h(x) = n_0$, $h(y) = n_K$ and for every $i \in \{1, \ldots, N\}$ there is a path ρ_i^h between $h(z_i)$ and $h(z_i')$ such that the sequence of edge labels λ_i^h associated to ρ_i^h, satisfies the expression r_i. We use the mapping h and the fact that K is sufficiently large compared with N to obtain a contradiction.

From h and $\varphi(x,y)$ we construct a graph $G_{h(\varphi(x,y))}$ as follows. Initially consider the values $h(z_1), h(z_1'), \ldots, h(z_N), h(z_N')$ as nodes in $G_{h(\varphi(x,y))}$. Now, for every conjunct (z_i, r_i, z_i') of $\varphi(x,y)$ let $\lambda_i^h = a_0 a_1 \ldots a_k$ be the sequence of edges (and inverse edges) in the path between $h(z_i)$ and $h(z_i')$ that satisfies the regular expression with inverse r_i when evaluated over G. Then, we include k fresh nodes $s_1, s_2, \ldots, s_k$ to $G_{h(\varphi(x,y))}$ and the following edges. Assuming that $s_0 = h(z_i)$ and that $s_{k+1} = h(z_i')$, then for every $j \in \{1, \ldots, k+1\}$:

– if $a_j = a$ or $a_j = b$ then add edge (s_{j-1}, a_j, s_j) to $G_{h(\varphi(x,y))}$, and
– if $a_j = a^-$ or $a_j = b^-$ then add edge (s_j, a_j, s_{j-1}) to $G_{h(\varphi(x,y))}$.

By the construction of $G_{h(\varphi(x,y))}$ it is easy to conclude that $(h(x), h(y)) = (n_0, n_K) \in [\![\varphi(x,y)]\!]_{G_{h(\varphi(x,y))}}$. We prove below that $(n_0, n_K) \notin [\![nexp]\!]_{G_{h(\varphi(x,y))}}$.

First notice that if n_0 and n_K are in different connected components in $G_{h(\varphi(x,y))}$, then clearly $(n_0, n_K) \notin [\![nexp]\!]_{G_{h(\varphi(x,y))}}$. Thus, assume that n_0 and n_K are in the same connected component of $G_{h(\varphi(x,y))}$. Notice that since n_0 and n_K are at distance K in G, then any *semi-path* (that is a path using forward and backward edges [6]) connecting n_0 and n_K in $G_{h(\varphi(x,y))}$ should have at least K edges (each edge either a forward or backward edge). Now, given that $\varphi(x,y)$ has N conjuncts and $K > N+1$ we know that everyone of the paths connecting n_0 and n_K in $G_{h(\varphi(x,y))}$ should contain a portion that was constructed by converting a sequence of edge labels of G into a *linear* semi-path in $G_{h(\varphi(x,y))}$. All this implies that every possible (semi) path from n_0 to n_K in $G_{h(\varphi(x,y))}$ should contain some intermediate nodes such that the sum of the out and in-degrees of the node is at most 2 (they are part of a line). On the other hand, given the semantics of NREs, every path that satisfies the expression $(a \cdot [b])^+$ should be such that all intermediate nodes (that is without considering the first and last node), should have in-degree at least 1 (one edge going into the node with label a) and out-degree at least 2 (one edge going out with label a, and another going out with label b), which implies that no path from n_0 to n_K in $G_{h(\varphi(x,y))}$ satisfies expression $(a \cdot [b])^+$. Thus we have that $(n_0, n_K) \in [\![\varphi(x,y)]\!]_{G_{h(\varphi(x,y))}}$ and then given that $\varphi(x,y)$ is a disjunct in $\psi(x,y)$, we also have that $(n_0, n_K) \in [\![\psi(x,y)]\!]_{G_{h(\varphi(x,y))}}$. But $(n_0, n_K) \notin [\![nexp]\!]_{G_{h(\varphi(x,y))}}$, which contradicts the fact that $\psi(x,y)$ expresses *nexp*. This completes the proof of the theorem. $\square$

4 Expressiveness of NREs with respect to acyclic C2RPQs

The reason why NREs do not capture C2RPQs is because they are *memoryless*, and that is why they cannot express CRPQs that form complex joins, in particular, *cyclic* joins. On the other hand, *acyclic* C2RPQs syntactically restrict C2RPQs by forbidding the existence of those cycles. This suggests that the expressive power of NREs may be better understood in terms of this class of queries. In fact, we prove in this section that NREs properly extend the class of acyclic UC2RPQs, under a mild syntactic restriction that asks for each disjunct of a UC2RPQ to be *connected*. However, in order to prove this we actually show something more interesting: the class of acyclic and connected UC2RPQs coincides with a natural syntactic restriction of the class of NREs.

In order to formally define acyclic C2RPQs, we need to recall the standard notion of the underlying undirected graph of a query. Consider a C2RPQ $\varphi(\bar{x})$ of the form $\exists \bar{y}((z_1, r_1, z_1') \wedge (z_2, r_2, z_2') \wedge \cdots \wedge (z_k, r_k, z_k'))$. The underlying graph of $\varphi(\bar{x})$ is $\mathcal{G}_{\varphi(\bar{x})} = (N, E)$, where $N = \bar{x} \cup \bar{y}$ and $E = \{\{z_i, z_i'\} \mid 1 \leq i \leq k\}$. That is, the nodes in $\mathcal{G}_{\varphi(\bar{x})}$ are the variables in $\varphi(\bar{x})$, and there is an edge between two (not necessarily distinct) variables if they occur in the same conjunct in $\varphi(\bar{x})$. Then we say that $\varphi(\bar{x})$ is *acyclic* if $\mathcal{G}_{\varphi(\bar{x})}$ has no cycles. Similarly, we say that $\varphi(\bar{x})$ is connected if $\mathcal{G}_{\varphi(\bar{x})}$ is a connected graph. We extend these definitions to UC2RPQs, and say that a UC2RPQ is acyclic and connected if every one of its disjuncts is acyclic and connected.

We shall prove that every acyclic and connected UC2RPQ can be expressed as an NRE. But, more interestingly, we can actually prove that this class of queries precisely coincides with a fragment of NREs, specifically, those NREs that do not allow the nesting operator to be used inside an expression of the form s^*. Formally, an NRE is *nesting-star alternation free*, if for every subexpression of *nexp* of the form s^*, it holds that s does not use the nesting operator. For instance, expressions n_1 and n_2 in Example 1 are nesting-star alternation free, while expression n_3 in the same example is not. We can prove that acyclic and connected UC2RPQs and nesting-star alternation-free NREs have the same expressive power.

Theorem 2.

(a) Let $\varphi(x, y)$ be an acyclic and connected binary UC2RPQ over Σ. Then $\varphi(x, y)$ can be expressed as a nesting-star alternation free NRE of linear size in $|\varphi(x, y)|$.

(b) Let nexp be a nesting-star alternation-free NRE over Σ. Then nexp can be expressed as an acyclic and connected binary UC2RPQ of exponential size in $|nexp|$.

Proof (idea). In order to prove (a) we use a technique which is similar to the *roll-up* of an acyclic conjunctive query, which has been used to provide decidability results for query containment in the context of description logics (see [13, 4]). In this case we roll-up the acyclic conjunctive query into an NRE.

Before going into the details of the proof of (1), we introduce the notion of an *inverse* of an NRE. The *inverse* of an NRE *exp* over Σ, denoted by exp^{-1}, is defined inductively as follows: (1) $\varepsilon^{-1} = \varepsilon$, (2) $a^{-1} = a^-$, for each $a \in \Sigma$, (2) $(a^-)^{-1} = a$, for each $a \in \Sigma$, (3) $(exp_1 \cdot exp_2)^{-1} = exp_2^{-1} \cdot exp_1^{-1}$, (4) $(exp^*)^{-1} = (exp^{-1})^*$, (5) $(exp_1 + exp_2)^{-1} = exp_1^{-1} + exp_2^{-1}$, and (6) $[exp]^{-1} = [exp]$. It is straightforward that for each graph G over Σ, it holds that $(u, v) \in [\![exp]\!]_G$ iff $(v, u) \in [\![exp^{-1}]\!]_G$, and moreover, exp^{-1} can be constructed in polynomial time from *exp*.

In order to prove (a), let $\varphi(x, y)$ be an acyclic and connected C2RPQ, and consider the graph $\mathcal{G}_{\varphi(x,y)}$ constructed by having variables of $\varphi(x, y)$ as nodes. Moreover, for every conjunct of the form (z_i, r_i, z_i') we add an undirected edge $\{z_i, z_i'\}$ with two labels $\ell_{(z_i,z_i')}(\{z_i, z_i'\}) = r_i$ and $\ell_{(z_i',z_i)}(\{z_i, z_i'\}) = (r_i)^{-1}$, where $(r_i)^{-1}$ is the *inverse* of r_i. (Notice that if r_i is a regular expression with inverse, then $(r_i)^{-1}$ is also a regular expression with inverse of size linear in the size of r.) Intuitively, label $\ell_{(z_i,z_i')}(\{z_i, z_i'\})$ describes the regular expression with inverse that allows to go from z_i to z_i', and $\ell_{(z_i',z_i)}(\{z_i, z_i'\})$ the regular expression with inverse that allows to go from z_i' to z_i, when evaluating the query $\varphi(x, y)$. Notice that $\mathcal{G}_{\varphi(x,y)}$ is a tree (since it is acyclic and connected).

Now, for every node u in $\mathcal{G}_{\varphi(x,y)}$ we define an NRE $\nu(u)$ as follows. First consider the graph $\mathcal{G}'_{\varphi(x,y)}$ obtained from $\mathcal{G}_{\varphi(x,y)}$ by deleting all the edges in the unique path between x and y. Notice that $\mathcal{G}'_{\varphi(x,y)}$ is no longer a tree, but a *forest*, and every node of $\mathcal{G}_{\varphi(x,y)}$ belongs to a unique tree in $\mathcal{G}'_{\varphi(x,y)}$. Let T_u be the tree in $\mathcal{G}'_{\varphi(x,y)}$ to which node u belongs, and assume that T_u is a rooted tree, with root u. Then for every node v in T_u construct an expression $\tau_u(v)$ recursively as follows:

- if v is a leaf in T_u then $\tau_u(v) = \varepsilon$
- else, if v has $v_1, \ldots, v_k$ as children in T_u then

$$\tau_u(v) \;=\; [\ell_{(v,v_1)}(\{v, v_1\}) \cdot \tau_u(v_1)] \cdot \;\cdots\; \cdot [\ell_{(v,v_k)}(\{v, v_k\}) \cdot \tau_u(v_k)]$$

Finally we say that $\nu(u) = \tau_u(u)$. Notice that the size of $\nu(u)$ is linear in the size of the tree T_u. Also notice that the fact that $\mathcal{G}_{\varphi(x,y)}$ is actually a tree is crucial for defining $\nu(u)$ for every node in $\mathcal{G}_{\varphi(x,y)}$.

Now, with $\nu(u)$ for every possible variable of $\varphi(x, y)$ we are ready to describe the NRE that defines $\varphi(x, y)$. Let $x, u_1, u_2, \ldots, u_k, y$ be the unique path in $\mathcal{G}_{\varphi(x,y)}$ from x to y. Then we consider the expression

$$\begin{aligned} \mathit{nexp} \;=\; \nu(x) \cdot \ell_{(x,u_1)}(\{x, u_1\}) \cdot \nu(u_1) \cdot \ell_{(u_1,u_2)}(\{u_1, u_2\}) \cdot \nu(u_2) \cdot \;\cdots \\ \cdots\; \cdot \nu(u_k) \cdot \ell_{(u_k,y)}(\{u_k, y\}) \cdot \nu(y). \end{aligned}$$

It is not difficult to see that *nexp* is of size linear in the size of $\varphi(x, y)$. Moreover, it is not difficult to prove by induction that for every graph G it holds that $[\![\mathit{nexp}]\!]_G = [\![\varphi(x, y)]\!]_G$ (see the details in the appendix). Essentially, expression *nexp* is testing for the existence of the unique path (in the query $\varphi(x, y)$) between x and y, and also existentially testing for all the branches that start from the variables in this path as described in $\varphi(x, y)$. In particular, let $\psi_u(u)$ be the formula obtained by considering the conjunction of all formulas (z_i, r_i, z'_i) obtained from the edges of graph T_u with all variables except for u existentially quantified. It is not difficult to prove that for every graph G, it holds that $(a) \in [\![\psi_u(u)]\!]_G$ if and only if $(a, a) \in [\![\nu(u)]\!]_G$ (details can be found in the appendix).

Now if we start with a UC2RPQ of the form $\varphi_1(x, y) \vee \cdots \vee \varphi_k(x, y)$, then we can apply the above described process to every disjunct to construct NREs $\mathit{nexp}_1, \ldots, \mathit{nexp}_k$ and then consider the expression $\mathit{nexp}_1 + \cdots + \mathit{nexp}_k$.

We now prove (b). Given an NRE *nexp*, we construct a UC2RPQ $\varphi_{\mathit{nexp}}(x, y)$ by an inductive process on the construction of NREs. We consider a construction in two steps. First for an NRE *nexp* that do not use the nesting operator we just have that $\varphi_{\mathit{nexp}}(x, y) = (x, \mathit{nexp}, y)$. Now for the NRE $\mathit{nexp} = [\mathit{nexp}']$ we construct the formula

$$\varphi_{\mathit{nexp}}(x, y) \;=\; \exists u \Big(\varphi_{\mathit{nexp}'}(x, u) \Big) \wedge (x, \varepsilon, y)$$

Notice that in this case we have used the conjunct (x, ε, y) to simulate an equality just to ensure that the constructed formula is binary (one can also eliminate this conjunct and consider unary formulas but this only complicates the inductive argument). We only have two remaining cases:

- if $nexp = nexp_1 \cdot nexp_2$ with $nexp_1$ or $nexp_2$ NREs that use the nesting operator we have that
$$\varphi_{nexp}(x,y) = \exists u \Big(\varphi_{nexp_1}(x,u) \wedge \varphi_{nexp_2}(u,y) \Big).$$
- if $nexp = nexp_1 + nexp_2$ with $nexp_1$ or $nexp_2$ NREs that use the nesting operator we have that
$$\varphi_{nexp}(x,y) = \varphi_{nexp_1}(x,y) \vee \varphi_{nexp_2}(x,y).$$

Notice that we do not need to consider the case $nexp = (nexp')^*$ since we are assuming that the expression is nesting-star alternation free. Also notice that the formula constructed is not exactly a UC2RPQ (since it does not necessarily have the disjunction always as a top-level logical operator) but can be transformed into a UC2RPQ by distribution. This implies that given a NRE *nexp* the UC2RPQ constructed by this process is of size exponential in the size of *nexp*. Details on the correctness of the transformation can be found in the appendix. □

We left open whether the exponential blow-up when translating from nesting-star alternation free NREs into acyclic and connected UC2RPQs is necessary. We end up this section by showing that both unions and the connectedness condition are essential in Theorem 2; that is, that nesting-star alternation free NREs are neither captured by the class of acyclic (but not necessarily connected) UC2RPQs nor by the class of acyclic and connected C2RPQs. In view of Theorem 2, it is enough to prove the following simple proposition. The proof can be found in the appendix.

Proposition 3. *The following holds:*

(a) There is a binary and acyclic CRPQ that cannot be expressed as a connected UC2RPQ.
(b) There is a binary, acyclic and connected UCRPQ that cannot be expressed as a C2RPQ.

Notice that while part (a) is very simple to prove, part (b) is not as straightforward as it may seem at first sight, since C2RPQs allow for some form of disjunction by means of the union (+) operator of regular expressions.

5 Concluding remarks

While both C2RPQs and NREs strictly subsume the class of acyclic C2RPQs, only NREs share the good query evaluation properties of acyclic C2RPQs. In fact, it is known that the query evaluation problem for both NREs and acyclic C2RPQs can be solved linearly in both the size of the graph database and the query, but it is intractable for CRPQs. Hence, our results suggest NREs to be a good language for graph databases, exhibiting a good balance between expressiveness and efficiency.

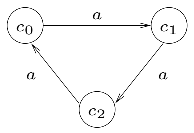
References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. Pablo Barceló, Carlos A. Hurtado, Leonid Libkin, and Peter T. Wood. Expressive languages for path queries over graph-structured data. In *PODS*, pages 3–14, 2010.
3. D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Containment of conjunctive regular path queries with inverse. In *7th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 176–185, 2000.
4. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive query containment and answering under description logic constraints. *ACM Trans. Comput. Log.*, 9(3), 2008.
5. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Rewriting of regular expressions and regular path queries. In *PODS*, pages 194–204, 1999.
6. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing for regular path queries with inverse. In *PODS*, pages 58–66, 2000.
7. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Simplifying schema mappings. In *ICDT*, pages 114–125, 2011.
8. Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.
9. M. Consens and A.O. Mendelzon. GraphLog: A visual formalism for real life recursion. In *9th ACM Symposium on Principles of Database Systems (PODS)*, pages 404–416, 1990.
10. D2R DBLP bibliography database hosted at L3S research center. `http://dblp.l3s.de/d2r/`.
11. George H. L. Fletcher, Marc Gyssens, Dirk Leinders, Jan Van den Bussche, Dirk Van Gucht, Stijn Vansummeren, and Yuqing Wu. Relative expressive power of navigational querying on graphs. In *ICDT*, pages 197–207, 2011.
12. D Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *17th ACM Symposium on Principles of Database Systems (PODS)*, pages 139–148, 1998.
13. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for description logics with transitive roles. In *Description Logics*, 2006.
14. P. Hayes. RDF Semantics, W3C Recommendation. `http://www.w3.org/TR/rdf-mt`, February 2004.
15. S. DeRose. J. Clark. Xml path language (xpath). W3C Recommendation, November 1999, `http://www.w3.org/TR/xpath`.
16. Christos Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.
17. Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. nSPARQL: A navigational language for RDF. *J. Web Sem.*, 8(4):255–270, 2010.
18. Mihalis Yannakakis. Algorithms for acyclic database schemes. In *Proc. VLDB 1981*, pages 82–94. IEEE Computer Society, 1981.

Appendix

Proof of Proposition 2

Consider the CRPQ $\varphi(x, y)$ given by the expression $(x, a, y) \wedge (y, a, x)$. We next show that $\varphi(x, y)$ cannot be expressed as an NRE. Consider the following graph G_1 over alphabet Σ:



Notice that $[\![\varphi(x, y)]\!]_{G_1} = \emptyset$, that is, there are no values u, v such that $G_1 \models \varphi(u, v)$. We show that every NRE *nexp* is such that $[\![nexp]\!]_{G_1} \neq \emptyset$ which proves that φ cannot be expressed as an NRE. In particular, we prove something stronger: for every NRE *nexp* there exists values $x_0, x_1, x_2, y_0, y_1, y_2 \in \{c_0, c_1, c_2\}$ such that $(c_0, x_0), (c_1, x_1), (c_2, x_2), (y_0, c_0), (y_1, c_1), (y_2, c_2) \in [\![nexp]\!]_{G_1}$. We prove this by induction on the construction of the expression.

We have three base cases:

1. if $nexp = \varepsilon$ then $[\![nexp]\!]_{G_1} = \{(c_0, c_0), (c_1, c_1), (c_2, c_2)\}$, and the property holds.
2. if $nexp = a$ then $[\![nexp]\!]_{G_1} = \{(c_0, c_1), (c_1, c_2), (c_2, c_0)\}$, and the property holds.
3. if $nexp = a^-$ then $[\![nexp]\!]_{G_1} = \{(c_0, c_2), (c_1, c_0), (c_2, c_1)\}$, and the property holds.

Assume now that $nexp_1$ and $nexp_2$ satisfy the property. We have several cases.

4. if $nexp = nexp_1 + nexp_2$ then $[\![nexp]\!]_{G_1} = [\![nexp_1]\!]_{G_1} \cup [\![nexp_2]\!]_{G_2}$ and then the property clearly holds.
5. assume that $nexp = nexp_1 \cdot nexp_2$. Then we know that that there exist values x_i's such that $(c_i, x_i) \in [\![nexp_1]\!]_{G_1}$ for $i \in \{0, 1, 2\}$. Notice that every $x_i \in \{c_0, c_1, c_2\}$ and thus, we know that there exists values u_0, u_1, u_2 such that $(x_i, u_i) \in [\![nexp_2]\!]_{G_1}$ for $i \in \{0, 1, 2\}$. This implies that $(c_i, u_i) \in [\![nexp]\!]_{G_1}$ for $i \in \{0, 1, 2\}$. By a symmetric argument we can show that there exists values v_0, v_1, v_2 such that $(v_i, c_i) \in [\![nexp]\!]_{G_1}$ for $i \in \{0, 1, 2\}$.
6. if $nexp = nexp_1^*$ the property holds by using 1), 4), 5) and the definition of $[\![nexp_1^*]\!]_{G_1}$.
7. if $nexp = [nexp_1]$ then since $nexp_1$ satisfies the property we have $[\![nexp]\!]_{G_1} = \{(c_0, c_0), (c_1, c_1), (c_2, c_2)\}$ and then *nexp* also satisfies the property.

This completes the proof. □

Proof of Theorem 2

Before going into the details of the proof we recall the notion of an *inverse* of an NRE. The *inverse* of an NRE *exp* over Σ, denoted by exp^{-1}, is an expression such that for each graph G over Σ, it holds that $(u, v) \in [\![exp]\!]_G$ iff $(v, u) \in [\![exp^{-1}]\!]_G$. As we have discussed, exp^{-1} can be constructed in polynomial time from exp.

We now prove (a). For completeness, we repeat the construction given in the body of the paper. Let $\varphi(x, y)$ be an acyclic and connected C2RPQ, and consider the graph $\mathcal{G}_{\varphi(x,y)}$ constructed by having variables of $\varphi(x, y)$ as nodes. Moreover, for every conjunct of the form (z_i, r_i, z_i') we add an undirected edge $\{z_i, z_i'\}$ with two labels $\ell_{(z_i,z_i')}(\{z_i, z_i'\}) = r_i$ and $\ell_{(z_i',z_i)}(\{z_i, z_i'\}) = (r_i)^{-1}$, where $(r_i)^{-1}$ is the *inverse* of r_i. (Notice that if r_i is a regular expression with inverse, then $(r_i)^{-1}$ is also a regular expression with inverse of size linear in the size of r.) Intuitively, label $\ell_{(z_i,z_i')}(\{z_i, z_i'\})$ describes the regular expression with inverse that allows to go from z_i to z_i', and $\ell_{(z_i',z_i)}(\{z_i, z_i'\})$ the regular expression with inverse that allows to go from z_i' to z_i, when evaluating the query $\varphi(x, y)$. Notice that $\mathcal{G}_{\varphi(x,y)}$ is a tree (since it is acyclic and connected).

Now, for every node u in $\mathcal{G}_{\varphi(x,y)}$ we define an NRE $\nu(u)$ as follows. First consider the graph $\mathcal{G}'_{\varphi(x,y)}$ obtained from $\mathcal{G}_{\varphi(x,y)}$ by deleting all the edges in the unique path between x and y. Notice that $\mathcal{G}'_{\varphi(x,y)}$ is no longer a tree, but a *forest*, and every node of $\mathcal{G}_{\varphi(x,y)}$ belongs to a unique tree in $\mathcal{G}'_{\varphi(x,y)}$. Let T_u be the tree in $\mathcal{G}'_{\varphi(x,y)}$ to which node u belongs, and assume that T_u is a rooted tree, with root u. Then for every node v in T_u construct an expression $\tau_u(v)$ recursively as follows:

– if v is a leaf in T_u then $\tau_u(v) = \varepsilon$
– else, if v has $v_1, \ldots, v_k$ as children in T_u then

$$\tau_u(v) \;=\; [\ell_{(v,v_1)}(\{v, v_1\}) \cdot \tau_u(v_1)] \cdot \;\cdots\; \cdot [\ell_{(v,v_k)}(\{v, v_k\}) \cdot \tau_u(v_k)]$$

Finally we say that $\nu(u) = \tau_u(u)$. Notice that the size of $\nu(u)$ is linear in the size of the tree T_u. Also notice that the fact that $\mathcal{G}_{\varphi(x,y)}$ is actually a tree is crucial for defining $\nu(u)$ for every node in $\mathcal{G}_{\varphi(x,y)}$.

Now, with $\nu(u)$ for every possible variable of $\varphi(x, y)$ we are ready to describe the NRE that defines $\varphi(x, y)$. Let $x, u_1, u_2, \ldots, u_k, y$ be the unique path in $\mathcal{G}_{\varphi(x,y)}$ from x to y. Then we consider the expression

$$\begin{aligned} nexp \;=\; \nu(x) \cdot \ell_{(x,u_1)}(\{x, u_1\}) \cdot \nu(u_1) \cdot \ell_{(u_1,u_2)}(\{u_1, u_2\}) \cdot \nu(u_2) \cdot\; \cdots \\ \cdots\; \cdot \nu(u_k) \cdot \ell_{(u_k,y)}(\{u_k, y\}) \cdot \nu(y). \end{aligned}$$

It is not difficult to see that $nexp$ is of size linear in the size of $\varphi(x, y)$. Moreover, it is not difficult to prove by induction that for every graph G it holds that $[\![nexp]\!]_G = [\![\varphi(x, y)]\!]_G$ (see the details in the appendix). Essentially, expression $nexp$ is testing for the existence of the unique path (in the query $\varphi(x, y)$) between x and y, and also existentially testing for all the branches that start from the variables in this path as described in $\varphi(x, y)$. In particular, let $\psi_u(u)$ be the formula obtained by considering the conjunction of all formulas (z_i, r_i, z_i') obtained from the edges of graph T_u with all variables except for u existentially quantified. It can be easily proved by induction in the construction of $\nu(u)$ that for every graph G, it holds that $(a) \in [\![\psi_u(u)]\!]_G$ if and only if $(a, a) \in [\![\nu(u)]\!]_G$. Moreover, it is easy to see that if $(x, r_0, u_1), (u_1, r_1, u_2), \ldots, (u_k, r_k, y)$ are the conjuncts defining the single path from x to y in $\varphi(x, y)$, then $\varphi(x, y)$ is equivalent to the formula

$$\exists u_1 \cdots \exists u_k \Bigg(\psi_x(x) \wedge (x, r, u_1) \wedge \psi_{u_1}(u_1) \wedge (u_1, r_1, u_2) \wedge \psi_{u_2}(u_2) \cdots \\ \cdots \wedge \psi_{u_k}(u_k) \wedge (u_k, r_k, y) \wedge \psi_y(y) \Bigg)$$

From this we obtain that our construction of *nexp* is correct and then *nexp* is equivalent to $\varphi(x, y)$. Notice that the expression *nexp* constructed above, uses several levels of nesting (that essentially depend on the level of branching in $\varphi(x, y)$) but does not use the Kleene-star operator over the nesting operator, and thus, it is nesting-star alternation free.

Now if we start with a UC2RPQ of the form $\varphi_1(x, y) \vee \cdots \vee \varphi_k(x, y)$, then we can apply the above described process to every disjunct to construct NREs $\mathit{nexp}_1, \ldots, \mathit{nexp}_k$ and then consider the expression $\mathit{nexp}_1 + \cdots + \mathit{nexp}_k$.

We now prove (b). For completeness, we repeat the construction given in the body of the paper. Given an NRE *nexp*, we construct a UC2RPQ $\varphi_{\mathit{nexp}}(x, y)$ by an inductive process on the construction of NREs. We consider a construction in two steps. First for an NRE *nexp* that do not use the nesting operator we just have that $\varphi_{\mathit{nexp}}(x, y) = (x, \mathit{nexp}, y)$. Now for the NRE $\mathit{nexp} = [\mathit{nexp}']$ we construct the formula

$$\varphi_{\mathit{nexp}}(x, y) \;=\; \exists u \Big(\varphi_{\mathit{nexp}'}(x, u) \Big) \wedge (x, \varepsilon, y)$$

Notice that in this case we have used the conjunct (x, ε, y) to simulate an equality just to ensure that the constructed formula is binary (one can also eliminate this conjunct and consider unary formulas but this only complicates the inductive argument). We only have two remaining cases:

– if $\mathit{nexp} = \mathit{nexp}_1 \cdot \mathit{nexp}_2$ with nexp_1 or nexp_2 NREs that use the nesting operator we have that

$$\varphi_{\mathit{nexp}}(x, y) = \exists u \Big(\varphi_{\mathit{nexp}_1}(x, u) \wedge \varphi_{\mathit{nexp}_2}(u, y) \Big).$$

– if $\mathit{nexp} = \mathit{nexp}_1 + \mathit{nexp}_2$ with nexp_1 or nexp_2 NREs that use the nesting operator we have that

$$\varphi_{\mathit{nexp}}(x, y) = \varphi_{\mathit{nexp}_1}(x, y) \vee \varphi_{\mathit{nexp}_2}(x, y).$$

Notice that we do not need to consider the case $\mathit{nexp} = (\mathit{nexp}')^*$ since we are assuming that the expression is nesting-star alternation free. Also notice that the formula constructed is not exactly a UC2RPQ (since it does not necessarily have the disjunction always as a top-level logical operator) but can be transformed into a UC2RPQ by distribution. This implies that given a NRE *nexp* the UC2RPQ constructed by this process is of size exponential in the size of *nexp*.

We show next that $[\![nexp]\!]_G = [\![\varphi_{nexp}(x,y)]\!]_G$ for all G by using an inductive argument. If *nexp* does not mention the nesting operator, then *nexp* is a regular expression with inverse, and it is clear that $[\![nexp]\!]_G = [\![(x, nexp, y)]\!]_G$ for every G. Now assume that $[\![nexp']\!]_G = [\![\varphi_{nexp'}(x,y)]\!]_G$ for every G and let $nexp = [nexp']$. Then by definition of NREs, we have that $[\![[nexp']]\!]_G = \{(a,a) \mid$ there exists b such that $(a,b) \in [\![nexp']\!]_G\}$. Thus, by applying the induction hypothesis we have that

$$[\![[nexp']]\!]_G = \{(a,a) \mid \text{there exists } b \text{ such that } (a,b) \in [\![\varphi_{nexp'}(x,y)]\!]_G\}.$$

which implies that

$$\begin{aligned}[\![[nexp']]\!]_G = \{(a,a) \mid a \in [\![\exists u(\varphi_{nexp'}(x,u))]\!]_G\} = \\ \{(a,a) \mid (a,a) \in [\![\exists u(\varphi_{nexp'}(x,u)) \wedge (x,\varepsilon,y)]\!]_G\} = [\![\varphi_{nexp}(x,y)]\!]_G.\end{aligned}$$

and then $[\![nexp]\!]_G = [\![\varphi_{nexp}(x,y)]\!]_G$. The cases $nexp = nexp_1 \cdot nexp_2$ and $nexp = nexp_1 + nexp_2$ are straightforward. □

5.1 Proof of Proposition 3

The proof of part (a) is simple: we only have to consider alphabet $\Sigma = \{a,b\}$ and the query $Q(x,y) := \exists z \exists u((x,a,z) \wedge (u,b,y))$ over Σ. Clearly, Q is a binary and acyclic CRPQ. Notice, on the other hand, that Q is not connected. We claim that Q cannot be expressed as a connected UC2RPQ $Q'(x,y)$. Assume otherwise. Consider the graph database G over Σ whose set of nodes is $\{1,2,3,4\}$, there is an a-labeled edge in G from 1 to 2, and there is an b-labeled edge in G from 3 to 4. Clearly, $(1,4) \in [\![Q]\!]_G$ but $(1,4) \notin [\![Q']\!]_G$. This is because each disjunct of Q' is connected, while 1 and 4 belong to different connected components of G. The latter contradicts the fact that Q and Q' are equivalent.

Now we prove part (b). Consider alphabet $\Sigma = \{a,b,c,d,e,f\}$ and the query

$$Q(x,y) \;=\; \exists u \exists v((x,a,u) \wedge (u,b,v) \wedge (u,c,y)) \vee \exists u \exists v((x,d,u) \wedge (u,e,v) \wedge (u,f,y)).$$

Clearly, Q is a binary, acyclic and connected UCRPQ. We claim that Q cannot be expressed as a C2RPQ $Q'(x,y)$. Assume for the sake of contradiction that this is the case. We construct below a graph G over Σ such that $[\![Q]\!]_G \neq [\![Q']\!]_G$, which is a contradiction.

Consider graph databases G_1 and G_2 over Σ, such that the set of edges of G_1 is $\{(1,a,2),(2,b,3),(2,c,4)\}$ and that of G_2 is $\{(1,d,2),(2,e,3),\ (2,f,4)\}$. Clearly, $(1,4) \in [\![Q]\!]_{G_1}$ and $(1,4) \in [\![Q]\!]_{G_2}$. Hence $(1,4) \in [\![Q']\!]_{G_1}$ and $(1,4) \in [\![Q']\!]_{G_2}$. Assume without loss of generality that the conjunctive part of Q' is of the form $(z_1, r_1, z_1') \wedge \cdots \wedge (z_N, r_N, z_N')$. Then by the semantics of C2RPQs, we know that there exists a mapping h_1 (resp. h_2) from $\{z_1, z_1', \ldots, z_N, z_N'\}$ to $\{1,2,3,4\}$ such that $h_1(x) = 1$, $h_1(y) = 4$ (resp. $h_2(x) = 1$, $h_2(y) = 4$) and for every $i \in \{1, \ldots, N\}$ it holds that there is a path $\rho_i^{h_1}$ (resp. $\rho_i^{h_2}$) between $h_1(z_i)$ and $h_1(z_i')$ in G_1 (resp. between $h_2(z_i)$ and $h_2(z_i')$ in G_2) such that the sequence of edge labels $\lambda_i^{h_1}$ (resp. $\lambda_i^{h_2}$) associated to $\rho_i^{h_1}$ (resp. $\rho_i^{h_2}$), satisfies r_i.

From h_1 and Q' we construct a graph database $G_{h_1(Q')}$ over alphabet $\Sigma_1 = \{a, b, c, \varepsilon\}$ as follows. Initially consider the variables $z_1, z_1', \ldots, z_N, z_N'$ as nodes in $G_{h_1(Q')}$. Now, for every conjunct (z_i, r_i, z_i') of $Q'(x, y)$ assume that $\lambda_i^{h_1} = a_0 a_1 \ldots a_k$ is the sequence of edge labels (and inverse edge labels) in the path between $h_1(z_i)$ and $h_1(z_i')$ that satisfies the regular expression with inverse r_i when evaluated over G_1. We consider two cases:

1. Assume first that $\lambda_i^{h_1}$ is the empty string. Then we connect z_i to z_i' in G_1 by an edge labeled ε.
2. Assume otherwise. Then, we include k fresh nodes $s_1, s_2, \ldots, s_k$ to $G_{h_1(Q')}$ and the following edges. Assuming that $s_0 = z_i$ and that $s_{k+1} = z_i'$, then for every $j \in \{1, \ldots, k+1\}$:
   - if $a_j = a$, $a_j = b$ or $a_j = c$ then add edge (s_{j-1}, a_j, s_j) to $G_{h_1(Q')}$, and
   - if $a_j = a^-$, $a_j = b^-$ or $a_j = c^-$ then add edge (s_j, a_j, s_{j-1}) to $G_{h_1(Q')}$.

We denote by $\nu_i^{h_1}$ the path $z_i = s_0, a_0, s_1, a_1, s_2, \ldots, s_k, a_k, s_{k+1} = z_i'$ (or z_i, ε, z_i', if $\lambda_i^{h_1}$ is the empty string). In an analogous way, we construct a graph database $G_{h_2(Q')}$ over $\Sigma_2 = \{d, e, f, \varepsilon\}$ from h_2 and Q'.

By the construction of $G_{h_1(Q')}$ it is easy to conclude that $(x, y) \in [\![Q']\!]_{G_{h_1(Q')})}$, if we assume that ε-labeled edges contribute to labels of paths as the empty string. In fact, the identity mapping from the variables of Q' into the nodes of G_1 can be used to witness the values for the variables of Q' when evaluated over G_1. For the same reason, $(x, y) \in [\![Q']\!]_{G_{h_2(Q')})}$.

Consider G to be the graph database over Σ that is obtained from $G_{h_1(Q')}$ by iteratively applying the following process:

- For each $1 \leq i \leq N$ such that $z_i = x$ or $z_i' = x$ and the path $\nu_i^{h_1}$ belongs to G, replace such path with $\nu_i^{h_2}$.
- If there is an ε-labeled edge connecting x to a node y, delete such an edge and declare y to be equal to x.

Notice that the symbols of $\Sigma_2 \setminus \{\varepsilon\}$ appear in G only in those paths that connect x with a variable of Q' without going through an intermediate variable of Q'. Further, every edge that leaves or starts in x in G is labeled in $\Sigma_2 \setminus \{\varepsilon\}$.

It is easy to see that for every $1 \leq i \leq N$ it is the case that node z_i is connected to z_i' in G by a path that satisfies r_i. Hence $(x, y) \in [\![Q']\!]_G$. We prove next that $(x, y) \notin [\![Q]\!]_G$. Assume otherwise. Since every edge that leaves x in G is labeled in $\Sigma_2 \setminus \{\varepsilon\}$, it must be the case that $(G, (x, y)) \models \exists u \exists v((x, d, u) \wedge (u, e, v) \wedge (u, f, y))$. We consider three possibilities:

1. There is a path $xdsfy$ in G, where s is a fresh node (i.e. s has been introduced as an internal node in a path of the form $\nu_i^{h_2}$), and s has an outgoing edge labeled e. But this is not possible since fresh nodes have exactly one outgoing edge, and in this case s must have one outgoing edge labeled e and another one labeled f.

2. There is a path $xdzfy$ in G, where z is a variable of Q', and z has an outgoing edge labeled e. But then either $x = z$ or $y = x$, since the only edges of G labeled in $\Sigma_2 \setminus \{\varepsilon\}$ that connect two nodes of G named after a variable of Q' are the ones that have at least one endpoint in x. This implies in the first that there is a path labeled d from 1 into itself in G_2, and in the second case that there is a path labeled df from 1 into itself in G_2. Clearly none of these cases can happen.

This is our desired contradiction. $\square$