Expressiveness and Static Analysis of Extended Conjunctive Regular Path Queries

Dominik D. Freydenberger and Nicole Schweikardt

Institut für Informatik, Goethe-Universität, Frankfurt am Main, Germany,
freydenberger@em.uni-frankfurt.de, schweika@informatik.uni-frankfurt.de

Abstract. We study the expressiveness and the complexity of static analysis of *extended conjunctive regular path queries (ECRPQs)*, introduced by Barceló et al. (PODS '10). ECRPQs are an extension of *conjunctive regular path queries (CRPQs)*, a well-studied language for querying graph structured databases. Our first main result shows that query containment and equivalence of a CRPQ in an ECRPQ is undecidable. This settles one of the main open problems posed by Barceló et al. As a second main result, we prove a non-recursive succinctness gap between CRPQs and the CRPQ-expressible fragment of ECRPQs. Apart from this, we develop a tool for proving inexpressibility results for CRPQs and ECRPQs. In particular, this enables us to show that there exist queries definable by regular expressions with backreferencing, but not expressible by ECRPQs.

1 Introduction

Many application areas (e. g., concerning the Semantic Web or biological applications) consider *graph structured data*, where the data consists of a finite set of nodes connected by labeled edges. For querying such data, one usually needs to specify types of paths along which nodes are connected. A widely studied class of queries for graph structured databases are the *conjunctive regular path queries (CRPQs)* (cf., e. g., [5, 7, 8]), where types of paths can be described by regular expressions specifying labels along the paths. For modern applications, however, also more expressive query languages are desirable, allowing not only to specify regular properties of path labels, but also to compare paths based on, e. g., their lengths, labels, or similarity.

To start a formal investigation of such concepts, Barceló et al. [4] introduced the class of *extended conjunctive regular path queries (ECRPQs)*, allowing to use not only regular languages to express properties of individual paths, but also *regular relations* among several paths, capable of expressing certain associations between paths. The authors of [4] investigated the complexity of query evaluation and static analysis of ECRPQs. While query containment is known to be decidable and EXPSPACE-complete for CRPQs [8, 5], it was shown to be undecidable for ECRPQs [4]. However, checking containment of an ECRPQ in a CRPQ still is decidable and EXPSPACE-complete [4]. (Un)Decidability of checking containment (or, equivalence) of a CRPQ in an ECRPQ was posed as an open question in [4].

In the present paper, we answer this question by showing that containment of a CRPQ in an ECRPQ is undecidable — even if the ECRPQ is, in fact, a CRPQ extended only by relations for checking equality of path labels (or, similarly, equal lengths of paths). Our proof proceeds by (a) simulating Turing machine runs by so-called *H-systems*, a concept from formal language theory generalizing pattern languages, and (b) using CRPQs and ECRPQs to represent languages described by H-systems. Our proof generalizes to (i) the case where one of the two queries is fixed, (ii) the case where all queries are Boolean and acyclic, and (iii) the problem of deciding equivalence rather than containment of CRPQs and ECRPQs.

Apart from the static analysis of queries, the present paper also investigates the expressiveness and succinctness of ECRPQs. Using the machinery developed for proving our undecidability results concerning static analysis, we show that CRPQ-definability of a given ECRPQ is undecidable, and that there is no recursive function f such that every CRPQ-definable ECRPQ of length n is equivalent to a CRPQ of length $f(n)$.

Concerning the expressivity of (E)CRPQs, to the best of our knowledge, tools for showing inexpressibility results have not been presented in the literature yet. We develop such tools, enabling us to show, for example, that no ECRPQ-query can return exactly those tuples of nodes between which there is a path whose length is a composite number (i. e., a number of the from nm for $n, m \geq 2$). Since these paths can be easily described by a *regular expression with backreferencing* (cf. [1]) of the form $(a\, a^+)\%x\, x^+$, this refutes a claim of [4] stating that all regular expressions with backreferencing can be expressed by ECRPQs.

Structure of the paper. We start with the necessary notations and definitions in Section 2 where, in particular, the syntax and semantics of ECRPQs (and restrictions thereof) are defined. Section 3 is devoted to the static analysis of ECRPQs and CRPQs, showing that containment and equivalence of CRPQs in ECRPQs are undecidable. Section 4 investigates the relative succinctness between CRPQs and CRPQ-expressible ECRPQs and provides tools for proving limitations to the expressive power of CRPQs and ECRPQs. Due to space limitations, many technical details of the proofs had to be deferred to the full version.

2 Preliminaries

Let $\mathbb{N}$ denote the set of non-negative integers. We denote the *empty word* by ε. Let A, B be alphabets. A *morphism* (between A^* and B^*) is a function $h : A^* \to B^*$ with $h(uv) = h(u)h(v)$ for all $u, v \in A^*$. For every word $w \in A^*$, $|w|$ stands for the length of w, and for every letter $a \in A$, $|w|_a$ denotes the number of occurrences of a in w.

DB-Graphs and Queries. A *Σ-labeled db-graph* is a directed graph $G = (V, E)$, where V is a finite set of nodes, and $E \subseteq V \times \Sigma \times V$ is a finite set of directed edges with labels from Σ. A *path* ρ between two nodes v_0 and v_n in G with $n \geq 0$ is a sequence $v_0 a_1 v_1 \cdots v_{n-1} a_n v_n$ with $v_0, \ldots, v_n \in V$, $a_1, \ldots, a_n \in \Sigma$,

and $(v_i, a_{i+1}, v_{i+1}) \in E$ for $0 \leq i < n$. We define the *label* $\lambda(\rho)$ of the path ρ by $\lambda(\rho) := a_1 \cdots a_n$. Furthermore, for every $v \in V$, we define the *empty path* $v\varepsilon v$, with $\lambda(v\varepsilon v) = \varepsilon$.

A central concept considered in the present paper are *regular relations* (cf. [4] and the references therein). Let Σ be a finite alphabet, let $\perp$ be a new symbol with $\perp \notin \Sigma$, and let $\Sigma_\perp := \Sigma \cup \{\perp\}$. Let $\overline{w} = (w_1, \ldots, w_k) \in (\Sigma^*)^k$, where $w_i = a_{i,1} \cdots a_{i,|w_i|}$ (and all $a_{i,j} \in \Sigma$). We define the string $[\overline{w}] \in (\Sigma_\perp^*)^k$ by $[\overline{w}] := b_1 \cdots b_n$, where n is the maximum of all $|w_i|$, and $b_j := (b_{j,1}, \ldots, b_{j,k})$, with $b_{j,i} = a_{i,j}$ if $j \leq |w_i|$, and $b_{j,i} = \perp$ if $j > |w_i|$. In other words, $[\overline{w}]$ is obtained by aligning all w_i to the left, and padding the unfilled space with $\perp$ symbols. A k-ary relation $R \subseteq (\Sigma^*)^k$ is called *regular* if the language $\{[\overline{r}] \mid \overline{r} \in R\}$ is regular.

Obviously, every regular language is a (unary) regular relation. In addition to this, the present paper focuses on the following k-ary regular relations ($k \geq 2$):

1. the *equality relation* $\mathrm{eq}_{\cdot} = \{(w_1, \ldots, w_k) \mid w_1 = \ldots = w_k\}$,
2. the *length equality relation* $\mathrm{el} := \{(w_1, \ldots, w_k) \mid |w_1| = \ldots = |w_k|\}$.

Note that each of these relations needs to be defined w. r. t. a finite alphabet Σ, which we usually omit for the sake of brevity.

We now define ECRPQs and CPRQs, following the definitions from [4]. Fix a countable set of *node variables* and a countable set of *path variables*. Let Σ be a finite alphabet. An *extended conjunctive regular path query (ECRPQ)* Q over Σ is an expression of the form

$$\mathrm{Ans}(\overline{z}, \overline{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \bigwedge_{1 \leq j \leq l} R_j(\overline{\omega}_j), \tag{1}$$

such that $m \geq 1$, $l \geq 0$, and

1. each R_j is a regular expression that defines a regular relation over Σ,
2. $\overline{x} = (x_1, \ldots, x_m)$ and $\overline{y} = (y_1, \ldots, y_m)$ are tuples of (not necessarily distinct) node variables,
3. $\overline{\pi} = (\pi_1, \ldots, \pi_m)$ is a tuple of distinct path variables,
4. $\overline{\omega}_1, \ldots, \overline{\omega}_l$ are tuples of path variables, such that each $\overline{\omega}_j$ is a tuple of variables from $\overline{\pi}$, of the same arity as R_j,
5. $\overline{z}$ is a tuple of node variables among $\overline{x}$, $\overline{y}$, and
6. $\overline{\chi}$ is a tuple of path variables among those in $\overline{\pi}$.

The expression $\mathrm{Ans}(\overline{z}, \overline{\chi})$ is the *head*, and the expression to the right of $\leftarrow$ is the *body* of Q. If $\overline{z}$ and $\overline{\chi}$ are the empty tuple (i. e., the head is of the form $\mathrm{Ans}()$), Q is a *Boolean query*. The *relational* part of an ECRPQ Q is $\bigwedge_{1 \leq i \leq m}(x_i, \pi_i, y_i)$, and the *labeling* part is $\bigwedge_{1 \leq j \leq l} R_j(\overline{\omega}_j)$. We denote the set of node variables in Q by $\mathrm{nvar}(Q)$.

Intuitively, all variables are quantified existentially, and the words formed by the labels along the paths have to satisfy the respective relations. Formally, for every Σ-labeled *db*-graph G, every ECRPQ Q (of the form described in (1)) over Σ, every mapping σ from the node variables of Q to nodes in G, and every mapping μ from the path variables of Q to paths in G, we write $(G, \sigma, \mu) \models Q$ if

1. $\mu(\pi_i)$ is a path from $\sigma(x_i)$ to $\sigma(y_i)$ for every $1 \leq i \leq m$,
2. for each $\overline{\omega}_j = (\pi_{j_1}, \ldots, \pi_{j_k})$, $1 \leq j \leq l$, the tuple $(\lambda(\mu(\pi_{j_1})), \ldots, \lambda(\mu(\pi_{j_k})))$ belongs to the relation R_j.

Finally, we define the output of Q (of the form described in (1)) on G by

$$Q(G) \quad := \quad \{ \ \big(\sigma(\overline{z}), \mu(\overline{\chi})\big) \ \mid \ \sigma, \mu \text{ such that } (G, \sigma, \mu) \models Q \ \}.$$

As usual, if Q is Boolean, we model the Boolean constants `true` and `false` by the empty tuple () and the empty set $\emptyset$, respectively. In other words, $Q(G) = \texttt{true}$ iff there exist assignments σ and μ with $(G, \sigma, \mu) \models Q$.

Two queries Q and Q' are called *equivalent* ($Q \equiv Q'$, for short) if $Q(G) = Q'(G)$ for all *db*-graphs G. A query Q is said to be *contained* in a query Q' ($Q \subseteq Q'$, for short) if $Q(G) \subseteq Q'(G)$ for all *db*-graphs G.

With an ECRPQ Q we associate an edge-labeled directed graph H_Q^{lab} whose vertex set is the set of node variables occurring in Q, and where there is an edge from x to y labeled π iff (x, π, y) occurs in the relational part of Q. As in [4], we write H_Q to denote the (unlabeled) directed graph obtained from H_Q^{lab} by deleting the edge-labels (and removing duplicate edges). A query Q is called *acyclic* if H_Q is acyclic.

In accordance with [4], a *conjunctive regular path query (CRPQ)* Q over Σ is an ECRPQ over Σ of the form described in (1), where all relations R_j are unary relations, and (hence), all tuples $\overline{\omega}_j$ are singletons.

Thus, CRPQs can only refer to the languages that are allowed to occur along the paths, while ECRPQs can also describe relations between different paths.

The present paper devotes special attention to two classes of queries with an expressive power that lies strictly between CRPQs and ECRPQs: A *CRPQ with equality relations* is an ECRPQ where every relation in the labeling part is either of arity 1 (i. e., a regular language), or a k-ary eq-relation for some $k \geq 2$. Analogously, a *CRPQ with equal length relations* is an ECRPQ where every relation in the labeling part is either of arity 1, or a k-ary el-relation.

It is easy to see that ECRPQs and CRPQs can be transformed into queries in the following normal forms (note, though, that these transformations might increase the size of the queries):

Lemma 1. *For every ECRPQ* $Q = \text{Ans}(\overline{z}, \overline{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \ \bigwedge_{1 \leq j \leq l} R_j(\overline{\omega}_j)$, *there exists a regular relation R of arity m such that Q is equivalent to the ECRPQ* $Q' := \text{Ans}(\overline{z}, \overline{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \ R(\pi_1, \ldots, \pi_m)$.

Lemma 2. *For every CRPQ* $Q = \text{Ans}(\overline{z}, \overline{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \ \bigwedge_{1 \leq j \leq l} L_j(\pi_{i_j})$ *(where $i_j \in \{1, \ldots, m\}$), there exist regular languages $L'_1, \ldots, L'_m \subseteq \Sigma^*$ such that Q is equivalent to the CRPQ* $Q' := \text{Ans}(\overline{z}, \overline{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \ \bigwedge_{1 \leq i \leq m} L'_i(\pi_i)$.

Hence, for ECRPQs it suffices to consider just one regular relation of arity m; and for CRPQs, it suffices to consider just one regular language per path variable.

Turing Machines and H-Systems. Let $\mathcal{M}$ be a (deterministic) Turing machine with state set Q, initial state $q_0 \in Q$, halting state $q_H \in Q$, tape alphabet Γ (including the blank symbol), such that $Q \cap \Gamma = \emptyset$, and an input alphabet $\Gamma_I \subset \Gamma$ that does not include the blank symbol. We adopt the conventions that $\mathcal{M}$ accepts by halting, and does not halt in the first step (i. e., $q_0 \neq q_H$).

A *configuration* of $\mathcal{M}$ is a word $w_1 q w_2$, with $w_1, w_2 \in \Gamma^*$ and $q \in Q$. We interpret $w_1 q w_2$ as $\mathcal{M}$ being in state q, while the tape contains w_1 on the left side, and w_2 on the right side. The head is on the position of the first (leftmost) letter of w_2 (if $w_2 = \varepsilon$, $\mathcal{M}$ reads the blank symbol). We denote the successor relation on configurations of $\mathcal{M}$ by $\vdash_{\mathcal{M}}$. An *accepting run* of $\mathcal{M}$ is a sequence $C_0, \ldots, C_n$ of configurations of $\mathcal{M}$ (with $n \geq 1$), such that $C_0 \in q_0 \Gamma_I^*$ (C_0 is an initial configuration), $C_n \in \Gamma^* q_H \Gamma^*$ (C_n is an accepting configuration), and $C_i \vdash_{\mathcal{M}} C_{i+1}$ holds for all $0 \leq i < n$. Let $\Sigma := \Gamma \cup Q \cup \{\#\}$, where $\#$ is a new letter that does not occur in Γ or Q. We define the set of *valid computations* of $\mathcal{M}$ by $\mathrm{VALC}(\mathcal{M}) := \{\#C_0\# \cdots \#C_n\# \mid C_0, \ldots, C_n \text{ is an accepting run of } \mathcal{M}\}$, and denote its complement by $\mathrm{INVALC}(\mathcal{M}) := \Sigma^* \setminus \mathrm{VALC}(\mathcal{M})$. Finally, we define $\mathrm{dom}(\mathcal{M})$ to be the set of all $w \in \Gamma_I^*$ such that $\mathcal{M}$ halts after a finite number of steps when started in the configuration $q_0 w$.

By definition, $\mathrm{INVALC}(\mathcal{M}) = \Sigma^*$ holds if and only if $\mathrm{dom}(\mathcal{M}) = \emptyset$; and note that (given $\mathcal{M}$), the question if $\mathrm{dom}(\mathcal{M}) = \emptyset$ is undecidable.

As a technical tool for our proofs, we use the notion of *H-systems* to describe the sets $\mathrm{INVALC}(\mathcal{M})$ for Turing machines $\mathcal{M}$. Our notion of H-systems can be viewed as a generalization of pattern languages (cf. Salomaa [15]), or as a restricted version of the H-systems introduced by Albert and Wegner [3].

Definition 3. *An* H-system *(over the alphabet Σ) is a 4-tuple $H := (\Sigma, X, \mathcal{L}, \alpha)$, where (i) X and Σ are finite, disjoint alphabets, (ii) $\mathcal{L}$ is a function that maps every $x \in X$ to a regular language $\mathcal{L}(x) \subseteq \Sigma^*$ with $\varepsilon \in \mathcal{L}(x)$, and (iii) $\alpha \in (X \cup \Sigma)^+$.*

A morphism $h : (\Sigma \cup X)^* \to \Sigma^*$ is H-compatible *if $h(a) = a$ for every $a \in \Sigma$, and $h(x) \in \mathcal{L}(x)$ for every $x \in X$. We then define the language $L(H)$ that is generated by $H = (\Sigma, X, \mathcal{L}, \alpha)$ as $L(H) := \{h(\alpha) \mid h \text{ is an } H\text{-compatible morphism}\}$.*

For every finite, nonempty set of H-systems $\mathcal{H} = \{H_1, \ldots, H_k\}$, we define $L(\mathcal{H}) = \bigcup_{i=1}^{k} L(H_i)$.

In other words, the letters from Σ are constants, the letters from X are variables, and $L(H)$ is obtained from α by uniformly replacing every variable x with a word from $\mathcal{L}(x)$. We assume w.l.o.g. that X is chosen minimally; i. e., every $x \in X$ occurs in α. It is easy to see that H-systems are able to generate non-regular languages; e. g., the system $H = (\Sigma, \{x\}, \mathcal{L}, xx)$ with $\mathcal{L}(x) = \Sigma^*$ generates the language of all ww, $w \in \Sigma^*$.We use unions of H-system languages to describe the sets $\mathrm{INVALC}(\mathcal{M})$:

Lemma 4. *Given a Turing machine $\mathcal{M}$, one can effectively construct a set $\mathcal{H} = \{H_1, \ldots, H_k\}$ of H-systems (for some $k \geq 1$) such that* $\mathrm{INVALC}(\mathcal{M}) = L(\mathcal{H})$.

Proof (sketch). Let $\mathcal{M}$ be a Turing machine with state set Q and tape alphabet Γ, and define $\Sigma := Q \cup \Gamma \cup \{\#\}$. We approach the process of defining $\mathcal{H}$ from the

following angle: Every word $w \in \mathrm{INVALC}(\mathcal{M})$ contains at least one error that prevents w from being an element of $\mathrm{VALC}(\mathcal{M})$. Most of these error conditions can be described using regular expressions (similar to the construction used in the proof of Lemma 10.2 in Aho et al. [2]).

As $\mathrm{INVALC}(\mathcal{M})$ might be non-regular (if $\mathrm{dom}(\mathcal{M})$ is infinite), regular languages alone are not sufficient to describe all possible errors in a run of $\mathcal{M}$. More specifically, we cannot handle arbitrary errors in the preservation of the tape contents from one configuration to the other. As an example, assume $\mathcal{M}$ reads some $a \in \Gamma$ while in state $q \in Q$ and is supposed to write some $b \in \Gamma$, move the head to the right, and enter some state $p \in Q$. In all these cases, a configuration $C = w_1 q a w_2$ with $w_1, w_2 \in \Gamma^*$ is followed by the configuration $C' = w_1 b p w_2$.

Our goal is to construct H-expressions that capture all cases where a word encodes a sequence of configurations $C_0, \ldots, C_n$ that contains configurations $C_i = w_1 q a w_2$, $C_{i+1} = w_3 b p w_4$ where $w_1 \neq w_3$, or $w_2 \neq w_4$ holds (with $w_1, \ldots, w_4 \in \Gamma^*$). Note that, for all words $w, w' \in \Gamma^*$, $w \neq w'$ holds if and only if there exist words $u, v, v' \in \Gamma^*$ and letters $c, d \in \Gamma$ with $c \neq d$, $w = ucv$, and $w' = udv'$, or exactly one of w, w' is the empty word.

As errors described in the latter case (i. e., that exactly one of w_1, w_3 or of w_2, w_4 is empty) can be expressed using regular languages, we focus on the former case. In order to express these errors, for every $c \in \Gamma$, we define languages

$$L_{c,1} := \bigcup_{v \in \Gamma^*} \Sigma^* \# \, v \, c \Gamma^* q a \Gamma^* \# \, v \, (\Gamma \setminus \{c\}) \Sigma^*,$$

$$L_{c,2} := \bigcup_{v \in \Gamma^*} \Sigma^* \# \Gamma^* q a \, v \, c \Gamma^* \# \Gamma^* b p \, v \, (\Gamma \setminus \{c\}) \Sigma^*.$$

The corresponding H-systems for each c can be constructed straightforwardly. A more extensive explanation can be found in the full version. □

As we shall see in the next section, it is possible to reduce decision problems on unions of H-systems (and, hence, on the domains of Turing machines) to decision problems on CRPQs and ECRPQs.

3 Query Containment and Equivalence

Query Containment. The *query containment problem* is the problem to decide for two input queries Q and Q' whether $Q \subseteq Q'$.

The containment of CRPQs in CPRQs and of ECRPQs in CRPQs is known to be decidable and EXPSPACE-complete (cf. [8, 5] and [4], resp.). In [4], the authors proved the undecidability of the containment problem for ECRPQs, and mentioned the decidability of containment of CRPQs in ECRPQs as an important open problem. Our first main result states that this problem is undecidable, even if the ECRPQs are of a comparatively restricted form:

Theorem 5. *For every alphabet Σ with $|\Sigma| \geq 2$, the containment problem of CRPQs in CRPQs with equality relations over Σ is undecidable.*

The proof is a consequence of Lemma 4, the undecidability of the emptiness of $\mathrm{dom}(\mathcal{M})$ for Turing machines $\mathcal{M}$, and the following lemma:

Lemma 6. *Let Σ be an alphabet. For every set $\mathcal{H} = \{H_1, \ldots, H_k\}$ of H-systems over Σ, one can effectively construct an alphabet Σ', a CRPQ Q_1 over Σ', and a CRPQ with equality relations Q_2 over Σ' such that $Q_1 \subseteq Q_2$ if and only if $L(\mathcal{H}) = \Sigma^*$.*

Proof. Let $\Sigma = \{a_1, \ldots, a_s\}$ for some $s \geq 1$. Let $\mathcal{H}$ be a set of k H-systems $\mathcal{H} = \{H_1, \ldots, H_k\}$ over Σ (with $k \geq 1$). We define $\Sigma' := \Sigma \cup \{\bigstar, \$\}$, where $\bigstar$ and \$ are distinct letters that do not occur in Σ. Next, we define

$$Q_1 \;:=\; \mathrm{Ans}() \leftarrow (x, \pi, y), L(\pi),$$

where $L := \$\bigstar a_1 \cdots a_s \bigstar \$ \bigstar \Sigma^* \bigstar \$$, and x and y are distinct variables. Thus, $Q_1(G) = \texttt{true}$ if and only if G contains a path ρ with $\lambda(\rho) \in L$.

The definition of Q_2 is more involved. Informally explained, Q_2 uses the structure provided by Q_1 to implement the union of the languages $L(H_i)$. We define Q_2 such that, for every *db*-graph G with $Q_1(G) = \texttt{true}$, $Q_2(G) = \texttt{true}$ holds if and only if there is a path ρ in G with $\lambda(\rho) = \$\bigstar a_1 \cdots a_s \bigstar \$ \bigstar w \bigstar \$$, where $w \in L(\mathcal{H})$ (i. e., $w \in L(H_i)$ for some $H_i \in \mathcal{H}$).

Note that the paths ρ described by Q_1 contain exactly three occurrences of the \$ symbol, which can be understood to divide ρ into two parts, where the left part is labeled $\bigstar a_1 \cdots a_s \bigstar$. Likewise, the query Q_2 can be understood as consisting of two parts, which are to be defined in the subqueries $\bigwedge_{1 \leq i \leq k} \phi_i^{sel}$ and $\bigwedge_{1 \leq i \leq k} \phi_i^{cod}$, respectively. Our goal is to construct Q_2 in such a way that, when matching Q_2 to ρ, the ϕ_i^{sel} are used to *select* which H-system H_i is simulated in Q_2, while the actual *encoding* of that H-system is achieved by ϕ_i^{cod} (hence, the superscripts *sel* and *cod*). We define Q_2 as

$$\begin{aligned} Q_2 \;:=\; \mathrm{Ans}() \leftarrow\; & (x_0, c_1^{\$}, x_1), (x_{k+1}, c_2^{\$}, \hat{x}_1), (\hat{x}_{k+1}, c_3^{\$}, \hat{x}_{k+2}), \\ & L_{\$}(c_1^{\$}), L_{\$}(c_2^{\$}), L_{\$}(c_3^{\$}), \bigwedge_{1 \leq i \leq k} \phi_i^{sel}, \bigwedge_{1 \leq i \leq k} \phi_i^{cod} \end{aligned}$$

where $L_{\$} = \{\$\}$, and the ϕ_i^{sel} and ϕ_i^{cod} consist of relational and labeling atoms that shall be defined further down. As explained above, the subqueries ϕ_i^{sel} are used to select which H-system is active when matching Q_2 to a graph. These queries are defined by

$$\begin{aligned} \phi_i^{sel} \;:=\; & (x_i, c_{i,1}^{\bigstar}, y_{i,1}), (y_{i,1}, c_i^{a_1}, y_{i,2}), \ldots, (y_{i,s}, c_i^{a_s}, y_{i,s+1}), (y_{i,s+1}, c_{i,2}^{\bigstar}, x_{i+1}), \\ & L_{\bigstar}(c_{i,1}^{\bigstar}), L_{a_1}(c_i^{a_1}), \ldots, L_{a_s}(c_i^{a_s}), L_{\bigstar}(c_{i,2}^{\bigstar}), \mathrm{eq}(c_{i,1}^{\bigstar}, c_{i,2}^{\bigstar}) \end{aligned}$$

where $L_a := \{\varepsilon, a\}$ for each $a \in \{\bigstar, a_1, \ldots, a_s\}$.

In order to define each ϕ_i^{cod}, we need to consider the respective H-system H_i: Let $H_i = (\Sigma, X_i, \mathcal{L}_i, \alpha_i)$, where $\alpha_i = \beta_{i,1} \cdots \beta_{i,m_i}$ for some $m_i \geq 1$ and $\beta_{i,1}, \ldots, \beta_{i,m_i} \in (X \cup \Sigma)$. We define the relational part of ϕ_i^{cod} to be

$$(\hat{x}_i, c_{i,3}^{\bigstar}, z_{i,1}), (z_{i,1}, d_{i,1}, z_{i,2}), \ldots, (z_{i,m_i}, d_{i,m_i}, z_{i,m_i+1}), (z_{i,m_i+1}, c_{i,4}^{\bigstar}, \hat{x}_{i+1}),$$

where $c^{\bigstar}_{i,3}$, $c^{\bigstar}_{i,4}$, and all $d_{i,j}$ are (pairwise distinct) new path variables. We start the construction of the labeling part of ϕ^{cod}_i with the labeling atoms $L_{\bigstar}(c^{\bigstar}_{i,3})$, $L_{\bigstar}(c^{\bigstar}_{i,4})$, $\mathrm{eq}(c^{\bigstar}_{i,1}, c^{\bigstar}_{i,3})$, and $\mathrm{eq}(c^{\bigstar}_{i,1}, c^{\bigstar}_{i,4})$. Furthermore, we define a regular language $L_{i,j}$ for every $1 \leq j \leq m_i$ by $L_{i,j} := \mathcal{L}_i(\beta_{i,j})$ if $\beta_{i,j} \in X$, and $L_{i,j} := \{\varepsilon, \beta_{i,j}\}$ if $\beta_{i,j} \in \Sigma$. In addition to this, we add a label atom $\mathrm{eq}(c_i^{\beta_{i,j}}, d_{i,j})$ for every j with $\beta_{i,j} \in \Sigma$. Finally, for every j with $\beta_{i,j} \in X$ such that $\beta_{i,j}$ occurs more than once in α_i, we add a relation $\mathrm{eq}(d_{i,j}, d_{i,l})$ for every $l \neq j$ with $\beta_{i,l} = \beta_{i,j}$.

Note that the relation graph H_{Q_2} consists only of a path from x_0 to $\hat{x}_{k+1}$, where each node (except $\hat{x}_{k+1}$, the last node) has exactly one successor. Thus, the relation graph is acyclic and has no branches.

We claim that $L(\mathcal{H}) = \Sigma^*$ holds if and only if $Q_1 \subseteq Q_2$, which completes the proof of Lemma 6. Due to space limitations, the proof of this claim has been omitted from this version. □

By using standard encoding techniques for representing arbitrary finite alphabets by an alphabet of size 2, the proof of Theorem 5 now easily follows from Lemma 4, the undecidability of the emptiness of $\mathrm{dom}(\mathcal{M})$ for Turing machines $\mathcal{M}$, and Lemma 6. By using universal Turing machines instead of arbitrary Turing machines, we also obtain the following strengthening of Theorem 5:

Theorem 7. *For every alphabet Σ with $|\Sigma| \geq 2$, there are a fixed CRPQ Q_1 over Σ and a fixed CRPQ with equality relations Q_2 over Σ such that (i) the containment problem of Q_1 in CRPQs with equality relations, and (ii) the containment problem of CRPQs in Q_2 are both undecidable. This holds even if all queries are Boolean and acyclic.*

Applying slight modifications to the proof of Lemma 6, we observe the same situation for ECRPQs that use length equality instead of equality relations:

Theorem 8. *For every alphabet Σ with $|\Sigma| \geq 2$, there are a fixed CRPQ Q_1 over Σ and a fixed CRPQ with length equality relations Q_2 over Σ such that (i) the containment problem of Q_1 in CRPQs with length equality relations, and (ii) the containment problem of CRPQs in Q_2, are both undecidable. This holds even if all queries are Boolean and acyclic.*

Query Equivalence. The *query equivalence problem* is the problem to decide for two input queries Q and Q' whether $Q \equiv Q'$.

Another question specifically posed in [4] is whether the equivalence problem for CRPQs and ECRPQs is decidable. Using a variant of the proof of Theorem 7, we can answer this question negatively:

Theorem 9. *For every alphabet Σ with $|\Sigma| \geq 2$, there are a fixed CRPQ Q_1 over Σ and a fixed ECRPQ Q_2 over Σ such that (i) the equivalence problem of Q_1 and ECRPQs, and (ii) the equivalence problem of CRPQs and Q_2, are both undecidable. This holds even if all queries are Boolean and acyclic.*

4 Expressiveness and Relative Succinctness

(E)CRPQ Expressibility. We say that a query function F is *CRPQ-expressible* (or *ECRPQ-expressible*) if there is a CRPQ (or ECRPQ, resp.) Q such that $Q(G) = F(G)$ for every Σ-labeled *db*-graph G.

For every language $L \subseteq \Sigma^*$, we define a query function F_L by

$$F_L(G) \;:=\; \{(x, y) \mid G \text{ contains a path } \rho \text{ from } x \text{ to } y \text{ with } \lambda(\rho) \in L\}$$

for every Σ-labeled *db*-graph G. Analogously, we define a Boolean query function F_L^B by $F_L^B(G) := \texttt{true}$ if and only if $F_L(G) \neq \emptyset$.

The proofs presented in this section will use specific *db*-graphs G_w representing strings $w \in \Sigma^*$ as follows: If $w = b_1 \cdots b_{|w|}$ (with all $b_i \in \Sigma$), we define the *db*-graph $G_w := (V_w, E_w)$ by $V_w := \{v_0, \ldots, v_{|w|}\}$ (where all v_i are distinct nodes), and $E_w = \{(v_i, b_{i+1}, v_{i+1}) \mid 0 \leq i < |w|\}$. Thus, G_w consists of a path from v_0 to $v_{|w|}$ that is labeled with w.

Clearly, if $L \subseteq \Sigma^*$ such that F_L is expressible by an ECRPQ Q_L, then for all words $w \in \Sigma^*$ we have $w \in L$ iff $(v_0, v_{|w|}) \in Q_L(G_w)$.

Lemma 10. *Let Σ be an alphabet, let $L \subseteq \Sigma^*$. Then F_L is CRPQ-expressible if and only if L is regular.*

Proof (sketch). The "if-direction" is trivial. For the "only-if-direction", let $L \subseteq \Sigma^*$ and let Q_L be a CRPQ such that $Q_L(G) = F_L(G)$ for every Σ-labeled *db*-graph G. For showing that L is regular, we proceed along the following steps:

(1) Rewrite Q_L into a CRPQ Q'_L such that (i) for all $w \in \Sigma^*$ we have $(v_0, v_{|w|}) \in Q'_L(G_w)$ iff $(v_0, v_{|w|}) \in Q_L(G_w)$ (i. e., in some sense, Q'_L is equivalent to Q_L on *db*-graphs representing strings), and (ii) the graph $H_{Q'_L}$ associated with the relational part of Q'_L (cf., Section 2) is a connected directed acyclic graph with x as its single source node and y as its single sink node, where $\mathrm{Ans}(x, y)$ is the head of Q'_L.
(2) Use Q'_L and a variant of the product construction to construct a nondeterministic finite automaton that accepts exactly those words $w \in \Sigma^*$ for which $(v_0, v_{|w|}) \in Q'_L(G_w)$.

The proof details can be found in the full version of the paper. □

The situation is not strictly the same for Boolean queries (e. g., if L contains every single letter of Σ, $F_L^B(G) = \texttt{true}$ holds for all non-empty *db*-graphs G); but a similar result can be observed:

Lemma 11. *Let Σ be an alphabet with $|\Sigma| \geq 2$, let $a \in \Sigma$, and let $L \subseteq (\Sigma \setminus \{a\})^*$. Then F_{aLa}^B is CRPQ-expressible if and only if L is regular.*

For alphabets Σ of size ≥ 2, ECRPQs can express queries F_L for non-regular $L \subseteq \Sigma^*$ which, according to Lemma 10, are not CRPQ-expressible. For example, for $L := \{a^n b^n \mid n \in \mathbb{N}\}$, F_L is not CRPQ-expressible, but is expressed by the ECRPQ $\mathrm{Ans}(x, y) \leftarrow (x, \pi_1, z), (z, \pi_2, y), L_1(\pi_1), L_2(\pi_2), \mathrm{el}(\pi_1, \pi_2)$, where $L_1 := a^*$ and $L_2 := b^*$. For *unary* alphabets (i. e., alphabets of size 1), however, we can show the following:

Lemma 12. *Let Σ be a unary alphabet, let $L \subseteq \Sigma^*$. Then F_L is ECRPQ-expressible if and only if it is CRPQ-expressible.*

Before giving a proof sketch of this lemma, let us note that, in spite of Lemma 12, there exist ECRPQ-queries over unary alphabets that are *not* CRPQ-expressible. For example, consider the ECRPQ

$$Q \;:=\; \mathrm{Ans}(x,y) \leftarrow (x,\pi_1,z),(y,\pi_2,z),\mathrm{el}(\pi_1,\pi_2),$$

selecting all pairs of nodes (u,v) in a db-graph G, for which there exists a node w such that there are paths from u to w and from v to w of the same length. It should be not too difficult to see that this query is not CRPQ-expressible.

Proof (Sketch of the proof of Lemma 12).
The "if-direction" is trivial. For the "only-if-direction" let $\Sigma := \{a\}$, let $L \subseteq \{a\}^*$, and Q_L be an ECRPQ expressing F_L. By Lemma 10 it suffices to show that L is regular. Due to Lemma 1, w.l.o.g. Q_L is of the form

$$\mathrm{Ans}(x,y) \;\leftarrow\; \bigwedge_{1\le i\le k} (x_i,\pi_i,y_i) \;\wedge\; R(\pi_1,\ldots,\pi_k)$$

for a k-ary regular relation R over $\{a\}^*$. With R we associate a relation $R_{\mathrm{len}} \subseteq \mathbb{N}^k$ as follows: $R_{\mathrm{len}} := \{\, (|w_1|,\ldots,|w_k|) \mid (w_1,\ldots,w_k) \in R \,\}$. The proof of the lemma proceeds along the following steps:

(1) Note that R_{len} is *semi-linear* (since R is a regular relation over a unary alphabet). The notion of semi-linear relations is defined as follows (cf., e. g., [10]): For every $k \ge 1$ and every vector $a \in \mathbb{N}^k$, define $a\mathbb{N} := \{ai \mid i \in \mathbb{N}\}$. For all sets $A, B \subseteq \mathbb{N}^k$, let $A + B := \{a + b \mid a \in A, b \in B\}$. A set $A \subseteq \mathbb{N}^k$ is *linear* if there exist $a_0,\ldots,a_n \in \mathbb{N}^k$ for some $n \ge 0$ such that $A = a_0 + a_1\mathbb{N} + \ldots + a_n\mathbb{N}$. A set is *semi-linear* if it is a finite union of linear sets.
(2) Consider all non-empty acyclic directed paths in the labeled query graph $H^{\mathrm{lab}}_{Q_L}$, and let $\mathcal{P} = \{p_1,\ldots,p_l\}$ be the set of all these paths. For each such path p_j let $\overline{\pi}_j$ be the sequence of path variables labeling the edges of p_j in $H^{\mathrm{lab}}_{Q_L}$. Furthermore, let $\mathrm{start}(p_j)$ and $\mathrm{end}(p_j)$ denote the start node and the end node of p_j. By definition, for each p_j, each path variable π_i occurs at most once in $\overline{\pi}_j$.
(3) With each $p_j \in \mathcal{P}$ we assume a function $\hat{p}_j : \mathbb{N}^k \to \mathbb{N}$ such that $\hat{p}_j(r_1,\ldots,r_k)$ is the sum of all r_i for which π_i occurs in $\overline{\pi}_j$; and let $\hat{p} : \mathbb{N}^k \to \mathbb{N}^l$ be defined as $\hat{p}(r_1,\ldots,r_k) := \big(\hat{p}_1(r_1,\ldots,r_k),\ldots,\hat{p}_l(r_1,\ldots,r_k)\big)$ for all $(r_1,\ldots,r_k) \in \mathbb{N}^k$. Note that $\hat{p}_j(r_1,\ldots,r_k)$ is the length of the path in G_w corresponding to the path p_j in $H^{\mathrm{lab}}_{Q_L}$.
(4) Since R_{len} is semi-linear, $\hat{p}(R_{\mathrm{len}}) := \{\hat{p}(\overline{r}) \mid \overline{r} \in R_{\mathrm{len}}\}$ is also semi-linear.
(5) Assume w.l.o.g. that for the path p_1 we have $\mathrm{start}(p_1) = x$ and $\mathrm{end}(p_1) = y$. Let $T := \hat{p}(R_{\mathrm{len}}) \cap B \cap \bigcap_{1\le j\le l} S_j$,
for $B := \{(s_1,\ldots,s_l) \in \mathbb{N}^l \mid s_j \le s_1 \text{ for all } 1 \le j \le l\}$

and $S_j := \{(s_1, \ldots, s_l) \in \mathbb{N}^l \mid s_{j'} = s_j \text{ for all } 1 \leq j' \leq l \text{ with } \text{start}(p_{j'}) = \text{start}(p_j) \text{ and } \text{end}(p_{j'}) = \text{end}(p_j)\}$.
Note that T is semi-linear (since each of the sets $\hat{p}(R_{\text{len}})$, B, S_j is semi-linear, and the class of semi-linear sets is closed under intersection, see [10]).

(6) Show that for all $w \in \{a\}^*$ we have: $|w| \in \text{proj}_1(T)$ iff $(v_0, v_{|w|}) \in Q_L(G_w)$, where proj_1 projects each element of T to its first component.

As a consequence, $L = \{w \in \{a\}^* \mid |w| \in \text{proj}_1(T)\}$ is regular, since $\text{proj}_1(T)$ is semi-linear (cf. Harrison [11]). This completes the proof sketch of Lemma 12. A detailed proof can be found in the full version. □

In Section 3.1 of [4], Barceló et al. mention that ECRPQs are able to express queries corresponding to *regular expressions with backreferencing* (or *extended regular expressions*) (cf. Aho [1], Freydenberger [9]). These expressions extend the regular expressions with variable binding and repetition operators; e. g., for every expression α, the extended expression $(\alpha)\%x\, xx$ generates the language of all www with $w \in L(\alpha)$ (α generates some $w \in L(\alpha)$, $\%x$ assigns that w to x, and the subsequent uses of x repeat this w – hence, xx generates ww).

Let $L := \{a^n \mid n \geq 4,\ n \text{ is a composite number}\}$. According to Lemma 12, F_L is not ECRPQ-expressible (as L is not regular). On the other hand, L is generated by the extended regular expression $(a\, a^+)\%x\ x^+$ (cf. Câmpeanu et al. [6]). This demonstrates that ECRPQs are not able to express all queries that correspond to extended regular expressions.

Relative Succinctness. We can adapt Lemma 6 to observe the following result on the decidability of expressibility:

Theorem 13. *CRPQ-expressibility for ECRPQs is not co-semi-decidable.*

Proof. This follows from the proof of Theorem 9, a variation of Lemma 11, and the observation that $\text{INVALC}(\mathcal{M})$ is regular iff $\text{dom}(\mathcal{M})$ is finite. Regarding the latter, note that if $\text{dom}(\mathcal{M})$ is finite, $\text{INVALC}(\mathcal{M})$ is co-finite; if $\text{dom}(\mathcal{M})$ is infinite, non-regularity of $\text{INVALC}(\mathcal{M})$ can be established using standard tools. This allows us to effectively construct an ECPRQ Q from a Turing machine $\mathcal{M}$ such that Q is CRPQ-expressible if and only if $\text{dom}(\mathcal{M})$ is finite.

Finiteness of $\text{dom}(\mathcal{M})$ is a Σ_2^0-complete problem in the arithmetical hierarchy (cf. Kozen [13]); hence, CRPQ-expressibility is Σ_2^0-hard, which means that this problem is neither semi-decidable, nor co-semi-decidable. □

Using Theorem 13 in conjunction with a technique that is due to Hartmanis [12] and has been widely used in Formal Language Theory (cf. Kutrib [14]), we obtain a result on the relative succinctness of ECRPQs and CRPQs. One of the benefits of that technique is that it applies to a wide range of different reasonable definitions of the size of an ECRPQ.

In order to be as general as possible, we define a *complexity measure* for ECRPQs as a computable function c from the set of all ECRPQs to $\mathbb{N}$, such that for every finite alphabet Σ, the set of all ECRPQs Q over Σ (i) can be

effectively enumerated in order of increasing $c(Q)$, and (ii) does not contain infinitely many ECRPQs with the same value $c(Q)$. As the following theorem demonstrates, no matter which complexity measure we choose, the size tradeoff between ECRPQs and CRPQs is not bounded by any recursive function:

Theorem 14. *Let Σ be a finite alphabet with $|\Sigma| \geq 2$. For every recursive function $f : \mathbb{N} \to \mathbb{N}$ and every complexity measure c, there exists an ECRPQ Q over Σ such that Q is CRPQ-expressible, but for every CRPQ Q' with $Q' \equiv Q$, $c(Q') > f(c(Q))$.*

References

1. A. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A. Elsevier, 1990.
2. A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.
3. J. Albert and L. Wegner. Languages with homomorphic replacements. *Theoretical Computer Science*, 16:291–305, 1981.
4. P. Barceló, C. Hurtado, L. Libkin, and P. Wood. Expressive languages for path queries over graph-structured data. In *Proc. PODS'10*, pages 3–14, 2010.
5. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR'00*, pages 176–185, 2000.
6. C. Câmpeanu, K. Salomaa, and S. Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14:1007–1018, 2003.
7. A. Deutsch and V. Tannen. Optimization properties for classes of conjunctive regular path queries. In *Proc. DBPL'01*, volume 2397 of *LNCS*, pages 21–39. Springer, 2001.
8. D. Florescu, A. Y. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *Proc. PODS'98*, pages 139–148, 1998.
9. D. Freydenberger. Extended regular expressions: Succinctness and decidability. In *Proc. STACS 2011*, pages 507–518, 2011.
10. S. Ginsburg and E. Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.
11. M. Harrison. *Introduction to Formal Language Theory*. Addison Wesley Publishing Company, 1978.
12. J. Hartmanis. On Gödel speed-up and succinctness of language representations. *Theoretical Computer Science*, 26(3):335–342, 1983.
13. D. Kozen. *Theory of Computation*. Springer-Verlag, London, 2006.
14. M. Kutrib. The phenomenon of non-recursive trade-offs. *International Journal of Foundations of Computer Science*, 16(5):957–973, 2005.
15. K. Salomaa. Patterns. In *Formal Languages and Applications*, number 148 in Studies in Fuzziness and Soft Computing, pages 367–379. Springer, 2004.