Journal of Computer and System Sciences 79 (2013) 892–909

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Expressiveness and static analysis of extended conjunctive regular path queries ☆

Dominik D. Freydenberger *, Nicole Schweikardt

Institut für Informatik, Goethe-Universität, Frankfurt am Main, Germany

A R T I C L E I N F O

Article history:
Received 12 December 2011
Received in revised form 6 August 2012
Accepted 16 January 2013
Available online 21 January 2013

Keywords:
Graph databases
Conjunctive queries
Regular relations
Regular path queries
Query containment

A B S T R A C T

We study the expressiveness and the complexity of static analysis of *extended conjunctive regular path queries* (*ECRPQs*), introduced by Barceló et al. (2010) [3]. 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 are 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., [4,6,7]), 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. [3] 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 [3] investigated the complexity of query evaluation and static analysis of ECRPQs. While query containment is known to be decidable and Expspace-complete for CRPQs [7,4], it was shown to be undecidable for ECRPQs [3]. However, checking containment of an ECRPQ in a CRPQ still is decidable and Expspace-complete [3]. (Un)Decidability of checking containment (or, equivalence) of a CRPQ in an ECRPQ was posed as an open question in [3].

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.

☆ The present paper is the full version of the conference contribution (Freydenberger and Schweikardt, 2011 [9]).

* Corresponding author.
E-mail addresses: freydenberger@em.uni-frankfurt.de (D.D. Freydenberger), schweika@informatik.uni-frankfurt.de (N. Schweikardt).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2013.01.008

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-expressible 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 a tool, 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 form nm for $n, m \geqslant 2$). Since these paths can be easily described by a *regular expression with backreferencing* (cf. [1,8]) of the form $(aa^+)\%xx^+$, this refutes a claim of [3] 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 a tool for proving limitations to the expressive power of CRPQs and ECRPQs.

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.

2.1. 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 $n \geqslant 0$ 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 $0 \leqslant i < n$. 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. [3] 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 $j \leqslant |w_i|$, 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 $\{[\bar{r}] \mid \bar{r} \in R\}$ 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, for $k \geqslant 2$:

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

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.

Before we define ECRPQs and CRPQs, note that our definitions are not completely identical to the definitions in [3]. There, it is also required that the tuples $\overline{\omega_i}$ of path variables (introduced further down) are distinct (i.e., for all i, j with $i \neq j$, the tuple $\overline{\omega_i}$ and the tuple $\overline{\omega_j}$ have no variable in common). This difference does not affect the topics discussed in the present paper: As shown in Lemmas 2.1 and 2.2 below, dropping this restriction does not increase the expressive power. Moreover, while the transformation from our form of queries to the ones described in [3] might lead to exponential increases of the size of the queries, the lower bound presented in Section 4.2 is non-recursive and, hence, not influenced by this difference.

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

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

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

1. each R_j is a regular expression that defines a regular relation over Σ,
2. $\bar{x} = (x_1, \ldots, x_m)$ 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. $\bar{z}$ is a tuple of node variables among $\bar{x}$, $\overline{y}$, and
6. $\overline{\chi}$ is a tuple of path variables among those in $\overline{\pi}$.

The expression $\mathrm{Ans}(\bar{z}, \overline{\chi})$ is the *head*, and the expression to the right of $\leftarrow$ is the *body* of Q. If $\bar{z}$ 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 $\bigwedge_{1\leqslant i\leqslant m}(x_i, \pi_i, y_i)$, and the *labeling* part is $\bigwedge_{1\leqslant j\leqslant l} R_j(\overline{\omega}_j)$. 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 $i \in \{1, \ldots, m\}$,
2. for each $\overline{\omega}_j = (\pi_{j_1}, \ldots, \pi_{j_k})$ with $1 \leqslant j \leqslant l$, 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

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

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, $Q(G) = \mathtt{true}$ 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 [3], we write H_Q to denote the unlabeled directed graph obtained from H_Q^{lab} by deleting the edge-labels and removing duplicate edges. We call H_Q and H_Q^{lab} the *query graph* of Q, respectively the *labeled query graph* of Q. A query Q is called *acyclic* if its query graph H_Q is acyclic.

In accordance with [3], 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 $k \geqslant 2$. 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 2.1. *For every ECRPQ $Q = \mathrm{Ans}(\bar{z}, \overline{\chi}) \leftarrow \bigwedge_{1\leqslant i\leqslant m}(x_i, \pi_i, y_i), \bigwedge_{1\leqslant j\leqslant l} R_j(\overline{\omega}_j)$, there exists a regular relation R of arity m such that Q is equivalent to the ECRPQ $Q' := \mathrm{Ans}(\bar{z}, \overline{\chi}) \leftarrow \bigwedge_{1\leqslant i\leqslant m}(x_i, \pi_i, y_i), R(\pi_1, \ldots, \pi_m)$.*

Proof. First, note that every relation R_i of arity m_i can be interpreted as a regular language over the alphabet $\{a, \perp\}^{m_i}$ that is recognized by some finite automaton M_i. Furthermore, we can assume w.l.o.g. that $m_i \leqslant m$ holds for all $i \in \{1 \ldots, l\}$. We then obtain the relation R from the relations R_i by constructing a finite automaton M over the alphabet $\{a, \perp\}^m$ that simulates all M_i in parallel. □

Lemma 2.2. *For every CRPQ $Q = \mathrm{Ans}(\bar{z}, \overline{\chi}) \leftarrow \bigwedge_{1\leqslant i\leqslant m}(x_i, \pi_i, y_i), \bigwedge_{1\leqslant j\leqslant l} L_j(\pi_{i_j})$ (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 $Q' := \mathrm{Ans}(\bar{z}, \overline{\chi}) \leftarrow \bigwedge_{1\leqslant i\leqslant m}(x_i, \pi_i, y_i), \bigwedge_{1\leqslant i\leqslant m} L'_i(\pi_i)$.*

Proof. Let Q be a CRPQ over Σ. For every path variable π_i, $1 \leqslant i \leqslant m$, we define $I_i := \{j \mid i_j = i\}$. We construct the labeling part of Q' by defining atoms $L'_i(\pi_i)$ for $1 \leqslant i \leqslant m$ in the following way:

1. If I_i is empty, let $L'_i := \Sigma^*$,
2. if I_i contains exactly one element j, let $L'_i := L_j$,
3. if I_i contains more than one element, let $L'_i := \bigcap_{j\in I_i} L_j$. As every language L_j is regular, L'_i is also regular.

The relational part of Q' is identical to the relational part of Q; and it is easy to see that $Q \equiv Q'$ holds. □

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.

2.2. 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 $C_0, \dots, C_n$ of configurations of $\mathcal{M}$ with $n \geqslant 1$, 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 $0 \leqslant i < n$. 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 $\mathrm{VALC}(\mathcal{M}) := \{\#C_0\# \cdots \#C_n\# \mid C_0, \dots, C_n$ 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 $\mathrm{dom}(\mathcal{M}) = \emptyset$. By Rice's Theorem (cf., e.g., [13]) we know 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 [17]), or as a restricted version of the H-systems introduced by Albert and Wegner [2].

Definition 2.3. 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 define the language $L(H)$ that is generated by $H = (\Sigma, X, \mathcal{L}, \alpha)$ as $L(H) := \{h(\alpha) \mid h$ is an H-compatible morphism$\}$.

For every finite, non-empty set of H-systems $\mathcal{H} = \{H_1, \dots, H_k\}$, 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 $\{ww \mid w \in \Sigma^*\}$.

Note that, due to condition (ii) in Definition 2.3, single H-systems are not able to express all regular languages. For example, the regular languages $L_1 := \{\mathtt{a}, \mathtt{b}\}$ and $L_2 := \{\mathtt{a}\}^+ \cup \{\mathtt{b}\}^+$ cannot be generated by any H-system. (This is easily seen by first showing that if $L_i = L(H)$ holds for some H-system $H = (\Sigma, X, \mathcal{L}, \alpha)$, $\alpha \in X^+$ must hold. As the definition of H-systems requires $\varepsilon \in \mathcal{L}(x)$ for all $x \in X$, $\varepsilon \in L(H)$ follows, which contradicts $\varepsilon \notin L_i$.) Constructions for similar examples can be found in the proof of the following lemma.

Lemma 2.4. *Given a Turing machine $\mathcal{M}$, one can effectively construct a set $\mathcal{H} = \{H_1, \dots, H_k\}$ of H-systems (for some $k \geqslant 1$) such that* $\mathrm{INVALC}(\mathcal{M}) = L(\mathcal{H})$.

Proof. 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 conditions can be described by regular languages which can be turned into H-systems in a straightforward manner; e. g, if

$$w \notin \# q_0 (\Gamma_I)^* (\# \Gamma^* Q \Gamma^*)^* \# \Gamma^* q_H \Gamma^* \#,$$

w is not an encoding of a sequence of configurations of $\mathcal{M}$, or it is such an encoding, but the first configuration is not an initial configuration, or the last configuration is not a halting configuration. Hence, we can define an H-system $H_1 := (\Sigma, \{x\}, \mathcal{L}_1, x)$, where $\mathcal{L}_1$ maps x to the complement of the language

$$\# q_0 (\Gamma_I)^* (\# \Gamma^* Q \Gamma^*)^* \# \Gamma^* q_H \Gamma^* \#.$$

Note that, as required by the definition of an H-system, $\varepsilon \in \mathcal{L}_1(x)$ holds. Moreover, if $w \notin L(H_1)$, we know that w is an encoding of configurations $C_0, \dots, C_n$ for some $n \geqslant 1$, such that C_0 is an initial configuration, and C_n is a halting configuration. All that remains is to describe all possible transition errors, i.e., C_i, C_{i+1} for which $C_i \vdash_{\mathcal{M}} C_{i+1}$ does not hold. Most of these errors can be described using only regular languages, which can be expressed using unions of H-systems.

Assume, e.g., that when reading some $a \in \Gamma$ in a state $q \in Q$, $\mathcal{M}$ is supposed to enter a state $p \in Q$. Then we can describe each invalid successor state $\overline{p} \in (Q \setminus \{p\})$ using an H-system $H_{\overline{p}}$ with

$$L(H_{\overline{p}}) = \Sigma^* \,\#\, \Gamma^* \, qa \, \Gamma^* \,\#\, \Gamma^* \, \overline{p} \, \Sigma^*$$

by defining $H_{\overline{p}} := (\Sigma, X, \mathcal{L}, \alpha)$ with

$$\alpha := x_1 \,\#\, y_1 \, qa \, y_2 \,\#\, y_3 \, \overline{p} \, x_2,$$

while $X := \{x_1, x_2, y_1, y_2, y_3\}$, and $\mathcal{L}$ is defined by $\mathcal{L}(x_i) := \Sigma^*$ and $\mathcal{L}(y_j) := \Gamma^*$.

It is easy to see that $w \in L(H_{\overline{p}}) \setminus L(H_1)$ if and only if w includes a sequence of configurations that contains a transition with the aforementioned error, and all these $H_{\overline{p}}$ together describe all possible state errors of $\mathcal{M}$ when reading a in state q.

All other state transition errors can be described analogously, as can be all errors regarding the symbols that $\mathcal{M}$ is supposed to write. For example, if $\mathcal{M}$ 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 $p \in Q$, the regular language

$$\Sigma^* \,\#\, \Gamma^* \, qa \, \Gamma^* \,\#\, \Gamma^* \, \bar{b}p \, \Sigma^*$$

describes the error where a symbol $\bar{b} \in (\Gamma \setminus \{b\})$ was written. This language is generated by the H-system $H_{\bar{b}} = (\Sigma, X, \mathcal{L}, \alpha)$, where

$$\alpha := x_1 \,\#\, y_1 \, qa \, y_2 \,\#\, y_3 \, \bar{b}p \, x_2,$$

while $X := \{x_1, x_2, y_1, y_2, y_3\}$, and $\mathcal{L}$ is defined by $\mathcal{L}(x_1) := \mathcal{L}(x_2) = \Sigma^*$ and $\mathcal{L}(y_1) := \mathcal{L}(y_2) := \mathcal{L}(y_3) := \Gamma^*$.

All types of errors that we have considered so far are described by regular languages, where each of these is generated by either a single H-system, or a finite union of H-systems. Of course, as $\mathrm{INVALC}(\mathcal{M})$ can 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, regular languages cannot handle arbitrary errors in the preservation of the tape contents from one configuration to the other.

Again, 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 $C = w_1 qa w_2$ with $w_1, w_2 \in \Gamma^*$ is followed by the configuration $C' = w_1 bp w_2$.

Our goal is now to define H-systems that capture all cases where the encoding of configurations $C_0, \ldots, C_n$ contains configurations $C_i = w_1 qa w_2$, $C_{i+1} = w_3 bp w_4$ 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.

We first consider the special case of types of errors of the latter case, i.e., where exactly one of w_1, w_3 or of w_2, w_4 is empty. For example, errors where $w_1 = \varepsilon \neq w_3$ are described by the language

$$\bigcup_{c \in \Gamma} \Sigma^* \,\#\, qa \, \Gamma^* \,\#\, c \, \Gamma^* \, bp \, \Gamma^* \,\#\, \Sigma^*.$$

For every $c \in \Gamma$, we define an H-system $H_c := (\Sigma, X, \mathcal{L}, \alpha)$ with

$$\alpha := x_1 \,\#\, qa \, y_1 \,\#\, c \, y_2 \, bp \, y_3 \;\#\, x_2,$$

where $X := \{x_1, x_2, y_1, \ldots, y_3\}$, and $\mathcal{L}$ is defined by $\mathcal{L}(x_i) := \Sigma^*$ and $\mathcal{L}(y_j) = \Gamma^*$.

The errors of the other case (i.e., where $w \neq w'$ holds without exactly one of the words being empty) are handled in a similar but slightly more involved way. In order to express these errors, for every $c \in \Gamma$, we consider the non-regular languages

$$L_{c,1} := \bigcup_{v \in \Gamma^*} \Sigma^* \,\#\, vc \, \Gamma^* \, qa \, \Gamma^* \,\#\, v\bigl(\Gamma \setminus \{c\}\bigr) \, \Sigma^*,$$

$$L_{c,2} := \bigcup_{v \in \Gamma^*} \Sigma^* \,\#\, \Gamma^* \, qa \, vc \, \Gamma^* \,\#\, \Gamma^* \, bp \, v\bigl(\Gamma \setminus \{c\}\bigr) \, \Sigma^*,$$

where $L_{c,1}$ describes the cases of $w_1 \neq w_3$, and $L_{c,2}$ the cases of $w_2 \neq w_4$.

Each of these languages can be generated by a union of H-systems, one for each $\bar{c} \in (\Gamma \setminus \{c\})$. We consider the definitions of H-systems for $L_{c,1}$; the H-systems for $L_{c,2}$ can be constructed analogously. For every $\bar{c} \in (\Gamma \setminus \{c\})$, we define an H-system $H_{\bar{c}} := (\Sigma, X, \mathcal{L}, \alpha)$ with $X := \{v, x_1, x_2, y_1, y_2\}$,

$$\alpha := x_1 \,\#\, v \, c \, y_1 \, qa \, y_2 \,\#\, v \, \bar{c} \, x_2,$$

and $\mathcal{L}(v) := \mathcal{L}(y_1) := \mathcal{L}(y_2) := \Gamma^*$ and $\mathcal{L}(x_1) := \mathcal{L}(x_2) := \Sigma^*$. Then

$$L_{c,1} = \bigcup_{\bar{c} \in (\Gamma \setminus \{c\})} L(H_{\bar{c}})$$

holds.

Analogously, we can define H-systems for those cases where $\mathcal{M}$ is supposed to move to the left instead of the right. Hence, by defining appropriate H-systems for all possible tape letters $a \in \Gamma$ and states $q \in Q$ and the corresponding actions of $\mathcal{M}$, $\mathcal{H}$ can be constructed effectively. □

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

3. Query containment and equivalence

3.1. 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 CRPQs and of ECRPQs in CRPQs is known to be decidable and EXPSPACE-complete (cf. [7,4] and [3], resp.). In [3], 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 3.1. *For every alphabet Σ with $|\Sigma| \geqslant 2$, the containment problem of CRPQs in CRPQs with equality relations over Σ is undecidable.*

The proof is a consequence of Lemma 2.4, the undecidability of the emptiness of $\operatorname{dom}(\mathcal{M})$ for Turing machines $\mathcal{M}$, and the following lemma:

Lemma 3.2. *Let Σ be an alphabet. For every set $\mathcal{H} = \{H_1, \dots, H_k\}$ 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 $\Sigma = \{a_1, \dots, a_s\}$ for some $s \geqslant 1$. Let $\mathcal{H} = \{H_1, \dots, H_k\}$ be a set of H-systems over Σ, with $k \geqslant 1$. We define $\Sigma' := \Sigma \cup \{\bigstar, \$\}$, where $\bigstar$ and \$ are distinct letters that do not occur in Σ. We let

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

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 $\bigstar a_1 \cdots a_s \bigstar$. The query Q_2 consists of two parts, which are to be defined in the subqueries $\bigwedge_{1 \leqslant i \leqslant k} \phi_i^{sel}$ and $\bigwedge_{1 \leqslant i \leqslant k} \phi_i^{cod}$, 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

$$Q_2 := \mathrm{Ans}() \leftarrow \left(x_0, c_1^{\$}, x_1\right), \left(x_{k+1}, c_2^{\$}, \hat{x}_1\right), \left(\hat{x}_{k+1}, c_3^{\$}, \hat{x}_{k+2}\right), L_{\$}\left(c_1^{\$}\right), L_{\$}\left(c_2^{\$}\right), L_{\$}\left(c_3^{\$}\right), \bigwedge_{1 \leqslant i \leqslant k} \phi_i^{sel}, \bigwedge_{1 \leqslant i \leqslant k} \phi_i^{cod}$$

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

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

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

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 $m_i \geqslant 1$ and $\beta_{i,1}, \dots, \beta_{i,m_i} \in (X \cup \Sigma)$. We define the relational part of ϕ_i^{cod} to be

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

where $c_{i,3}^{\bigstar}$, $c_{i,4}^{\bigstar}$, and all $d_{i,j}$ are (pairwise distinct) new path variables. We start the construction of the labeling part of ϕ_i^{cod} with the labeling atoms $L_{\bigstar}(c_{i,3}^{\bigstar})$, $L_{\bigstar}(c_{i,4}^{\bigstar})$, $\mathrm{eq}(c_{i,1}^{\bigstar}, c_{i,3}^{\bigstar})$, and $\mathrm{eq}(c_{i,1}^{\bigstar}, c_{i,4}^{\bigstar})$. Furthermore, for every $1 \leqslant j \leqslant m_i$, we add an atom $L_{i,j}(d_{i,j})$, where the regular language $L_{i,j}$ is defined 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 $\beta_{i,j} \in \Sigma$.

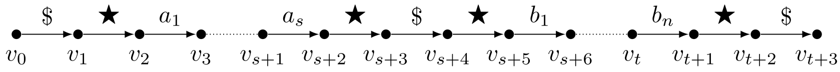

Fig. 1. An illustration of the path ρ that is characteristic for all graphs G with $Q_1(G) = \texttt{true}$. To increase readability, this figure uses $t := s + n + 4$.

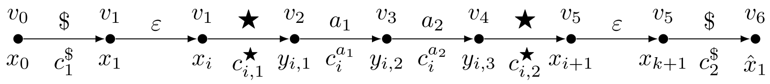

Fig. 2. An illustration of the first half of the path ρ, compared to Q_2 under the assignments σ and μ, for the special case $s = 2$. The bottom row shows the node and path variables, while the top row contains the respective nodes and path labels. See also Fig. 3 for an illustration of the second half.

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 query graph H_{Q_2} (cf. Section 2.1) consists only of a path from x_0 to $\hat{x}_{k+2}$, where each node (except $\hat{x}_{k+2}$, the last node) has exactly one successor. Thus, the query 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 3.2.

"$\Longrightarrow$": Assume that $L(\mathcal{H}) = \Sigma^*$, and let $G = (V, E)$ be a db-graph over Σ' with $Q_1(G) = \texttt{true}$. By definition of Q_1, G contains a path ρ with $\lambda(\rho) = \$\bigstar a_1 \cdots a_s \bigstar\$\bigstar w \bigstar\$$ for some $w \in \Sigma^*$. Let $w = b_1 \cdots b_n$ with $n \geqslant 0$ and $b_j \in \Sigma$ for $1 \leqslant j \leqslant n$. Accordingly, there are nodes $v_0, \ldots, v_{s+n+7} \in V$ such that

$$\rho = v_0 \, \$ \, v_1 \bigstar v_2 \, a_1 \, v_3 \cdots v_{s+1} \, a_s \, v_{s+2} \bigstar v_{s+3} \, \$ \, v_{s+4} \bigstar v_{s+5} \, b_1 \, v_{s+6} \cdots v_{s+n+4} \, b_n \, v_{s+n+5} \bigstar v_{s+n+6} \, \$ \, v_{s+n+7}.$$

See Fig. 1 for an illustration of this path. Although this does not matter for our considerations, note that the v_i are not necessarily distinct. In order to show that $Q_2(G) = \texttt{true}$, we construct a node mapping σ and a path mapping μ such that $(G, \sigma, \mu) \models Q_2$. We first define

$$\begin{aligned}
&\sigma(x_0) := v_0, && \mu\big(c_1^{\$}\big) := v_0 \, \$ \, v_1,\\
&\sigma(x_1) := v_1, && \\
&\sigma(x_{k+1}) := v_{s+3}, && \mu\big(c_2^{\$}\big) := v_{s+3} \, \$ \, v_{s+4},\\
&\sigma(\hat{x}_1) := v_{s+4}, && \\
&\sigma(\hat{x}_{k+1}) := v_{s+n+6}, && \mu\big(c_3^{\$}\big) := v_{s+n+6} \, \$ \, v_{s+n+7},\\
&\sigma(\hat{x}_{k+2}) := v_{s+n+7}. &&
\end{aligned}$$

As $L(\mathcal{H}) = \Sigma^*$, there is an i with $1 \leqslant i \leqslant k$ such that $w \in L(H_i)$. We now want to map the path described in ϕ_i^{sel} to the path between v_1 and v_{s+3} (for an illustration, see Fig. 2). In order to achieve this, we define

$$\begin{aligned}
&\sigma(x_i) := v_1, && \mu\big(c_{i,1}^{\bigstar}\big) := v_1 \bigstar v_2,\\
&\sigma(y_{i,1}) := v_2, && \mu\big(c_i^{a_1}\big) := v_2 \, a_1 \, v_3,\\
&\quad\vdots && \quad\vdots\\
&\sigma(y_{i,s+1}) := v_{s+2}, && \mu\big(c_i^{a_s}\big) := v_{s+1} \, a_s \, v_{s+2},\\
&\sigma(x_{i+1}) := v_{s+3}, && \mu\big(c_{i,2}^{\bigstar}\big) := v_{s+2} \bigstar v_{s+3}.
\end{aligned}$$

As all other ϕ_j^{sel} are not needed, we define

$$\sigma(x_j) := \begin{cases} v_1 & \text{if } 1 \leqslant j < i,\\ v_{s+3} & \text{if } i < j \leqslant k,\end{cases}$$

and $\sigma(y_{j,l}) := \sigma(x_j)$ for all $j \neq i$, $1 \leqslant j \leqslant k$ and $1 \leqslant l \leqslant s+1$. Accordingly, for all $\pi_j \in \{c_{j,1}^{\bigstar}, c_{j,2}^{\bigstar}, c_j^{a_1}, \ldots, c_j^{a_s}\}$ with $j \neq i$, we define

$$\mu(\pi_j) := \begin{cases} v_1 \, \varepsilon \, v_1 & \text{if } 1 \leqslant j < i,\\ v_{s+3} \, \varepsilon \, v_{s+3} & \text{if } i < j \leqslant k.\end{cases}$$

We can already observe that $(G, \sigma, \mu) \models Q_2$ holds modulo the subquery $\bigwedge_{1 \leqslant j \leqslant k} \phi_j^{cod}$, using the following reasoning: As $\lambda(\mu(c_j^{\$})) = \$$ for $j \in \{1, 2, 3\}$, $\lambda(\mu(c^{\$})) \in L_{\$} = \{\$\}$ is true, and $L_{\$}(c_j^{\$})$ is satisfied. Furthermore, for every $j \neq i$ with $1 \leqslant j \leqslant k$, we observe

$$\lambda\big(\mu\big(c_{j,1}^{\bigstar}\big)\big) = \lambda\big(\mu\big(c_j^{a_1}\big)\big) = \cdots = \lambda\big(\mu\big(c_j^{a_s}\big)\big) = \lambda\big(\mu\big(c_{j,2}^{\bigstar}\big)\big) = \varepsilon.$$

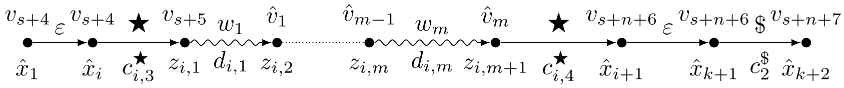

Fig. 3. An illustration of the assignments σ and μ that are defined in the only-if-direction of the proof of Lemma 3.2. As in Fig. 2, the bottom row shows the node and path variables, while the top row contains the respective nodes and path labels.

Due to $\varepsilon \in L_{\bigstar}, L_{a_1}, \ldots, L_{a_s}$, each of

$$L_{\bigstar}\big(c^{\bigstar}_{j,1}\big), L_{a_1}\big(c^{a_1}_{j}\big), \ldots, L_{a_s}\big(c^{a_s}_{j}\big), L_{\bigstar}\big(c^{\bigstar}_{j,2}\big), \mathrm{eq}\big(c^{\bigstar}_{j,1}, c^{\bigstar}_{j,2}\big)$$

is satisfied. Thus, ϕ^{sel}_j is satisfied for all $j \in \{1, \ldots, k\}$ with $j \neq i$.

Similarly, we observe

$$\begin{aligned}
&\lambda\big(\mu\big(c^{\bigstar}_{i,1}\big)\big) = \lambda\big(\mu\big(c^{\bigstar}_{i,2}\big)\big) = \bigstar \in L_{\bigstar},\\
&\lambda\big(\mu\big(c^{a_1}_{i}\big)\big) = a_1 \in L_{a_1},\\
&\quad\vdots\\
&\lambda\big(\mu\big(c^{a_s}_{i}\big)\big) = a_s \in L_{a_s},
\end{aligned}$$

which demonstrates that ϕ^{sel}_i is satisfied as well.

All that remains is to find a proper assignment of the variables in ϕ^{cod}_i that describes the second half of ρ, while all other variables describe only the empty path. An illustration of the underlying idea can be found in Fig. 3.

We define

$$\sigma(\hat{x}_j) := \begin{cases} v_{s+4} & \text{if } 1 \leqslant j < i,\\ v_{s+n+6} & \text{if } i < j \leqslant k, \end{cases}$$

and, likewise,

$$\sigma(z_{j,l}) := \begin{cases} v_{s+4} & \text{if } 1 \leqslant j < i,\\ v_{s+n+6} & \text{if } i < j \leqslant k, \end{cases}$$

for all l such that $z_{j,l}$ occurs in Q_2. Consequently, we define

$$\mu\big(c^{\bigstar}_{j,3}\big) = \mu\big(c^{\bigstar}_{j,4}\big) = \mu(d_{j,l}) = \sigma(\hat{x}_j)\ \varepsilon\ \sigma(\hat{x}_j)$$

for all $1 \leqslant j \leqslant k$, $j \neq i$.

We observe that for all $j \neq i$ with $1 \leqslant j \leqslant k$, ϕ^{cod}_j is satisfied: First, observe that

$$\lambda\big(\mu\big(c^{\bigstar}_{j,3}\big)\big) = \lambda\big(\mu\big(c^{\bigstar}_{j,4}\big)\big) = \varepsilon$$

holds. As $\lambda(\mu(c^{\bigstar}_{j,1})) = \varepsilon$, all of

$$L_{\bigstar}\big(c^{\bigstar}_{j,3}\big), L_{\bigstar}\big(c^{\bigstar}_{j,4}\big), \mathrm{eq}\big(c^{\bigstar}_{j,1}, c^{\bigstar}_{j,3}\big), \mathrm{eq}\big(c^{\bigstar}_{j,1}, c^{\bigstar}_{j,4}\big)$$

are satisfied. Furthermore, for all $d_{j,l}$ that occur in ϕ^{cod}_j, $\lambda(\mu(d_{j,l})) = \varepsilon$. Therefore, every $L_{j,l}(d_{j,l})$ is satisfied, as $\varepsilon \in L_{j,l}$ holds by Definition 2.3. Moreover, as each of these paths is an empty path, all relations $\mathrm{eq}(d_{j,l}, d_{j,l'})$ in ϕ^{cod}_j are satisfied as well, which means that ϕ^{cod}_j is satisfied.

As the last remaining task, we need to complete the definition of σ and μ such that ϕ^{cod}_i is satisfied. In order to examine H_i in detail, assume that $H_i = (\Sigma, X_i, \mathcal{L}_i, \alpha_i)$, and let $\alpha_i = \beta_{i,1} \cdots \beta_{i,m_i}$ for some $m_i \geqslant 1$ with $\beta_{i,j} \in (X_i \cup \Sigma)$ for $1 \leqslant j \leqslant m_i$. By definition of $w \in L(H_i)$, there is an H_i-compatible morphism $h : (X_i \cup \Sigma)^* \to \Sigma^*$.

As $w = h(\alpha_i)$, there is a natural decomposition of w into factors $w_1 \cdots w_{m_i}$, which are defined by $w_j := h(\beta_j)$ for $1 \leqslant j \leqslant m_i$. We take special note of the subpaths of ρ that can be derived from these w_j, and define

$$\begin{aligned}
n_0 &:= s + 5, & \hat{v}_0 &:= v_{n_0} = v_{s+5},\\
n_j &:= s + 5 + |w_1 \cdots w_j|, & \hat{v}_j &:= v_{n_j} = v_{s+5+|w_1 \cdots w_j|}
\end{aligned}$$

for each $1 \leqslant j \leqslant m_i$. Hence, for each j, the subpath of ρ between $\hat{v}_{j-1}$ and $\hat{v}_j$ is labeled with w_j. Note that $\hat{v}_{j-1} = \hat{v}_j$ might hold, in particular if $w_j = \varepsilon$. Also note that, by definition, $\hat{v}_{m_i} = v_{s+n+5}$.

Hence, we define

$$\begin{aligned}
&\sigma(\hat{x}_i) := v_{s+4}, && \mu\big(c^{\bigstar}_{i,3}\big) := v_{s+4} \bigstar v_{s+5},\\
&\sigma(z_{i,1}) := \hat{v}_0 = v_{s+5}, &&\\
&\quad\vdots &&\\
&\sigma(z_{i,m_i}) := \hat{v}_{m_i-1}, &&\\
&\sigma(z_{i,m_i+1}) := \hat{v}_{m_i} = v_{s+n+5}, && \mu\big(c^{\bigstar}_{i,4}\big) = v_{s+n+5} \bigstar v_{s+n+6},\\
&\sigma(\hat{x}_{i+1}) := v_{s+n+6}. &&
\end{aligned}$$

Finally, we define each $\mu(d_{i,j})$ with $1 \leqslant j \leqslant m_i$ to correspond to the subpath of ρ between $\hat{v}_{j-1}$ and $\hat{v}_j$ that is labeled with w_j.

We now prove that $L_{i,j}(d_{i,j})$ is satisfied for every $1 \leqslant j \leqslant m_i$. As in the definition of $L_{i,j}$, we distinguish the following cases:

1. If $\beta_{i,j} \in X_i$, $L_{i,j} = \mathcal{L}_i(\beta_{i,j})$. As $w_j = h(\beta_{i,j})$, and due to $h(\beta_{i,j}) \in \mathcal{L}_i(\beta_{i,j})$, $\lambda(\mu(d_{i,j})) \in L_{i,j}$,
2. if $\beta_{i,j} \in \Sigma$, $L_{i,j} = L_{\beta_{i,j}}$. As $w_j = h(\beta_{i,j}) = \beta_{i,j}$, $\lambda(\mu(d_{i,j})) \in L_{i,j}$ holds.

This also proves that, for every j with $\beta_{i,j} \in \Sigma$, $\lambda(\mu(d_{i,j})) = \beta_{i,j} = \lambda(\mu(c_i^{\beta_{i,j}}))$. Hence, the relations $\mathrm{eq}(c_i^{\beta_{i,j}}, d_{i,j})$ are satisfied as well. Finally, for every $\beta_{i,j} \in X_i$ that occurs more than once in α_i, we need to consider the relations $\mathrm{eq}(d_{i,j}, d_{i,l})$ for all l, j with $l \neq j$ and $\beta_{i,l} = \beta_{i,j}$. As h is a morphism, $\beta_{i,j} = \beta_{i,l}$ implies $h(\beta_{i,j}) = h(\beta_{i,l})$, and thus,

$$\lambda\big(\mu(d_{i,j})\big) = w_j = w_l = \lambda\big(\mu(d_{i,l})\big).$$

Obviously, $\mathrm{eq}(d_{i,j}, d_{i,l})$ is satisfied. We now have demonstrated that $(\sigma, \mu, G) \models Q_2$. Hence, $Q_2(G) = \texttt{true}$, and as G was chosen arbitrarily with $Q_1(G) = \texttt{true}$, $Q_1 \subseteq Q_2$ follows.

"$\Longleftarrow$": We prove this direction through its contraposition; i.e., we show that $L(\mathcal{H}) \neq \Sigma^*$ implies $Q_1 \nsubseteq Q_2$. Assume there is a $w \in \Sigma^*$ with $w \notin L(\mathcal{H})$. Let $w = b_1 \cdots b_n$ for some $n \geqslant 0$ with $b_i \in \Sigma$ for all $i \in \{1, \ldots, n\}$. We define $G := (V, E)$, where $V := \{v_0, \ldots, v_{s+n+7}\}$ (and all elements of V are pairwise distinct), and

$$\begin{aligned}
E := \big\{&(v_0, \$, v_1), (v_1, \bigstar, v_2), (v_2, a_1, v_3), \ldots, (v_{s+1}, a_s, v_{s+2}), (v_{s+2}, \bigstar, v_{s+3}), (v_{s+3}, \$, v_{s+4}),\\
&(v_{s+4}, \bigstar, v_{s+5}), (v_{s+5}, b_1, v_{s+6}), \ldots, (v_{s+n+4}, b_n, v_{s+n+5}), (v_{s+n+5}, \bigstar, v_{s+n+6}), (v_{s+n+6}, \$, v_{s+n+7})\big\}.
\end{aligned}$$

In other words, G is an acyclic graph that consists solely of a path from v_0 to v_{s+n+7} labeled $\$\bigstar a_1 \cdots a_s \bigstar \$ \bigstar w \bigstar \$$. As $w \in \Sigma^*$, $Q_1(G) = \texttt{true}$ holds. For convenience, we denote this path by ρ.

For the sake of contradiction, assume $Q_1(G) \subseteq Q_2(G)$, which necessarily implies $Q_2(G) = \texttt{true}$. Thus, there are assignments σ, μ such that $(\sigma, \mu, G) \models Q_2$. As $\lambda(\rho)$ contains exactly three occurrences of \$, and as $L_\$(c_i^\$)$ occurs in Q_2 for $1 \leqslant i \leqslant 3$, we know that σ and μ must satisfy the following conditions:

$$\begin{aligned}
&\sigma(x_0) = v_0, && \mu\big(c_1^\$\big) = v_0 \,\$\, v_1,\\
&\sigma(x_1) = v_1, &&\\
&\sigma(x_{k+1}) = v_{s+3}, && \mu\big(c_2^\$\big) = v_{s+3} \,\$\, v_{s+4},\\
&\sigma(\hat{x}_1) = v_{s+4}, &&\\
&\sigma(\hat{x}_{k+1}) = v_{s+n+6}, && \mu\big(c_3^\$\big) = v_{s+n+6} \,\$\, v_{s+n+7},\\
&\sigma(\hat{x}_{k+2}) = v_{s+n+7}. &&
\end{aligned}$$

As $\mathrm{eq}(c^{\bigstar}_{i,1}, c^{\bigstar}_{i,2})$ needs to be satisfied for all $i \in \{1, \ldots, k\}$, and as the subpath between v_1 and v_{s+5} contains exactly two occurrences of $\bigstar$, there must be exactly one i with $\lambda(\mu(c^{\bigstar}_{i,1})) = \bigstar$. We shall see that our assumption allows us to conclude that $w \in L(H_i)$, which leads to the intended contradiction. Due to our previous observations, the following must hold:

$$\begin{aligned}
&\sigma(x_i) = v_1, && \mu\big(c^{\bigstar}_{i,1}\big) = v_1 \bigstar v_2,\\
&\sigma(y_{i,1}) = v_2, && \mu\big(c_i^{a_1}\big) = v_2 \, a_1 \, v_3,\\
&\quad\vdots && \quad\vdots\\
&\sigma(y_{i,s+1}) = v_{s+2}, && \mu\big(c_i^{a_s}\big) = v_{s+1} \, a_s \, v_{s+2},\\
&\sigma(x_{i+1}) = v_{s+3}, && \mu\big(c^{\bigstar}_{i,2}\big) = v_{s+2} \bigstar v_{s+3}.
\end{aligned}$$

Now, note that Q_2 is acyclic. Therefore, the structure of G permits no other assignments than

$$\sigma(x_j) = \begin{cases} v_1 & \text{if } 1 \leqslant j < i, \\ v_{s+3} & \text{if } i < j \leqslant k, \end{cases}$$

and $\sigma(y_{j,l}) = \sigma(x_j)$ for all $j \neq i$ and all $1 \leqslant l \leqslant s+1$. Accordingly,

$$\mu\big(c^{\bigstar}_{j,1}\big) = \mu\big(c^{a_1}_j\big) = \cdots = \mu\big(c^{a_s}_j\big) = \mu\big(c^{\bigstar}_{j,2}\big) = \sigma(x_j)\ \varepsilon\ \sigma(x_j)$$

holds for all these j. Thus, only the path variables from ϕ^{sel}_i are mapped to a non-empty path. The same phenomenon occurs for the variables of ϕ^{cod}_i: As Q_2 contains relations $\mathrm{eq}(c^{\bigstar}_{i,1}, c^{\bigstar}_{i,3})$ and $\mathrm{eq}(c^{\bigstar}_{i,1}, c^{\bigstar}_{i,4})$, we conclude that

$$\begin{aligned} &\sigma(\hat{x}_i) = v_{s+4}, && \mu\big(c^{\bigstar}_{i,3}\big) = v_{s+4} \bigstar v_{s+5}, \\ &\sigma(z_{i,1}) = v_{s+5}, && \\ &\sigma(z_{i,m+1}) = v_{s+n+5}, && \mu\big(c^{\bigstar}_{i,4}\big) = v_{s+n+5} \bigstar v_{s+n+6}, \\ &\sigma(\hat{x}_{i+1}) = v_{s+n+6} && \end{aligned}$$

holds. Let the H-system H_i be defined by $H_i = (\Sigma, X_i, \mathcal{L}_i, \alpha_i)$, where $\alpha_i = \beta_{i,1} \cdots \beta_{i,m_i}$ for some $m_i \geqslant 1$. By definition of Q_2, we know that ϕ^{cod}_i contains the path variables $d_{i,1}, \ldots, d_{i,m_i}$ (in addition to $c^{\bigstar}_{i,3}$ and $c^{\bigstar}_{i,4}$). This implies

$$\lambda\big(\mu(d_{i,1})\big)\lambda\big(\mu(d_{i,2})\big) \cdots \lambda\big(\mu(d_{i,m_i})\big) = b_1 \cdots b_n = w.$$

We define words $w_j := \lambda(\mu(d_{i,j}))$ for $1 \leqslant j \leqslant m_i$. In order to prove that $w \in L(H_i)$, we show that these words can be used to define an H_i-compatible morphism h with $h(\alpha_i) = w$. First, we distinguish two possible cases for every $j \in \{1, \ldots, m_i\}$:

1. If $\beta_{i,j} \in X_i$, $L_{i,j} = \mathcal{L}_i(\beta_{i,j})$ holds by definition of Q_2. This implies that $w_j \in \mathcal{L}_i(\beta_{i,j})$.
2. If $\beta_{i,j} \in \Sigma$, $w_j = \beta_{i,j}$ must hold, as Q_2 contains label atoms $L_{\beta_{i,j}}(d_{i,j})$ and $\mathrm{eq}(c^{\beta_{i,j}}_i, d_{i,j})$, and $\lambda(\mu(c^{\beta_{i,j}}_i)) = \beta_{i,j}$.

Furthermore, for all $j \neq l$ with $\beta_{i,j} = \beta_{i,l} \in X_i$, Q_2 contains a label atom $\mathrm{eq}(d_{i,j}, d_{i,l})$. Hence, $w_j = w_l$ holds. This allows us to define a morphism $h : (X_i \cup \Sigma)^* \to \Sigma^*$ by $h(\beta_{i,j}) := w_j$ for all $j \in \{1, \ldots, m_i\}$. Furthermore, as shown above, h is H_i-compatible. Finally, $h(\alpha_i) = h(\beta_{i,1} \cdots \beta_{i,m_i}) = w$ holds by definition.

Thus, $w \in L(H_i) \subseteq L(\mathcal{H})$, which contradicts our initial assumption. This concludes the if-direction of the proof. □

By using standard encoding techniques for representing arbitrary finite alphabets by an alphabet of size 2, the proof of Theorem 3.1 now easily follows from Lemma 2.4, the undecidability of the emptiness of $\mathrm{dom}(\mathcal{M})$ for Turing machines $\mathcal{M}$, and Lemma 3.2. By using universal Turing machines instead of arbitrary Turing machines, we also obtain the following strengthening of Theorem 3.1:

Theorem 3.3. *For every alphabet Σ with $|\Sigma| \geqslant 2$, 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.

Proof. The first claim follows from the proof of Theorem 3.1 and Lemma 3.2, as Q_1 is fixed (i.e., does not depend on $\mathcal{H}$). In order to prove the second claim, we choose $\mathcal{M}$ to be a certain kind of universal Turing machine, and we use Q_1 to choose the program number of the universal machine we want to simulate.

More precisely, let $\Psi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a universal partially recursive function, i.e., for every partially recursive function $\phi : \mathbb{N} \to \mathbb{N}$, there is an $m \geqslant 0$ such that $\Psi_m(n) := \Psi(m, n) = \phi(n)$ for every $n \geqslant 0$. It is an elementary fact of recursion theory that such a function exists (cf., e.g., [13]), and moreover, there is a Turing machine $\mathcal{U}$ over some tape alphabet Γ such that

$$\mathrm{dom}(\mathcal{U}) = \big\{ \mathtt{a}^m \mathtt{b}^n \bigm| \Psi(m, n) \text{ is defined} \big\},$$

where $\mathtt{a}$, $\mathtt{b}$ are two distinct letters in the input alphabet of $\mathcal{U}$. The machine $\mathcal{U}$ can be understood as simulating, on input $\mathtt{a}^m \mathtt{b}^n$, the partial recursive function Ψ_m on the input n.

We define $\Sigma := \Gamma \cup Q \cup \{\#\} = \{a_1, \ldots, a_s\}$ (for some $s \geqslant 2$) and $\Sigma' := \Sigma \cup \{\bigstar, \$\}$, and construct Q_2 and $\mathcal{H}$ from $\mathcal{U}$ as in the proof of Lemmas 3.2 and 2.4. For every $m \geqslant 0$, we define a CRPQ $Q_{1,m}$ by

$$Q_{1,m} := \mathrm{Ans}() \leftarrow (x, \pi, y), L_m(\pi),$$

where

$$L_m := \$\bigstar a_1 \cdots a_s \bigstar \$ \bigstar \# q_0 \mathtt{a}^m \mathtt{b}^* \# \Sigma^* \bigstar \$,$$

and proceeding as in the proof of Lemma 3.2, *mutatis mutandis*. In other words, Q_1 does not "generate" arbitrary sequences, but sequences that start with the encoding of possible initial sequences of simulations of the function Ψ_m.

Then, $Q_{1,m} \subseteq Q_2$ holds if and only if $\mathrm{dom}(\mathcal{U}) \cap \mathtt{a}^m\mathtt{b}^*$ is empty, which holds if and only if $\Psi_m(n)$ is undefined on all inputs n. As decidability of this problem would allow to decide the emptiness of the domain of partial-recursive functions (an undecidable problem), the second claim follows. Again, the usual encoding techniques can be applied to replace the alphabet Σ' with a binary alphabet.

Furthermore, all queries used are Boolean and acyclic by definition. □

Applying slight modifications to the proof of Lemma 3.2, we observe the same situation for ECRPQs that use length equality instead of equality relations:

Theorem 3.4. *For every alphabet Σ with $|\Sigma| \geqslant 2$, 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.

Proof. As we shall see, it suffices to replace all occurrences of eq in the queries that are constructed in the proof of Theorem 3.3 (and the other proofs referenced therein) with el.

Note that, in order to prove Theorem 3.1, we do not need to express all possible unions of H-systems, but only those $\mathcal{H}$ that are derived from a Turing machine $\mathcal{M}$ as explained in the proof of Lemma 2.4. Furthermore, the construction in Lemma 3.2 uses the eq-predicates in two different contexts: First, on path variables that are associated with languages $\{a, \varepsilon\}$ for some $a \in \Sigma$, and second, on path variables that simulate variables x in H-systems $(\Sigma, X, \mathcal{L}, \alpha)$ such that $|\alpha|_x \geqslant 2$.

For the first case, we can simply replace eq with el without changing the behavior of the query: Obviously, for all $w, w' \in \{a, \varepsilon\}$, $w = w'$ holds if and only if $|w| = |w'|$.

Regarding the second case, note that almost all H-systems that are derived from the proof of Lemma 2.4 describe regular languages. The only non-regular languages that are constructed describe cases where $w \neq w'$ holds for certain words $w, w' \in \Gamma^*$, and characterizes this relation by $w \neq w'$ if and only if

1. there exist words $u, v, v' \in \Gamma^*$ and letters $c, d \in \Gamma$ with $c \neq d$, $w = ucv$, and $w' = udv'$, or
2. exactly one of w, w' is the empty word.

The first condition holds if and only if there exist words $u, u'v, v' \in \Gamma^*$ and letters $c, d \in \Gamma$ with $c \neq d$, $w = ucv$, $w' = u'dv'$, and $|u| = |u'|$, which demonstrate that in this case, the replacement of eq with el leads to the same results.

Hence, in order to prove Theorem 3.4, it suffices to replace eq with el in the proof of Lemma 3.2 and to proceed as in the proof of Theorem 3.3. □

3.2. Query equivalence

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

An open question raised in [3] is whether the equivalence problem for CRPQs and ECRPQs is decidable. Using a variant of the proof of Theorem 3.3, we can answer this negatively:

Theorem 3.5. *For every alphabet Σ with $|\Sigma| \geqslant 2$, 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.

Proof. Theorem 3.5 can be obtained from the proof of Theorem 3.3 by using the following lemma instead of Lemma 3.2:

Lemma 3.6. *Let Σ be an alphabet. For every regular language $L \subseteq \Sigma^*$ and every set $\mathcal{H} = \{H_1, \ldots, H_k\}$ of H-systems over Σ, one can effectively construct a CRPQ Q_1 and an ECRPQ Q_2 such that $Q_1 \equiv Q_2$ if and only if $L(\mathcal{H}) = L$.*

Proof. Let Σ, Σ', and Q_1 be defined as in the proof of Lemma 3.2, and let $L \subseteq \Sigma^*$ be a regular language.

We define $Q_1 := \mathrm{Ans}() \leftarrow (x, \pi, y), L'(\pi)$, where $L' := \$\bigstar a_1 \cdots a_s \bigstar \$ \bigstar L \bigstar \$$ (as L is regular, L' is regular as well). In order to define Q_2, we introduce the regular k-ary relation $\mathrm{xor}(w_1, \ldots, w_k)$, which is defined by

$$\mathrm{xor} := \{(w_1, \ldots, w_k) \mid \text{there is exactly one } i \in \{1, \ldots, k\} \text{ with } w_i \neq \varepsilon\}.$$

We now obtain Q_2 by adding $\mathrm{xor}(c^\bigstar_{1,1}, \ldots, c^\bigstar_{k,1})$ to the query Q_2 used in the proof of Lemma 3.2. Then $Q_2(G) = \mathtt{true}$ holds if and only if G contains a path ρ with $\lambda(\rho) \in \$\bigstar\ a_1 \cdots a_s\ \bigstar\$\bigstar L(\mathcal{H})\bigstar\$$. This concludes the proof of Lemma 3.6. □

If query equivalence were decidable, we could use Lemma 3.6 to decide whether $\mathrm{INVALC}(\mathcal{M}) = L$ for every Turing machine $\mathcal{M}$ and every regular language L. As this problem is undecidable, query equivalence must be undecidable. Hence, Theorem 3.5 follows. □

Note that the ECRPQs in the proof of Theorem 3.5 use only the relatively simple xor-relation in addition to the equality relations that are from the proof of Theorem 3.1. As in the proof of Theorem 3.4, this construction can be adapted to use length equality relations instead of equality relations.

4. Expressiveness and relative succinctness

4.1. Expressiveness of (E)CRPQs

In this section, we examine the expressive power of CRPQs and ECRPQs. In particular, we present a class of query functions for which we characterize expressibility in CRPQs, and in ECRPQs over unary alphabets (i.e., alphabets containing only one letter).

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^B_L by $F^B_L(G) := \mathtt{true}$ 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 $w = a_1 \cdots a_{|w|}$ (with all $a_i \in \Sigma$), 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 $E_w = \{(v_i, a_{i+1}, v_{i+1}) \mid 0 \leqslant i < |w|\}$. 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 $(v_0, v_{|w|}) \in Q_L(G_w)$.

Theorem 4.1. *Let Σ be an alphabet, let $L \subseteq \Sigma^*$. Then F_L is CRPQ-expressible if and only if L is regular.*

Proof. The *if*-direction is obvious: If L is regular, the CRPQ

$$\mathrm{Ans}(x, y) \leftarrow (x, \pi, y), L(\pi)$$

expresses F_L.

To prove the *only if*-direction, we will use the following notation: If $w = a_1 \cdots a_n \in \Sigma^+$ with $a_i \in \Sigma$ for all $i \in \{1, \ldots, n\}$, and $1 \leqslant p < q \leqslant n+1$, we write $w[p, q)$ to denote the string $a_p \cdots a_{q-1}$, and $w[p, p)$ denotes the empty string.

Now let $L \subseteq \Sigma^*$, and assume there exists a CRPQ Q_L with $Q_L(G) = F_L(G)$ for every Σ-labeled db-graph G. Our goal is to show that L is regular.

Due to Lemma 2.2 we can assume w.l.o.g. that Q_L is of the form

$$\mathrm{Ans}(x, y) \leftarrow \bigwedge_{1 \leqslant i \leqslant m} (x_i, \pi_i, y_i), \bigwedge_{1 \leqslant i \leqslant m} L_i(\pi_i).$$

Let $l := |\mathrm{nvar}(Q)|$ be the number of node variables occurring in Q, and let $z_1, \ldots, z_l$ be an enumeration of the elements of $\mathrm{nvar}(Q)$. Let $j_0, k_0 \in \{1, \ldots, l\}$ such that $x = z_{j_0}$ and $y = z_{k_0}$, and for each $i \in \{1, \ldots, m\}$ let $j_i, k_i \in \{1, \ldots, l\}$ be such that $x_i = z_{j_i}$ and $y_i = z_{k_i}$.

Since Q_L defines F_L, we know that for all strings $w \in \Sigma^+$ we have

$$w \in L \iff (v_0, v_{|w|}) \in Q_L(G_w). \tag{2}$$

$$\iff \text{There exist nodes } b_1, \ldots, b_l \text{ in } G_w \text{ such that } b_{j_0} = v_0,\ b_{k_0} = v_{|w|}, \text{ and for every } i \in \{1, \ldots, m\}, \text{ there is a path } \pi_i \text{ from } b_{j_i} \text{ to } b_{k_i} \text{ in } G_w \text{ that is labeled by a string belonging to } L_i. \tag{3}$$

$$\iff \quad \text{There exist } p_1, \ldots, p_l \text{ with } 1 \leqslant p_i \leqslant |w|+1 \text{ such that } p_{j_0} = 1,\ p_{k_0} = |w|, \text{ and for every } i \in \{1, \ldots, m\}, \text{ we have } p_{j_i} \leqslant p_{k_i} \text{ and the string } w[p_{j_i}, p_{k_i}) \text{ belongs to } L_i. \tag{4}$$

By *Büchi's Theorem* (cf., e.g., [15]), a language $L' \subseteq \Sigma^+$ is regular if, and only if, it is definable in *monadic second-order logic* MSO.

Since each of the languages L_i (for $1 \leqslant i \leqslant m$) is regular, we therefore obtain MSO-formulas $\phi_i(u, v)$ (with free first-order variables u, v) such that the following is true for all strings $w \in \Sigma^+$ and all positions p, q in w with $p \leqslant q$:

$$w \models \phi_i(p,q) \quad \iff \quad w[p,q) \in L_i.$$

Furthermore, since MSO is closed under Boolean combinations and first-order quantification, it is straightforward to see that the statement of Eq. (4) can be expressed by an MSO-formula ϕ. Thus, for all strings $w \in \Sigma^+$ we have $w \in L$ iff $w \models \phi$. By Büchi's Theorem, we therefore obtain that the language $L \setminus \{\varepsilon\}$ is regular. Consequently, also the language L is regular. This concludes the proof of Theorem 4.1. $\square$

For *Boolean* queries, the situation is somewhat different. For example, if $\Sigma \subseteq L$, then $F_L^B(G) = \texttt{true}$ holds for *all* non-empty db-graphs G. However, Theorem 4.1 immediately implies the following:

Corollary 4.2. *Let Σ be an alphabet with $|\Sigma| \geqslant 2$, 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 $\geqslant 2$, ECRPQs can express queries F_L for non-regular $L \subseteq \Sigma^*$ which, according to Theorem 4.1, 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:

Theorem 4.3. *Let Σ be a unary alphabet, let $L \subseteq \Sigma^*$. Then F_L is ECRPQ-expressible if and only if it is CRPQ-expressible.*

Theorem 4.3 follows immediately from the stronger Theorem 4.4. For formulating Theorem 4.4, we need to introduce the following notion of *rational relations* (cf. [16]), which was also examined in [3]: While k-ary regular relations over an alphabet Σ are defined using regular expressions over the alphabet $(\Sigma_\perp)^k$, k-ary rational relations are defined over the alphabet $(\Sigma_{\perp,\varepsilon})^k$, where $\Sigma_{\perp,\varepsilon} := \Sigma \cup \{\perp, \varepsilon\}$. For example, the rational relation $R = \{(w_1, w_2) \mid w_1, w_2 \in \{a\}^*,\ |w_2| = 2|w_1|\}$ corresponds to the regular expression $((a,a)(\varepsilon,a))^*$.

Note that in the literature (e.g. [16]), k-ary regular relations are usually defined to be those relations that are accepted by a *k-synchronous transducer* – a finite automaton with k tapes and one head for each tape, where all heads read the input synchronously from left to right. For *rational relations*, the requirement of synchronous movement is dropped. Using standard techniques, it is easily seen that the definition via regular expressions and the definition via k-tape transducers are of equivalent power.

An ECRPQ that uses a rational relation is defined as an ECRPQ whose labeling part consists of exactly one relation – but this relation is allowed to be a rational relation (instead of a regular relation).

By Lemma 2.1 and by the fact that every rational relation is a regular relation, every query that is ECRPQ-expressible can also be expressed by an ECRPQ that uses a rational relation. Hence, Theorem 4.3 follows from the following result:

Theorem 4.4. *Let Σ be a unary alphabet, let $L \subseteq \Sigma^*$. Then F_L can be expressed by an ECRPQ that uses a rational relation if and only if F_L is CRPQ-expressible.*

Proof. The *if*-direction holds by definition and by Lemma 2.1, as every CRPQ is an ECRPQ, and every ECRPQ can be rewritten to use a single relation that is regular (and, hence, rational). Before we proceed to the proof of the *only if*-direction, we introduce some basic definitions. For every $k \geqslant 1$ and every vector $a \in \mathbb{N}^k$, define $a\mathbb{N} := \{a \cdot i \mid i \in \mathbb{N}\}$. For all sets $A, B \subseteq \mathbb{N}^k$, let $A + B := \{a + b \mid a \in A,\ b \in B\}$, where addition of k-tuples is defined component-wise. A set $A \subseteq \mathbb{N}^k$ is *linear* if there exist $a_0, \ldots, a_n \in \mathbb{N}^k$ for some $n \geqslant 0$ such that $A = a_0 + a_1\mathbb{N} + \cdots + a_n\mathbb{N}$. A set is *semi-linear* if it is a finite union of linear sets.

Let $A = \{a_1, \ldots, a_k\}$ be a (finite) alphabet. The *Parikh mapping* (for A) is the function $\psi : A^* \to \mathbb{N}^k$ that is defined as $\psi(w) := (|w|_{a_1}, \ldots, |w|_{a_k})$ for all $w \in A^*$. We extend this to the *Parikh image* of a language $L \subseteq A^*$ by $\psi(L) := \{\psi(w) \mid w \in L\}$. A language L is called semi-linear iff its Parikh image $\psi(L)$ is semi-linear.

Let A be any set, and let $k \geqslant 1$. For every $(a_1, \ldots, a_k) \in A^k$, we define functions $\mathrm{proj}_i(a_1, \ldots, a_k) := a_i$ for all $i \in \{1, \ldots, k\}$. Thus, the function proj_i projects an element of A^k to its i-th component.

For the *only if*-direction, let $\Sigma := \{a\}$, and let $L \subseteq \Sigma^*$ be such that F_L can be expressed with an ECRPQ that uses a rational relation. Our aim throughout the remainder of this proof is to show that L is regular. Note that by (the easy direction of) Theorem 4.1, regularity of L implies that F_L is CRPQ-expressible.

Since F_L can be expressed by an ECRPQ that uses a rational relation, there exists a Q_L of the form

$$\mathrm{Ans}(x, y) \leftarrow \bigwedge_{1 \leqslant i \leqslant k} (x_i, \pi_i, y_i), R(\pi_1, \dots, \pi_k),$$

such that $Q_L(G) = F_L(G)$ for every Σ-labeled db-graph G.

We interpret the rational relation R as a regular language over the alphabet $A := \{a, \perp, \varepsilon\}^k$. Let $\psi : A^* \to \mathbb{N}$ denote the Parikh mapping for A. As R is a rational relation over Σ, it represents a regular language over A, which means that its Parikh set $\psi(R) \subseteq \mathbb{N}^{3^k}$ is semi-linear (cf., e.g., [11]).

We define the set $R_{\mathrm{len}} \subseteq \mathbb{N}^k$ by

$$R_{\mathrm{len}} := \bigl\{ \bigl(|w_1|, \dots, |w_k|\bigr) \bigm| (w_1, \dots, w_k) \in R \bigr\}.$$

Claim 1. R_{len} *is semi-linear.*

Proof. Let $b_1, \dots, b_{3^k}$ be the enumeration of $\{a, \perp, \varepsilon\}^k$ that corresponds to ψ (i.e., for every $1 \leqslant i \leqslant 3^k$, we have $b_i \in \{a, \perp, \varepsilon\}^k$, $\mathrm{proj}_i(\psi(b_i)) = 1$, and all other positions of $\psi(b_i)$ are 0). We define functions $f_i : \mathbb{N}^{3^k} \to \mathbb{N}$ with $1 \leqslant i \leqslant k$ by

$$f_i(n_1, \dots, n_{3^k}) := \sum_{j:\ \mathrm{proj}_i(b_j) = a} n_j,$$

and extend this to a function $f : \mathbb{N}^{3^k} \to \mathbb{N}^k$ by

$$f(\bar{n}) := \bigl(f_1(\bar{n}), \dots, f_k(\bar{n})\bigr)$$

for every $\bar{n} \in \mathbb{N}^{3^k}$. It is easy to see that $R_{\mathrm{len}} = f(\psi(R))$. As $\psi(R)$ is semi-linear, there exist an $m \geqslant 0$ and linear sets $R_1, \dots, R_m \subseteq \mathbb{N}$ such that $\psi(R) = \bigcup_{i=1}^m R_i$. Thus, $R_{\mathrm{len}} = \bigcup_{i=1}^m f(R_i)$.

Every R_i is a linear set, hence, for every $1 \leqslant i \leqslant m$, there exist an $n \geqslant 0$ and $c_0, \dots, c_n \in \mathbb{N}^{3^k}$ with $R_i = c_0 + c_1\mathbb{N} + \dots + c_n\mathbb{N}$. Therefore, $f(R_i) = f(c_0) + f(c_1)\mathbb{N} + \dots + f(c_n)\mathbb{N}$, which demonstrates that $f(R_i)$ is a linear subset of $\mathbb{N}^k$. Hence, R_{len} is semi-linear, concluding the proof of Claim 1. □

The next step is the construction of a relation that extends R_{len} by not only describing the lengths of paths that are obtained from a single path variable, but to paths that are formed by connecting these single paths.

A *label sequence* (*in* Q_L) is a sequence $i_1, \dots, i_m$ with $m \geqslant 1$, and

1. $1 \leqslant i_j \leqslant k$ for all $1 \leqslant j \leqslant m$ (every i_j corresponds to the path variable π_{i_j} in Q_L),
2. $i_j \neq i_{j'}$ if $j \neq j'$,
3. there exist $z_0, \dots, z_m \in \mathrm{nvar}(Q_L)$ such that $(z_j, \pi_{i_{j+1}}, z_{j+1})$ is an atom in Q_L for every $0 \leqslant j < m$.

Hence, every label sequence describes a non-empty, acyclic path through the labeled query graph $H^{lab}_{Q_L}$; moreover, for every label sequence, the corresponding node variables $z_0, \dots, z_m$ are uniquely defined, as every path variable occurs exactly once in the relational part of Q_L.

For every label sequence p with corresponding node variables $z_0, \dots, z_{m_p}$, we define $\mathrm{start}(p) := z_0$, $\mathrm{end}(p) := z_{m_p}$, and let $\mathrm{lab}(p) \subseteq \{1, \dots, k\}$ denote all i_j that occur in p.

Let $\mathcal{P} = \{p_1, \dots, p_l\}$ with $l \geqslant 1$ denote the set of all label sequences in Q_L (as there is only a finite number of path variables in Q_L, and no index i_j occurs twice in a label sequence, $\mathcal{P}$ is finite). Without loss of generality, assume that $\mathrm{start}(p_1) = x$ and $\mathrm{end}(p_1) = y$ hold; i.e., p_1 corresponds to a path from x to y in $H^{lab}_{Q_L}$.

For each p_i in $\mathcal{P}$, we define a function $\hat{p}_i : \mathbb{N}^k \to \mathbb{N}$ by

$$\hat{p}_i(r_1, \dots, r_k) := \sum_{j \in \mathrm{lab}(p_i)} r_j.$$

Hence, if $r \in R_{\mathrm{len}}$ (and, hence, corresponds to the path lengths in an assignment that satisfies Q_L), $\hat{p}_i(r)$ is the length of the path between $\mathrm{start}(p_i)$ and $\mathrm{end}(p_i)$ along the edges labeled with π_j for $j \in \mathrm{lab}(p_i)$.

We combine the functions $\hat{p}_i$ to a function $\hat{p} : \mathbb{N}^k \to \mathbb{N}^l$ by

$$\hat{p}(\bar{r}) := \bigl(\hat{p}_1(\bar{r}), \dots, \hat{p}_l(\bar{r})\bigr)$$

for all $\bar{r} \in \mathbb{N}^k$, and define

$$\hat{p}(R_{\mathrm{len}}) := \bigl\{ \hat{p}(\bar{r}) \bigm| \bar{r} \in R_{\mathrm{len}} \bigr\}.$$

Since R_{len} is semi-linear (cf. Claim 1), it is easy to see that also the following is true:

Claim 2. *$\hat{p}(R_{\text{len}})$ is semi-linear.*

We now define the equivalence relation $\equiv$ on $\mathcal{P}$ by $p_i \equiv p_j$ if $\text{start}(p_i) = \text{start}(p_j)$ and $\text{end}(p_i) = \text{end}(p_j)$. For every $1 \leqslant i \leqslant l$, let

$$S_i := \{(s_1, \ldots, s_l) \in \mathbb{N}^l \mid s_j = s_i \text{ for all } j \text{ with } p_i \equiv p_j\},$$

and define

$$B := \{(s_1, \ldots, s_l) \in \mathbb{N}^l \mid s_j \leqslant s_1 \text{ for all } j\},$$

$$T := \hat{p}(R_{\text{len}}) \cap B \cap \bigcap_{i=1}^{l} S_i.$$

Intuitively, $\bigcap_{i=1}^{l} S_i$ enforces that all paths with the same exterior nodes are assigned paths of the same lengths. Furthermore, the set B ensures that no path is longer than the path described by p_1.

Note that B and all S_i are semi-linear. Due to Claim 2 and the fact that the class of semi-linear sets is closed under intersection and projection (cf. Ginsburg and Spanier [10]), we obtain:

Claim 3. *The set $T_1 := \{\text{proj}_1(t) \mid t \in T\}$ is semi-linear.*

We are now able to state the claim that shall allow us to finish the proof.

Claim 4. *$T_1 = \psi_a(L)$, where $\psi_a : \{a\}^* \to \mathbb{N}$ is the Parikh mapping of $\{a\}$.*

Before proving Claim 4, note that by Claims 4 and 3 we obtain that $\psi_a(L)$ is semi-linear and thus, since $L \subseteq \{a\}^*$, L must be regular. To conclude the proof of Theorem 4.3, it therefore suffices to prove Claim 4.

Proof of Claim 4. To prove the claim, we examine the behavior of Q_L on the db-graphs $G_n := G_w$ with $w = a^n$ for $n \geqslant 0$.

To show that $\psi_a(L) \subseteq T_1$, let $n \geqslant 0$ such that $n \in \psi_a(L)$, i.e., $a^n \in L$. Then $(v_0, v_n) \in Q_L(G_n)$ holds by definition, and there exist assignments σ, μ such that $(G_n, \sigma, \mu) \models Q_L$, $\sigma(x) = v_0$, and $\sigma(y) = v_n$ hold. We define

$$\bar{r} := (\lambda(\mu(\pi_1)), \ldots, \lambda(\mu(\pi_k))),$$

and observe that $\bar{r} \in R$ holds by definition. Hence, for

$$\bar{r}_{\text{len}} := (|\lambda(\mu(\pi_1))|, \ldots, |\lambda(\mu(\pi_k))|),$$

we have $\bar{r}_{\text{len}} \in R_{\text{len}}$, and, consequently, $\hat{p}(\bar{r}_{\text{len}}) \in \hat{p}(R_{\text{len}})$. As the path that corresponds to p_1 (the path from v_0 to v_n) is the longest possible path in G_n,

$$\text{proj}_j(\hat{p}(\bar{r}_{\text{len}})) \leqslant \text{proj}_1(\hat{p}(\bar{r}_{\text{len}}))$$

holds for all $j \in \{1, \ldots, l\}$. Thus, $\hat{p}(\bar{r}_{\text{len}}) \in B$.

Furthermore, for all $p_i, p_j \in \mathcal{P}$ with $p_i \equiv p_j$, there is exactly one path in G_n between $\sigma(\text{start}(p_i))$ and $\sigma(\text{end}(p_i))$. Hence, the two paths that result from the assignment of paths to their path variables under μ are identical, which means that $\hat{p}(\bar{r}_{\text{len}}) \in S_i$ holds for all $i \in \{1 \ldots, l\}$.

In summary, $\hat{p}(\bar{r}_{\text{len}}) \in T$, and hence $\psi_a(a^n) = n = \text{proj}_1(\hat{p}(\bar{r}_{\text{len}})) \in T_1$.

To show that $T_1 \subseteq \psi_a(L)$, let $n \geqslant 0$ with $n \in T_1$. Our goal is to show that $n \in \psi_a(L)$, i.e., $a^n \in L$. Since $n \in T_1$, by definition of T_1 and T there exists a $\bar{t} \in T$ with $n = \text{proj}_1(\bar{t})$, and there exists an $\bar{r}_{\text{len}} \in R_{\text{len}}$ with $\bar{t} = \hat{p}(\bar{r}_{\text{len}})$.

We now use $\bar{r}_{\text{len}}$ to define assignments σ, μ with $(G_n, \sigma, \mu) \models Q_L$, $\sigma(x) = v_0$, $\sigma(y) = v_n$ as follows: First, we choose $\sigma(x) := v_0$ and $\sigma(y) := v_n$. We then follow p_1 and assign paths and nodes according to the respective path lengths in $\bar{r}_{\text{len}}$. We then proceed analogously for all other $p_i \in \mathcal{P}$ with $p_i \equiv p_1$. As $\bar{t} \in S_j$ holds for all $1 \leqslant j \leqslant l$, this process is well-defined.

In order to assign the remaining variables and paths, we first process all $p_i \in \mathcal{P}$ that start at x, but end in variables z such that there is no $p_j \in \mathcal{P}$ with $\text{start}(p_j) = z$ and $\text{end}(p_j) = y$. Again, we assign node variables and path variables accordingly. As $\bar{t} \in B$, we know that the resulting paths cannot have a length of more than n; hence, these assignments are possible. Analogously, we work backwards from y, and process all remaining variables that lead to y.

Next, observe that for all label sequences $p_i \in \mathcal{P}$ with $\text{end}(p_i) = x$ or $\text{start}(p_i) = y$, $\text{proj}_i(\bar{t}) = 0$ must hold, as otherwise, this label sequence and p_1 could be concatenated to form a label sequence $p_j \in \mathcal{P}$ with $\text{proj}_j(\bar{t}) > \text{proj}_1(\bar{t})$, which would contradict $\bar{t} \in B$. Hence, all respective node variables can be assign to x or y, and all these paths are assigned the empty path.

In terms of the labeled query graph $H_{Q_L}^{lab}$, this process yields assignments for all node variables $z \in \text{nvar}(Q)$ that are connected to x (or y), and the respective path variables that occur on the edges. Any unassigned variable must occur in a

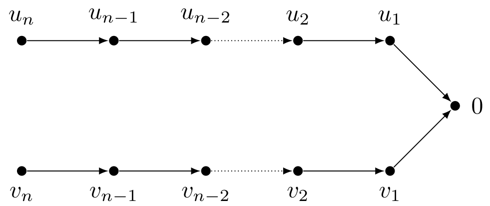

Fig. 4. The graph G_n used in the proof of Proposition 4.5.

subgraph of $H^{lab}_{Q_L}$ that is disconnected from the subgraph that contains x. For each such subgraph, pick a node of in-degree 0 and treat it like x, or a node of out-degree 0 and treat it like y, again working forwards or backwards. Again, $\bar{t} \in B$ ensures that such an assignment is possible, and $\bar{t} \in \bigcap_{i=1}^{l} S_i$ prevents inconsistencies as well as problems with cycles.

As σ and μ were derived from R_{len} (and, hence, R), (G_n, σ, μ) holds. Hence, $(v_0, v_n) \in Q_L(G_n)$. Since Q_L defines F_L, we thus have $a^n \in L$, i.e., $n \in \psi_a(L)$, concluding the proof of Claim 4 and the proof of Theorem 4.3. □

In Section 3.1 of [3], Barceló et al. mention that ECRPQs are able to express queries corresponding to *regular expressions with backreferencing* (or *extended regular expressions*) (cf., e.g., [1,8]). These expressions extend the regular expressions with variable binding and repetition operators. For example, for every expression α, the extended expression $(\alpha)\%x\ xx$ generates the language $\{www \mid w \in L(\alpha)\}$: the expression α generates some $w \in L(\alpha)$, $\%x$ assigns this w to x, and the subsequent uses of x repeat this w – hence, xx generates ww.

Let $L := \{a^n \mid n \geqslant 4,\ n \text{ is a composite number}\}$. According to Theorem 4.3, 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. [5]). This demonstrates that ECRPQs are not able to express all queries that correspond to extended regular expressions.

Note that, in spite of Theorem 4.3, there exist ECRPQ-queries over unary alphabets that are *not* CRPQ-expressible:

Proposition 4.5. *Let Σ be a unary alphabet, and define the ECRPQ Q_E as*

$$Q_E := \text{Ans}(x, y) \leftarrow (x, \pi_1, z), (y, \pi_2, z), \text{el}(\pi_1, \pi_2).$$

Then Q_E is not CRPQ-expressible.

Proof. Let $\Sigma = \{\text{a}\}$. Assume for the sake of a contradiction that there exists a CRPQ Q with $Q \equiv Q_E$, where (without loss of generality)

$$Q = \text{Ans}(x, y) \leftarrow \bigwedge_{1 \leqslant i \leqslant m} (x_i, \pi_i, y_i), L_i(\pi_i)$$

for some $m \in \mathbb{N}$.

For every $n \in \mathbb{N}$, we consider the Σ-labeled db-graph G_n illustrated in Fig. 4. That is, $G_n = (\mathcal{U}_n \cup \mathcal{V}_n \cup \{0\}, E)$, where $\mathcal{U}_n = \{u_1, \dots, u_n\}$, $\mathcal{V}_n = \{v_1, \dots, v_n\}$, and

$$E = \{(u_i, \text{a}, u_{i-1}) \mid 2 \leqslant i \leqslant n\} \cup \{(u_1, \text{a}, 0)\} \cup \{(v_i, \text{a}, v_{i-1}) \mid 2 \leqslant i \leqslant n\} \cup \{(v_1, \text{a}, 0)\}.$$

Since $Q \equiv Q_E$, we know that $(u_k, v_k) \in Q(G_n)$, for every $k \in \{1, \dots, n\}$. Furthermore, due to the structure of the graph G_n, for all assignments σ, μ such that $(G_n, \sigma, \mu) \models Q$, and for every path variable π_i with $1 \leqslant i \leqslant m$, the nodes in $\mu(\pi_i)$ belong either to the set $\mathcal{U}_n \cup \{0\}$ (we say they belong to the *$\mathcal{U}$-part* of G_n), or to the set $\mathcal{V}_n \cup \{0\}$ (we say they belong to the *$\mathcal{V}$-part* of G_n). Thus, every π_i is assigned within either the $\mathcal{U}$-part or the $\mathcal{V}$-part of G_n.

Note that, for sufficiently large n, there must be $k, \ell \in \{1, \dots, n\}$ with $k \neq \ell$ such that the assignments σ_k, μ_k and σ_ℓ, μ_ℓ proving that (u_k, v_k) and (u_ℓ, v_ℓ) belong to $Q(G_n)$ have the following property:

(∗) For each $i \in \{1, \dots, m\}$, $\mu_k(\pi_i)$ and $\mu_\ell(\pi_i)$ either both belong to the $\mathcal{U}$-path of G_n, or they both belong to the $\mathcal{V}$-path of G_n.

This allows us to obtain a contradiction by constructing a new path assignment μ by combining the "$\mathcal{U}$-part" of μ_k, and the "$\mathcal{V}$-part" of μ_ℓ: We define μ by

$$\mu(\pi_i) := \begin{cases} \mu_k(\pi_i) & \text{if } \mu_k(\pi_i) \text{ belongs to the } \mathcal{U}\text{-part of } G_n, \\ \mu_\ell(\pi_i) & \text{otherwise} \end{cases}$$

for all $i \in \{1, \dots, m\}$. The node assignment σ is chosen accordingly.

Since $(G_n, \sigma_k, \mu_k) \models Q$ and $(G_n, \sigma_\ell, \mu_\ell) \models Q$, we know by the construction of σ, μ that also $(G_n, \sigma, \mu) \models Q$. Furthermore, since σ_k, μ_k and σ_ℓ, μ_ℓ certify that (u_k, v_k) and (u_ℓ, v_ℓ) belong to $Q(G_n)$, we know that $u_k = \sigma_k(x) = \sigma(x)$ and

$v_\ell = \sigma_\ell(y) = \sigma(y)$. Thus, (σ, μ) certifies that (u_k, v_ℓ) belongs to $Q(G_n)$. Since $k \neq \ell$, however, the (u_k, v_ℓ) does not belong to $Q_E(G_n)$. □

4.2. Relative succinctness

In this section, we first obtain an undecidability result on the CRPQ-expressibility of ECRPQs. From this result, we derive a statement of the relative succinctness of ECRPQs in comparison to CRPQs.

We can adapt Lemma 3.2 to determine the location of CRPQ-expressibility for ECRPQs in the arithmetical hierarchy (cf., e.g., [13]); namely on its second level, in Σ_2^0. Recall that Σ_2^0 contains problems that are neither semi-decidable, nor co-semi-decidable (cf. [13]).

Theorem 4.6. *CRPQ-expressibility of ECRPQs is Σ_2^0-complete. Hence, this problem is neither semi-decidable, nor co-semi-decidable.*

Proof. First, note that for any Turing machine $\mathcal{M}$, dom($\mathcal{M}$) is finite if, and only if, INVALC($\mathcal{M}$) is regular: if dom($\mathcal{M}$) is finite, INVALC($\mathcal{M}$) is co-finite and thus regular; if dom($\mathcal{M}$) is infinite, non-regularity of INVALC($\mathcal{M}$) can be established using standard tools.

Furthermore, finiteness of dom($\mathcal{M}$) (and thus, regularity of INVALC($\mathcal{M}$)) is a Σ_2^0-complete problem in the arithmetical hierarchy (again, cf. [13]).

Clearly, INVALC($\mathcal{M}$) is regular if, and only if, the language

$$L := \bigstar\, a_1 \cdots a_s \bigstar \$ \bigstar \mathrm{INVALC}(\mathcal{M}) \bigstar$$

is regular.

Now note that by Lemma 2.4 and by the proof of Lemma 3.6 we obtain an ECRPQ $Q_{\mathcal{M}}$ such that for any db-graph G, $Q_{\mathcal{M}}(G) = \texttt{true}$ iff G contains a path whose label belongs to $\$L\$$. By modifying the first and the last \$ symbol into a new symbol \$′, we obtain that the Boolean query $F^B_{\$' L \$'}$ is ECRPQ-expressible. From Corollary 4.2 we know that this query is CRPQ-expressible if, and only if, L is regular.

In summary, the ECRPQ-expressible query $F^B_{\$' L \$'}$ is CRPQ-expressible if, and only if, INVALC($\mathcal{M}$) is regular. As the latter problem is Σ_2^0-complete, CRPQ-expressibility is Σ_2^0-hard.

To show membership in Σ_2^0, it suffices to observe that CRPQ-expressibility can be expressed with a Σ_2^0-formula: An ECRPQ Q is CRPQ-expressible if and only if there exists a CRPQ Q' such that for all db-graphs G, $Q(G) = Q'(G)$ holds. This concludes the proof of Theorem 4.6. □

Using Theorem 4.6 in conjunction with a technique that is due to Hartmanis [12] and has been widely used in Formal Language Theory (cf., e.g., [14]), we obtain a result on the relative succinctness of ECRPQs and CRPQs. One of the benefits of this 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 4.7. *Let Σ be a finite alphabet with $|\Sigma| \geqslant 2$. 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 $Q' \equiv Q$, $c(Q') > f(c(Q))$.*

Proof. Let Σ be a finite alphabet with $|\Sigma| \geqslant 2$, and let c be a complexity measure for ECRPQs. Assume, to the contrary, that there exists a recursive function $f_c : \mathbb{N} \to \mathbb{N}$ such that, for every CRPQ-expressible ECRPQ Q over Σ, there is a CRPQ Q' with $Q \equiv Q'$ and $c(Q') \leqslant f(c(Q))$. We shall now demonstrate that this implies that the set

$$\Delta := \{Q \mid Q \text{ is an ECRPQ over } \Sigma \text{ that is not CRPQ-expressible}\}$$

is semi-decidable. This, in turn, would imply that CRPQ-expressibility for ECRPQs is co-semi-decidable, and contradict Theorem 4.6.

Under our assumptions, the semi-decision procedure for Δ can be defined as follows: Given an ECRPQ Q, compute $n := f_c(c(Q))$, and let F_n be the set of all CRPQs Q' over Σ with $c(Q') \leqslant n$. As c is a complexity measure, F_n is finite. Furthermore, as we can decide whether an ECRPQ is a CRPQ, we can compute a list of all elements of F_n (as we can effectively enumerate all ECRPQs Q'' with $c(Q'') \leqslant n$).

For every $Q' \in F_n$, we semi-decide $Q \not\equiv Q'$ by searching for a Σ-labeled db-graph $G_{Q'}$ with $Q(G_{Q'}) \neq Q'(G_{Q'})$. If $Q' \not\equiv Q$ holds, such a $G_{Q'}$ can be found in finite time, and if we have found a graph $G_{Q'}$ for *every* $Q' \in F_n$, we let the procedure return the output "$\texttt{yes}$".

By our choice of f_c (and, hence, F_n), Q is not CRPQ-expressible if and only if $Q \not\equiv Q'$ holds for every $Q' \in F_n$. Hence, this procedure is a semi-decision procedure for Δ, which implies that CRPQ-expressibility for ECRPQs over Σ is co-semi-decidable. This contradicts Theorem 4.6. $\square$

5. Final remarks

As shown in Section 3, the containment problem of CRPQs in ECRPQs and the equivalence problem of CRPQs and ECRPQs are both undecidable. This undecidability persists even under various restrictions: In every case, one of the two involved queries can be fixed, all queries are Boolean and acyclic, and the ECRPQs use only very restricted relations (length equality or equal lengths relations, and, in the case of query equivalence, additionally the xor relation).

In contrast to this, it remains open whether these results still hold if the alphabet contains only a single letter:

Open Problem 5.1. Let Σ be an alphabet with $|\Sigma| = 1$.
Is the query containment problem for CRPQs in ECRPQs over Σ decidable?
Is the query equivalence problem for CRPQs and ECRPQs over Σ decidable?

Likewise, the proof of the non-recursive tradeoff between ECRPQs and CRPQs given in Section 4.2 relies on the presence of at least two different letters in Σ. The relative succinctness of ECRPQs and CRPQs over a *unary* alphabet remains open.

Acknowledgments

We thank Joachim Bremer as well as an anonymous referee for helpful comments on an earlier version of this article.

References

[1] A. Aho, Algorithms for finding patterns in strings, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. A, Elsevier, 1990.
[2] J. Albert, L. Wegner, Languages with homomorphic replacements, Theoret. Comput. Sci. 16 (1981) 291–305.
[3] P. Barceló, C. Hurtado, L. Libkin, P. Wood, Expressive languages for path queries over graph-structured data, in: Proc. 29th ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS'10), 2010, pp. 3–14.
[4] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, Containment of conjunctive regular path queries with inverse, in: Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'00), 2000, pp. 176–185.
[5] C. Câmpeanu, K. Salomaa, S. Yu, A formal study of practical regular expressions, Internat. J. Found. Comput. Sci. 14 (2003) 1007–1018.
[6] A. Deutsch, V. Tannen, Optimization properties for classes of conjunctive regular path queries, in: Proc. 6th International Symposium on Database Programming Languages (DBPL'01), in: Lecture Notes in Comput. Sci., vol. 2397, Springer, 2001, pp. 21–39.
[7] D. Florescu, A.Y. Levy, D. Suciu, Query containment for conjunctive queries with regular expressions, in: Proc. 17th ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS'98), 1998, pp. 139–148.
[8] D.D. Freydenberger, Extended regular expressions: Succinctness and decidability, Theory Comput. Syst., http://dx.doi.org/10.1007/s00224-012-9389-0, in press.
[9] D.D. Freydenberger, N. Schweikardt, Expressiveness and static analysis of extended conjunctive regular path queries, in: Proc. 5th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW'11), in: CEUR Workshop Proc., vol. 749, 2011.
[10] S. Ginsburg, E. Spanier, Bounded ALGOL-like languages, Trans. Amer. Math. Soc. 113 (2) (1964) 333–368.
[11] M. Harrison, Introduction to Formal Language Theory, Addison Wesley, 1978.
[12] J. Hartmanis, On Gödel speed-up and succinctness of language representations, Theoret. Comput. Sci. 26 (3) (1983) 335–342.
[13] D. Kozen, Theory of Computation, Springer, London, 2006.
[14] M. Kutrib, The phenomenon of non-recursive trade-offs, Internat. J. Found. Comput. Sci. 16 (5) (2005) 957–973.
[15] L. Libkin, Elements of Finite Model Theory, Springer, 2004.
[16] J. Sakarovitch, Elements of Automata Theory, Cambridge University Press, 2009.
[17] K. Salomaa, Patterns, in: Formal Languages and Applications, in: Stud. Fuzziness Soft Comput., vol. 148, Springer, 2004, pp. 367–379.