Graph Logics with Rational Relations

Logical Methods in Computer Science
Vol. 9(3:01)2013, pp. 1–44
www.lmcs-online.org

Submitted Nov. 15, 2012
Published Jul. 4, 2013

GRAPH LOGICS WITH RATIONAL RELATIONS *

PABLO BARCELÓ [a], DIEGO FIGUEIRA [b], AND LEONID LIBKIN [c]

[a] Department of Computer Science, University of Chile
e-mail address: pbarcelo@dcc.uchile.cl

[b,c] Laboratory for Foundations of Computer Science, University of Edinburgh
e-mail address: {dfigueir, libkin}@inf.ed.ac.uk

Abstract. We investigate some basic questions about the interaction of regular and rational relations on words. The primary motivation comes from the study of logics for querying graph topology, which have recently found numerous applications. Such logics use conditions on paths expressed by regular languages and relations, but they often need to be extended by rational relations such as subword or subsequence. Evaluating formulae in such extended graph logics boils down to checking nonemptiness of the intersection of rational relations with regular or recognizable relations (or, more generally, to the generalized intersection problem, asking whether some projections of a regular relation have a nonempty intersection with a given rational relation).

We prove that for several basic and commonly used rational relations, the intersection problem with regular relations is either undecidable (e.g., for subword or suffix, and some generalizations), or decidable with non-primitive-recursive complexity (e.g., for subsequence and its generalizations). These results are used to rule out many classes of graph logics that freely combine regular and rational relations, as well as to provide the simplest problem related to verifying lossy channel systems that has non-primitive-recursive complexity. We then prove a dichotomy result for logics combining regular conditions on individual paths and rational relations on paths, by showing that the syntactic form of formulae classifies them into either efficiently checkable or undecidable cases. We also give examples of rational relations for which such logics are decidable even without syntactic restrictions.

2012 ACM CCS: [**Theory of computation**]: Formal languages and automata theory; Theory and algorithms for application domains—Database theory—Database query languages (principles); Design and analysis of algorithms.

Key words and phrases: Regular relations; Rational relations; Recognizable relations; intersection problem; RPQ; graph databases; non primitive recursive.

* This is the full version of the conference paper [3].

[a,b,c] Partial support provided by Fondecyt grant 1110171 for Barceló and EPSRC grants G049165 and J015377 for Figueira and Libkin.

DOI:10.2168/LMCS-9(3:01)2013

1. Introduction

The motivation for the problems investigated in this paper comes from the study of logics for querying graphs. Such logics form the basis of query languages for graph databases, that have recently found numerous applications in areas including biological networks, social networks, Semantic Web, crime detection, etc. (see [1] for a survey) and led to multiple systems and prototypes. In such applications, data is usually represented as a labeled graph. For instance, in social networks, people are nodes, and labeled edges represent different types of relationship between them; in RDF – the underlying data model of the Semantic Web – data is modeled as a graph, with RDF triples naturally representing labeled edges.

The questions that we address are related to the interaction of various classes of relations on words, for instance, rational relations (examples of those include subword and subsequence) or regular relations (such as prefix, or equality of words). An example of a question we are interested in is as follows: is it decidable whether a given regular relation contains a pair (w, w') so that w is a subword/subsequence of w'? Problems like this are very basic and deserve a study on their own, but they are also necessary to answer questions on the power and complexity of querying graph databases. We now explain how they arise in that setting.

Logical languages for querying graph data have been developed since the late 1980s (and some of them became precursors of languages later used for XML). They query the topology of the graph, often leaving querying data that might be stored in the nodes to a standard database engine. Such logics are quite different in their nature and applications from another class of graph logics based on spatial calculi [11, 18]. Their formulae combine various reachability patterns. The simplest form is known as *regular path queries (RPQs)* [17, 16]; they check the existence of a path whose label belongs to a regular language. Those are typically used as atoms and then closed under conjunction and existential quantification, resulting in the class of *conjunctive regular path queries (*CRPQ*s)*, which have been the subject of much investigation [9, 19, 22]. For instance, a CRPQ may ask for a node v such that there exist nodes v_1 and v_2 and paths from v to v_i with the label in a regular language L_i, for $i = 1, 2$.

The expressiveness of these queries, however, became insufficient in applications such as the Semantic Web or biological networks due to their inability to *compare* paths. For instance, it is a common requirement in RDF languages to compare paths based on specific semantic associations [2]; biological sequences often need to be compared for similarity, based, for example, on the edit distance.

To address this, an extension of CRPQs with relations on paths was proposed [4]. It used *regular* relations on paths, i.e., relations given by synchronized automata [21, 23]. Equivalently, these are the relations definable in automatic structures on words [5, 7, 8]. They include prefix, equality, equal length of words, or fixed edit distance between words. The extension of CRPQs with them, called ECRPQs, was shown to have acceptable complexity (NLogSpace with respect to data, PSpace with respect to query).

However, the expressive power of ECRPQs is still short of the expressiveness needed in many applications. For instance, semantic associations between paths used in RDF applications often deal with subwords or subsequences, but these relations are *not* regular. They are *rational*: they are still accepted by automata, but those whose heads move asynchronously.

Adding them to a query language must be done with extreme care: simply replacing regular relations with rational in the definition of ECRPQs makes query evaluation undecidable!

So we set out to investigate the following problem: given a class of graph queries, e.g., CRPQs or ECRPQs, what happens if one adds the ability to test whether pairs of paths belong to a rational relation S, such as subword or subsequence? We start by observing that this problem is a generalization of the *intersection problem*: given a regular relation R, and a rational relation S, is $R \cap S \neq \emptyset$? It is well known that there exist rational relations S for which it is undecidable [6]; however, we are not interested in artificial relations obtained by encoding PCP instances, but rather in very concrete relations used in querying graph data.

The intersection problem captures the essence of graph logics ECRPQs and CRPQs (for the latter, when restricted to the class of recognizable relations [6, 15]). In fact, query evaluation can be cast as the *generalized intersection problem*. Its input includes an m-ary regular relation R, a binary rational relation S, and a set I of pairs from $\{1, \ldots, m\}$. It asks whether there is a tuple $(w_1, \ldots, w_m) \in R$ so that $(w_i, w_j) \in S$ whenever $(i, j) \in I$. For $m = 2$ and $I = \{(1, 2)\}$, this is the usual intersection problem.

Another motivation for looking at these basic problems comes from verification of lossy channel systems (finite-state processes that communicate over unbounded, but lossy, FIFO channels). Their reachability problem is known to be decidable, although the complexity is not bounded by any multiply-recursive function [14]. In fact, a "canonical" problem used in reductions showing this enormous complexity [13, 14] can be restated as follows: given a binary rational relation R, does it have a pair (w, w') so that w is a subsequence of w'? This naturally leads to the question whether the same bounds hold for the simpler instance of the intersection problem when we use regular relations instead of rational ones. We actually show that this is true.

Summary of results. We start by showing that evaluating CRPQs and ECRPQs extended with a rational relation S can be cast as the generalized intersection problem for S with recognizable and regular relations respectively. Moreover, the complexity of the basic intersection problem is a lower bound for the complexity of query evaluation.

We then study the complexity of the intersection problem for fixed relations S. For recognizable relations, it is well known to be efficiently decidable for every rational S. For regular relations, we show that if S is the subword, or the suffix relation, then the problem is undecidable. That is, it is undecidable to check, given a binary regular relation R, whether it contains a pair (w, w') so that w is a subword of w', or even a suffix of w'. We also present a generalization of this result.

The analogous problem for the subsequence relation is known to be decidable, and, if the input is a rational relation R, then the complexity is non-multiply-recursive [13]. We extend this in two ways. First, we show that the lower bound remains true even for regular relations R. Second, we extend decidability to the class of all rational relations for which one projection is closed under subsequence (the subsequence relation itself is trivially such, obtained by closing the first projection of the equality relation).

In addition to establishing some basic facts about classes of relations on words, these results tell us about the infeasibility of adding rational relations to ECRPQs: in fact adding subword makes query evaluation undecidable, and while it remains decidable with subsequence, the complexity is prohibitively high.

So we then turn to the generalized intersection problem with recognizable relations, corresponding to the evaluation of CRPQs with an extra relation S. We show that the shape of the relation I holds the key to decidability. If its underlying undirected graph is acyclic, then the problem is decidable in PSPACE for every rational relation S (and for a fixed formula the complexity drops to NLOGSPACE). In the cyclic case, the problem is undecidable for some rational relation S. For relations generalizing subsequence, we have decidability when I is a DAG, and for subsequence itself, as well as for suffix, query evaluation is decidable regardless of the shape of CRPQs.

Thus, under the mild syntactic restriction of acyclicity of comparisons with respect to rational relations, such relations can be added to the common class CRPQ of graph queries, without incurring a high complexity cost.

Organization. We give basic definitions in Section 2 and define the main problems we study in Section 3. Section 4 introduces graph logics and establishes their connection with the (generalized) intersection problem. Section 5 studies decidable and undecidable cases of the intersection problem. Section 6 looks at the case of recognizable relations and CRPQs and establishes decidability results based on the intersection pattern.

2. Preliminaries

Let $\mathbb{N} = \{1, 2, \dots\}$, $[i..j] = \{i, i+1, \dots, j\}$ (if $i > j$, $[i..j] = \emptyset$), $[i] = [1..i]$. Given, $A, B \subseteq \mathbb{N}$, an *increasing* function $f : A \to B$ is one such that $f(i) \geq f(j)$ whenever $i > j$. If $f(i) > f(j)$ we call it *strictly* increasing.

Alphabets, languages, and morphisms. We shall use letters Σ, Γ to denote finite alphabets. The set of all finite words over an alphabet Σ is denoted by Σ^*. We write ε for the empty word, $w \cdot w'$ for the concatenation of two words, and $|w|$ for the length of a word w. Given a word $w \in \Sigma^*$, $w[i..j]$ stands for the substring in positions $[i..j]$, $w[i]$ for $w[i..i]$, and $w[i..]$ for $w[i..|w|]$. Positions in the word start with 1.

If $w = w' \cdot u \cdot w''$, then

- u is a *subword* of w (also called *factor* in the literature, written as $u \preceq w$),
- w' is a *prefix* of w (written as $w' \preceq_{\text{pref}} w$), and
- w'' is a *suffix* of w (written as $w'' \preceq_{\text{suff}} w$).

We say that w' is a *subsequence* of w (also called *subword embedding* or *scattered subword* in the literature, written as $w' \sqsubseteq w$) if w' is obtained by removing some letters (perhaps none) from w, i.e., $w = a_1 \dots a_n$, and $w' = a_{i_1} a_{i_2} \dots a_{i_k}$, where $1 \leq i_1 < i_2 < \dots < i_k \leq n$.

If $\Sigma \subset \Gamma$ and $w \in \Gamma^*$, then by w_Σ we denote the projection of w on Σ. That is, if $w = a_1 \dots a_n$ and $a_{i_1}, \dots, a_{i_k}$ are precisely the letters from Σ, with $i_1 < \dots < i_k$, then $w_\Sigma = a_{i_1} \dots a_{i_k}$.

Recall that a *monoid* $M = \langle U, \cdot, 1 \rangle$ has an associative binary operation $\cdot$ and a neutral element 1 satisfying $1x = x1 = x$ for all x (we often write xy for $x \cdot y$). The set Σ^* with the operation of concatenation and the neutral element ε forms a monoid $\langle \Sigma^*, \cdot, \varepsilon \rangle$, the free monoid generated by Σ. A function $f : M \to M'$ between two monoids is a *morphism* if it sends the neutral element of M to the neutral element of M', and if $f(xy) = f(x)f(y)$ for all $x, y \in M$. Every morphism $f : \langle \Sigma^*, \cdot, \varepsilon \rangle \to M$ is uniquely determined by the values $f(a)$,

for $a \in \Sigma$, as $f(a_1 \dots a_n) = f(a_1) \cdots f(a_n)$. A morphism $f : \langle \Sigma^*, \cdot, \varepsilon \rangle \to \langle \Gamma^*, \cdot, \varepsilon \rangle$ is called *alphabetic* if $f(a) \in \Gamma \cup \{\varepsilon\}$, and *strictly alphabetic* if $f(a) \in \Gamma$ for each $a \in \Sigma$, see [6].

A language L is a subset of Σ^*, for some finite alphabet Σ. It is *recognizable* if there is a finite monoid M, a morphism $f : \langle \Sigma^*, \cdot, \varepsilon \rangle \to M$, and a subset M_0 of M such that $L = f^{-1}(M_0)$.

A language L is *regular* if there exists an NFA (non-deterministic finite automaton) $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F \rangle$ such that $L = \mathcal{L}(\mathcal{A})$, the language of words accepted by $\mathcal{A}$. We use the standard notation for NFAs, where Q is the set of states, q_0 is the initial state, F is the set of final states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

A language is *rational* if it is denoted by a regular expression; such expressions are built from $\emptyset$, ε, and alphabet letters by using operations of concatenation ($e \cdot e'$), union ($e \cup e'$), and Kleene star (e^*). It is of course a classical result of formal language theory that the classes of recognizable, regular, and rational languages coincide.

Recognizable, regular, and rational relations. While the notions of recognizability, regularity, and rationality coincide over languages $L \subseteq \Sigma^*$, they differ over relations over Σ, i.e., subsets of $\Sigma^* \times \ldots \times \Sigma^*$. We now define those (see [6, 12, 15, 21, 23, 34]).

Since $\langle \Sigma^*, \cdot, \varepsilon \rangle$ is a monoid, the product $(\Sigma^*)^n$ has the structure of a monoid too. We can thus define *recognizable n-ary relations* over Σ as subsets $R \subseteq (\Sigma^*)^n$ so that there exists a finite monoid M and a morphism $f : (\Sigma^*)^n \to M$ such that $R = f^{-1}(M_0)$ for some $M_0 \subseteq M$. The class of n-ary recognizable relations will be denoted by REC_n; when n is clear or irrelevant, we write just REC.

It is well-known that a relation $R \subseteq (\Sigma^*)^n$ is in REC_n iff it is a finite union of the sets of the form $L_1 \times \ldots \times L_n$, where each L_i is a regular language over Σ, see [6, 21].

Next, we define the class of regular relations. Let $\perp \notin \Sigma$ be a new alphabet letter, and let $\Sigma_\perp$ be $\Sigma \cup \{\perp\}$. Each tuple $\bar{w} = (w_1, \dots, w_n)$ of words from Σ^* can be viewed as a word over $\Sigma_\perp^n$ as follows: pad words w_i with $\perp$ so that they all are of the same length, and use as the kth symbol of the new word the n-tuple of the kth symbols of the padded words. Formally, let $\ell = \max_i |w_i|$. Then $w_1 \otimes \ldots \otimes w_n$ is a word of length ℓ whose kth symbol is $(a_1, \dots, a_n) \in \Sigma_\perp^n$ such that

$$a_i = \begin{cases} \text{the } k\text{th letter of } w_i & \text{if } |w_i| \geq k \\ \perp & \text{otherwise.} \end{cases}$$

We shall also write $\otimes \bar{w}$ for $w_1 \otimes \ldots \otimes w_n$. We define $\pi_i(u_1 \otimes \cdots \otimes u_k) = u_i$ for all $i \in [k]$. A relation $R \subseteq (\Sigma^*)^n$ is called a *regular n-ary relation* over Σ if there is a finite automaton $\mathcal{A}$ over $\Sigma_\perp^n$ that accepts $\{\otimes \bar{w} \mid \bar{w} \in R\}$. The class of n-ary regular relations is denoted by REG_n; as before, we write REG when n is clear or irrelevant.

Finally, we define rational relations. There are two equivalent ways of doing it. One uses regular expressions, which are now built from tuples $\bar{a} \in (\Sigma \cup \{\varepsilon\})^n$ using the same operations of union, concatenation, and Kleene star. Binary relations $\preceq_{\text{suff}}$, $\preceq$, and $\sqsubseteq$ are all rational: the expression $\big(\bigcup_{a \in \Sigma}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a \in \Sigma}(a, a)\big)^*$ defines $\preceq_{\text{suff}}$, the expression $\big(\bigcup_{a \in \Sigma}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a \in \Sigma}(a, a)\big)^* \cdot \big(\bigcup_{a \in \Sigma}(\varepsilon, a)\big)^*$ defines $\preceq$, and the expression $\big(\bigcup_{a \in \Sigma}(\varepsilon, a) \cup (a, a)\big)^*$ defines $\sqsubseteq$.

Alternatively, n-ary rational relations can be defined by means of n-tape automata, that have n heads for the tapes and one additional control; at every step, based on the state and the letters it is reading, the automaton can enter a new state and move some (but

not necessarily all) tape heads. The classes of n-ary relations so defined are called *rational n-ary relations*; we use the notation RAT_n or just RAT, as before.

Relationships between classes of relations. While it is well known that $\mathsf{REC}_1 = \mathsf{REG}_1 = \mathsf{RAT}_1$, we have strict inclusions

$$\mathsf{REC}_k \subsetneq \mathsf{REG}_k \subsetneq \mathsf{RAT}_k$$

for every $k > 1$ (see for example [6]). For instance, $\preceq_{\text{pref}} \in \mathsf{REG}_2 - \mathsf{REC}_2$ and $\preceq_{\text{suff}} \in \mathsf{RAT}_2 - \mathsf{REG}_2$.

The classes of recognizable and regular relations are closed under intersection; however the class of rational relations is not. In fact, one can find $R \in \mathsf{REG}_2$ and $S \in \mathsf{RAT}_2$ so that $R \cap S \notin \mathsf{RAT}_2$. However, if $R \in \mathsf{REC}_m$ and $S \in \mathsf{RAT}_m$, then $R \cap S \in \mathsf{RAT}_m$.

Binary rational relations can be characterized as follows [6, 30]. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is rational iff there is a finite alphabet Γ, a regular language $L \subseteq \Gamma^*$ and two alphabetic morphisms $f, g : \Gamma^* \to \Sigma^*$ such that $R = \{(f(w), g(w)) \mid w \in L\}$. If we require f and g to be strictly alphabetic morphisms, we get the class of *length-preserving* regular relations, i.e., $R \in \mathsf{REG}_2$ so that $(w, w') \in R$ implies $|w| = |w'|$. Regular binary relations are then finite unions of relations of the form $\{(w \cdot u, w') \mid (w, w') \in R,\ u \in L\}$ and $\{(w, w' \cdot u) \mid (w, w') \in R,\ u \in L\}$, where R ranges over length-preserving regular relations, and L over regular languages.

Properties of classes of relations. Since relations in REC and REG are given by NFAs, they inherit all the closure/decidability properties of regular languages. If $R \in \mathsf{RAT}$, then each of its projections is a regular language, and can be effectively constructed (e.g., from the description of R as an n-tape automaton). Hence, the nonemptiness problem is decidable for rational relations. However, testing nonemptiness of the intersection of two rational relations is undecidable [6]. Also, for $R, R' \in \mathsf{RAT}$, the following are undecidable: checking whether $R \subseteq R'$ or $R = R'$, universality ($R = \Sigma^* \times \Sigma^*$), and checking whether $R \in \mathsf{REG}$ or $R \in \mathsf{REC}$ [6, 12, 28].

Remark. We defined recognizable, regular, and rational relations over the same alphabet, i.e., as subsets of $(\Sigma^*)^n$. Of course it is possible to define them as subsets of $\Sigma_1 \times \ldots \times \Sigma_n$, with the Σ_i's not necessarily distinct. Technically, there are no differences and all the results will continue to hold. Indeed, one can simply consider a new alphabet Σ as the disjoint union of Σ_i's, and enforce the condition that the ith projection only use the letters from Σ_i (this is possible for all the classes of relations we consider). In fact, in the proofs we shall be using both types of relations.

Well-quasi-orders. A well-quasi-order $\leq \subseteq A \times A$ is a reflexive and transitive relation such that for every infinite sequence $(a_i)_{i\in\mathbb{N}}$ over A there are $i < j$ with $a_i \leq a_j$. We will make use of the following two lemmas.

Lemma 2.1 (Higman's Lemma [25])**.** *For every alphabet Σ, the subsequence relation $\sqsubseteq \subseteq \Sigma^* \times \Sigma^*$ is a well quasi-order.*

Lemma 2.2 (Dickson's Lemma [20])**.** *For every well-quasi-order $\leq \subseteq A \times A$, the product order $\leq^k \subseteq A^k \times A^k$ (where $(a_1, \ldots, a_k) \leq^k (a'_1, \ldots, a'_k)$ iff $a_i \leq a'_i$ for all $i \in [k]$) is a well-quasi-order.*

3. Generalized intersection problem

We now formalize the main technical problem we study. Let $\mathcal{R}$ be a class of relations over Σ, and $\mathcal{S}$ a class of binary relations over Σ. We use the notation $[m]$ for $\{1, \ldots, m\}$. If R is an m-ary relation, S is a binary relation, and $I \subseteq [m]^2$, we write $R \cap_I S$ for the set of tuples $(w_1, \ldots, w_m)$ in R such that $(w_i, w_j) \in S$ whenever $(i, j) \in I$.

The *generalized intersection problem* $(\mathcal{R} \cap_I \mathcal{S}) \stackrel{?}{=} \emptyset$ is defined as:

Problem:	$(\mathcal{R} \cap_I \mathcal{S}) \stackrel{?}{=} \emptyset$
Input:	an m-ary relation $R \in \mathcal{R}$, a relation $S \in \mathcal{S}$, and $I \subseteq [m]^2$
Question:	is $R \cap_I S \neq \emptyset$?

If $\mathcal{S} = \{S\}$, we write S instead of $\{S\}$. We write $\textsc{GenInt}_S(\mathcal{R})$ for the class of all problems $(\mathcal{R} \cap_I S) \stackrel{?}{=} \emptyset$ where S is fixed, i.e., the input consists of $R \in \mathcal{R}$ and I. As was explained in the introduction, this problem captures the essence of evaluating queries in various graph logics, e.g., CRPQs or ECRPQs extended with rational relations S. The classes $\mathcal{R}$ will typically be REC and REG.

If $m = 2$ and $I = \{(1, 2)\}$, the generalized intersection problem becomes simply the *intersection problem* for the classes $\mathcal{R}$ and $\mathcal{S}$ of binary relations:

Problem:	$(\mathcal{R} \cap \mathcal{S}) \stackrel{?}{=} \emptyset$
Input:	$R \in \mathcal{R}$ and $S \in \mathcal{S}$
Question:	is $R \cap S \neq \emptyset$?

The problem $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ is decidable for every rational relation S, simply by constructing $R \cap S$, which is a rational relation, and testing its nonemptiness. However, $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ could already be undecidable (we shall give one particularly simple example later).

4. Graph logics and the generalized intersection problem

In this section we show how the (generalized) intersection problems provide us with upper and lower bounds on the complexity of evaluating a variety of logical queries over graphs. We start by recalling the basic classes of logics used in querying graph data, and show that extending them with rational relations allows us to cast the query evaluation problem as an instance of the generalized intersection problem. The key observations are that:

- the complexity of $\textsc{GenInt}_S(\mathsf{REC})$ and $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ provide an upper and a lower bound for the complexity of evaluating $\mathrm{CRPQ}(S)$ queries; and
- for $\mathrm{ECRPQ}(S)$, these bounds are provided by the complexity of $\textsc{GenInt}_S(\mathsf{REG})$ and of $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$.

The standard abstraction of graph databases [1] is finite Σ-labeled graphs $G = \langle V, E\rangle$, where V is a finite set of nodes, or vertices, and $E \subseteq V \times \Sigma \times V$ is a set of labeled edges. A *path* ρ from v_0 to v_m in G is a sequence of edges $(v_0, a_0, v_1), (v_1, a_1, v_2), \cdots, (v_{m-1}, a_{m-1}, v_m)$ from E, for some $m \geq 0$. The *label* of ρ, denoted by $\lambda(\rho)$, is the word $a_0 \cdots a_{m-1} \in \Sigma^*$.

The main building blocks for graph queries are *regular path queries*, or *RPQ*s [17]; they are expressions of the form $x \xrightarrow{L} y$, where L is a regular language. We normally assume that L is represented by a regular expression or an NFA. Given a Σ-labeled graph $G = \langle V, E\rangle$, the answer to an RPQ above is the set of pairs of nodes (v, v') such that there is a path ρ from v to v' with $\lambda(\rho) \in L$.

Conjunctive RPQs, or CRPQ*s* [9, 10, 16] are the closure of RPQs under conjunction and existential quantification. Formally, they are expressions of the form

$$\varphi(\bar{x}) \quad = \quad \exists \bar{y} \bigwedge_{i=1}^{m} (u_i \xrightarrow{L_i} u'_i) \tag{4.1}$$

where variables u_i, u'_is come from $\bar{x}, \bar{y}$. The semantics naturally extends the semantics of RPQs: $\varphi(\bar{a})$ is true in G iff there is a tuple $\bar{b}$ of nodes such that for every $i \leq m$ and every v_i, v'_i interpreting u_i and u'_i, respectively, we have a path ρ_i between v_i and v'_i whose label $\lambda(\rho_i)$ is in L_i.

CRPQs can further be extended to *compare* paths. For that, we need to name path variables, and choose a class of allowed relations on paths. The simplest such extension is the class of $\mathrm{CRPQ}(S)$ queries, where S is a binary relation over Σ^*. Its formulae are of the form

$$\varphi(\bar{x}) \quad = \quad \exists \bar{y} \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i : L_i} u'_i) \quad \wedge \quad \bigwedge_{(i,j) \in I} S(\chi_i, \chi_j) \Big) \tag{4.2}$$

where $I \subseteq [m]^2$. We use variables $\chi_1, \ldots, \chi_m$ to denote paths; these are quantified existentially. That is, the semantics of $G \models \varphi(\bar{a})$ is that there is a tuple $\bar{b}$ of nodes and paths ρ_k, for $k \leq m$, between v_k and v'_k (where, as before, v_k, v'_k are elements of $\bar{a}, \bar{b}$ interpreting u_k, u'_k) such that $(\lambda(\rho_i), \lambda(\rho_j)) \in S$ whenever $(i, j) \in I$. For instance, the query

$$\exists y, y' \ \big((x \xrightarrow{\chi : \Sigma^* a} y) \wedge (x \xrightarrow{\chi' : \Sigma^* b} y') \wedge \chi \sqsubseteq \chi' \big)$$

finds nodes v so that there are two paths starting from v, one ending with an a-edge, whose label is a subsequence of the other one, that ends with a b-edge.

The input to the *query evaluation problem* consists of a graph G, a tuple $\bar{v}$ of nodes, and a query $\varphi(\bar{x})$; the question is whether $G \models \varphi(\bar{v})$. This corresponds to the *combined complexity* of query evaluation. In the context of query evaluation, one is often interested in *data complexity*, when the typically small formula φ is fixed, and the input consists of the typically large graph $(G, \bar{v})$. We now relate it to the complexity of $\textsc{GenInt}_S(\mathsf{REC})$.

Lemma 4.1. *Fix a* $\mathrm{CRPQ}(S)$ *query* φ *as in (4.2). Then there is a* DLogSpace *algorithm that, given a graph* G *and a tuple* $\bar{v}$ *of nodes, constructs an* m*-ary relation* $R \in \mathsf{REC}$ *so that the answer to the generalized intersection problem* $(R \cap_I S) \stackrel{?}{=} \emptyset$ *is 'yes' iff* $G \models \varphi(\bar{v})$.

Proof. Given a Σ-labeled graph $G = \langle V, E\rangle$ and two nodes v, v', we write $\mathcal{A}(G, v, v')$ for G viewed as an NFA with the initial state v and the final state v' (that is, the set of states is V, the transition relation is E, and the alphabet is Σ). The language of such an automaton, $\mathcal{L}(\mathcal{A}(G, v, v'))$, is the set of labels of all paths between v and v'.

Now consider a CRPQ(S) query $\varphi(\bar{x})$ given by

$$\exists \bar{y} \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i : L_i} u'_i) \quad \wedge \quad \bigwedge_{(i,j)\in I} S(\chi_i, \chi_j) \Big),$$

as in (4.2). Suppose we are given a graph G as above and a tuple of nodes $\bar{v}$, of the same length as the length of $\bar{x}$. The DLogSpace algorithm works as follows.

First we enumerate all tuples $\bar{b}$ of nodes of G of the same length as $\bar{y}$; since φ is fixed, this can be done in DLogSpace. For each $\bar{b}$, we construct an m-ary relation $R_{\bar{b}}$ in REC as follows. Let n_i and n'_i be the interpretations of u_i and u'_i, when $\bar{x}$ is interpreted as $\bar{v}$ and $\bar{y}$ as $\bar{b}$. Then

$$R_{\bar{b}} \;=\; \prod_{i=1}^{m} (\mathcal{L}(\mathcal{A}(G, n_i, n'_i)) \cap L_i).$$

Note that it can be constructed in DLogSpace; indeed each coordinate of $R_{\bar{b}}$ is simply a product of the automaton $\mathcal{A}(G, n_i, n'_i)$ and a fixed automaton defining L_i. Next, let $R = \bigcup_{\bar{b}} R_{\bar{b}}$. This is constructed in DLogSpace too. Now it follows immediately from the construction that $R \cap_I S \neq \emptyset$ iff for some $\bar{b}$, there exist paths ρ_i between n_i, n'_i, for $i \leq m$, such that $(\lambda(\rho_l), \lambda(\rho_j)) \in S$ whenever $(l, j) \in I$, i.e., iff $G \models \varphi(\bar{v})$. □

Conversely, the intersection problem for recognizable relations and S can be encoded as answering CRPQ(S) queries.

Lemma 4.2. *For any given binary relation S, there is a* CRPQ*(S) query $\varphi(x, x')$ and a* DLogSpace *algorithm that, given a relation $R \in \mathsf{REC}_2$, constructs a graph G and two nodes v, v' so that $G \models \varphi(v, v')$ iff $R \cap S \neq \emptyset$.*

Proof. Let R be in REC_2. It is given as $\bigcup_{i=1}^{n} (L_i \times K_i)$, where the L_is and the K_is are regular languages over Σ. These languages are given by their NFAs which we can view as Σ-labeled graphs. Let $\langle V_i, E_i\rangle$ be the underlying graph of the NFA defining L_i, such that v_0^i is the initial state, and F_i is the set of final states. Likewise, let $\langle W_i, H_i\rangle$ be the underlying graph of the NFA defining K_i, such that w_0^i is the initial state, and C_i is the set of final states.

We now construct the graph G. Its labeling alphabet is the union of Σ and $\{\#, \$, !\}$. Its set of vertices is the disjoint union of all the V_is, W_is, as well as two distinguished nodes *start* and *end*. Its edges include all the edges from E_is and H_is, and the following:

- #-labeled edges from *start* to each initial state, i.e., to each v_i^0 and w_i^0 for all $i \leq n$.
- \$-labeled edges between the initial states of automata with the same index, i.e., edges $(v_0^i, \$, w_0^i)$ for all $i \leq n$.
- !-labeled edges from final states to *end*, i.e., edges $(v, !, \mathit{end})$, where $v \in \bigcup_{i\leq n} F_i \cup \bigcup_{i\leq n} C_i$.

We now define a CRPQ(S) query $\varphi(x,y)$ (omitting path variables for paths that are not used in comparisons):

$$\exists x_1,x_2,z_1,z_2 \left(\begin{array}{lllll} & x \xrightarrow{\#} x_1 & \wedge & x \xrightarrow{\#} x_2 & \\ \wedge & x_1 \xrightarrow{\$} x_2 & & & \\ \wedge & x_1 \xrightarrow{\chi:\Sigma^*} z_1 & \wedge & x_2 \xrightarrow{\chi':\Sigma^*} z_2 & \\ \wedge & z_1 \xrightarrow{!} y & \wedge & z_2 \xrightarrow{!} y & \\ \wedge & S(\chi,\chi') & & & \end{array}\right)$$

The query says that from *start*, we have #-edges to the initial states v_0^i and w_0^i: they must have the same index since there is a \$-edge between them. From there we have two paths, ρ and ρ', corresponding to the variables χ and χ', which are Σ-labeled, and thus are paths in the automata for L_i and K_i, respectively. From the end nodes of those paths we have !-edges to *end*, so they must be final states; in particular, $\lambda(\rho) \in L_i$ and $\lambda(\rho') \in K_i$. We finally require $(\lambda(\rho),\lambda(\rho')) \in S$, i.e., $(\lambda(\rho),\lambda(\rho')) \in (L_i \times K_i) \cap S$. Hence, if $G \models \varphi(\mathit{start},\mathit{end})$ then for some $i \leq n$ we have two words (w,w') that belong to $(L_i \times K_i) \cap S$, i.e., $R \cap S \neq \emptyset$. Conversely, if $R \cap S \neq \emptyset$, then $(L_i \times K_i) \cap S \neq \emptyset$ for some $i \leq n$, and the witnessing paths of the nonemptiness of $(L_i \times K_i) \cap S$ will witness the formula $\varphi(\mathit{start},\mathit{end})$ (together with initial states of the automata of L_i and K_i and some of their final states). □

Combining the lemmas, we obtain:

Theorem 4.3. *Let $\mathcal{K}$ be a complexity class closed under* DLogSpace *reductions. Then:*

(1) *If the problem* $\textsc{GenInt}_S(\mathsf{REC})$ *is in $\mathcal{K}$, then data complexity of* CRPQ*(S) queries is in $\mathcal{K}$; and*
(2) *If the problem* $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ *is hard for $\mathcal{K}$, then so is data complexity of* CRPQ*(S) queries.*

We now consider *extended CRPQs*, or *ECRPQs*, which enhance CRPQs with regular relations [4], and prove a similar result for them, with the role of REC now played by REG. Formally, ECRPQs are expressions of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y} \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i:L_i} u_i') \;\wedge\; \bigwedge_{j=1}^{k} R_j(\bar{\chi}_j) \Big) \tag{4.3}$$

where each R_j is a relation from REG, and $\bar{\chi}_j$ a tuple from $\chi_1,\ldots,\chi_m$ of the same arity as R_j. The semantics of course extends the semantics of CRPQs: the witnessing paths $\rho_1,\ldots,\rho_m$ should also satisfy the condition that for every atom $R(\rho_{i_1},\ldots,\rho_{i_l})$ in (4.3), the tuple $(\lambda(\rho_{i_1}),\ldots,\lambda(\rho_{i_l}))$ is in R.

Finally, we obtain ECRPQ(S) queries by adding comparisons with respect to a relation $S \in \mathsf{RAT}$, getting a class of queries $\varphi(\bar{x})$ of the form

$$\exists \bar{y} \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i:L_i} u_i') \wedge \bigwedge_{j=1}^{k} R_j(\bar{\chi}_j) \wedge \bigwedge_{(i,j)\in I} S(\chi_i,\chi_j) \Big) \tag{4.4}$$

Similarly to the case of CRPQs, we can establish a connection between data complexity of ECRPQ(S) queries and the complexity of the generalized intersection problem:

Theorem 4.4. *Let $\mathcal{K}$ be a complexity class closed under* DLogSpace *reductions. Then:*

(1) *If the problem* $\textsc{GenInt}_S(\mathsf{REG})$ *is in* $\mathcal{K}$*, then data complexity of* ECRPQ*(S) queries is in* $\mathcal{K}$*; and*
(2) *If the problem* $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ *is hard for* $\mathcal{K}$*, then so is data complexity of* ECRPQ*(S) queries.*

Similarly to the proof of Theorem 4.3, the result will be an immediate consequence of two lemmas. First, evaluation of ECRPQ(S) queries is reducible to the generalized intersection problem for regular relations.

Lemma 4.5. *Fix an* ECRPQ*(S) query* φ *as in (4.4). Then there is a* DLogSpace *algorithm that, given a graph* G *and a tuple* $\bar{v}$ *of nodes, constructs an* m*-ary relation* $R \in \mathsf{REG}$ *so that the answer to the generalized intersection problem* $(R \cap_I S) \stackrel{?}{=} \emptyset$ *is 'yes' iff* $G \models \varphi(\bar{v})$*.*

Conversely, the intersection problem for regular relations and S can be encoded as answering ECRPQ(S) queries.

Lemma 4.6. *For each binary relation* S*, there is an* ECRPQ*(S) query* $\varphi(x, x')$ *and a* DLogSpace *algorithm that, given a relation* $R \in \mathsf{REG}_2$*, constructs a graph* G *and two nodes* v, v' *so that* $G \models \varphi(v, v')$ *iff* $(R \cap S) \neq \emptyset$*.*

The proof of Lemma 4.5 is almost the same as the proof of Lemma 4.1: as before, we enumerate tuples $\bar{b}$, construct relations $R_{\bar{b}}$ and $R = \bigcup_{\bar{b}} R_{\bar{b}}$, but this time we take the product of this recognizable relation with regular relations mentioned in the query. Since the query is fixed, and hence we take a product with a fixed number of fixed automata, such a product construction can be done in DLogSpace. The result is now a regular m-ary relation. The rest of the proof is exactly the same as in Lemma 4.1.

We now prove Lemma 4.6. Let $R \in \mathsf{REG}_2$ be given by an NFA over $\Sigma_\perp \times \Sigma_\perp$ whose underlying graph is $G_R = \langle V_R, E_R \rangle$, where $E_R \subseteq V_R \times (\Sigma_\perp \times \Sigma_\perp) \times V_R$. Let v_0 be its initial state, and let F be the set of final states.

We now define the graph G. Its labeling alphabet Γ is the disjoint union of $\Sigma_\perp \times \Sigma_\perp$, the alphabet Σ itself, and a new symbol $\#$. Its nodes V include all nodes in V_R and two extra nodes, v_f and v'. The edges are:

- all the edges in E_R;
- edges $(v, \#, v_f)$ for every $v \in F$;
- edges (v', a, v') for every $a \in \Sigma$.

We now define two regular relations over Γ. The first, R_1, consists of pairs (w, w'), where $w \in (\Sigma_\perp \times \Sigma_\perp)^*$ and $w' \in \Sigma^*$. Furthermore, w is of the form $w' \otimes w''$ for some $w'' \in \Sigma^*$. It is straightforward to check that this relation is regular. The second one, R_2, is the same except w is of the form $w'' \otimes w'$. In other words, the first component is $w_1 \otimes w_2$, and the second is either w_1 or w_2, for R_1 or R_2, respectively.

Next, we define the ECRPQ(S) $\varphi(x, y)$:

$$\exists x_1, y_1, x_2, y_2, z \left(\begin{array}{ccccccc} & x & \stackrel{\chi:\Sigma_\perp \times \Sigma_\perp}{\rightarrow} & z & \wedge & z & \stackrel{\#}{\rightarrow} y \\ \wedge & x_1 & \stackrel{\chi_1:\Sigma^*}{\rightarrow} & y_1 & \wedge & x_2 & \stackrel{\chi_2:\Sigma^*}{\rightarrow} y_2 \\ \wedge & R_1(\chi, \chi_1) & & & \wedge & R_2(\chi, \chi_2) & \wedge \quad S(\chi_1, \chi_2) \end{array} \right)$$

Note that when this formula is evaluated over G, with x interpreted as v_0 and y interpreted as v_f, the paths χ_1 and χ_2 can have arbitrary labels from Σ^*. Paths χ can have arbitrary labels over $\Sigma_\perp \times \Sigma_\perp$; however, since they start in v_0 and must be followed by an $\#$-edge, they end in a final state of the automaton for R, and hence labels of these paths are precisely

words in $\Sigma_\perp \times \Sigma_\perp$ of the form $w_1 \otimes w_2$, where $(w_1, w_2) \in R$. Now R_1 ensures that the label of χ_1 is w_1 and that the label of χ_2 is w_2. Hence the labels of χ_1 and χ_2 are precisely the pairs of words in R, and the query asks whether such a pair belongs to S. Hence, $G \models \varphi(v_0, v_f)$ iff $R \cap S \neq \emptyset$. It is straightforward to check that the construction of G can be carried out in DLogSpace. This proves the lemma and the theorem.

Thus, our next goal is to understand the behaviors of the generalized intersection problem for various rational relations S which are of interest in graph logics; those include subword, suffix, subsequence. In fact to rule out many undecidable or infeasible cases it is often sufficient to analyze the intersection problem. We do this in the next section, and then analyze the decidable cases to come up with graph logics that can be extended with rational relations.

5. The intersection problem: decidable and undecidable cases

We now study the problem $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ for binary rational relations S such as subword and subsequence, and for classes of relations generalizing them. The input is a binary regular relation R over Σ, given by an NFA over $\Sigma_\perp \times \Sigma_\perp$. The question is whether $R \cap S \neq \emptyset$. We also derive results about the complexity of $\mathrm{ECRPQ}(S)$ queries. For all lower-bound results in this section, we assume that the alphabet contains at least two symbols.

As already mentioned, there exist rational relations S such that $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ is undecidable. However, we are interested in relations that are useful in graph querying, and that are among the most commonly used rational relations, and for them the status of the problem was unknown.

Note that the problem $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ is tractable: given $R \in \mathsf{REC}$, the relation $R \cap S$ is rational, can be efficiently constructed, and checked for nonemptiness.

5.1. Undecidable cases: subword and relatives.

We now show that even for such simple relations as subword and suffix, the intersection problem is undecidable. That is, given an NFA over $\Sigma_\perp \times \Sigma_\perp$ defining a regular relation R, the problem of checking for the existence of a pair $(w, w') \in R$ with $w \preceq_{\text{suff}} w'$ or $w \preceq w'$ is undecidable.

Theorem 5.1. *The problems* $(\mathsf{REG} \cap \preceq_{\text{suff}}) \stackrel{?}{=} \emptyset$ *and* $(\mathsf{REG} \cap \preceq) \stackrel{?}{=} \emptyset$ *are undecidable.*

As an immediate consequence of this, we obtain:

Corollary 5.2. *The query evaluation problem for* $\mathrm{ECRPQ}(\preceq_{\text{suff}})$ *and* $\mathrm{ECRPQ}(\preceq)$ *is undecidable.*

Thus, some of the most commonly used rational relations cannot be added to ECRPQs without imposing further restrictions.

We skip the proof of Theorem 5.1 for the time being and concentrate first on how to obtain a more general undecidability result out of it. As we will see below, the essence of the undecidability result is that relations such as $\preceq_{\text{suff}}$ and $\preceq$ can be decomposed in a way that one of the components of the decomposition is a graph of a nontrivial strictly alphabetic morphism. More precisely, let $R \cdot R'$ be the binary relation $\{(w \cdot w', u \cdot u') \mid (w, u) \in R \text{ and } (w', u') \in R'\}$. Let $\mathrm{Graph}(f)$ be the graph of a function $f : \Sigma^* \to \Sigma^*$, i.e., $\{(w, f(w)) \mid w \in \Sigma^*\}$.

Proposition 5.3. *Let R_0, R_1 be binary relations on Σ such that R_0 is recognizable and its second projection is Σ^*. Let f be a strictly alphabetic morphism that is not constant (i.e. the image of f contains at least two letters). Then, for $S = R_0 \cdot \mathrm{Graph}(f) \cdot R_1$, the problem $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ is undecidable.*

Note that both $\preceq_{\mathrm{suff}}$ and $\preceq$ are of the required shape: suffix is $(\{\varepsilon\} \times \Sigma^*) \cdot \mathrm{Graph}(\mathrm{id}) \cdot (\{\varepsilon\} \times \{\varepsilon\})$, and subword is $(\{\varepsilon\} \times \Sigma^*) \cdot \mathrm{Graph}(\mathrm{id}) \cdot (\{\varepsilon\} \times \Sigma^*)$, where id is the identity alphabetic morphism.

Proofs of Theorem 5.1 and Proposition 5.3. We present the proof for the suffix relation $\preceq_{\mathrm{suff}}$. The proofs for the subword relation, and more generally, for the relations containing the graph of an alphabetic morphism follow the same idea and will be explained after the proof for $\preceq_{\mathrm{suff}}$. The proof is by encoding nonemptiness for linearly bounded automata (LBA). Recall that an LBA $\mathcal{A}$ has a tape alphabet Γ that contains two distinguished symbols, α and β, which are the left and the right marker. The input word $w \in (\Gamma - \{\alpha, \beta\})^*$ is written between them, i.e., the content of the input tape is $\alpha \cdot w \cdot \beta$. The LBA behaves just like a Turing machine, except that when it is reading α or β, it cannot rewrite them, and it cannot move left of α or right of β. The problem of checking whether the language of a given LBA is nonempty is undecidable.

We encode this as follows. The alphabet Σ is the disjoint union of the tape alphabet Γ of the LBA $\mathcal{A}$, the set of its states Q, and the designated symbol \$ (we assume, of course, that these are disjoint). A configuration C of the LBA consists of the tape content $a_0 \dots a_n$, where $a_0 = \alpha$ and $a_n = \beta$, and all the a_is, for $0 < i < n$, are letters from $\Gamma - \{\alpha, \beta\}$, the state q, and the position i, for $0 \leq i \leq n$, that the head is pointing to. We encode this as a word

$$w_C \;=\; \$a_0 \dots a_{i-1} q a_i \dots a_n \$ \in \Sigma^*$$

of length $n + 4$. Of course if the head is pointing to α, the configuration is $\$qa_0 \dots a_n\$$. Note that if we have a run of the LBA with configurations $C_0, C_1, \dots$, then the lengths of all the w_{C_i}s are the same.

Next, note that the relation

$$R^{\mathcal{A}}_{\mathrm{imm}} \;=\; \{(w_C, w_{C'}) \mid C' \text{ is an immediate successor of } C\}$$

is regular (in fact such a relation is well-known to be regular even for arbitrary Turing machines [5, 7, 8]). Since all configurations are of the same length, we obtain that the relation

$$R'_{\mathcal{A}} \;=\; \{(w_{C_0} w_{C_1} \dots w_{C_m}, w_{C'_1} \dots w_{C'_m}) \mid C'_{i+1} \text{ is an immediate successor of } C_i \text{ for } i < m\}$$

is regular too (since only one configuration in the first projection does not correspond to a configuration in the second projection). By taking the product with a regular language that ensures that the first symbol from Q in a word is q_0, and the last such symbol is from F, we have a regular relation

$$R_{\mathcal{A}} \;=\; \left\{ (w_{C_0} w_{C_1} \dots w_{C_m}, w_{C'_1} \dots w_{C'_m}) \;\middle|\; \begin{array}{l} C'_{i+1} \text{ is an immediate successor of } C_i \text{ for } i < m; \\ C_0 \text{ is an initial configuration ;} \\ C_m \text{ is a final configuration} \end{array} \right\}$$

which can be effectively constructed from the description of the LBA.

Now assume that $R_{\mathcal{A}} \cap \preceq_{\mathrm{suff}}$ is nonempty. Then, since all encodings of configurations are of the same length, it must contain a pair $(w_{C_0} w_{C_1} \dots w_{C_m}, w_{C_1} \dots w_{C_m})$ such that C_{i+1}

is an immediate successor of C_i for all $i < m$. Since C_0 is an initial configuration and C_m is a final configuration, this implies that the LBA has an accepting computation. Conversely, if there is an accepting computation with a sequence of configurations $C_0, C_1, \ldots, C_m$ of the LBA, then the pair $(w_{C_0} w_{C_1} \ldots w_{C_m}, w_{C_1} \ldots w_{C_m})$ is both in $R_{\mathcal{A}}$ and in the suffix relation. Hence, $R_{\mathcal{A}} \cap \preceq_{\text{suff}}$ is nonempty iff there is an accepting computation of the LBA, proving undecidability.

The proof for the subword relation is practically the same. We change the definition of relation $R_{\mathcal{A}}$ so that there is an extra \$ symbol inserted between w_{C_0} and w_{C_1}, and two extra \$ symbols after w_{C_m} in the first projection; in the second projection we insert extra two \$ symbols before $w_{C'_1}$ and after $w_{C'_m}$. Note that the relation remains regular: even if the components are not fully synchronized, at every point there is a constant delay between them (either 2 or 1), and this can be captured by simply encoding one or two alphabet symbols into the state. Since in each word there are precisely two places where the subword \$\$\$ appears, the subword relation in this case becomes the suffix relation, and the previous proof applies.

The same proof can be applied to deduce Proposition 5.3. Note that we can encode letters of alphabet Σ within the alphabet $\{0, 1\}$ so that the encodings of each letter of Σ will have the same length, namely $\lceil \log_2(|\Gamma| + |Q| + 1) \rceil$. Then the same proof as before will apply to show undecidability over the alphabet $\{0, 1\}$, since the encodings of configurations still have the same length.

Since R_0 is regular, it is of the form $\bigcup_i L_i \times K_i$, and by the assumption, $\bigcup_i K_i = \Sigma^*$. Thus, the encoding of the initial configuration will belong to one of the K_is, say K_j. We then take a fixed word $w_0 \in L_j$ and assume that the second component of the relation starts with w_0 (which can be enforced by the regular relation). Likewise, we take a fixed pair $(w_1, w_2) \in R_1$, and assume that w_1 is the suffix of the first component of the relation, and w_2 is the suffix of the second. This too can be enforced by the regular relation.

Now if we have a non-constant alphabetic morphism f, we have two letters, say a and b, so that $f(a) \neq f(b)$. We now simply use these letters, with a playing the role of 0, and b playing the role of 1 in the first projection of relation R, and $f(a), f(b)$ playing the roles of 0 and 1 in the second projection, to encode the run of an LBA as we did before. The only difference is that instead of a sequence of \$ symbols to specify the positions of the encoding we use a (fixed-length) sequence that is different from w_0, w_1, w_2 above, to identify its position uniquely. Then the proof we have presented above applies verbatim. □

5.2. **Decidable cases: subsequence and relatives.** We now show that the intersection problem is decidable for the subsequence relation $\sqsubseteq$ and, much more generally, for a class of relations that do not, like the relations considered in the previous section, have a "rigid" part. More precisely, the problem is also decidable for any relation so that its projection on the first component is closed under subsequence. However, the complexity bounds are extremely high. In fact we show that the complexity of checking whether $(R \cap \sqsubseteq) \neq \emptyset$, when R ranges over REG_2, is not bounded by any multiply-recursive function. This was previously known for R ranging over RAT_2, and was viewed as the simplest problem with non-multiply-recursive complexity [13]. We now push it further and show that this high complexity is already achieved with regular relations.

Some of the ideas for showing this come from a decidable relaxation of the Post Correspondence Problem (PCP), namely the *regular Post Embedding Problem*, or PEP^{reg}, introduced in [13]. An instance of this problem consists of two morphisms $\sigma, \sigma' : \Sigma^* \to \Gamma^*$ and a regular language $L \subseteq \Sigma^*$; the question is whether there is some $w \in L$ such that $\sigma(w) \sqsubseteq \sigma'(w)$ (recall that in the case of the PCP the question is whether $\sigma(w) = \sigma'(w)$ with $L = \Sigma^+$). We call w a *solution* to the instance (σ, σ', L). The PEP^{reg} problem is known to be decidable, and as hard as the reachability problem for lossy channel systems [13] which cannot be bounded by any primitive-recursive function —in fact, by any multiply-recursive function (a generalization of primitive recursive functions with hyper-Ackermannian complexity, see [31]). More precisely, it is shown in [32] to be precisely at the level $\mathrm{F}_{\omega^\omega}$ of the fast-growing hierarchy of recursive functions [29, 31].[1]

The problem PEP^{reg} is just a reformulation of the problem $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$. Indeed, relations of the form $\{(f(w), g(w)) \mid w \in L\}$, where $L \subseteq \Sigma^*$ ranges over regular languages and f, g over morphisms $\Sigma^* \to \Gamma^*$ are precisely the relations in RAT_2 [6, 30]. Hence, $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ is decidable, with non-multiply-recursive complexity.

Proposition 5.4 ([13])**.** $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ *is decidable, non-multiply-recursive.*

We show that the lower bound already applies to regular relations.

Theorem 5.5. *The problem* $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ *is decidable, and its complexity is not bounded by any multiply-recursive function.*

The proof of the theorem above will be shown further down, after some preparatory definitions and lemmas are introduced.

It is worth noticing that one cannot solve the problem $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ by simply reducing to nonemptiness of rational relations due to the following.

Proposition 5.6. *There is a binary regular relation* R *such that* $(R \cap \sqsubseteq)$ *is not rational.*

Proof. Let $\Sigma = \{a, b\}$, and consider the following regular relation,

$$R = \{(a^m, b^m \cdot a^{m'}) \mid m, m' \in \mathbb{N}\}.$$

Note that the relation $R \cap \sqsubseteq$ is then $\{(a^m, b^m \cdot a^{m'}) \mid m, m' \in \mathbb{N}, m' \geq m\}$. We show that $R \cap \sqsubseteq$ is not rational by means of contradiction. Suppose that it is, and let $\mathcal{A}$ be an NFA over $\{a, b, \varepsilon\} \times \{a, b, \varepsilon\}$ that recognizes $R \cap \sqsubseteq$. Suppose Q is the set of states of $\mathcal{A}$, and $|Q| = n$.

Consider the following pair

$$(a^{n+1}, b^{n+1} \cdot a^{n+1}) \quad \in \quad R \cap \sqsubseteq.$$

Then there must be some $u \in (\{a, b, \varepsilon\} \times \{a, b, \varepsilon\})^*$ such that

$$(\pi_1(u), \pi_2(u)) = (a^{n+1}, b^{n+1} \cdot a^{n+1})$$

and $u \in \mathcal{L}(\mathcal{A})$. Let $\rho_{\mathcal{A}} : [0..|u|] \to Q$ be the accepting run of $\mathcal{A}$ on u, and let $1 \leq i_1 < \cdots < i_{n+1} \leq |u|$ be such that $\pi_2(u[i_j]) = a$ for all $j \in [n+1]$. Clearly, among $\rho_{\mathcal{A}}(i_1), \ldots, \rho_{\mathcal{A}}(i_{n+1})$ there must be two repeating elements by the pigeonhole principle. Let $1 \leq j_1 < j_2 \leq n+1$

[1] In this hierarchy—also known as the Extended Grzegorczyk Hierarchy—, the classes of functions F_α are closed under elementary-recursive reductions, and are indexed by ordinals. Ackermannian complexity corresponds to level $\alpha = \omega$, and level $\alpha = \omega^\omega$ corresponds to some hyper-Ackermannian complexity.

be such elements, where $\rho_{\mathcal{A}}(i_{j_1}) = \rho_{\mathcal{A}}(i_{j_2})$. Hence $u' = u[1..i_{j_1} - 1] \cdot u[i_{j_2}..] \in \mathcal{L}(\mathcal{A})$, and therefore

$$\big(\pi_1(u'), \pi_2(u')\big) \quad \in \quad R \cap \sqsubseteq.$$

Notice that $\pi_2(u') = b^{n+1} \cdot a^{n+1-(j_2-j_1)}$. But by definition of $R \cap \sqsubseteq$ we have that $\pi_1(u') = a^{n+1}$ with $n + 1 - (j_2 - j_1) \geq n + 1$, which is clearly false. The contradiction comes from the assumption that $R \cap \sqsubseteq$ is rational. □

As already mentioned, the decidability part of Theorem 5.5 follows from Proposition 5.4. We prove the lower bound by reducing PEP^{reg} into $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

This reduction is done in two phases. First, we show that there is a reduction from PEP^{reg} into the problem of finding solutions of PEP^{reg} with a certain shape, which we call a *strict codirect solutions* (Lemma 5.7). Second, we show that there is a reduction from the problem of finding strict codirect solutions of a PEP^{reg} instance into $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ (Proposition 5.8). Both reductions are elementary and thus the hardness result of Theorem 5.5 follows.

In the next section we define the strict codirect solutions for PEP^{reg}, showing that we can restrict to this kind of solutions. In the succeeding section we show how to reduce the problem into $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

5.2.1. *Codirect solutions of* PEP^{reg}. There are some variations of the PEP^{reg} problem that result being equivalent problems. These variations restrict the solutions to have certain properties. Given a PEP^{reg} instance (σ, σ', L), we say that $w \in L$ with $|w| = m$ is a *codirect solution* if there are (possibly empty) words $v_1, \ldots, v_m$ such that

1. $v_k \sqsubseteq \sigma'(w[k])$ for all $1 \leq k \leq m$,
2. $\sigma(w[1..m]) = v_1 \cdots v_m$, and
3. $|\sigma(w[1..k])| \geq |v_1 \cdots v_k|$ for all $1 \leq k \leq m$.

If furthermore

4. $|\sigma(w[1..k])| > |v_1 \cdots v_k|$ for all $1 \leq k < m$,

we say that it is a *strict codirect solution*. In this case we say that the solution w is *witnessed by* $v_1, \ldots, v_m$. In [13] it has been shown that the problem of whether an instance of the PEP^{reg} problem has a codirect solution is equivalent to the problem of whether it has a solution. Moreover, it can be shown that this also holds for strict codirect solutions.

Lemma 5.7. *The problem of whether a* PEP^{reg} *instance has a strict codirect solution is as hard as whether a* PEP^{reg} *instance has a solution.*

Proof. We only show how to reduce from finding a codirect solution problem to finding a strict codirect solution problem. The other direction is trivial, since a strict codirect solution is in particular a solution. Let (σ, σ', L) be a PEP^{reg} instance, and $w \in L$ be a codirect solution with $|w| = m$, minimal in size, and witnessed by $v_1, \ldots, v_k$. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an NFA representing L, where $|Q| = n$. Let $\rho : [0..m] \to Q$ be an accepting run of $\mathcal{A}$ on w. Let $0 \leq k_1 < \cdots < k_t \leq m$ be all the elements of $\{s \geq 0 : |\sigma(w[1..s])| = |v_1 \cdots v_s|\}$. Observe that $k_1 = 0$, and $k_t = m$ by condition 2. It is not difficult to show that by minimality of m there cannot be more than n indices.

Claim 5.0.1. $t \leq n$.

Proof. Suppose ad absurdum that $t \geq n+1$. Then, there must be two $k_l < k_{l'}$ such that $\rho(k_l) = \rho(k_{l'})$. Hence, $w' = w[1..k_l] \cdot w[k_{l'}+1..] \in L$ is also a codirect solution, contradicting that w is a minimal size solution. □

Let $L[q, q']$ be the regular language denoted by the NFA $(Q, \Sigma, q, \delta, \{q'\})$.

Claim 5.0.2. For every $i < t$, $(\sigma, \sigma', L[\rho(k_i), \rho(k_{i+1})])$ has a strict codirect solution.

Proof. We show that for every $i < t$, $w[k_i+1..k_{i+1}]$ is a solution for $(\sigma, \sigma', L[\rho(k_i), \rho(k_{i+1})])$, witnessed by $v_{k_i+1}, \ldots, v_{k_{i+1}}$.

Clearly, condition 1 still holds. Further, since

$$|\sigma(w[1..k_i])| = |v_1 \cdots v_{k_i}| \qquad \text{and} \qquad |\sigma(w[1..k_{i+1}])| = |v_1 \cdots v_{k_{i+1}}|,$$

we have that $|\sigma(w[k_i+1..k_{i+1}])| = |v_{k_i+1} \cdots v_{k_{i+1}}|$ and then

$$\sigma(w[k_i+1..k_{i+1}]) = v_{k_i+1} \cdots v_{k_{i+1}},$$

verifying condition 2.

Finally, by the fact that k_i and k_{i+1} are consecutive indices we cannot have some k' with $k_i + 1 < k' < k_{i+1}$ so that $|\sigma(w[k_i+1..k'])| = |v_{k_i+1} \cdots v_{k'}|$ since it would imply $|\sigma(w[1..k'])| = |v_1 \cdots v_{k'}|$ and in this case $k' \geq k_{i+1}$. Then, conditions 3 and 4 hold. □

Therefore, we obtain the following reduction.

Claim 5.0.3. (σ, σ', L) has a codirect solution if, and only if, there exist $\{q_1, \ldots, q_t\} \subseteq Q$ with $q_1 = q_0$ and $q_t \in F$, such that for every i, $(\sigma, \sigma', L[q_i, q_{i+1}])$ has a strict codirect solution.

This reduction being exponential is outweighed by the fact that we are dealing with a much harder problem. □

With the help of Lemma 5.7 we prove Theorem 5.5 in the next section.

5.2.2. *Proof of Theorem 5.5.* Since decidability follows from Proposition 5.4, we only show the lower bound. To this end, we show how to code the existence of a strict codirect solution as an instance of $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Proposition 5.8. *There is an elementary reduction from the existence of strict codirect solutions of* PEP^{reg} *into* $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Given a PEP^{reg} instance (σ, σ', L), remember that the presence of a strict codirect solution enforces that if there is a pair $(u, v) = (\sigma(w), \sigma'(w))$ with $w \in L$ and $u \sqsubseteq v$, it is such that for every proper prefix u' of u the smallest prefix v' of v such that $u' \sqsubseteq v'$ must be so that $|v'| > |u'|$. In the proof, we convert the *rational* relation $R = \{(\sigma(w), \sigma'(w)) \mid w \in L\}$ into a length-preserving *regular* relation R' over an extended alphabet $\Gamma \cup \{\#\}$, defined as the set of all pairs $(u, v) \in (\Gamma \cup \{\#\})^* \times (\Gamma \cup \{\#\})^*$ so that $|u| = |v|$ and $(u_\Gamma, v_\Gamma) \in R$. If we now let R'' to be the regular relation $R' \cdot \{(\varepsilon, v) \mid v \in \{\#\}^*\}$, we obtain that:

(i) if $w \in R'' \cap \sqsubseteq$ then $w' \in R \cap \sqsubseteq$, where w' is the projection of w onto $\Gamma^* \times \Gamma^*$; and
(ii) if there is some strict codirect solution $w' \in R \cap \sqsubseteq$, then there is some $w \in R'' \cap \sqsubseteq$ such that w' is the projection of w onto $\Gamma^* \times \Gamma^*$.

Whereas (i) is trivial, (ii) follows from the fact that w' is a strict codirect solution. If $w' = (u, v) \in R''$, where $f(w) = (u)_\Gamma$, $g(w) = (v)_\Gamma$, the complication is now that, since $u \in \Gamma \cup \{\#\}$, it could be that $u \not\sqsubseteq v$ just because there is some $\#$ in u that does not appear in v. But we show how to build (u, v) such that whenever $u[i] = \#$ forces $v[j] = \#$ with $j > i$ then we also have that $u[j] = \#$. This repeats, forcing $v[k] = \#$ for some $k > j$ and so on, until we reach the tail of v that has sufficiently many $\#$'s to satisfy all the accumulated demands for occurrences of $\#$.

Proof of Proposition 5.8. Let (σ, σ', L) be a PEP^{reg} instance. For every $a \in \Sigma$, consider the binary relation R_a consisting of all pairs $(u, u') \in (\Gamma \cup \{\#\})^* \times (\Gamma \cup \{\#\})^*$ such that $u_\Gamma = \sigma(a)$, $u'_\Gamma = \sigma'(a)$ and $|u| = |u'|$. Note that R_a is a length-preserving regular relation. Let R' be the set of pairs $(u_1 \cdots u_m, u'_1 \cdots u'_m)$ such that there exists $w \in L$ where $|w| = m$ and $(u_i, u'_i) \in R_{w[i]}$ for all i. Note that R' is still a length-preserving regular relation. Finally, we define R as the set of pairs $(u, u' \cdot u'')$ such that $(u, u') \in R'$ and $u'' \in \{\#\}^*$. R is no longer a length-preserving relation, but it is regular. Observe that if $R \cap \sqsubseteq \neq \emptyset$, then (σ, σ', L) has a solution. Conversely, we show that if (σ, σ', L) has a strict codirect solution, then $R \cap \sqsubseteq \neq \emptyset$.

Suppose that the PEP^{reg} instance (σ, σ', L) has a strict codirect solution $w \in L$ with $|w| = m$, witnessed by $v_1, \ldots, v_m$. Assume, without any loss of generality, that σ and σ' are alphabetic morphisms and that $m > 1$. We exhibit a pair $(u, u') \in R$ such that $u \sqsubseteq u'$. We define $(u, u') = (u_1 \cdots u_m, u'_1 \cdots u'_m \cdot u'_{m+1})$, where $(u_i, u'_i) \in R_{w[i]}$ for every $i \leq m$, and $u'_{m+1} \in \{\#\}^*$. In order to give the precise definition of (u, u'), we need to introduce some concepts first.

Let $\sigma_\#(a) \in \Gamma \cup \{\#\}$ be $\#$ if $\sigma(a) = \epsilon$, or $\sigma(a)$ otherwise; likewise for $\sigma'_\#$. By definition of strict codirect solution, we have the following.

Claim 5.0.4. $\sigma(w[1]) \in \Gamma$.

Proof. Indeed, if $\sigma(w[1]) \neq \Gamma$, then $\sigma(w[1]) = \varepsilon$ and $|\sigma(w[1])| = 0$, and then condition 4 of strict codirectness stating that $|\sigma(w[1])| > |v_1|$, would be falsified. □

Let us define the function $g : [m] \to [m]$ so that $g(i)$ is the minimum j such that $v_1 \cdots v_j = \sigma(w[1..i])$. Note that there is always such a j, since $|\sigma(w[1..i])| > 0$ by Claim 5.0.4. Now we show some easy properties of g, necessary to correctly define the witnessing pair $(u, u') \in R$ such that $u \sqsubseteq u'$.

Claim 5.0.5. $g(i) > i$ for all $1 \leq i < m$, and $g(m) = m$.

Proof. Let $g(i) = j$ and hence $|\sigma(w[1..i])| = |v_1 \cdots v_j|$. First, notice that $|v_1 \cdots v_j| = |\sigma(w[1..i])| \geq |v_1 \cdots v_i|$ by condition 3 of codirectness, and then that $j \geq i$. If $i < m$, $|v_1 \cdots v_i| < |\sigma(w[1..i])|$ by condition 4, and thus $|v_1 \cdots v_i| < |v_1 \cdots v_j|$ which implies $i < j$. If $i = m$, then $j = i$ by the fact that $j \geq i = m$. □

Claim 5.0.6. g is increasing: $g(i) \geq g(j)$ if $i \geq j$.

Proof. Given $m \geq i \geq j \geq 1$, we have that

$$\begin{aligned} |v_1 \cdots v_{g(i)}| &= |\sigma(w[1..i])| && \text{(by definition of } g) \\ &\geq |\sigma(w[1..j])| && \text{(since } i \geq j) \\ &= |v_1 \cdots v_{g(j)}| && \text{(by definition of } g) \end{aligned}$$

which implies that $g(i) \geq g(j)$. □

Observation 5.1. For all $i \leq m$, if $\sigma(w[i]) \in \Gamma$ then $\sigma(w[i]) = \sigma'(w[g(i)])$.

The most important pairs of positions $(i, j) \in [m] \times [m]$ that witness $u \sqsubseteq u'$, are those so that $j = g(i)$ and $\sigma(w[i]) \neq \varepsilon$. Once those are fixed, the remaining elements in the definition of g are also fixed. Let us call G to this set, and let us state some simple facts for later use.

$$G = \{(i, g(i)) \in [m] \times [m] \mid \sigma(w[i]) \in \Gamma\}$$

Observation 5.2. For every $(i, j), (i', j') \in G$, if $i \neq i'$ then $j \neq j'$. In other words, g restricted to $\{i \mid \sigma(w[i]) \in \Gamma\}$ is injective.

Claim 5.2.1. Given i, j with $(i, j) \in G$ and $i < m$, then $|\sigma(w[i..j])| \geq 2$.

Proof. This is because $i < j$ by Claim 5.0.5, $\sigma(w[i]) \in \Gamma$ by definition of G, and $\sigma(w[j]) = \sigma(w[g(i)]) \in \Gamma$ by definition of g. □

Since our coding uses the letter # as some sort of blank symbol, it will be useful to define the factors $\tilde{u}_1, \tilde{u}_2, \ldots$ of u that contain exactly one letter from Γ. We then define $\tilde{u}_i$ as the maximal prefix of $u_i \cdots u_m$ belonging to the following regular expression: $\Gamma \cdot \{\#\}^*$.

We are now in good shape to define precisely u_j, u'_j for every $j \in [m]$. For every $j < m$,

- if $(i, j) \in G$ for some i, then
$$u'_j = \tilde{u}_i \quad \text{and} \quad u_j = \sigma_{\#}(w[j]) \cdot u'_j[2..]; \text{ and}$$
- if there is no i so that $(i, j) \in G$, then
$$(u_j, u'_j) = (\sigma_{\#}(w[j]), \sigma'_{\#}(w[j])).$$

And on the other hand, $(u_m, u'_m) = (\sigma_{\#}(w[m]), \sigma'_{\#}(w[m]))$ and $u'_{m+1} = \#^{|u_1 \cdots u_m|}$. Figure 1 contains an example with all the previous definitions. Notice that the definition of u_j makes use of $\tilde{u}_j$ and the definition of $\tilde{u}_j$ seems to make use of u_j. We next show that in fact $\tilde{u}_j$ does not depend on u_j, and that the strings above are well defined.

Observation 5.3. For $i < m$, $\tilde{u}_i$ is a prefix of $u_i \cdots u_{g(i)-1}$.

Proof. By Claim 5.0.5 and Claim 5.2.1, $\sigma(w[i..g(i)])$ contains at least two elements and hence $u_i \cdots u_{g(i)}$ contains at least two elements from Γ, namely $u_i[1]$ and $u_{g(i)}[1]$. Then, $\tilde{u}_i$ cannot contain $u_i \cdots u_{g(i)-1} \cdot (u_{g(i)}[1])$ as a prefix. □

By the above Observation 5.3, to compute $\tilde{u}_i$ we only need u_j's and u'_j's with $j < i$, and hence (u, u') is well defined.

Observation 5.4. All the u_i's, u'_i's and $\tilde{u}_i$'s are of the form $a \cdot \# \cdots \#$ or $\# \cdots \#$, for $a \in \Gamma$.

From the definition of (u, u') we obtain the following.

Observation 5.5. For every $n \leq m$,

(1) $|(u_1 \cdots u_n)_\Gamma| = \{i \in [n] \mid \exists j.(i, j) \in G\} = |\sigma(w[1..n])|$, and
(2) $|(u'_1 \cdots u'_n)_\Gamma| = \{j \in [n] \mid \exists i.(i, j) \in G\} = |\sigma'(w[1..n])|$.

We now show that $(u, u') \in R$ and that $u \sqsubseteq u'$.

Claim 5.5.1. $(u, u') \in R$.

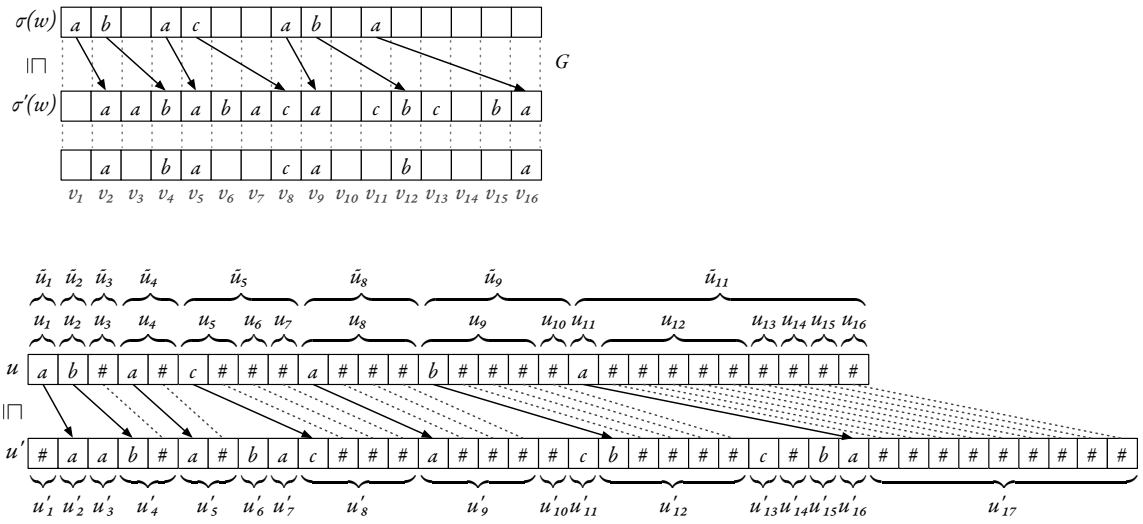

Figure 1: Exemplary reduction from PEP^{reg} to $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$, for the case $\sigma(w) = abacaba$, $\sigma'(w) = aababacacbcba$.

Proof. Note that $u_i = \sigma_{\#}(w[i])$ for all i and then $(u_i)_\Gamma = \sigma(w[i])$.

We also show that $(u'_i)_\Gamma = \sigma'(w[i])$. If u'_j is such that there is no $(i,j) \in G$, or $j = m$, then it is plain that $(u'_j)_\Gamma = \sigma'(w[j])$ by definition of u'_j. On the other hand, if $u'_j = \tilde{u}_i$ for $(i,j) \in G$, then

$$\begin{aligned}
(u'_j)_\Gamma &= (\tilde{u}_i)_\Gamma = (u_i)_\Gamma = (u_i[1])_\Gamma && \text{(by Observation 5.4)}\\
&= (\sigma(w[i]))_\Gamma && \text{(by def. of } u_i)\\
&= \sigma(w[i]) && \text{(since } \sigma(w[i]) \in \Gamma \text{ by def. of } G)\\
&= \sigma'(w[g(i)]) = \sigma'(w[j]). && \text{(by Observation 5.1)}
\end{aligned}$$

Thus, every (u_i, v_i) with $i \leq m$ is such that $(u_i)_\Gamma = \sigma(w[i])$ and $(u'_i)_\Gamma = \sigma'(w[i])$, meaning that $(u_i, v_i) \in R_{w[i]}$ for every $i \leq m$. Hence, we have that $(u_1 \cdots u_m, u'_1 \cdots u'_m) \in R'$ and since $u'_{m+1} \in \{\#\}^*$, $(u, u') \in R$. □

Next, we prove that $u \sqsubseteq u'$, but before doing so, we need an additional straightforward claim. Let $\{i_1 < \cdots < i_{|G|}\} = \{i \mid (i, g(i)) \in G\}$. Note that $i_1 = 1$ by Claim 5.0.4.

Claim 5.5.2. $i_{j+1} \leq g(i_j)$

Proof. By means of contradiction, suppose $g(i_j) < i_{j+1}$. Then,

$$\begin{aligned}
|\sigma(w[1..g(i_j)])| &= |\{i \in [g(i_j)] \mid \exists j.(i,j) \in G\}| && \text{(by Observation 5.5.1)}\\
&= |\{i \in [g(i_j)] \mid \exists j.(i,j) \in G\}| && \text{(since } g(i_j) < i_{j+1})\\
&= |\sigma'(w[1..g(i_j)])|. && \text{(by Observation 5.5.2)}
\end{aligned}$$

In other words, there is some $k < m$ such that $|\sigma(w[1..k])| = |\sigma'(w[1..k])|$. This is in contradiction with condition 4 of strict codirectness. Hence, $g(i_j) \geq i_{j+1}$. □

Claim 5.5.3. $u \sqsubseteq u'$.

Proof. We factorize $u = \hat{u}_1 \cdots \hat{u}_{|G|}$ and we show that each $\hat{u}_i$ is a substring of u' that appears in an increasing order.

We define $\hat{u}_j = u_{i_j} \cdots u_{i_{(j+1)}-1}$ for every $j < |G|$, and $\hat{u}_{|G|} = u_{i_{|G|}} \cdots u_m$. Hence, the $\hat{u}_i$'s form a factorization of u. Indeed, this is the unique factorization in which each $\hat{u}_i$ is of the form $b \cdot \# \cdots \#$ for $b \in \Gamma$.

For every $j < |G|$, we show that $\hat{u}_j \sqsubseteq u'_{g(i_j)}$.

$$\begin{aligned}
\hat{u}_j &= u_{i_j} \cdots u_{i_{(j+1)}-1} \\
&\sqsubseteq u_{i_j} \cdots u_{g(i_j)-1} && \text{(by Claim 5.5.2)} \\
&\sqsubseteq \tilde{u}_{i_j} && \text{(by Observation 5.3)} \\
&= \tilde{u}_{g^{-1}(g(i_j))} && \text{(by Observation 5.2)} \\
&= u'_{g(i_j)} && \text{(by def. of } u')
\end{aligned}$$

On the other hand, $\hat{u}_{|G|} \sqsubseteq u'_{g(i_{|G|})} \cdot u'_{m+1} = u'_m \cdot u'_{m+1}$. By Claim 5.0.6, g is increasing. Hence, $u \sqsubseteq u'$. □

By Claims 5.5.1 and 5.5.3, we conclude that $R \cap \sqsubseteq \neq \emptyset$. □

5.2.3. *Subsequence-closed relations.* The next question is how far we can extend the decidability of $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$. It turns out that if we allow one projection of a rational relation to be closed under taking subsequences, then we retain decidability.

Let $R \subseteq \Sigma^* \times \Gamma^*$ be a binary relation. Define another binary relation

$$R_{\sqsubseteq} = \{(u, w) \mid u \sqsubseteq u' \text{ and } (u', w) \in R \text{ for some } u'\}$$

Then the class of *subsequence-closed relations*, or SCR, is the class $\{R_{\sqsubseteq} \mid R \in \mathsf{RAT}\}$. Note that the subsequence relation itself is in SCR, since it is obtained by closing the (regular) equality relation under subsequence. That is, $\sqsubseteq \;=\; \{(w, w) \mid w \in \Sigma^*\}_{\sqsubseteq}$. Not all rational relations are subsequence-closed (for instance, subword is not).

The following summarizes properties of subsequence-closed relations.

Proposition 5.9.

(1) $\mathsf{SCR} \subsetneq \mathsf{RAT}$.
(2) $\mathsf{SCR} \not\subseteq \mathsf{REG}$ *and* $\mathsf{REG} \not\subseteq \mathsf{SCR}$.
(3) *A relation R is in* SCR *iff* $\{w \otimes w' \mid (w, w') \in R\}$ *is accepted by an NFA* $\mathcal{A} = \langle Q, \Sigma_\perp \times \Sigma_\perp, q_0, \delta, F\rangle$ *such that* $(q, (a, b), q') \in \delta$ *implies* $(q, (\perp, b), q') \in \delta$ *for all* $q, q' \in Q$ *and* $a, b \in \Sigma_\perp$. *We call an automaton with such property a* subsequence-closed automaton.

Note that (3) is immediate by definition of $R_{\sqsubseteq}$, (1) is a consequence of (3), and (2) is due to the fact that $\sqsubseteq$ is not regular and that, for example, the identity $\{(u, u) \mid u \in \Sigma^*\}$ is not a subsequence-closed relation.

When an SCR relation is given as an input to a problem, we assume that it is represented as a subsequence-closed automaton as defined in item (3) in the above proposition.

Note also that $(\mathsf{SCR} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ is decidable in polynomial time: if $R, R' \in \mathsf{SCR}$ and $R \cap R' \neq \emptyset$, then $(\varepsilon, w) \in R \cap R'$ for some w, and hence the problem reduces to simple NFA nonemptiness checking.

The main result about SCR relations generalizes decidability of $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Theorem 5.10. *The problem* $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ *is decidable, with non-mutiply recursive complexity.*

In order to prove Theorem 5.10 we use Lemmas 5.11 and 5.12, as shown below. But first we need to introduce some additional terminology. We say that $(\mathcal{A}_0, \mathcal{A}_1)$ is an *instance* of $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ over Σ, Γ if $\mathcal{A}_1$ is a subsequence-closed automaton over $\Sigma_\perp \times \Gamma_\perp$, and $\mathcal{A}_0$ is a NFA over $\Sigma_\perp \times \Gamma_\perp$. Given a $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ instance $(\mathcal{A}_0, \mathcal{A}_1)$ over Σ, Γ, we say that (w_1, w_2) is a *solution* if $w_1, w_2 \in (\Sigma_\perp \times \Gamma_\perp)^*$, $w_1 \in \mathcal{L}(\mathcal{A}_1), w_2 \in \mathcal{L}(\mathcal{A}_0)$. We say that a solution (w_0, w_1) of an instance $(\mathcal{A}_0, \mathcal{A}_1)$ over Σ, Γ is *synchronized* if $\pi_2(w_0) = \pi_2(w_1)$. We write $(\mathsf{RAT} \cap \mathsf{SCR})^{\text{syn}} \stackrel{?}{=} \emptyset$ for the problem of whether there is a synchronized solution.

Lemma 5.11. *There is a polynomial-time reduction from the problem* $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ *into* $(\mathsf{RAT} \cap \mathsf{SCR})^{syn} \stackrel{?}{=} \emptyset$.

Proof. We show that $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ is reducible to the problem of whether there exists a synchronized solution of $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$. Suppose that $(\mathcal{A}_0, \mathcal{A}_1)$ is an instance of $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ over the alphabets Σ, Γ. Consider the automata $\mathcal{A}'_0, \mathcal{A}'_1$ as the result of adding all transitions $(q, (\perp, \perp), q)$ for every possible state q to both automata. It is clear that the relations recognized by these remain unchanged, and that $\mathcal{A}'_0$ is still a subsequence-closed automaton. Moreover, this new instance has a synchronized solution if there is any, as stated in the following claim.

Claim 5.5.4. There is a synchronized solution for $(\mathcal{A}'_0, \mathcal{A}'_1)$ if, and only if, there is a solution for $(\mathcal{A}_0, \mathcal{A}_1)$.

The 'only if' part is immediate. For the 'if' part, let (w_0, w_1) be a solution for $(\mathcal{A}_0, \mathcal{A}_1)$. Let $w_0 = w_{0,1} \cdots w_{0,n}$, $w_1 = w_{1,1} \cdots w_{1,n}$ be factorizations of w_0 and w_1 such that for every $i \in \{0, 1\}$, $\pi_2(w_{i,1})$ is in $\{\perp\}^*$; and for each $j > 1, i \in 0, 1$, $\pi_2(w_{i,j})$ is in $\Gamma \cdot \{\perp\}^*$. It is plain that there is always such factorization and that it is unique.

For every $j \in [n]$, we define $w'_{0,j} = w_{0,j} \cdot (\perp, \perp)^k$ and $w'_{1,j} = w_{1,j} \cdot (\perp, \perp)^{-k}$, with $k = |w_{1,j}| - |w_{0,j}|$, where we assume that $(\perp, \perp)^m$ with $m \leq 0$ is the empty string. We define $w'_0 = w'_{0,1} \cdots w'_{0,n}$, $w'_1 = w'_{1,1} \cdots w'_{1,n}$. Note that (w'_0, w'_1) is a solution of $(\mathcal{A}'_0, \mathcal{A}'_1)$ since it is the result of adding letters $(\perp, \perp)$ to (w_0, w_1), which is also a solution of $(\mathcal{A}'_0, \mathcal{A}'_1)$. We have that $\pi_2(w'_0) = \pi_2(w'_1)$, and therefore that (w'_0, w'_1) is a synchronized solution for $(\mathcal{A}'_0, \mathcal{A}'_1)$. □

Lemma 5.12. *There is a polynomial-time reduction from* $(\mathsf{RAT} \cap \mathsf{SCR})^{syn} \stackrel{?}{=} \emptyset$ *into* $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Proof. The problem of finding a synchronized solution for $\mathcal{A}_0, \mathcal{A}_1$ can be then formulated as the problem of finding words $v, u_0, u_1 \in \Sigma_\perp^*$ with $|v| = |u_0| = |u_1|$, so that $(u_0 \otimes v, u_1 \otimes v)$ is a solution. We can compute an NFA $\mathcal{A}$ over $\Sigma_\perp^2 \times \Gamma_\perp$ from $\mathcal{A}_0, \mathcal{A}_1$, such that $(u_0, u_1, v) \in \mathcal{L}(\mathcal{A})$ if, and only if, $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_1)$ and $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_0)$. Consider now an automaton $\mathcal{A}'$ over $\Sigma_\perp^2$ such that $\mathcal{L}(\mathcal{A}') = \{(u_0, u_1) \mid \exists v \ (u_0, u_1, v) \in \mathcal{L}(\mathcal{A})\}$. It corresponds to the rational automaton of the projection onto the first and second components of the ternary relation of $\mathcal{A}$, and it can be computed from $\mathcal{A}$ in polynomial time. We then deduce that there exists $u_0 \otimes u_1 \in \mathcal{L}(\mathcal{A}')$ so that $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$ if, and only if, there is $v \in \Gamma_\perp^*$ with $|v| = |u_0| = |u_1|$ so that $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_0)$ and $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_1)$, where $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$. But this condition is in fact equivalent to $R_0 \cap R_1 \neq \emptyset$ (where $R_i = \{((u)_\Sigma, (v)_\Sigma) \mid u \otimes v \in \mathcal{L}(\mathcal{A}_i)\}$), since

- if $((u_1)_\Sigma, (v)_\Sigma) \in R_1$ and $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$, then $((u_0)_\Sigma, (v)_\Sigma) \in R_1$ (since $R_1 \in \mathsf{SCR}$) and hence $((u_0)_\Sigma, (v)_\Sigma) \in R_0 \cap R_1$; and
- if $R_0 \cap R_1 \neq \emptyset$, then there exists a synchronized solution $(u_0 \otimes v, u_1 \otimes v)$ of $\mathcal{A}_0, \mathcal{A}_1$; in other words, there are $|v| = |u_0| = |u_1|$ so that $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_0)$, $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_1)$, and $(u_0)_\Sigma = (u_1)_\Sigma$.

We have thus reduced the problem to an instance of $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$: whether there is (u, v) in the relation denoted by $\mathcal{A}'$ so that $u \sqsubseteq v$. □

Proof of Theorem 5.10. The decidability part of Theorem 5.10 follows as a corollary of Lemmas 5.11 and 5.12, and Proposition 5.4. Of course the complexity is non-multiply-recursive, since the problem subsumes $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ of Theorem 5.5. □

Coming back to graph logics, we obtain:

Corollary 5.13. *The complexity of evaluation of* ECRPQ*($\sqsubseteq$) queries is not bounded by any multiply-recursive function.*

Another corollary can be stated in purely language-theoretic terms.

Corollary 5.14. *Let $\mathcal{C}$ be a class of binary relations on Σ^* that is closed under intersection and contains* REG*. Then the nonemptiness problem for $\mathcal{C}$ is:*

- *undecidable if $\preceq$ or $\preceq_{\text{suff}}$ is in $\mathcal{C}$;*
- *non-multiply-recursive if $\sqsubseteq$ is in $\mathcal{C}$.*

5.3. **Discussion.** In addition to answering some basic language-theoretic questions about the interaction of regular and rational relations, and to providing the simplest yet problem with non-multiply-recursive complexity, our results also rule out logical languages for graph databases that freely combine regular relations and some of the most commonly used rational relations, such as subword and subsequence. With them, query evaluation becomes either undecidable or non-multiply-recursive (which means that no realistic algorithm will be able to solve the hard instances of this problem).

This does not yet fully answer our questions about the evaluation of queries in graph logics. First, in the case of subsequence (or, more generally, SCR relations) we still do not know if query evaluation of ECRPQs with such relations is decidable (i.e., what happens with $\textsc{GenInt}_S(\mathsf{REG})$ for such relations S).

Even more importantly, we do not yet know what happens with the complexity of CRPQs (i.e., $\textsc{GenInt}_S(\mathsf{REC})$) for various relations S. These questions are answered in the next section.

6. Restricted logics and the generalized intersection problem

The previous section already ruled out some graph logics with rational relations as either undecidable or decidable with extremely high complexity. This was done merely by analyzing the intersection problem for binary rational and regular relations. We now move to the study of the generalized intersection problem, and use it to analyze the complexity of graph logics in full generality. We first deal with the generalization of the decidable case (SCR relations), and then consider the problem $\textsc{GenInt}_S(\mathsf{REC})$, corresponding to CRPQs extended with relations S on paths.

6.1. **Generalized intersection problem and subsequence.** We know that $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ is decidable, although not multiply-recursive. What about its generalized version? It turns out it remains decidable.

Theorem 6.1. *The problem* $\textsc{GenInt}_{\sqsubseteq}(\mathsf{REG})$ *is decidable. That is, there is an algorithm that decides, for a given* m*-ary regular relation* R *and* $I \subseteq [m]^2$*, whether* $R \cap_I \sqsubseteq \neq \emptyset$*.*

Proof. Let $k \in \mathbb{N}$, $I \subseteq [k] \times [k]$ and $R \in \mathsf{REG}_k$ be an instance of the problem. Let us define $G = \{(w_1, \ldots, w_k) \mid \forall (i,j) \in I, w_i \sqsubseteq w_j\}$. We show how to compute if $R \cap G$ is empty or not. Let $\mathcal{A} = (Q, (\Sigma_\perp)^k, q_0, \delta, F)$ be a NFA over $(\Sigma_\perp)^k$ corresponding to R, for simplicity we assume that it is complete. Remember that every $w \in \mathcal{L}(\mathcal{A})$ is such that $\pi_i(w)$ is in $\Sigma^*; \{\perp\}^*$ for every $i \in [k]$.

Given $u, v \in \Sigma^*$, we define $u \setminus v$ as $u[i..]$, where i is the maximal index such that $u[1..i-1] \sqsubseteq v$. In other words, $u \setminus v$ is the result of removing from u the maximal prefix that is a subsequence of v.

We define a finite tree $\mathbf{t}$ whose every node is labeled with

- a depth $n \geq 0$,
- k words $w_1, \ldots, w_k \in \Sigma_\perp^n$,
- for every $(i,j) \in I$, a word $\alpha_{ij} \in \Sigma^*$, and
- a state $q \in Q$.

For a node x we denote these labels by $x.n$, $x.w_1, \ldots, x.w_k$, $x.\alpha_{ij}$ for every $(i,j) \in I$ and $x.q$ respectively. The tree is such that the following conditions are met.

- The root is labeled by $x.n = 0$, $x.w_1 = \cdots = x.w_k = \varepsilon$, for very $(i,j) \in I$, $x.\alpha_{ij} = \varepsilon$, and $x.q = q_0$.
- A node x has a child y in $\mathbf{t}$ if and only if
  - $y.n = x.n + 1$,
  - $x.w_i = y.w_i[1..y.x - 1]$ for every $i \in [k]$,
  - there is a transition $(x.q, \bar{a}, y.q) \in \delta$ with $\bar{a} = (y.w_i[y.n])_{i \in [k]}$, and
  - $y.\alpha_{ij} = (w_i)_\Sigma \setminus (w_j)_\Sigma$ for every $(i,j) \in I$.
- A node x is a leaf in $\mathbf{t}$ if and only if is final or saturated (as defined below).

A node x is **final** if $x.q \in F$ and $x.\alpha_{ij} = \varepsilon$ for all $(i,j) \in I$. It is **saturated** if it is not final and there is an ancestor $y \neq x$ such that $y.q = x.q$ and $y.\alpha_{ij} \sqsubseteq x.\alpha_{ij}$ for all $(i,j) \in I$.

Lemma 6.2. *The tree* $\mathbf{t}$ *is finite and computable.*

Proof. The root is obviously computable, and for every branch, one can compute the list of children nodes of the bottom-most node of the branch. Indeed these are finite and bounded. The tree $\mathbf{t}$ cannot have an infinite branch. If there was an infinite branch, then as a result of Higman's Lemma *cum* Dickson's Lemma (and the Pigeonhole principle) there would be two nodes $x \neq y$, where x is an ancestor of y, $x.q = y.q$, and for all $(i,j) \in I$, $x.\alpha_{ij} \sqsubseteq y.\alpha_{ij}$. Therefore, y is saturated and it does not have children, contradicting the fact that x and y are in an infinite branch of $\mathbf{t}$. Since all the branches are finite and the children of any node are finite, by Kőnig's Lemma, $\mathbf{t}$ is finite, and computable. □

Lemma 6.3. *If* $\mathbf{t}$ *has a final node,* $R \cap G \neq \emptyset$*.*

Proof. If a leaf x is final, consider all the $x.n$ ancestors of x: $x_0, \ldots, x_{n-1}$, such that $x_i.n = i$ for every $i \in [n-1]$. Consider the run $\rho : [0..x.n] \to Q$ defined as $\rho(x.n) = x.q$ and $\rho(i) = x_i.q$ for $i < x.n$. It is easy to see that ρ is an accepting run of $\mathcal{A}$ on $x.w_1 \otimes \ldots \otimes x.w_k$ and therefore

that $((x.w_1)_\Sigma, \dots, (x.w_k)_\Sigma) \in R$. On the other hand, for every $(i,j) \in I$, $(x.w_i)_\Sigma \sqsubseteq (x.w_j)_\Sigma$ since $\alpha_{ij} = \varepsilon$. Hence, $((x.w_1)_\Sigma, \dots, (x.w_k)_\Sigma) \in G$ and thus $R \cap G \neq \emptyset$. □

Lemma 6.4. *If all the leaves of* $\mathbf{t}$ *are saturated,* $R \cap G = \emptyset$.

Proof. By means of contradiction suppose that there is $w = w_1 \otimes \cdots \otimes w_k \in (\Sigma_\perp^k)^*$ such $w \in \mathcal{L}(\mathcal{A})$ through an accepting run $\rho : [0..n] \to Q$, and for every $(i,j) \in I$, $(w_i)_\Sigma \sqsubseteq (w_j)_\Sigma$. Let $|w| = n$ be of minimal size.

By construction of $\mathbf{t}$, the following claims follow.

Claim 6.0.5. There is a maximal branch $x_0, \dots, x_m$ in $\mathbf{t}$ such that $x_\ell.n = \ell$, $x_\ell.w_j = w_j[1..\ell]$, $x_\ell.q = \rho(\ell)$ for every $\ell \in [0..m]$ and $j \in [k]$.

Claim 6.0.6. For every $\ell \in [0..m]$ and $(i,j) \in I$,

$$x_\ell.\alpha_{ij} \cdot (w_i[\ell+1..])_\Sigma \sqsubseteq (w_j[\ell+1..])_\Sigma, \tag{6.1}$$

$$(w_i[1..\ell - |x_\ell.\alpha_{ij}|])_\Sigma \sqsubseteq (w_j[1..\ell])_\Sigma. \tag{6.2}$$

Since we assume that all the leaves of $\mathbf{t}$ are saturated, in particular x_m is saturated and there must be some $m' < m$ such that x_m and $x_{m'}$ verify the saturation conditions.

Consider the following word.

$$w' = w[1..m'] \cdot w[m+1..]$$

The run ρ trimmed with the positions $[m'+1..m]$ is still an accepting run on w' (since $\rho(m') = \rho(m)$), and therefore $((\pi_1(w'))_\Sigma, \dots, (\pi_k(w'))_\Sigma) \in R$.

For an arbitrary $(i,j) \in I$, we show that $(\pi_i(w'))_\Sigma \sqsubseteq (\pi_j(w'))_\Sigma$. First, note that by (6.2) we have that

$$\begin{aligned}(\pi_i(w')[1..m' - |x_{m'}.\alpha_{ij}|])_\Sigma &= (w_i[1..m' - |x_{m'}.\alpha_{ij}|])_\Sigma \\ &\sqsubseteq (w_j[1..m'])_\Sigma && \text{(by (6.2))} \\ &= (\pi_j(w')[1..m'])_\Sigma.\end{aligned}$$

Since $x_{m'}$ and x_m verify the saturation conditions, $x_{m'}.\alpha_{ij} \sqsubseteq x_m.\alpha_{ij}$. Therefore,

$$\begin{aligned}(\pi_i(w')[m' - |x_{m'}.\alpha_{ij}| + 1..])_\Sigma &= (\pi_i(w')[m' - |x_{m'}.\alpha_{ij}| + 1..m'])_\Sigma \cdot (\pi_i(w')[m'+1..])_\Sigma \\ &= x_{m'}.\alpha_{ij} \cdot (w_i[m+1..])_\Sigma \\ &\sqsubseteq x_m.\alpha_{ij} \cdot (w_i[m+1..])_\Sigma && \text{(since } x_{m'}.\alpha_{ij} \sqsubseteq x_m.\alpha_{ij}\text{)} \\ &\sqsubseteq (w_j[m+1..])_\Sigma && \text{(by (6.1))} \\ &= (\pi_j(w')[m'+1..])_\Sigma\end{aligned}$$

Hence, we showed that there are some ℓ, ℓ' such that $(\pi_i(w')[1..\ell])_\Sigma \sqsubseteq (\pi_j(w')[1..\ell'])_\Sigma$ and $(\pi_i(w')[\ell+1..])_\Sigma \sqsubseteq (\pi_j(w')[\ell'+1..])_\Sigma$, for $\ell = m' - |x_{m'}.\alpha_{ij}|$ and $\ell' = m'$. Thus, $(\pi_i(w'))_\Sigma \sqsubseteq (\pi_j(w'))_\Sigma$.

This means that $((\pi_1(w'))_\Sigma, \dots, (\pi_k(w'))_\Sigma) \in G$ and thus $((\pi_1(w'))_\Sigma, \dots, (\pi_k(w'))_\Sigma) \in R \cap G$. But this cannot be since $|w'| < |w|$ and w is of minimal length. The contradiction arises from the assumption that $R \cap G \neq \emptyset$. Then, $R \cap G = \emptyset$. □

Hence, by Lemmas 6.2, 6.3 and 6.4, $R \cap G \neq \emptyset$ if and only if $\mathbf{t}$ has a final node, which is computable. □

Corollary 6.5. *The query evaluation problem for* ECRPQ$(\sqsubseteq)$ *queries is decidable.*

Of course the complexity is extremely high as we already know from Corollary 5.13.

Note that while the intersection problem of $\sqsubseteq$ with rational relations is decidable, as is $\textsc{GenInt}_{\sqsubseteq}(\mathsf{REG})$, we lose the decidability of $\textsc{GenInt}_{\sqsubseteq}(\mathsf{RAT})$ even in the simplest cases that go beyond the intersection problem (that is, for ternary relations in RAT and any I that does not force two words to be the same).

Proposition 6.6. *The problem* $(\mathsf{RAT} \cap_I \sqsubseteq) \stackrel{?}{=} \emptyset$ *is undecidable even over ternary relations when I is one of the following:*
(1) $\{(1,2),(2,3)\}$,
(2) $\{(1,2),(1,3)\}$, *or*
(3) $\{(1,2),(3,2)\}$.

Proof. The three proofs use a reduction from the PCP problem. Recall that this is defined as follows. The input are two equally long lists $u_1, u_2, \ldots, u_n$ and $v_1, v_2, \ldots, v_n$ of strings over alphabet Σ. The PCP problems asks whether there exists a solution for this input, that is, a sequence of indices $i_1, i_2, \ldots, i_k$ such that $1 \leq i_j \leq n$ $(1 \leq j \leq k)$ and $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$.
(1) $\{(1,2),(2,3)\}$: The proof goes by reduction from an arbitrary PCP instance given by lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of strings over alphabet Σ. The following relation

$$R = \{(u_{i_1} \cdots u_{i_m}, v_{i_1} \cdots v_{i_m}, u_{i_1} \cdots u_{i_m}) \mid m \in \mathbb{N} \text{ and } i_1, \ldots, i_m \in [n]\}$$

is rational and $R \cap \{(x,y,z) \mid x \sqsubseteq y \sqsubseteq z\}$ is non-empty if and only if the instance has a solution.
(2) $\{(1,2),(1,3)\}$: The proof again goes by reduction from an arbitrary PCP instance given by lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of strings over alphabet Σ. For simplicity, and without any loss of generality, we assume that $|u_i|, |v_i| \leq 1$ for every i. Let $\hat{\Sigma} = \{\hat{a} \mid a \in \Sigma\}$, and for every $w = a_1 \cdots a_\ell \in \Sigma^*$, let $\hat{w} = \hat{a}_1 \cdots \hat{a}_\ell$. Consider

$$\begin{aligned} R = \{(x,y,z) \mid\ & m \in \mathbb{N},\ i_1, \ldots, i_m \in [n],\ w_1, w'_1, \ldots, w_{m+1}, w'_{m+1} \in \Sigma^*, \\ & x = u_{i_1} \hat{v}_{i_1} u_{i_2} \hat{v}_{i_2} \cdots u_{i_m} \hat{v}_{i_m}, \\ & y = w'_1 \hat{u}_{i_1} w'_2 \cdots w'_m \hat{u}_{i_m} w'_{m+1}, \\ & z = \hat{w}_1 v_{i_1} \hat{w}_2 \cdots \hat{w}_m v_{i_m} \hat{w}_{m+1}\} \end{aligned}$$

which is a rational relation. Note that there is some $(x,y,z) \in R$ with $x \sqsubseteq y$ if and only if there is some $v_{i_1} \cdots v_{i_m} \sqsubseteq u_{i_1} \cdots u_{i_m}$. Similarly for $x \sqsubseteq z$. Therefore, there is $(x,y,z) \in R$ with $x \sqsubseteq y$, $x \sqsubseteq z$ if and only if $v_{i_1} \cdots v_{i_m} = u_{i_1} \cdots u_{i_m}$ for some choice of $i_1, \ldots, i_m$.
(3) $\{(1,2),(3,2)\}$: This is similar to (2), but this time we consider the following rational relation.

$$\begin{aligned} R = \{(x,y,z) \mid\ & m \in \mathbb{N},\ i_1, \ldots, i_m \in [n],\ w_1, w'_1, \ldots, w_{m+1}, w'_{m+1} \in \Sigma^*, \\ & y = u_{i_1} \hat{v}_{i_1} u_{i_2} \hat{v}_{i_2} \cdots u_{i_m} \hat{v}_{i_m}, \\ & x = w'_1 \hat{u}_{i_1} w'_2 \cdots w'_m \hat{u}_{i_m} w'_{m+1}, \\ & z = \hat{w}_1 v_{i_1} \hat{w}_2 \cdots \hat{w}_m v_{i_m} \hat{w}_{m+1}\} \end{aligned}$$

Analogously as before, there is $(x,y,z) \in R$ with $x \sqsubseteq y$, $z \sqsubseteq y$ if and only if the PCP instance has a solution. □

6.2. **Generalized intersection problem for recognizable relations.** We now consider the problem of answering CRPQs with rational relations S, or, equivalently, the problem $\textsc{GenInt}_S(\mathsf{REC})$. Recall that an instance of such a problem consists of an m-ary recognizable relation R and a set $I \subseteq [m]^2$. The question is whether $R \cap_I S \neq \emptyset$, i.e., whether there exists a tuple $(w_1, \ldots, w_m) \in R$ so that $(w_i, w_j) \in S$ whenever $(i, j) \in I$. It turns out that the decidability of this problem hinges on the graph-theoretic properties of I. In fact we shall present a *dichotomy result*, classifying problems $\textsc{GenInt}_S(\mathsf{REC})$ into PSPACE-complete and undecidable depending on the structure of I.

Before stating the result, we need to decide how to represent a recognizable relation R. Recall that an m-ary $R \in \mathsf{REC}$ is a union of relations of the form $L_1 \times \ldots \times L_m$, where each L_i is a regular language. Hence, as the representation of R we take the set of all such L_is involved, and as the measure of its complexity, the total size of NFAs defining the L_is.

With a set $I \subseteq [m]^2$ we associate an *undirected* graph G_I whose nodes are $1, \ldots, m$ and whose edges are $\{i, j\}$ such that either $(i, j) \in I$ or $(j, i) \in I$. We call an instance of $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *acyclic* if G_I is an acyclic graph.

Now we can state the dichotomy result.

Theorem 6.7.

- *Let S be a binary rational relation. Then acyclic instances of* $\textsc{GenInt}_S(\mathsf{REC})$ *are decidable in* PSPACE. *Moreover, there is a fixed binary relation S_0 such that the problem $(\mathsf{REC} \cap_I S_0) \stackrel{?}{=} \emptyset$ is* PSPACE*-complete.*
- *For every I such that G_I is not acyclic, there exists a binary rational relation S such that the problem $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ is undecidable.*

Proof. For PSPACE-hardness we can do an easy reduction from nonemptiness of the intersection of m given NFA's, which is known to be PSPACE-complete [26]. Given m NFAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$, define the (acyclic) relation $I = \{(i, i+1) \mid 1 \leq i < m\}$. Then $\bigcap_i \mathcal{L}(\mathcal{A}_i)$ is nonempty if and only if $\prod_i \mathcal{L}(\mathcal{A}_i) \cap_I S_0 \neq \emptyset$, where S_0 is the regular relation $\{(w, w) \mid w \in \Sigma^*\}$.

For the upper bound, we use the following idea: First we show how to construct, in exponential time, the following for each m-ary recognizable relation R, binary rational relation S and acyclic $I \subseteq [m]^2$: An m-tape automaton $\mathcal{A}(R, S, I)$ that accepts precisely those $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ such that $\bar{w} \in R$ and $(w_i, w_j) \in S$, for each $(i, j) \in I$. Intuitively, $\mathcal{A}(R, S, I)$ represents the "synchronization" of the transducer that accepts R with a copy of the 2-tape automaton that recognizes S over each projection defined by the pairs in I. Such synchronization is possible since I is acyclic. Hence, in order to solve $\textsc{GenInt}_S(\mathsf{REC})$ we only need to check $\mathcal{A}(R, S, I)$ for nonemptiness. The latter can be done in PSPACE by the standard "on-the-fly" reachability analysis. We proceed with the details of the construction below.

Recall that rational relations are the ones defined by n-tape automata. We start by formally defining the class of n-tape automata that we use in this proof. An n-tape automaton, $n > 0$, is a tuple $\mathcal{A} = (Q, \Sigma, Q_0, \delta, F)$, where Q is a finite set of control states, Σ is a finite alphabet, $Q_0 \subseteq Q$ is the set of initial states, $\delta : Q \times (\Sigma \cup \{\varepsilon\})^n \to 2^{Q \times ([n] \cup \{[n]\})}$ is the transition function with ε a symbol not appearing in Σ, and $F \subseteq Q$ is the set of final states. Intuitively, the transition function specifies how $\mathcal{A}$ moves in a situation when it is in state q reading symbol $\bar{a} \in \Sigma^n$: If $(q', j) \in \delta(q, \bar{a})$, where $j \in [n]$, then $\mathcal{A}$ is allowed to enter state q' and move its j-th head one position to the right of its tape. If $(q', [n]) \in \delta(q, \bar{a})$

then $\mathcal{A}$ is allowed to enter state q' and move each one of its heads one position to the right of its tape.

Given a tuple $\bar{w} = (w_1, \ldots, w_n) \in (\Sigma^*)^n$ such that w_i is of length $p_i \geq 0$, for each $1 \leq i \leq n$, a *run* of $\mathcal{A}$ over $\bar{w}$ is a sequence $q_0\, P_0\, q_1\, P_1 \cdots q_{k-1}\, P_{k-1}\, q_k$, for $k \geq 0$, such that:

(1) $q_i \in Q$, for each $0 \leq i \leq k$,
(2) $q_0 \in Q_0$,
(3) P_i is a tuple in $([p_1] \cup \{0\}) \times \cdots \times ([p_n] \cup \{0\})$, for each $0 \leq i \leq k-1$ (intuitively, the P_i's represent the positions of the n heads of $\mathcal{A}$ at each stage of the run. In particular, the j-th component of P_i represents the position of the j-th head of $\mathcal{A}$ in stage i of the run),
(4) $P_0 = (b_1, \ldots, b_n)$, where $b_i := 0$ if w_i is the empty word ε (that is, $p_i = 0$) and $b_i := 1$ otherwise (that is, the run starts by initializing each one of the n heads to be in the initial position of its tape, if possible),
(5) $P_{k-1} = (p_1, \ldots, p_n)$, that is, the run ends when each head scans the last position of its head, and
(6) for each $0 \leq i \leq k-1$, if $P_i = (r_1, \ldots, r_n)$ and
$$\big((\pi_1(\bar{w}))[r_1], \ldots, (\pi_n(\bar{w}))[r_n] \big) \;=\; (a_1, \ldots, a_n),$$
where we assume by definition that $w[0] = \varepsilon$, then $\delta(q_i, (a_1, \ldots, a_n))$ contains a pair of the form (q_{i+1}, j) such that:
   (a) if $i < k-1$ then $j \in [n]$ and P_{i+1} is the tuple $(r_1, \ldots, r_{j-1}, r_j + 1, r_{j+1}, \ldots, r_n)$. In such case we say that (q_{i+1}, P_{i+1}) is a *valid transition from* (q_i, P_i) *over* $\bar{w}$ *in the* j*-th head*, and
   (b) if $i = k-1$ then $j = [n]$. This is a technical condition that ensures that each head of $\mathcal{A}$ should leave its tape after the last transition in the run is performed.

   That is, each run is forced to respect the transition function δ when the n-tape automaton $\mathcal{A}$ is in state q reading the symbols in the corresponding positions of its n heads. Further, the positions of the n heads are updated in the run also according to what is allowed by δ. Notice that each transition in a run moves a single head, except for the last one that moves all of them at the same time.

The run is *accepting* if $q_k \in F$ (that is, $\mathcal{A}$ enters an accepting state after each one of its heads scans the last position of its own tape).

Each n-tape automaton $\mathcal{A}$ defines the language $L(\mathcal{A}) \subseteq (\Sigma^*)^n$ of all those $\bar{w} = (w_1, \ldots, w_n) \in (\Sigma^*)^n$ such that there is an accepting run of $\mathcal{A}$ over $\bar{w}$. It can be proved with standard techniques that languages defined by n-ary rational relations are precisely those defined by n-tape automata. Notice that there is an alternative, more general model of n-tape automata that allows each transition to move an arbitrary number of heads. It is easy to see that this model is equivalent in expressive power to the one we present here, as transitions that move an arbitrary number of heads can easily be encoded by a a series of single-head transitions. We have decided to use this more restricted version of n-tape automata here, as it will allow us simplifying some of the technical details in our proof.

Now we continue with the proof that the problem $\textsc{GenInt}_S(\mathsf{REC})$ can be solved in PSpace if I is acyclic (that is, it defines an acyclic undirected graph). The main technical tool for proving this is the following lemma:

Lemma 6.8. *Let R be an m-ary relation in* REC*, S a binary rational relation, and I a subset of $[m] \times [m]$ that defines an acyclic undirected graph. It is possible to construct,*

in exponential time, an m-tape automaton $\mathcal{A}(R,S,I)$ such that the language defined by $\mathcal{A}(R,S,I)$ is precisely the set of words $\bar{w} = (w_1,\ldots,w_m) \in (\Sigma^*)^m$ such that $\bar{w} \in R$ and $(w_i,w_j) \in S$ for all $(i,j) \in I$.

We start by proving the lemma. The intuitive idea is that $\mathcal{A}(R,S,I)$ is an m-tape automaton that at the same time recognizes R and represents the "synchronization" of the $|I|$ copies of the 2-tape automaton S over the projections corresponding to the pairs in I. Since I is acyclic, such synchronization is possible.

Assume that $|I| = \ell$. Let $t_1,\ldots,t_\ell$ be an arbitrary enumeration of the pairs in I. Also, assume that the recognizable relation R is given as

$$\bigcup_i \mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m},$$

where each $\mathcal{N}_{i_j}$ is an NFA over Σ (without transitions on the empty word). Assume that the set of states of $\mathcal{N}_{i_j}$ is U_{i_j}, its set of initial states is $U^0_{i_j}$ and its set of final states is $U^F_{i_j}$. Further, assume that the 2-tape transducer S is given by the tuple $(Q_S,\Sigma,Q^0_S,\delta_S,Q^F_S)$, where Q_S is the set of states, the set of initial states is Q^0_S, the set of final states is Q^F_S, and $\delta_S : Q_S \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to 2^{Q \times (\{1,2\} \cup \{\{1,2\}\})}$ is the transition function. We take $|I| = \ell$ disjoint copies $S_1,\ldots,S_\ell$ of S, such that S_i, for each $1 \leq i \leq \ell$, is the tuple $(Q_{S_i},\Sigma,Q^0_{S_i},\delta_{S_i},Q^F_{S_i})$. Without loss of generality we assume that if $t_i = (j,j') \in [m] \times [m]$ then δ_{S_i} is a function from $Q_{S_i} \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})$ into $2^{Q \times (\{j,j'\} \cup \{\{j,j'\}\})}$. We can do this because I is acyclic, and hence $j \neq j'$.

The m-tape automaton $\mathcal{A}(R,S,I)$ is defined as the tuple (Q,Σ,Q_0,δ,F), where:

(1) The set of states Q is

$$\bigcup_i \big(U_{i_1} \times \cdots \times U_{i_m} \times Q_{S_1} \times \cdots \times Q_{S_\ell}\big).$$

(2) The initial states in Q_0 are precisely those in

$$\bigcup_i \big(U^0_{i_1} \times \cdots \times U^0_{i_m} \times Q^0_{S_1} \times \cdots \times Q^0_{S_\ell}\big).$$

(3) The final states in F are precisely those in

$$\bigcup_i \big(U^F_{i_1} \times \cdots \times U^F_{i_m} \times Q^F_{S_1} \times \cdots \times Q^F_{S_\ell}\big).$$

(4) The transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\})^m \to 2^{Q \times ([m] \cup \{[m]\})}$ is defined as follows on state $\bar{q} \in Q$ and symbol $\bar{a} \in (\Sigma \cup \{\varepsilon\})^m$. Assume that $\bar{q} = (u_{i_1},\ldots,u_{i_m},q_1,\ldots,q_\ell)$, where $u_{i_j} \in U_{i_j}$ for each $1 \leq j \leq m$, and $q_j \in Q_{S_j}$ for each $1 \leq j \leq \ell$. Further, assume that $\bar{a} = (a_1,\ldots,a_m)$, where $a_j \in (\Sigma \cup \{\varepsilon\})$ for each $1 \leq j \leq m$. Then $\delta(\bar{q},\bar{a})$ consists of all pairs of the form $\big((u'_{i_1},\ldots,u'_{i_m},q'_1,\ldots,q'_\ell),\, j\,\big)$, for $j \in [m]$, such that:
   (a) $u'_{i_k} = u_{i_k}$ for each $k \in [m] \setminus \{j\}$, and there is a transition in $\mathcal{N}_{i_j}$ from u_{i_j} into u'_{i_j} labeled a_j; and
   (b) for each $1 \leq k \leq \ell$, if t_k is the pair $(k_1,k_2) \in [m] \times [m]$ then the following holds: (1) If $j \notin \{k_1,k_2\}$ then $q_k = q'_k$, and (2) if $j \in \{k_1,k_2\}$ then (q'_k,j) belongs to $\delta_{S_k}(q_k,(a_{k_1},a_{k_2}))$,

   plus all pairs of the form $\big((u'_{i_1},\ldots,u'_{i_m},q'_1,\ldots,q'_\ell),\,[m]\,\big)$ such that:
   (a) for each $1 \leq k \leq m$ there is a transition in $\mathcal{N}_{i_k}$ from u_{i_k} into u'_{i_k} labeled a_k; and

(b) for each $1 \leq k \leq \ell$, if t_k is the pair $(k_1, k_2) \in [m] \times [m]$ then $(q'_k, \{\{k_1, k_2\}\})$ belongs to $\delta_{S_k}(q_k, (a_{k_1}, a_{k_2}))$.

Intuitively, δ defines possible transitions of $\mathcal{A}(R, S, I)$ that respect the transition function of each one of the copies of S over its respective projection. Further, while scanning its tapes the automaton $\mathcal{A}(R, S, I)$ also checks that there is an i such that for each $1 \leq j \leq m$ the j-th tape contains a word in the language defined by $\mathcal{N}_{i_j}$.

Clearly, $\mathcal{A}(R, S, I)$ can be constructed in exponential time from R, S and I. Notice, however, that states of $\mathcal{A}(R, S, I)$ are of polynomial size.

We prove next that for every $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ it is the case that $\bar{w}$ is accepted by $\mathcal{A}(R, S, I)$ if and only if $\bar{w}$ belongs to the language of R and $(w_i, w_j) \in S$, for each $(i, j) \in I$.

$\Longrightarrow$) Assume first that $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ is accepted by $\mathcal{A}(R, S, I)$. It is easy to see from the way $\mathcal{A}(R, S, I)$ is defined that, for some i, the projection of the accepting run of $\mathcal{A}(R, S, I)$ on each $1 \leq j \leq m$ defines an accepting run of $\mathcal{N}_{i_j}$ over w_j. Further, for each $(j, k) \in I$ it is the case that the projection of the accepting run of $\mathcal{A}(R, S, I)$ on (j, k) defines an accepting run of S over (w_j, w_k). We conclude that $\bar{w}$ belongs to the language of R and $(w_j, w_k) \in S$, for each $(j, k) \in I$.

$\Longleftarrow$) Assume, on the other hand, that $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ belongs to the language of R and $(w_i, w_j) \in S$, for each $(i, j) \in I$. Further, assume that the length of w_i is $p_i \geq 0$, for each $1 \leq i \leq m$. We prove next that $\bar{w}$ is accepted by $\mathcal{A}(R, S, I)$.

Since $\bar{w} \in R$ it must be the case that $\bar{w}$ is accepted by $\mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m}$, for some i. Let us assume that

$$\rho_{i_j} \;:=\; u_{i_j,0}\,(1)\,u_{i_j,1}\,(2)\,\cdots\,u_{i_j,p_j-1}\,(p_j)\;u_{i_j,p_j}$$

is an accepting run of the 1-tape automaton $\mathcal{N}_{i_j}$ over w_j, for each $1 \leq j \leq m$. Since for every t_j $(1 \leq j \leq \ell)$ of the form $(k, k') \in [m] \times [m]$ it is the case that $(w_k, w_{k'}) \in S$, there is an accepting run

$$\lambda_j \;:=\; q_{j,0}\,P_{j,0}\,q_{j,1}\,P_{j,1}\,\cdots\,q_{j,r_j}\,P_{j,r_j}\,q_{j,r_j+1}$$

of S_j over $(w_k, w_{k'})$. We then inductively define a sequence

$$\bar{q}_0\,P_0\,\bar{q}_1\,P_1\,\cdots$$

where each $\bar{q}_j$ is a state of Q and each P_j is a tuple in $([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, as follows:

(1) $\bar{q}_0 := (u_{i_1,0}, \ldots, u_{i_m,0}, q_{1,0}, \ldots, q_{\ell,0})$.
(2) $P_0 = (b_1, \ldots, b_m)$, where $b_i := 0$ if w_i is the empty word and $b_i := 1$ otherwise.
(3) Let $j \geq 0$. Assume that $\bar{q}_j = (u_{i_1}, \ldots, u_{i_m}, q_1, \ldots, q_\ell)$, where each u_{i_k} is a state in $\mathcal{N}_{i_k}$ and each q_k is a state in S_k, and that $P_j = (r_1, \ldots, r_m) \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$.

If for every $1 \leq k \leq m$ it is the case that $r_k = p_k$ then the sequence stops. Otherwise it proceeds as follows.

If for some $1 \leq k \leq m$ it is the case that $u_{i_k}(r_k)$ is not a subword of the accepting run ρ_{i_k},[2] or that for some $1 \leq k \leq \ell$ such that $t_k = (k_1, k_2) \in [m] \times [m]$ it is the case that $q_k(r_{k_1}, r_{k_2})$ is not a subword of the accepting run λ_k,[3] then the sequence simply fails.

[2]Notice that ρ_{i_k} is a word in the language defined by $(U_{i_k} \cdot [p_k])^* \cdot U_{i_k}$, and hence it is completely well-defined whether a word in $U_{i_k} \cdot [p_k]$ is or not a subword of ρ_{i_k}.

[3]This is well-defined for essentially the same reasons given in the previous footnote.

Otherwise check whether there is a $1 \leq k \leq m$ such that the following holds:

(a) $r_k \neq p_k$.
(b) For each pair $t_{k_1} \in I$ of the form $(k, k') \in [m] \times [m]$ it is the case that if $q'_{k_1}(r'_k, r'_{k'})$ is the subword in $Q_{S_{k_1}} \cdot ([p_k] \times [p_{k'}])$ that immediately follows $q_{k_1}(r_k, r_{k'})$ in the run λ_{k_1},[4] then $r'_k = r_k + 1$, and $r'_{k'} = r_{k'}$.
(c) For each pair $t_{k_1} \in I$ of the form $(k', k) \in [m] \times [m]$ it is the case that if $q'_{k_1}(r'_{k'}, r'_k)$ is the subword in $Q_{S_{k_1}} \cdot ([p_{k'}] \times [p_k])$ that immediately follows $q_{k_1}(r_{k'}, r_k)$ in the run λ_{k_1}, then $r'_k = r_k + 1$, and $r'_{k'} = r_{k'}$.

Intuitively, this states that we can move the k-th head of $\mathcal{A}(R, S, I)$ and preserve the transitions on each run of the form λ_{k_1} such that S_{k_1} is a copy of S that has one of its components reading tape k.

If no such k exists the sequence fails. Otherwise pick the least $1 \leq k \leq m$ that satisfies the conditions above, and continue the sequence by defining the pair $(\bar{q}_{j+1}, P_{j+1})$ as

$$\big((u_{i_1}, \cdots, u_{i_{k-1}}, u'_{i_k}, u_{i_{k+1}}, \cdots, u_{i_m}, q'_1, \cdots, q'_\ell), (r_1, \cdots, r_{k-1}, r_k + 1, r_{k+1}, \cdots, r_m) \big),$$

where the following holds:

(a) $u'_{i_k}(r_k + 1)$ is the subword in $U_{i_k} \cdot [p_k]$ that immediately follows $u_{i_k}(r_k)$ in ρ_{i_k}.
(b) For each pair $t_{k_1} \in I$ of the form $(k, k') \in [m] \times [m]$, it is the case that q'_{k_1} satisfies that $q'_{k_1}(r_k + 1, r_{k'})$ is the subword in $Q_{S_{k_1}} \cdot ([p_k] \times [p_{k'}])$ that immediately follows $q_{k_1}(r_k, r_{k'})$ in the run λ_{k_1}.
(c) For each pair $t_{k_1} \in I$ of the form $(k', k) \in [m] \times [m]$, it is the case that q'_{k_1} satisfies that $q'_{k_1}(r_{k'}, r_k + 1)$ is the subword in $Q_{S_{k_1}} \cdot ([p_{k'}] \times [p_k])$ that immediately follows $q_{k_1}(r_{k'}, r_k)$ in the run λ_{k_1}.
(d) For each pair $t_{k_1} \in I$ of the form $(k', k'') \in [m] \times [m]$ such that $k' \neq k$ and $k'' \neq k$, it is the case that $q'_{k_1} = q_{k_1}$.

In this case we say that $(\bar{q}_{j+1}, P_{j+1})$ is *obtained from* $(\bar{q}_j, P_j)$ *by performing a transition on the k-th head.*

We first prove by induction the following crucial property of the sequence $\bar{q}_0 P_0 \bar{q}_1 P_1 \cdots$: The sequence does not fail at any stage $j \geq 0$. Clearly, the sequence does not fail in stage 0 given by pair $(\bar{q}_0, P_0)$. Assume now by induction that the sequence has not failed until stage $j \geq 0$ given by pair $(\bar{q}_j, P_j)$, and, further, that the sequence does not stop in stage j. We prove next that the sequence does not fail in stage $j + 1$.

If the sequence stops in stage $j + 1$ it clearly does not fail. Assume then that the sequence does not stop in stage $(j + 1)$. Also, assume that $q_j = (u_{i_1}, \ldots, u_{i_m}, q_1, \ldots, q_\ell)$, where each u_{i_k} is a state in $\mathcal{N}_{i_k}$ and each q_k is a state in S_k. Further, assume that $P_j = (r_1, \ldots, r_m) \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$. Since the sequence did not stop in stage j it must be the case that for every $1 \leq k \leq m$ the sequence $u_{i_k}(r_k)$ is a subword of the accepting run ρ_{i_k}, and that for every $1 \leq k \leq \ell$ such that $t_k = (k_1, k_2) \in [m] \times [m]$ the sequence $q_k(r_{k_1}, r_{k_2})$ is a subword of the accepting run λ_k.

Assume that $(\bar{q}_{j+1}, P_{j+1})$ is obtained from $(\bar{q}_j, P_j)$ by performing a transition on the k-th head, for $1 \leq k \leq m$. Then the pair $(\bar{q}_{j+1}, P_{j+1})$ is of the form:

$$\big((u'_{i_1}, \cdots, u'_{i_k}, \cdots, u'_{i_m}, q'_1, \cdots, q'_\ell), (r'_1, \cdots, r'_k, \cdots, r'_m) \big),$$

[4]Notice, since $\mathcal{A}(R, S, I)$ does not allow empty transitions, that $q'_{k_1}(r'_k, r'_{k'})$ is well-defined since the subword $q_{k_1}(r_k, r_{k'})$ appears exactly once in the run λ_{k_1} and, further, $q_{k_1}(r_k, r_{k'})$ is followed in λ_{k_1} by a subword in $Q_{S_{k_1}} \cdot ([p_k] \times [p_{k'}])$ because $r_k \neq p_k$.

where the following holds:

(1) $u'_{i_{k'}} = u_{i_{k'}}$, for each $k' \in [m] \setminus \{k\}$,
(2) $u'_{i_k}(r_k + 1)$ is the subword in $U_{i_k} \cdot [p_k]$ that immediately follows $u_{i_k}(r_k)$ in ρ_{i_k},
(3) $r'_{k'} = r_{k'}$, for each $k' \in [m] \setminus \{k\}$,
(4) $r'_k = r_k + 1$,
(5) for each pair $t_{k_1} \in I$ of the form $(k, k') \in [m] \times [m]$, it is the case that q'_{k_1} satisfies that $q'_{k_1}(r_k + 1, r_{k'})$ is the subword in $Q_{S_{k_1}} \cdot ([p_k] \times [p_{k'}])$ that immediately follows $q_{k_1}(r_k, r_{k'})$ in the run λ_{k_1},
(6) for each pair $t_{k_1} \in I$ of the form $(k', k) \in [m] \times [m]$, it is the case that q'_{k_1} satisfies that $q'_{k_1}(r_{k'}, r_k + 1)$ is the subword in $Q_{S_{k_1}} \cdot ([p_{k'}] \times [p_k])$ that immediately follows $q_{k_1}(r_{k'}, r_k)$ in the run λ_{k_1}, and
(7) for each pair $t_{k_1} \in I$ of the form $(k', k'') \in [m] \times [m]$ such that $k' \neq k$ and $k'' \neq k$, it is the case that $q'_{k_1} = q_{k_1}$.

Then, by inductive hypothesis, it is the case that for every $k' \in [m] \setminus \{k\}$ the sequence $u'_{i_{k'}}(r'_{k'})$ is a subword of the accepting run $\rho_{i_{k'}}$. For the same reason, for every $1 \leq k' \leq \ell$ such that $t_{k'} = (k_1, k_2) \in [m] \times [m]$, $k_1 \neq k$ and $k_2 \neq k$, it is the case that $q'_{k'}(r'_{k_1}, r'_{k_2})$ is a subword of the accepting run $\lambda_{k'}$. Further, simply by definition $u'_{i_k}(r'_k)$ is a subword of the accepting run ρ_{i_k}. Also, by definition, for each pair $t_{k_1} \in I$ of the form $(k', k) \in [m] \times [m]$, it is the case that $q'_{k_1}(r'_{k'}, r'_k)$ is a subword of the accepting run λ_{k_1}, and, similarly, for each pair $t_{k_1} \in I$ of the form $(k, k') \in [m] \times [m]$, it is the case that $q'_{k_1}(r'_k, r'_{k'})$ is a subword of the accepting run λ_{k_1}. Hence, in order to prove that the sequence does not fail in stage $j + 1$ it is enough to show that there is an $1 \leq h \leq m$ such that some pair of the form $(\bar{q}, P)$, where $\bar{q} \in Q$ and $P \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, can be obtained from $(\bar{q}_{j+1}, P_{j+1})$ by performing a transition on the h-th head.

Since the sequence does not stop in stage $j + 1$, the set $\mathcal{H} = \{1 \leq h' \leq m \mid r'_{h'} \neq p_{h'}\}$ must be nonempty. Let h_1 be the least element in $\mathcal{H}$. Since the underlying undirected graph of I is acyclic, the connected component of I to which h_1 belongs is a tree T. Without loss of generality we assume that T is rooted at h_1.

We start by trying to prove that there is pair of the form $(\bar{q}, P)$, where $\bar{q} \in Q$ and $P \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, that can be obtained from $(\bar{q}_{j+1}, P_{j+1})$ by performing a transition on the h_1-th head. If this is the case we are done and the proof finishes. Assume otherwise. Then we can assume without loss of generality that there is a pair of the form $t_{k'} \in I$ of the form $(h_1, h_2) \in [m] \times [m]$ such that the subword in $Q_{S_{k'}} \cdot ([p_{h_1}] \times [p_{h_2}])$ that immediately follows $q'_{k'}(r'_{h_1}, r'_{h_2})$ in the run $\lambda_{k'}$ is of the form $q''_{k'}(r'_{h_1}, r'_{h_2} + 1)$. (That is, the run $\lambda_{k'}$ continues from $q'_{k'}(r'_{h_1}, r'_{h_2})$ by moving its second head). The other possibility is that there is a pair of the form $t_{k''} \in I$ of the form $(h_2, h_1) \in [m] \times [m]$ such that the subword in $Q_{S_{k''}} \cdot ([p_{h_2}] \times [p_{h_1}])$ that immediately follows $q'_{k''}(r'_{h_2}, r'_{h_1})$ in the run $\lambda_{k''}$ is of the form $q''_{k''}(r'_{h_2} + 1, r'_{h_1})$. But this case is completely symmetric to the previous one.

We then continue by trying to show that there is pair of the form $(\bar{q}, P)$, where $\bar{q} \in Q$ and $P \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, that can be obtained from $(\bar{q}_{j+1}, P_{j+1})$ by performing a transition on the h_2-th head. If this is the case then we are ready and the proof finishes. Assume otherwise. Then again we can assume without loss of generality that there is a pair of the form $t_{k''} \in I$ of the form $(h_2, h_3) \in [m] \times [m]$ such that the subword in $Q_{S_{k''}} \cdot ([p_{h_2}] \times [p_{h_3}])$ that immediately follows $q'_{k''}(r'_{h_2}, r'_{h_3})$ in the run $\lambda_{k''}$ is of the form

$q''_{k''}(r'_{h_2}, r'_{h_3}+1)$. (That is, the run $\lambda_{k''}$ continues from $q'_{k''}(r'_{h_2}, r'_{h_3})$ by moving its second head).

Since T is acyclic and finite, if we iteratively continue in this way from h_2 we will either have to find some $h \in \mathcal{H}$ such that there is pair of the form $(\bar{q}, P)$, where $\bar{q} \in Q$ and $P \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, that can be obtained from $(\bar{q}_{j+1}, P_{j+1})$ by performing a transition on the h-th head, or we will have to stop in some $h \in \mathcal{H}$ that is a leaf in T. But clearly for this h it must be possible to show that there is pair of the form $(\bar{q}, P)$, where $\bar{q} \in Q$ and $P \in ([p_1] \cup \{0\}) \times \cdots \times ([p_m] \cup \{0\})$, that can be obtained from $(\bar{q}_{j+1}, P_{j+1})$ by performing a transition on the h-th head. This shows that the sequence does not fail in stage $j+1$.

We now continue with the proof of the first part of the theorem. Since the sequence does not fail, and from stage j into stage $j+1$ the position of at least one head moves to the right of its tape, the sequence must stop in some stage $j \geq 0$ with associated pair $(\bar{q}_j, P_j)$. Then $P_j = (p_1, \ldots, p_m)$. Assume that $\bar{q}_j = (u_{i_1}, \ldots, u_{i_m}, q_1, \ldots, q_\ell)$, where each u_{i_k} is a state in $\mathcal{N}_{i_k}$ and each q_k is a state in S_k. Then, from the properties of the sequence, it must be the case that $u_{i_k}(p_k)$ appears as a subword in the accepting run ρ_{i_k}, for each $1 \leq k \leq m$, and for each $1 \leq k \leq \ell$ such that $t_k = (k_1, k_2) \in [m] \times [m]$ it is the case that $q_k(p_{k_1}, p_{k_2})$ appears as a subword in the accepting run λ_k. Hence $u_{i_k} = u_{i_k, p_k - 1}$ and $q_k = q_{k, r_k}$.

It easily follows from the definition of the sequence $(\bar{q}_0, P_0)(\bar{q}_1, P_1) \cdots$ and the transition function δ of $\mathcal{A}(R, S, I)$, that the following holds for each $k < j$: If $(\bar{q}_{k+1}, P_{k+1})$ is obtained from $(\bar{q}_k, P_k)$ by performing a transition on the k'-the head, $1 \leq k' \leq m$, then $(\bar{q}_{k+1}, P_{k+1})$ is a valid transition from $(\bar{q}_k, P_k)$ over $\bar{w}$ in the k'-th head. Further, assume that

$$\bar{a} \;=\; \big((\pi_1(\bar{w}))[p_1], \ldots, (\pi_n(\bar{w}))[p_n] \big),$$

then $\delta(\bar{q}_j, \bar{a})$ contains a pair of the form $(\bar{q}_{j+1}, \{[m]\})$, where:

$$\bar{q}_{j+1} \;:=\; \big(u_{i_1, p_1}, \cdots, u_{i_m, p_m}, q_{1, r_1+1}, \cdots, q_{\ell, r_\ell+1} \big).$$

Clearly, $\bar{q}_{j+1} \in F$ (that is, $\bar{q}_{j+1}$ is a final state of $\mathcal{A}(R, S, I)$) and we conclude that $\bar{q}_0 P_0 \bar{q}_1 P_1 \cdots \bar{q}_j P_j \bar{q}_{j+1}$ is an accepting run of $\mathcal{A}(R, S, I)$ over $\bar{w}$, which was to be proved.

We now explain how Theorem 6.7 follows from Lemma 6.8. The lemma tells us that in order to solve acyclic instances of $\textsc{GenInt}_S(\mathsf{REC})$ we can construct, from the m-ary recognizable relation R, the binary rational relation S and the acyclic $I \subseteq [m] \times [m]$, the m-tape automaton $\mathcal{A}(R, S, I)$, and then check $\mathcal{A}(R, S, I)$ for nonemptiness. The latter can be done in polynomial time in the size of $\mathcal{A}(R, S, I)$ by performing a simple reachability analysis in the states of $\mathcal{A}(R, S, I)$. This gives us a simple exponential time bound for the complexity of solving acyclic instances of $\textsc{GenInt}_S(\mathsf{REC})$. However, as we mentioned before, each state in $\mathcal{A}(R, S, I)$ is of polynomial size. Thus, checking whether $\mathcal{A}(R, S, I)$ is nonempty can be done in nondeterministic PSpace by using a standard "on-the-fly" construction of $\mathcal{A}(R, S, I)$ as follows: Whenever the reachability algorithm for checking emptiness of $\mathcal{A}(R, S, I)$ wants to move from a state r_1 of $\mathcal{A}(R, S, I)$ to a state r_2, it guesses r_2 and checks whether there is a transition from r_1 to r_2. Once this is done, the algorithm can discard r_1 and follow from r_2. Thus, at each step, the algorithm needs to keep track of at most two states, each one of polynomial size. From Savitch's theorem, we know that PSpace equals nondeterministic PSpace. This shows that acyclic instances of $\textsc{GenInt}_S(\mathsf{REC})$ can be solved in PSpace.

The proof of the second part of the theorem is by an easy reduction from the PCP problem (e.g. in the style of the proof of the second part of Theorem 6.10). □

6.3. CRPQ**s with rational relations.** The acyclicity condition gives us a robust class of queries, with an easy syntactic definition, that can be extended with *arbitrary* rational relations. Note that acyclicity is a very standard restriction imposed on database queries to achieve better behavior, often with respect to complexity; it is in general known to be easy to enforce syntactically, and to yield benefits from both the semantics and query evaluation point of view. This is the approach we follow here.

Recall that CRPQ(S) queries are those of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}\, \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i : L_i} u_i') \quad \wedge \quad \bigwedge_{(i,j)\in I} S(\chi_i, \chi_j) \Big),$$

see (4.2) in Sec.4. We call such a query *acyclic* if G_I, the underlying undirected graph of I, is acyclic.

Theorem 6.9. *The query evaluation problem for acyclic* CRPQ*(S) queries is decidable for every binary rational relation S. Its combined complexity is* PSpace*-complete, and data complexity is* NLogSpace*-complete.*

Proof. We provide a nondeterministic PSpace algorithm that solves the query evaluation problem when we assume the query to be part of the input (i.e. combined complexity). Then the result will follow from Savitch's theorem, that states that PSpace equals nondeterministic PSpace.

Given a graph G, a tuple $\bar{a}$ of nodes, and acyclic CRPQ(S) query of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}\, \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\rho_i : L_i} u_i') \quad \wedge \quad \bigwedge_{(i,j)\in I} S(\rho_i, \rho_j) \Big),$$

the algorithm starts by guessing a polynomial size assignment $\bar{b}$ for the existentially quantified variables of $\varphi(\bar{x})$, that is, the variables in $\bar{y}$. It then checks that $G \models \psi(\bar{a}, \bar{b})$, assuming that $\psi(\bar{x}, \bar{y})$ is the CRPQ(S) formula

$$\Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\rho_i : L_i} u_i') \quad \wedge \quad \bigwedge_{(i,j)\in I} S(\rho_i, \rho_j) \Big).$$

If this is the case the algorithm accepts and declares that $G \models \varphi(\bar{a})$. Otherwise it rejects and declares that $G \not\models \varphi(\bar{a})$.

By using essentially the same techniques as in the proof of Lemma 4.1, one can show that there is a polynomial time translation that, given G and $\psi(\bar{a}, \bar{b})$, constructs an acyclic instance of GenInt$_S$(REC) such that the answer to this instance is 'yes' iff $G \models \psi(\bar{a}, \bar{b})$. From Theorem 6.7 we know that acyclic instances of GenInt$_S$(REC) can be solved in PSpace, and hence that the algorithm described above can be performed in nondeterministic PSpace.

With respect to the data complexity, we start with the following observation. Acyclic instances of GenInt$_S$(REC) can be solved in NLogSpace for m-ary relations in REC, if we assume m to be fixed. The proof of this fact mimicks the proof of the PSpace upper bound in Theorem 6.7, but this time we assume the arity of R to be fixed. In such case $\mathcal{A}(R, S, I)$

is of polynomial size, and each one of its states is of logarithmic size. We can easily check $\mathcal{A}(R, S, I)$ for nonemptiness in NLogSpace in this case, by performing a standard "on-the-fly" reachability analysis.

We provide an NLogSpace algorithm that solves the query evaluation problem when we assume the query to be fixed (i.e. data complexity). Consider a fixed acyclic CRPQ(S) query of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}\, \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\rho_i : L_i} u'_i) \;\; \wedge \;\; \bigwedge_{(i,j)\in I} S(\rho_i, \rho_j) \Big).$$

Given a graph G and tuple $\bar{a}$ of nodes, the algorithm constructs (using the proof of Lemma 4.1) in deterministic logarithmic space an acyclic instance of GenInt$_S$(REC), given by recognizable relation R of *fixed* arity m (this follows from the fact that $\varphi(\bar{x})$ is fixed), and fixed $I \subseteq [m] \times [m]$, such that the answer to this instance is 'yes' iff $G \models \varphi(\bar{a})$. Since the arity of R is fixed, our previous observation tells us that we can solve the instance of GenInt$_S$(REC) given by R and I in NLogSpace. But NLogSpace reductions compose, and hence the data complexity of the query evaluation problem for CRPQ(S) queries is also NLogSpace. □

Thus, we get not only the possibility of extending CRPQs with rational relations but also a good complexity of query evaluation. The NLogSpace-data complexity matches that of RPQs, CRPQs, and ECRPQs [16, 17, 4], and the combined complexity matches that of first-order logic, or ECRPQs without extra relations.

The next natural question is whether we can recover decidability for weaker syntactic conditions by putting restrictions on a class of relations S. The answer to this is positive if we consider *directed* acyclicity of I, rather than acyclicity of the underlying undirected graph of I. Then we get decidability for the class of SCR relations. In fact, we have a dichotomy similar to that of Theorem 6.7.

Theorem 6.10.

- *Let S be a relation from* SCR. *Then* $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *is decidable in* NExptime *if I is a directed acyclic graph.*
- *There is a relation I with a directed cycle and $S \in$ SCR such that* $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *is undecidable.*

Proof. We start by proving the first item. In order to do that, we first prove a small model property for the size of the witnesses of the instances in $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$, when S is a relation in SCR and I is a DAG. Let R be an m-ary recognizable relation, $m > 0$, and $I \subseteq [m] \times [m]$ that defines a DAG. Assume that both R and S are over Σ. Then the following holds: Assume $R \cap_I S \neq \emptyset$. There is $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ of at most exponential size that is accepted by R and such that $(w_i, w_j) \in S$, for each $(i, j) \in I$. We prove this small model property by applying usual cutting techniques.

Assume that R is given as

$$\bigcup_i \mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m},$$

where each $\mathcal{N}_{i_j}$ is an NFA over Σ. Further, assume that S is given as one of the 2-tape NFAs used in the PSpace upper bound of Theorem 6.7. That is, S defined by the tuple $(Q_S, \Sigma, Q_S^0, \delta_S, Q_S^F)$, where Q_S is the set of states, the set of initial states is Q_S^0, the set of

final states is Q_S^F, and $\delta_S : Q_S \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to 2^{Q \times (\{1,2\} \cup \{\{1,2\}\})}$ is the transition function. Assume also that there is $\bar{u} = (u_1, \ldots, u_m) \in (\Sigma^*)^m$ that is accepted by R such that $(u_i, u_j) \in S$, for each $(i,j) \in I$. Then $\bar{u}$ is accepted by $\mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m}$, for some i.

Since I is a DAG it has a topological order on $[m]$. We assume without loss of generality that such topological order is precisely the linear order on $[m]$. We prove the following invariant on $1 \leq \ell \leq m$: There exists $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ such that (1) $\bar{w}$ is accepted by R, (2) $(w_j, w_k) \in S$, for each $(j,k) \in I$, and (3) each $w_{\ell'}$ with $\ell' \leq \ell$ is of at most exponential size. Clearly this proves our small model property on $\ell = m$. The proof is by induction.

The basis case is $\ell = 1$. We start from $\bar{u}$ and "cut" its first component in order to satisfy the invariant. By using standard pumping techniques it is possible to show that there is a subsequence w_1 of u_1 of size at most $O(|\mathcal{N}_{i_1}|)$ that is accepted by $\mathcal{N}_{i_1}$. Clearly the tuple $(w_1, u_2, \ldots, u_m)$ belongs to R. Further, for each pair of the form $(1,j)$ in I it is the case that $(w_1, u_j) \in S$. This is the case because $(u_1, u_j) \in S$, $u_1 \sqsubseteq w_j$ and $S \in \mathsf{SCR}$. Notice that we do not need to consider pairs of the form $(j,1)$ since we are assuming that the linear order on $[m]$ is a topological order of I. This implies that $(w_1, u_2, \ldots, u_m)$ satisfies our invariant on $\ell = 1$.

Assume now that the invariant holds for $\ell < m$. Then there exists $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ such that (1) $\bar{w}$ is accepted by R, (2) $(w_j, w_k) \in S$, for each $(j,k) \in I$, and (3) each $w_{\ell'}$ with $\ell' \leq \ell$ is of at most exponential size. We proceed to "cut" $w_{\ell+1}$ while preserving the invariant. Let $I(\ell+1)$ be $\{1 \leq j \leq \ell \mid (j, \ell+1) \in I\}$. Let ρ_j be an accepting run of S over $(w_j, w_{\ell+1})$, for each $j \in I(\ell+1)$. Further, let $\mathcal{P}$ be the set of all positions $1 \leq k \leq |w_{\ell+1}|$ such that for some $j \in I(\ell+1)$ the accepting run ρ_j contains a subword of the form $q\,(k', k)\,q'\,(k'+1, k)$, where $q, q' \in Q_S$ and $1 \leq k' \leq |w_j|$. That is, $\mathcal{P}$ defines the set of positions over $w_{\ell+1}$, in which the accepting run ρ_j of S over $(w_j, w_{\ell+1})$, for some $j \in I(\ell+1)$, makes a move on the head positioned over w_j. Intuitively, these are the positions of $w_{\ell+1}$ that should not be "cut" in order to maintain the invariant. Notice that the size of $\mathcal{P}$ is bounded by $s := \Sigma_{1 \leq \ell' \leq \ell} |w_{\ell'}|$, and hence from the inductive hypothesis the size of $\mathcal{P}$ is exponentially bounded.

By using standard pumping techniques it is possible to show that there is a subsequence $w'_{\ell+1}$ of $w_{\ell+1}$ of size at most $|\mathcal{N}_{i_{\ell+1}}| \cdot |\mathcal{P}| \cdot |I(\ell+1)| \cdot |Q_S| \cdot |\Sigma| + 2$, such that $w'_{\ell+1}$ is accepted by $\mathcal{N}_{i_{\ell+1}}$ and $(w_j, w'_{\ell+1})$ is accepted by S, for each $j \in I(\ell+1)$. Assume this is not the case, and that the shortest subsequence $w'_{\ell+1}$ of $w_{\ell+1}$ that satisfies this condition is of length strictly bigger than $|\mathcal{N}_{i_{\ell+1}}| \cdot |\mathcal{P}| \cdot |I(\ell+1)| \cdot |Q_S| \cdot |\Sigma| + 2$. Then there exist two positions $1 \leq i < j \leq |w_{\ell+1}|$ such that (i) $k \notin \mathcal{P}$, for each $i \leq k \leq j$, (ii) the labels of i and j in $w_{\ell+1}$ coincide, (iii) the run ρ_s assigns the same state to both i and j, for each $s \in I(\ell+1)$, and (iv) some accepting run of $\mathcal{N}_{\ell+1}$ assigns the same state to both i and j. Let $w''_{\ell+1}$ be the subsequence of $w'_{\ell+1}$ that is obtained by cutting all positions $i \leq k \leq j-1$. Clearly, $w''_{\ell+1}$ is shorter than $w'_{\ell+1}$ and is accepted by $\mathcal{N}_{\ell+1}$. Further, $(w_s, w''_{\ell+1})$ is accepted by S, for every $s \in I(\ell+1)$. This is because $(w_s, w'_{\ell+1})$ is invariant with respect to the accepting run ρ_s, for each $s \in I(\ell+1)$, as the cutting does not include elements in $\mathcal{P}$ (that is, we only cut elements in which ρ_s does not need to synchronize with the head positioned over w_s) and ρ_s assigns the same state to both i and j, which have, in addition, the same label. This is a contradiction.

We claim that $\bar{w}' = (w_1, \ldots, w_\ell, w'_{\ell+1}, w_{\ell+2}, \cdots, w_m) \in (\Sigma^*)^m$ satisfies the invariant. Clearly, $\bar{w}'$ is accepted by R since $w'_{\ell+1}$ is accepted by $\mathcal{N}_{i_{\ell+1}}$ and, by inductive hypothesis, w_j

is accepted by $\mathcal{N}_{i_j}$, for each $j \in [m] \setminus \{\ell+1\}$. Further, simply by definition it is the case that $(w_j, w'_{\ell+1}) \in S$, for each $j \in I(\ell+1)$. Moreover, $(w'_{\ell+1}, w_j) \in S$, for each $(\ell+1, j) \in I$, simply because $w'_{\ell+1} \sqsubseteq w_{\ell+1}$ and $S \in \mathsf{SCR}$. The remaining pairs in I are satisfied by induction hypothesis. Finally, $w'_{\ell+1}$ is of size at most $O(|\mathcal{N}_{i_{\ell+1}}| \cdot |\mathcal{P}| \cdot |I(\ell+1)| \cdot |Q_S| \cdot |\Sigma|)$, and hence, by inductive hypothesis, it is of size at most exponential. By inductive hypothesis, each $w_{\ell'}$ with $\ell' \leq \ell$ is of size at most exponential.

It is now simple to prove the first part of the theorem using the small model property. In fact, in order to check whether $R \cap_I S \neq \emptyset$, for $S \in \mathsf{SCR}$, we only need to guess an exponential size witness $\bar{w}$, and then check in polynomial time that it satisfies R and each projection in I satisfies S. This algorithm clearly works in nondeterministic exponential time.

Now we prove the second item. We reduce from the PCP problem. Assume that the input to PCP are two equally long lists $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ of strings over alphabet Σ. Recall that we want to decide whether there exists a solution for this input, that is, a sequence of indices $i_1, i_2, \ldots, i_k$ such that $1 \leq i_j \leq n$ $(1 \leq j \leq k)$ and $a_{i_1} a_{i_2} \cdots a_{i_k} = b_{i_1} b_{i_2} \cdots b_{i_k}$.

Assume without loss of generality that Σ is disjoint from $\mathbb{N}$. Corresponding to every input $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ of PCP over alphabet Σ, we define the following:

- An alphabet $\Sigma(n) := \Sigma \cup \{1, 2, \ldots, n\}$;
- a regular language $R_{a,n} := (\bigcup_{1 \leq i \leq n} a_i \cdot i)^*$;
- a regular language $R_{b,n} := (\bigcup_{1 \leq j \leq n} b_j \cdot j)^*$.

Consider a ternary recognizable relation R over alphabet $\Sigma(n) \cup \{\star, \dagger\}$, where $\star$ and $\dagger$ are symbols not appearing in $\Sigma(n)$, defined as

$$\big(\star \cdot \Sigma^*\big) \times \big(\dagger \cdot R_{a,n}\big) \times \big(\dagger \cdot R_{b,n}\big).$$

Further, consider a binary relation S over $(\Sigma(n) \cup \{\star, \dagger\})^*$ defined as the union of the following sets:

(1) $\{(w, w') \in (\dagger \cdot (\Sigma(n))^*) \times (\dagger \cdot (\Sigma(n))^*) \mid w_{\{1,\ldots,n\}} \sqsubseteq w'_{\{1,\ldots,n\}}\}$.
(2) $\{(w, w') \in (\dagger \cdot (\Sigma(n))^*) \times (\star \cdot \Sigma^*) \mid w_\Sigma \sqsubseteq w'_\Sigma\}$.
(3) $\{(w, w') \in (\star \cdot \Sigma^*) \times (\dagger \cdot (\Sigma(n))^*) \mid w_\Sigma \sqsubseteq w'_\Sigma\}$.

The intuition is that S takes care that indices in the sequences are consistent. It is easy to see that S is a rational relation, which implies that $S_\sqsubseteq$ is in SCR.

From input $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ to the PCP problem, we construct an instance of $\textsc{GenInt}_{S_\sqsubseteq}(\mathsf{REC})$ defined by the recognizable relation R and

$$I \;=\; \{(1,2), (2,1), (1,3), (3,1), (2,3), (3,2)\}.$$

We claim that $R \cap_I S \neq \emptyset$ if and only if the PCP instance given by lists $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ has a solution.

Assume first that $R \cap_I S \neq \emptyset$. Hence there are words $w_1 \in (\star \cdot \Sigma^*)$, $w_2 \in (\dagger \cdot R_{a,n})$ and $w_3 \in (\dagger \cdot R_{b,n})$, such that (w_i, w_j) belongs to $S_\sqsubseteq$, for each $(i, j) \in I$. Since $(2, 3) \in I$, it must be the case that (w_2, w_3) belongs to $S_\sqsubseteq$. Thus, since the first symbol of both w_2 and w_3 is $\dagger$, it must be the case that $(w_2)_{\{1,\ldots,n\}} \sqsubseteq (w_3)_{\{1,\ldots,n\}}$. For the same reasons, and given that $(3, 2) \in I$, it must be the case that $(w_3)_{\{1,\ldots,n\}} \sqsubseteq (w_2)_{\{1,\ldots,n\}}$. We conclude that $(w_2)_{\{1,\ldots,n\}} = (w_3)_{\{1,\ldots,n\}}$.

Since $(1,2) \in I$, it must be the case that (w_1, w_2) belongs to $S_{\sqsubseteq}$. Thus, since the first symbol of w_1 is $\star$ and the first symbol of w_2 is $\dagger$, it must be the case that $(w_1)_\Sigma \sqsubseteq (w_2)_\Sigma$. For the same reasons, and given that $(2,1) \in I$, it must be the case that $(w_2)_\Sigma \sqsubseteq (w_1)_\Sigma$. We conclude that $(w_1)_\Sigma = (w_2)_\Sigma$.

Mimicking the same argument, but this time using the fact that $\{(1,3),(3,1)\} \subseteq I$, we conclude that $(w_1)_\Sigma = (w_3)_\Sigma$. But then $(w_2)_\Sigma = (w_3)_\Sigma$ (because $(w_1)_\Sigma = (w_2)_\Sigma$).

Assume $(w_2)_{\{1,\ldots,n\}} = (w_3)_{\{1,\ldots,n\}} = i_1 i_2 \cdots i_n$, where each $i_j \in [n]$. Then from the fact that $(w_2)_\Sigma = (w_3)_\Sigma$ we conclude that $a_{i_1} a_{i_2} \cdots a_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}$, and hence that the instance of the PCP problem given by $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ has a solution.

The other direction, that is, that the fact that the instance of the PCP problem given by $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ has a solution implies that $R \cap_I S \neq \emptyset$, can be proved using the same arguments. □

In particular, if we have a CRPQ(S) query of the form

$$\exists \bar{y} \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i : L_i} u_i') \quad \wedge \quad \bigwedge_{(i,j) \in I} S(\chi_i, \chi_j) \Big),$$

where I is acyclic (as a directed graph) and $S \in \mathsf{SCR}$, then query evaluation has NEXPTIME combined complexity.

The proof of this result is quite different from the upper bound proof of Theorem 6.7, since the set of witnesses for the generalized intersection problem is no longer guaranteed to be rational without the undirected acyclicity condition. Instead, here we establish the finite-model property, which implies the result.

Also, as a corollary to the proof of Theorem 6.10, we get the following result:

Proposition 6.11. *Let $S \in \mathsf{SCR}$ be a partial order. Then* GENINT$_S$(REC) *is decidable in* NEXPTIME.

Proof. As in the previous proof, we start by proving a small model property for the size of the witnesses of the instances in GENINT$_S$(REC), for S a partial order in SCR. Let R be an m-ary recognizable relation, $m > 0$, and $I \subseteq [m] \times [m]$. Assume that both R and S are over Σ. Then the following holds: Assume $R \cap_I S \neq \emptyset$. There is $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ of at most exponential size that is accepted by R and such that $(w_i, w_j) \in S$, for each $(i,j) \in I$. We prove this small model property by applying usual cutting techniques.

Assume that R is given as

$$\bigcup_i \mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m},$$

where each $\mathcal{N}_{i_j}$ is an NFA over Σ. Further, assume that S is given as the 2-tape transducer S defined by the tuple $(Q_S, \Sigma, Q_S^0, \delta_S, Q_S^F)$, where Q_S is the set of states, the set of initial states is Q_S^0, the set of final states is Q_S^F, and $\delta_S : Q_S \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \to 2^{Q \times (\{1,2\} \cup \{\{1,2\}\})}$ is the transition function. Assume also that there is $\bar{u} = (u_1, \ldots, u_m) \in (\Sigma^*)^m$ that is accepted by R and such that $(u_i, u_j) \in S$, for each $(i,j) \in I$. Then $\bar{u}$ is accepted by $\mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m}$, for some i.

Let I^+ be the transitive closure of I. Notice, since S defines a partial order over Σ^*, that $(u_j, u_k) \in S$, for each $(j,k) \in I^+$. Further, for every pair $(j,k) \in [m] \times [m]$ such that $\{(j,k),(k,j)\} \subseteq I^+$ we must have that $u_j = u_k$. We need to maintain such equality when

applying our cutting techniques over $\bar{u}$. In order to do that we define an equivalence relation $\mathcal{E}_I$ over $[m]$ as follows:

$$\mathcal{E}_I \ := \ \{(j,k) \in [m] \times [m] \mid j = k \text{ or } \{(j,k),(k,j)\} \subseteq I^+\}.$$

Hence $\mathcal{E}_I$ contains all pairs $(j,k) \in [m] \times [m]$ such that I implies $u_j = u_k$. Take the quotient $[m]/\mathcal{E}_I$, and consider the restriction $I([m]/\mathcal{E}_I)$ of I over $[m]/\mathcal{E}_I$, defined in the expected way: $([j]_{\mathcal{E}_I}, [k]_{\mathcal{E}_I}) \in I([m]/\mathcal{E}_I)$ if and only if $(j',k') \in I$, for some $j' \in [j]_{\mathcal{E}_I}$ and $k' \in [k]_{\mathcal{E}_I}$. Notice that $I([m]/\mathcal{E}_I)$ defines a DAG over $[m]/\mathcal{E}_I$.

Consider now a new input to $\textsc{GenInt}_S(\mathsf{REC})$, given this time by $I([m]/\mathcal{E}_I) \subseteq ([m]/\mathcal{E}_I) \times ([m]/\mathcal{E}_I)$, and the recognizable relation R' defined as

$$\prod_{[j]_{\mathcal{E}_I} \in [m]/\mathcal{E}_I} \mathcal{M}_i^{[j]_{\mathcal{E}_I}},$$

where $\mathcal{M}_i^{[j]_{\mathcal{E}_I}} = \bigcap_{k \in [j]_{\mathcal{E}_I}} \mathcal{N}_{i_k}$. Notice that this new input may be of exponential size in the size of R.

Assume that $[m]/\mathcal{E}_I$ consists of $p \leq m$ equivalence classes and, without loss of generality, that these correspond to the first p indices of $[m]$. Hence each product in R' is of the form $\prod \mathcal{M}_{i_1} \times \cdots \times \mathcal{M}_{i_p}$, where $\mathcal{M}_{i_j}$ is defined as the intersection of all NFAs in the equivalence class $[j]_{\mathcal{E}_I}$. Also, $I([m]/\mathcal{E}_I)$ is the restriction of I to $[p] \times [p]$. Then it must be the case that $(u_1, \ldots, u_p) \in (\Sigma^*)^p$ belongs to R' and $(u_j, u_k) \in S$, for each $(j,k) \in I([m]/\mathcal{E}_I)$. Further, from every witness to the fact that $R' \cap_{I([m]/\mathcal{E}_I)} S \neq \emptyset$ we can construct in polynomial time a witness to the fact that $R \cap_I S \neq \emptyset$. Hence, in order to prove our small model property it will be enough to prove the following: There is $\bar{w} = (w_1, \ldots, w_p) \in (\Sigma^*)^p$ of at most exponential size (in R) that is accepted by R' and such that $(w_j, w_k) \in S$, for each $(j,k) \in I([m]/\mathcal{E}_I)$.

The latter can be done by mimicking the inductive proof of the first part of Theorem 6.10. We only have to deal now with the issue that some of the NFAs that define R' may be exponential in the size of R. However, by following the inductive proof one observes that this is not a problem, and that the same exponential bound holds in this case.

It is now simple to prove the first part of the theorem using the small model property. In fact, in order to check whether $R \cap_I S \neq \emptyset$, for S a partial order in SCR, we only need to guess an exponential size witness $\bar{w}$, and then check in exponential time that it satisfies R and each projection in I satisfies S. This algorithm clearly works in nondeterministic exponential time. □

By applying similar techniques to those in the proof of Theorem 6.9 we obtain the following.

Corollary 6.12. *If $S \in \mathsf{SCR}$ is a partial order, then* CRPQ*(S) queries can be evaluated with* NExptime *combined complexity. In particular,* CRPQ*$(\sqsubseteq)$ queries have* NExptime *combined complexity.*

We do not have at this point a matching lower bound for the complexity CRPQ$(\sqsubseteq)$ queries. Notice that an easy PSpace lower bound follows by a reduction from the intersection problem for NFAs, as the one presented in the proof of Theorem 6.7.

The last question is whether these results can be extended to other relations considered here, such as subword and suffix. We do not know the result for subword (which appears to be hard), but we do have a matching complexity bound for the suffix relation.

Proposition 6.13. *The problem* $\textsc{GenInt}_{\preceq_{\text{suff}}}(\mathsf{REC})$ *is decidable in* NExptime. *In particular,* $\text{CRPQ}(\preceq_{\text{suff}})$ *queries can be evaluated with* NExptime *combined complexity.*

Proof. We only prove that $\textsc{GenInt}_{\preceq_{\text{suff}}}(\mathsf{REC})$ is decidable in NExptime. The fact that $\text{CRPQ}(\preceq_{\text{suff}})$ queries can be evaluated with NExptime combined complexity follows easily from this by applying the same techniques as in the proof of Theorem 6.9.

We start by proving a small model property for the size of the witnesses of the instances in $\textsc{GenInt}_{\preceq_{\text{suff}}}(\mathsf{REC})$. Let R be an m-ary recognizable relation, $m > 0$, and $I \subseteq [m] \times [m]$. Assume that both R and $\preceq_{\text{suff}}$ are over Σ. Then the following holds: Assume it is the case that $R \cap_I \{\preceq_{\text{suff}}\} \neq \emptyset$. There is $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ of at most exponential size that is accepted by R and such that $w_i \preceq_{\text{suff}} w_j$, for each $(i,j) \in I$. We prove this small model property by applying cutting techniques.

Assume that R is given as

$$\bigcup_i \mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m},$$

where each $\mathcal{N}_{i_j}$ is an NFA over Σ. We assume, without loss of generality, that I defines a DAG over $[m] \times [m]$. In fact, assume otherwise; that is, I does not define a DAG over $[m] \times [m]$. Since $\preceq_{\text{suff}}$ defines a partial order over Σ^*, we can always reduce in polynomial time the instance of $\textsc{GenInt}_{\preceq_{\text{suff}}}(\mathsf{REC})$ given by R and I to an "equivalent" instance of $\textsc{GenInt}_{\preceq_{\text{suff}}}(\mathsf{REC})$ given by recognizable relation R' of arity $m' \leq m$ and $I' \subseteq [m'] \times [m']$ such that I' defines a DAG. We already showed how to do this for an arbitrary partial order over Σ^* in the proof of Proposition 6.11, so we prefer not to repeat the argument here, and simply assume that I defines a DAG over $[m] \times [m]$. Since I defines a DAG it has a topological order over $[m]$. We assume without loss of generality that such topological order is precisely the linear order on $[m]$.

Assume then that there is $\bar{u} = (u_1, \ldots, u_m) \in (\Sigma^*)^m$ that is accepted by R and such that $u_i \preceq_{\text{suff}} u_j$, for each $(i,j) \in I$. Then $\bar{u}$ is accepted by $\mathcal{N}_{i_1} \times \cdots \times \mathcal{N}_{i_m}$, for some i. Assume that the length of u_j is $p_j \geq 0$, for each $1 \leq j \leq m$. Our goal is to "cut" $\bar{u}$ in order to obtain an exponential size witness to the fact that $R \cap_I \{\preceq_{\text{suff}}\} \neq \emptyset$.

We recursively define the set $\mathcal{M}_k$ of *marked* positions in string u_k, $1 \leq k \leq m$, as follows:

- No position in u_1 is marked.
- For each $1 < k \leq m$ the set $\mathcal{M}_k$ of marked positions in u_k is defined as the union of the marked positions in u_k *with respect to* j, for each $j < k$ such that $(j,k) \in I$, where the latter is defined as follows. Assume that $\mathcal{M}_j$ is the set of marked positions in u_j. Then the set $\mathcal{M}_k$ of positions $1 \leq \ell \leq p_k$ that are marked in u_k with respect to j is $\{r + p_k - p_j \mid r = 1 \text{ or } r \in \mathcal{M}_j\}$. (Notice that $p_k - p_j \geq 0$ since $u_j \preceq_{\text{suff}} u_k$, and hence $1 \leq r + p_k - p_j \leq p_k$ for each $r \in \mathcal{M}_j$ and for $r = 1$).

Intuitively, $\mathcal{M}_k$ consists of those positions $1 \leq \ell \leq p_k$ such that for some $j < k$ with $(j,k) \in I^+$, where I^+ is the transitive closure of I, it is the case that that $u_k = u_k[1, \ell-1] \cdot u_j$. Or, in other words, the fact that $u_j \preceq_{\text{suff}} u_k$ starts to be "witnessed" at position ℓ of u_k. We assume the $\mathcal{M}_k$'s to be linearly ordered by the restriction of the linear order $1 < 2 < \cdots < m$ to $\mathcal{M}_k$. By a simple inductive argument it is possible to prove that the size of $\mathcal{M}_k$ is polynomially bounded in m, for each $1 \leq k \leq m$.

Since $u_j \preceq_{\text{suff}} u_k$, for each $(j,k) \in I$, this implies that the labels in some positions of u_j are preserved in the respective positions of u_k that witness the fact that $u_j \preceq_{\text{suff}} u_k$. The important thing to notice is that, since we are dealing with $\preceq_{\text{suff}}$, the following holds:

For each position p that is "copied" from u_j into u_k in order to satisfy $u_j \preceq_{\text{suff}} u_k$, the distance from p to the last element of u_j equals the distance from the copy of p in u_k to the last position of u_k. That is, distances to the last element of the string are preserved when copying positions (and labels) in order to satisfy I. We need to take care of this information when "cutting" $\bar{u}$ in order to obtain an exponential size witness for the fact that $R \cap_I \{\preceq_{\text{suff}}\} \neq \emptyset$. In order to do this we define for each $0 \leq r \leq \max\{p_k \mid 1 \leq k \leq m\}$, a binary relation $\overset{r}{\rightharpoonup}$ on $\{u_1, \ldots, u_m\}$ such that $u_j \overset{r}{\rightharpoonup} u_k$ if $p_j - r > 0$ and $(j,k) \in I$. This implies that position $p_j - r$ of u_j is "copied" as position $p_k - r$ of u_k in order to satisfy the fact that $u_j \preceq_{\text{suff}} u_k$.

But in order to consistently "cut" $\bar{u}$, we need to preserve the suffix relation both with respect to forward and backward edges of the graph defined by I. In order to do that we define $\overset{r}{\rightleftharpoons}$ as $(\overset{r}{\rightharpoonup} \cup (\overset{r}{\rightharpoonup})^{-1})$. Further, since $\preceq_{\text{suff}}$ is a partial order over Σ^*, and hence it defines a transitive relation, it is important for us also to consider the transitive closure $(\overset{r}{\rightleftharpoons})^+$ of the binary relation $\overset{r}{\rightleftharpoons}$. Intuitively, $u_j(\overset{r}{\rightleftharpoons})^+u_k$, for $1 \leq j,k \leq m$, if position $p_j - r$ of u_j has to be "copied" into position $p_k - r$ of u_k in order for $\bar{u}$ to satisfy the pairs in I with respect to $\preceq_{\text{suff}}$.

Let $t := |\mathcal{N}_{i_1}| \cdot |\mathcal{N}_{i_2}| \cdots |\mathcal{N}_{i_m}|$ and $s := (\sum_{1 \leq k \leq m} |\mathcal{M}_k|) + 1$. We claim the following: There is $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ such that: (1) $\bar{w}$ is accepted by R, (2) $w_i \preceq_{\text{suff}} w_j$, for each $(i,j) \in I$, and (3) for each $1 \leq k \leq m$ the number of positions in w_k between any two consecutive positions in $\mathcal{M}_k$ is bounded by $s \cdot t \cdot 2^m \cdot |\Sigma|^m$. This clearly implies our small model property.

Assume that $\bar{u}$ does not satisfy this. Then there exists $1 \leq j \leq m$ and two consecutive positions p and p' in $\mathcal{M}_j$, such that the number of positions in u_j between p and p' is bigger than $s \cdot t \cdot 2^m \cdot |\Sigma|^m$. But this implies that there are two positions $p_j - r$ and $p_j - r'$ $(r > r')$ between p and p' in u_j such that the following hold:

(1) $\{1 \leq k \leq m \mid u_j(\overset{r}{\rightleftharpoons})^+u_k\} = \{1 \leq k \leq m \mid u_j(\overset{r'}{\rightleftharpoons})^+u_k\}$. Intuitively, this says that the set of strings in which position $p_j - r$ of u_j is "copied" coincides with the set of strings in which position $p_j - r'$ of u_j is "copied".
(2) For each k such that $u_j(\overset{r}{\rightleftharpoons})^+u_k$ it is the case that neither $p_k - r$ nor $p_k - r'$ is a marked position in $\mathcal{M}_k$, and there is no marked position in $\mathcal{M}_k$ in between $p_k - r$ and $p_k - r'$ in u_k.
(3) The state assigned by the accepting run of $\mathcal{N}_{i_j}$ over u_j to position $p_j - r$ of u_j is the same than the one assigned to position $p_j - r'$.
(4) The state assigned by the accepting run of $\mathcal{N}_{i_k}$ over u_k to the "copy" $p_k - r$ of position $p_j - r$ over u_k, for each k such that $u_j(\overset{r}{\rightleftharpoons})^+u_k$, is the same than the one assigned to the "copy" $p_k - r'$ of position $p_j - r'$ over u_k.
(5) The symbol in position $p_j - r$ of u_j is the same as the symbol in position $p_j - r'$ of u_j.
(6) For each k such that $u_j(\overset{r}{\rightleftharpoons})^+u_k$ it is the case that the symbol in position $p_k - r$ of u_k is the same as the symbol in position $p_k - r'$ of u_k.

Intuitively, this states that if we "cut" the string u_j from position $p_j - r + 1$ to $p_j - r'$, and string u_k from position $p_k - r + 1$ to $p_k - r'$, for each k such that $u_j(\overset{r}{\rightleftharpoons})^+u_k$, then the resulting $\bar{u}' = (u'_1, \ldots, u'_m) \in (\Sigma)^m$ satisfies the following: (1) $\bar{u}'$ is accepted by R, and (2) for each $(j,k) \in I$ it is the case that $u'_j \preceq_{\text{suff}} u'_k$. We formally prove this below. Notice for the time being that this implies our small model property. Indeed, if we recursively apply

this procedure to $\bar{u}$ we will end up with $\bar{w} = (w_1, \ldots, w_m) \in (\Sigma^*)^m$ such that: (1) $\bar{w}$ is accepted by R, (2) $w_j \preceq_{\text{suff}} w_k$, for each $(j,k) \in I$, and (3) for each $1 \leq k \leq m$ the number of positions in w_k between any two consecutive positions in $\mathcal{M}_k$ is bounded by $s \cdot t \cdot 2^m \cdot |\Sigma|^m$.

Let $\bar{u}' = (u'_1, \ldots, u'_m) \in (\Sigma)^m$ be the result of applying once the cutting procedure described above to $\bar{u} = (u_1, \ldots, u_m)$, starting from string $\bar{u}_j$ by cutting positions from $p_j - r + 1$ to $p_j - r'$ $(r > r')$. It is not hard to see that $\bar{u}'$ is accepted by R, since each u_k has been cut in a way that is invariant with respect to the accepting run of $\mathcal{N}_{i_k}$ over u_k. Assume that $(\ell, k) \in I$. We need to prove that $u'_\ell \preceq_{\text{suff}} u'_k$. If $u_\ell = u'_\ell$ and $u_k = u'_k$ then $u'_\ell \preceq_{\text{suff}} u'_k$ by assumption. Assume then that at least one of u_ℓ and u_k has been cut. Suppose first that u_ℓ has been cut from position $p_\ell - r + 1$ to position $p_\ell - r'$ in order to obtain u'_ℓ. Then $u_j(\overset{r}{\rightleftharpoons})^+ u_\ell$ and $u_j(\overset{r'}{\rightleftharpoons})^+ u_\ell$. Clearly, it is also the case that $u_\ell \overset{r}{\rightleftharpoons} u_k$ and $u_\ell \overset{r'}{\rightleftharpoons} u_k$, which implies that $u_j(\overset{r}{\rightleftharpoons})^+ u_k$ and $u_j(\overset{r'}{\rightleftharpoons})^+ u_k$. Thus, u_k is also cut from position $p_k - r + 1$ to $p_k - r'$ in order to obtain u'_k, and hence $u'_\ell \preceq_{\text{suff}} u'_k$. Suppose, on the other hand, that u_ℓ has not been cut but u_k has been cut from position $p_k - r + 1$ to position $p_k - r'$ in order to obtain u'_k. We consider three cases:

(1) $r' > p_j - 1$. Then clearly $u'_k \preceq_{\text{suff}} u'_j$.
(2) $r' \leq p_j - 1$ and $r > p_j - 1$. This cannot be the case since then either $p_k - r'$ is a marked position in $\mathcal{M}_k$ (when $r' = p_j - 1$), or $p_k - r$ and $p_k - r'$ have a marked position in $\mathcal{M}_k$ in between (namely, $p_k - p_j + 1$). Any of these contradicts the fact that a cutting of u_k could be applied from position $p_k - r$ to position $p_k - r'$ in order to obtain u'_k.
(3) $r' < p_j - 1$ and $r \geq p_j - 1$. Similar to the previous one.
(4) $r < p_j - 1$. But then clearly $u_\ell \overset{r}{\rightleftharpoons} u_k$ and $u_\ell \overset{r'}{\rightleftharpoons} u_k$, which implies that $u_j(\overset{r}{\rightleftharpoons})^+ u_\ell$ and $u_j(\overset{r'}{\rightleftharpoons})^+ u_\ell$. This implies that u_ℓ should have also been cut from position $p_\ell - r$ to position $p_\ell - r'$ in order to obtain u'_ℓ, which is a contradiction.

We can finally prove the theorem using the small model property. In fact, in order to check whether $R \cap_I \{\preceq_{\text{suff}}\} \neq \emptyset$ we only need to guess an exponential size witness $\bar{w}$, and then check in polynomial time that it satisfies R and each projection in I satisfies $\preceq_{\text{suff}}$. This algorithm clearly works in nondeterministic exponential time. □

7. Conclusions

Motivated by problems arising in studying logics on graphs (as well as some verification problems), we studied the intersection problem for rational relations with recognizable and regular relations over words. We have looked at rational relations such as subword $\preceq$, suffix $\preceq_{\text{suff}}$, and subsequence $\sqsubseteq$, which are often needed in graph querying tasks. The main results on the complexity of the intersection and generalized intersection problems, as well as the combined complexity of evaluating different classes of logical queries over graphs are summarized in Fig. 2. Several results generalizing those (e.g., to the class of SCR relations) were also shown. Two problems related to the interaction of the subword relation with recognizable relations remain open and appear to be hard.

From the practical point of view, as rational-relation comparisons are demanded by many applications of graph data, our results essentially say that such comparisons should not be used together with regular-relation comparisons, and that they need to form acyclic patterns (easily enforced syntactically) for efficient evaluation.

	$R \in \mathsf{REC}$	$R \in \mathsf{REG}$	$R \in \mathsf{RAT}$
$(R \cap \preceq) \stackrel{?}{=} \emptyset$	PTIME (cf. [6])	undecidable	undecidable
$(R \cap \preceq_{\text{suff}}) \stackrel{?}{=} \emptyset$		undecidable	undecidable
$(R \cap \sqsubseteq) \stackrel{?}{=} \emptyset$		decidable, NMR	decidable, NMR [13]
$(R \cap_I \preceq) \stackrel{?}{=} \emptyset$	?	undecidable	undecidable
$(R \cap_I \preceq_{\text{suff}}) \stackrel{?}{=} \emptyset$	NEXPTIME	undecidable	
$(R \cap_I \sqsubseteq) \stackrel{?}{=} \emptyset$	NEXPTIME	decidable, NMR	

	$S = \sqsubseteq$	$S = \preceq_{\text{suff}}$	$S = \preceq$	S arbitrary in RAT
ECRPQ(S)	decidable, NMR	undecidable	undecidable	undecidable
CRPQ(S)	NEXPTIME	NEXPTIME	?	undecidable
acyclic CRPQ(S)	PSPACE	PSPACE	PSPACE	PSPACE

Figure 2: Complexity of the intersection and generalized intersection problems, and combined complexity of graph queries for subword ($\preceq$), suffix ($\preceq_{\text{suff}}$), and subsequence ($\sqsubseteq$) relations. NMR stands for non-multiply-recursive lower bound.

So far we dealt with the classical setting of graph data [1, 9, 10, 16, 17] in which the model of data is that of a graph with labels from a finite alphabet. In both graph data and verification problems it is often necessary to deal with the extended case of infinite alphabets (say, with graphs holding data values describing its nodes), and languages that query both topology and data have been proposed recently [24, 27]. A natural question is to extend the positive results shown here to such a setting.

REFERENCES

[1] R. Angles, C. Gutiérrez. Survey of graph database models. *ACM Computing Surveys* 40(1), 2008.

[2] K. Anyanwu, A. P. Sheth. ρ-Queries: enabling querying for semantic associations on the semantic web. *12th International World Wide Web Conference* (WWW), pages 690–699, 2003.

[3] P. Barceló, D. Figueira, L. Libkin. Graph Logics with Rational Relations and the Generalized Intersection Problem. *27th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 115–124, 2012.

[4] P. Barceló, L. Libkin, A. W. Lin, P. Wood. Expressive languages for path queries over graph-structured data. *ACM Transactions on Database Systems*, 37(4) (2012).

[5] M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. Definable relations and first-order query languages over strings. *Journal of the ACM* 50(5):694-751, 2003.

[6] J. Berstel. *Transductions and Context-Free Languages*. B. G. Teubner, 1979.

[7] A. Blumensath and E. Grädel. Automatic structures. *15th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 51–62, 2000.

[8] V. Bruyère, G. Hansel, C. Michaux, R. Villemaire. Logic and p-recognizable sets of integers. *Bulletin of the Belgium Mathematical Society* 1, 191–238, 1994.

[9] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Containment of conjunctive regular path queries with inverse. *7th International Conference on Principles of Knowledge Representation and Reasoning* (KR), pages 176–185, 2000.

[10] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. View-based query processing and constraint satisfaction. *15th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 361-371, 2000.

[11] L. Cardelli, P. Gardner, G. Ghelli. A spatial logic for querying graphs. *29th International Colloquium on Automata, Languages and Programming* (ICALP), pages 597-610, 2002.

[12] O. Carton, C. Choffrut, S. Grigorieff. Decision problems among the main subfamilies of rational relations. *Informatique Théorique et Applications*, 40, pages 255–275, 2006.
[13] P. Chambart, Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. *27th International Conference on the Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 265–276, 2007.
[14] P. Chambart, Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. *23rd Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 205–216, 2008.
[15] C. Choffrut. Relations over words and logic: a chronology. *Bulletin of the EATCS* 89, 159–163, 2006.
[16] M. P. Consens, A. O. Mendelzon. GraphLog: a visual formalism for real life recursion. *9th ACM Symposium on Principles of Database Systems* (PODS), pages 404–416, 1990.
[17] I. Cruz, A. Mendelzon, P. Wood. A graphical query language supporting recursion. *ACM Special Interest Group on Management of Data* (SIGMOD), pages 323-330, 1987.
[18] A. Dawar, P. Gardner, G. Ghelli. Expressiveness and complexity of graph logic. *Information and Computation* 205, pages 263-310, 2007.
[19] A. Deutsch, V. Tannen. Optimization properties for classes of conjunctive regular path queries. *8th International Workshop on Database Programming Languages* (DBPL), pages 21–39, 2001.
[20] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *The American Journal of Mathematics*, 35(4), pages 413–422, 1913.
[21] C. Elgot and J. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development* 9, pages 47–68, 1965.
[22] D. Florescu, A. Levy, D. Suciu. Query containment for conjunctive queries with regular expressions. *17th ACM Symposium on Principles of Database Systems* (PODS), pages 139–148, 1998.
[23] C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Computer Science* 108, pages 45–82, 1993.
[24] O. Grumberg, O. Kupferman, S. Sheinvald. Variable automata over infinite alphabets. *4th International Conference on Language and Automata Theory and Applications* (LATA), pages 561–572, 2010.
[25] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society* (3), 2(7), pages 326–336, 1952.
[26] D. Kozen. Lower bounds for natural proof systems. *18th Annual Symposium on Foundations of Computer Science* (FOCS), pages 254-266, 1977.
[27] L. Libkin, D. Vrgoč. Regular path queries on graphs with data. *15th International Conference on Database Theory* (ICDT), 2012.
[28] L. Lisovik. The identity problem for regular events over the direct product of free and cyclic semigroups. *Doklady Akad. Nauk Ukr., ser. A*, 6 (1979), 410–413.
[29] M.H. Löb and S.S. Wainer. Hierarchies of number theoretic functions, I. *Archiv fr mathematische Logik und Grundlagenforschung*, 13:39–51, 1970.
[30] M. Nivat. Transduction des langages de Chomsky. *Annales de l'Institut Fourier* 18 (1968), 339–455.
[31] H. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.
[32] S. Schmitz and Ph. Schnoebelen Multiply-Recursive Upper Bounds with Higman's Lemma. *38th International Colloquium on Automata, Languages and Programming* (ICALP), pages 441–452, 2011.
[33] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters* 83, pages 251-261, 2002.
[34] W. Thomas. Infinite trees and automaton-definable relations over ω-words. *Theoretical Computer Science* 103, pages 143–159, 1992.