Graph Logics with Rational Relations and the Generalized Intersection Problem
Pablo Barceló, Diego Figueira, Leonid Libkin

To cite this version:
Pablo Barceló, Diego Figueira, Leonid Libkin. Graph Logics with Rational Relations and the Generalized Intersection Problem. Annual IEEE Symposium on Logic in Computer Science (LICS), Jun 2012, Dubrovnik, Croatia. pp.115-124, 10.1109/LICS.2012.23. hal-01799740

HAL Id: hal-01799740
https://hal.archives-ouvertes.fr/hal-01799740
Submitted on 25 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 (factor) 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-multiply-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-multiply-recursive complexity. We then prove a dichotomy result for logics related to verifying lossy channel systems that has non-multiply-recursive complexity. We state problems like this are very basic and deserve a study on their own right, 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 [10], [17]. Their formulae combine reachability patterns. The simplest form is known as regular path queries (RPQs) [16], [15]; 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 (CRPQs), which have been the subject of much investigation [8], [18], [21]. For instance, a CRPQ may ask for nodes \(v\) so that there exist nodes \(v_1\) and \(v_2\) and paths from \(v\) to \(v_1\) with the label in a regular language \(L_i\).

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 [3]. It used regular relations on paths, i.e., relations given by synchronized automata [20], [22]. Equivalently, these are the relations definable in automatic structures on words [4], [6], [7]. They include prefix, equality, equal length of words, or fixed edit distance between words. The extension of CRPQs with them, called ECPRQs, was shown to have acceptable complexity (NLOGSPACE with respect to data, PSPACE with respect to query).
However, it was 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 [5]; 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 [5], [14]). 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 $\{1, \ldots, n\}$. 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 channels). Their reachability problem is known to be decidable [12], [13] can be added to the common class CRPQ of graph queries, without incurring a high complexity cost.

### Organization

We give basic definitions in Section II and define the main problems we study in Section III. Section IV introduces graph logics and establishes their connection with the (generalized) intersection problem. Section V studies decidable and undecidable cases of the intersection problem. Section VI looks at the case of recognizable relations and CRPQs and establishes decidability results based on the intersection pattern. Complete proofs of all results are available in the full version of the paper.

### Preliminaries

**Alphabets, languages, and morphisms:** We shall use letters $\Sigma, \Gamma$ to denote finite alphabets. The set of all finite words over an alphabet $\Sigma$ is denoted by $\Sigma^*$. We write $\varepsilon$ for the empty word, $w \cdot w'$ for the concatenation of two words, and $|w|$ for the length of a word $w$.

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 $w'' \preceq_{\text{suffix}} w$).

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

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 $\Sigma^*$ with the operation of concatenation and the neutral element $\varepsilon$ forms a monoid $\langle \Sigma^*, \cdot, \varepsilon \rangle$, the free monoid generated by $\Sigma$. 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 $f(a_1 \ldots a_n) = f(a_1) \ldots f(a_n)$. A morphism $f: \langle \Sigma^*, \cdot, \varepsilon \rangle \to (\Gamma^*, \cdot, \varepsilon)$ is called alphabetic if $f(a) \in \Gamma \cup \{\varepsilon\}$, and strictly alphabetic if $f(a) \in \Gamma$ for each $a \in \Sigma$, see [5].

A language $L$ is a subset of $\Sigma^*$. 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) $A = (Q, \Sigma, q_0, \delta, F)$ such that $L = L(A)$, the language of words accepted by $A$. 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 $\emptyset, \varepsilon$, and alphabet letters by using operations of concatenation ($e \cdot e'$), union ($e \cup e'$), and Kleene star ($e^*$). It is of course the 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 $\Sigma$, i.e., subsets of $\Sigma^* \times \ldots \times \Sigma^*$. We now define those (see [5], [11], [14], [20], [22], [30]).

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

It is well-known that a relation $R \subseteq \langle \Sigma^* \rangle^n$ is in $\text{REC}_n$ 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 $\Sigma$, see [5], [20].

Next, we define the class of regular relations. Let $\downarrow \subseteq \Sigma$ be a new alphabet letter, and let $\Sigma_1$ be $\Sigma \cup \{\downarrow\}$. Each tuple $\bar{w} = (w_1, \ldots, w_n)$ of words from $\Sigma_1$ can be viewed as a word over $\Sigma_1^n$ as follows: pad words $w_i$ with $\downarrow$ so that they all are of the same length, and use as the $k$th symbol of the new word the $n$-tuple of the $k$th symbols of the padded words. Formally, let $\ell = \max\{|w_i|\}$. Then $w_1 \otimes \ldots \otimes w_n$ is a word of length $\ell$ whose $k$th symbol is $(a_1, \ldots, a_n) \in \Sigma_1^n$ such that

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

We shall also write $\otimes \bar{w}$ for $w_1 \otimes \ldots \otimes w_n$. A relation $R \subseteq \langle \Sigma^* \rangle^n$ is called a regular $n$-ary relation over $\Sigma$ if there is a finite automaton $A$ over $\Sigma_1^n$ that accepts $\{\otimes \bar{w} : \bar{w} \in R\}$. The class of $n$-ary regular relations is denoted by $\text{REG}_n$; as before, we write $\text{REG}$ 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 $\preceq_{\text{suffix}}, \preceq, \preceq_{\text{pref}}$ and $\preceq$ are all rational: the expression $(\{a \in \Sigma \mid a \varepsilon\})^* \preceq_{\text{suffix}} (\{a \in \Sigma \mid a \varepsilon\})^*$ defines $\preceq_{\text{suffix}}$, the expression $(\{a \in \Sigma \mid a \varepsilon\})^* \cdot (\{a \in \Sigma \mid a \varepsilon\})^*$ defines $\preceq$, and the expression $(\{a \in \Sigma \mid a \varepsilon\} \cup (a, a))^*$ defines $\preceq_{\text{pref}}$.

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 $\text{RAT}_n$, or just $\text{RAT}$, as before.

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

$$\text{REC}_k \subseteq \text{REG}_k \subseteq \text{RAT}_k$$

for every $k > 1$. For instance, $\preceq_{\text{pref}} \in \text{REG}_2 - \text{REG}_2$ and $\preceq_{\text{pref}} \subseteq \text{RAT}_2 - \text{REG}_2$.

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

Binary rational relations can be characterized as follows [5], [28]. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is rational iff there is a finite alphabet $\Gamma$, 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., $R \in \text{REG}_2$ so that $(w, w') \in R$ implies $|w| = |w'|$. Regular binary relations are then finite unions of relations of the form $\{(w, u, w') \mid (w, w') \in R, u \in L\}$ and $\{(w, w', u) \mid (w, w') \in R, u \in L\}$, where $R$ ranges over length-preserving regular relations, and $L$ over regular languages.

Properties of classes of relations: Since relations in $\text{REC}$ and $\text{REG}$ are given by NFAs, they inherit all the closure/decidability properties of regular languages. If $R \in \text{RAT}$, 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 [5]. Also, for $R, R' \in \text{RAT}$, the following are undecidable: checking whether $R \subseteq R'$ or $R = R'$, universality ($R = \Sigma^* \times \Sigma^*$), and checking whether $R \in \text{REG}$ or $R \in \text{REC}$ [5], [11], [26].
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 \(\Sigma_1 \times \ldots \times \Sigma_m\), with the \(\Sigma_i\)'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 \(\Sigma\) as the disjoint union of \(\Sigma_i\)'s, and enforce the condition that the \(i\)th projection only uses the letters from \(\Sigma_i\). This is possible for all the classes of relations we consider. In fact, in the proofs we shall be using both types of relations.

III. Generalized intersection problem

We now formalize the main technical problem we study. Let \(\mathcal{R}\) be a class of relations over \(\Sigma\), and \(\mathcal{S}\) a class of binary relations over \(\Sigma\). We use the notation \([m]\) for \(\{1, \ldots, m\}\). The generalized intersection problem \((\mathcal{R} \cap \mathcal{S}) \nsubseteq \emptyset\) is defined as:

**Problem:** 
\[ (\mathcal{R} \cap \mathcal{S}) \nsubseteq \emptyset \]

**Input:**
- \(\mathcal{R} \in \mathcal{R}\)
- \(\mathcal{S} \in \mathcal{S}\)

**Question:**
- \(\exists \bar{w} \in \mathcal{S}\) such that \(\bar{w} \in \mathcal{R}\) for all \((i, j) \in I\)?

If \(\mathcal{S} = \{\mathcal{S}\}\), we write \(\mathcal{S}\) instead of \(\{\mathcal{S}\}\). We write \(\text{GenInt}_S(\mathcal{R})\) for the class of all problems \((\mathcal{R} \cap \mathcal{S}) \nsubseteq \emptyset\) where \(\mathcal{S}\) is fixed, i.e., the input consists of \(\mathcal{R} \in \mathcal{R}\) 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 \(\mathcal{S}\). 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:** 
\[ (\mathcal{R} \cap \mathcal{S}) = \emptyset \]

**Input:**
- \(\mathcal{R} \in \mathcal{R}\)
- \(\mathcal{S} \in \mathcal{S}\)

**Question:**
- \(\mathcal{R} \cap \mathcal{S} \neq \emptyset\)?

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

IV. 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:

- for ECRPQ(S), these bounds are provided by the complexity of \(\text{GenInt}_S(\text{REG})\) and of \((\text{REG} \cap \mathcal{S}) \subseteq \emptyset\)

The standard abstraction of graph databases [1] is finite \(\Sigma\)-labeled graphs \(G = (V, E)\), 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 \(\rho\) from \(v_0\) to \(v_m\) in \(G\) is a sequence of edges \((v_0, a_0, v_1), (v_1, a_1, v_2), \ldots, (v_{m-1}, a_{m-1}, v_m)\) from \(E\), for some \(m \geq 0\).

The label of \(\rho\), 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 RPQs [16]; they are expressions of the form \(x \xrightarrow{} y\), where \(L\) is a regular language. We normally assume that \(L\) is represented by a regular expression or an NFA. Given a \(\Sigma\)-labeled graph \(G = (V, E)\), the answer to an RPQ above is the set of pairs of nodes \((v, v')\) such that there is a path \(\rho\) from \(v\) to \(v'\) with \(\lambda(\rho) \in L\).

Conjunctive RPQs, or CRPQs [8], [9], [15] are the closure of RPQs under conjunction and existential quantification. Formally, they are expressions of the form

\[
\varphi(\bar{x}) = \exists \bar{y} \left( \bigwedge_{i=1}^{m} \left( u_i \xrightarrow{L_i} u_i' \right) \land \bigwedge_{(i,j) \in I} S(\chi_i, \chi_j) \right) \tag{1}
\]

where variables \(u_i, u_i'\) 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 \(\rho_i\) 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 CRPQ(S) queries, where \(S\) is a binary relation over \(\Sigma^*\). Its formulae are of the form

\[
\varphi(\bar{x}) = \exists \bar{y} \left( \bigwedge_{i=1}^{m} \left( u_i \xrightarrow{L_i} u_i' \right) \land \bigwedge_{(i,j) \in I} S(\chi_i, \chi_j) \right) \tag{2}
\]

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 \(\rho_k\), 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 \((\lambda(\rho_k), \lambda(\rho'_k))\) \(\in S\) whenever \((i, j) \in I\). For instance, the query

\[
\exists y, y' \left( (x \xrightarrow{\Sigma^* a} y) \land (x' \xrightarrow{\Sigma^* b} y') \land \chi \subseteq \chi' \right)
\]

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 \(G \models \varphi(\bar{a})\). 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 \(\varphi\) is fixed, and the input consists of the typically large graph \((G, \bar{a})\). We now relate it to the complexity of \(\text{GenInt}_S(\text{REC})\).
Lemma IV.1. Fix a CRPQ\(S\) query \(\varphi\) as in (2). Then there is a DLOGSPACE algorithm that, given a graph \(G\) and a tuple \(\bar{v}\) of nodes, constructs an \(n\)-ary relation \(R \in \text{REC}\) so that the answer to the generalized intersection problem \((R \cap_1 S) \triangleq \emptyset\) is ‘yes’ iff \(G \models \varphi(\bar{v})\).

Proof idea. Given a \(\Sigma\)-labeled graph \(G = \langle V, E \rangle\) and two nodes \(v, v'\), we write \(A(G, v, v')\) for \(G\) viewed as an NFA with the initial state \(v\) and the final state \(v'\). Now consider a CRPQ\(S\) query \(\varphi(\bar{x})\) given by (2). Let \(\bar{v}\) be a tuple of nodes of \(G\), of the same length as \(\bar{x}\).

The algorithm first enumerates all tuples \(\bar{b}\) of nodes of \(G\) of the same length as \(\bar{y}\). Let \(v_i\) and \(v'_i\) be the interpretations of \(u_i\) and \(u'_i\), when \(\bar{x}\) is interpreted as \(\bar{v}\) and \(\bar{y}\) as \(\bar{b}\). Define \(R_{\bar{b}} = \prod_{i=1}^{m}(\mathcal{L}(A(G, v_i, v'_i)) \cap L_i)\); this is a relation in \(\text{REC}_m\). Hence, \(R = \bigcup_{\bar{b}} R_{\bar{b}}\) in \(\text{REC}_m\) too. It is now easy to see that \(R \cap_1 S \neq \emptyset\) iff \(G \models \varphi(\bar{v})\).

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

Lemma IV.2. For each binary relation \(S\), there is a CRPQ\(S\) query \(\varphi(x, x')\) and a DLOGSPACE algorithm that, given a relation \(R \in \text{REC}_2\), constructs a graph \(G\) and two nodes \(v, v'\) so that \(G \models \varphi(v, v')\) iff \(R \cap S \neq \emptyset\).

Proof idea. Let \(R \in \text{REC}_2\) be given as \(\bigcup_{i=1}^{n}(L_i \times K_i)\), where the \(L_i, K_i \subseteq \Sigma^*\) are regular languages for every \(i\). Let \((V_i, E_i)\) 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 we define \((W_i, H_i)\), nodes \(w_0^i\) and sets \(C_i \subseteq W_i\) for NFA defining \(K_i\).

We now construct the graph \(G\). Its labeling alphabet is the union of \(\Sigma\) and \(\{\#, \$, !\}\). Its set of vertices is the disjoint union of all the \(V_i\), \(W_i\), as well as two distinguished nodes \(\text{start}\) and \(\text{end}\). Its edges include all the edges from \(E_i, S_i\) and \(H_i, S_i\), and the following:
- \#-labeled edges from \(\text{start}\) to each initial state, i.e., to each \(v_0^i\) and \(w_0^i\) for all \(i \leq n\).
- \$-labeled edges between the initial states of automata with the same index, i.e., edges \((v_0^i, \$, w_0^j)\) for all \(i < j \leq n\).
- !-labeled edges from final states to \(\text{end}\), i.e., edges \((v, !, \text{end})\), where \(v \in \bigcup_{i \leq n} F_i \cup \bigcup_{i \leq n} C_i\).

The CRPQ\(S\) query \(\varphi(x, y)\) is given below; it omits path variables for paths that are not used in comparisons:

\[
\exists x_1, x_2, z_1, z_2 \left( x \xrightarrow{\#} x_1 \land x \xrightarrow{\$} x_2 \land \begin{array}{l}
\begin{array}{l}
x_1 \xrightarrow{\Sigma^*} z_1 \\
x_2 \xrightarrow{\Sigma^*} z_2 \\
z_1 \xrightarrow{!} y \\
z_2 \xrightarrow{!} y \\
x_1 \xrightarrow{\$} x_2 \\
S(x, \chi')
\end{array}
\end{array} \right)
\]

It is routine to verify that \(G \models \varphi(\text{start}, \text{end})\) iff \(R \cap S \neq \emptyset\).

Combining the lemmas, we obtain:

Theorem IV.3. Let \(K\) be a complexity class closed under DLOGSPACE reductions. Then:
1) If the problem \(\text{GENINTS}(\text{REC})\) is in \(K\), then data complexity of CRPQ\(S\) queries is in \(K\); and
2) If the problem \((\text{REC} \cap_1 S) \triangleq \emptyset\) is hard for \(K\), then so is data complexity of CRPQ\(S\) queries.

We now consider extended CRPQ\(S\), or ECRPQ\(S\), which enhance CRPQ\(S\) with regular relations \([3]\), and prove a similar result for them, with the role of \(\text{REC}\) now played by \(\text{REG}\). Formally, ECRPQ\(S\) are expressions of the form

\[
\varphi(\bar{x}) = \exists \bar{y} \left( \bigcap_{i=1}^{m}(u_i \xrightarrow{\chi_i} u'_i) \land \bigcap_{j=1}^{k} R_j(\bar{x}_j) \right)
\]

where each \(R_j\) is a relation from \(\text{REG}\), and \(\bar{x}_j\) a tuple from \(\Sigma_1, \ldots, \Sigma_m\) of the same arity as \(R_j\). The semantics of course extends the semantics of CRPQ\(S\): the witnessing paths \(\rho_1, \ldots, \rho_m\) should also satisfy the condition that for every atom \(R(\rho_1, \ldots, \rho_i)\) in (3), the tuple \((\lambda(\rho_1), \ldots, \lambda(\rho_i))\) is in \(R\).

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

\[
\exists \bar{y} \left( \bigcap_{i=1}^{m}(u_i \xrightarrow{\chi_i} u'_i) \land \bigcap_{j=1}^{k} R_j(\bar{x}_j) \land \bigcap_{(i,j) \in I} S(\bar{x}_i, \bar{x}_j) \right)
\]

Exact analogs of Lemmas IV.1 and IV.2 hold, with ECRPQ\(S\) replacing CRPQ\(S\) and \(\text{REG}\) replacing \(\text{REC}\). Hence, we get:

Theorem IV.4. Let \(K\) be a complexity class closed under DLOGSPACE reductions. Then:
1) If the problem \(\text{GENINTS}(\text{REC})\) is in \(K\), then data complexity of ECRPQ\(S\) queries is in \(K\); and
2) If the problem \((\text{REC} \cap_1 S) \triangleq \emptyset\) is hard for \(K\), then so is data complexity of ECRPQ\(S\) queries.

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.

V. THE INTERSECTION PROBLEM: DECIDABLE AND UNDECIDABLE CASES

We now study the problem \((\text{REC} \cap_1 S) \triangleq \emptyset\) 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 \(\Sigma\), given by an NFA over \(\Sigma \times \Sigma\). The question is whether \(R \cap S \neq \emptyset\). We also derive results about the complexity of 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 \((\text{REC} \cap_1 S) \triangleq \emptyset\) 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 $(\text{REC} \cap S) \not\subseteq \emptyset$ is tractable: given $R \in \text{REC}$, the relation $R \cap S$ is rational, can be efficiently constructed, and checked for nonemptiness.

A. 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 $\Sigma_1 \times \Sigma_1$ 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 V.1.** The problems $(\text{REG} \cap \preceq_{\text{suff}}) \not\subseteq \emptyset$ and $(\text{REG} \cap \preceq) \not\subseteq \emptyset$ are undecidable.

As an immediate consequence of this, we obtain:

**Corollary V.2.** The query evaluation problem for $\text{ECRPQ}(\preceq_{\text{suff}})$ and $\text{ECRPQ}(\preceq)$ is undecidable.

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

**Proof idea.** We sketch the idea of the proof for $\preceq_{\text{suff}}$. We encode nonemptiness for linearly bounded automata (LBA). The alphabet $\Sigma$ is the disjoint union of the tape alphabet of the LBA, its states, and the designated symbol $8$. Each configuration $C$ has the tape content $a_0 \ldots a_n$, where $a_0$ and $a_n$ are the left and right markers, the state is $q$, and the head points at $a_i$ is encoded as a word $w_C = \rho a_0 \ldots a_{i-1} q a_i \ldots a_n 8$. Note that the relation $\{(w_C, w_{C'}) \mid C'$ is an immediate successor of $C\}$ is regular and hence so is the relation $R = \{(w_{C_0}, w_{C_1}, \ldots, w_{C_m}, w_{C'_1}, \ldots, w_{C'_m}) \mid C_{i+1}$ is an immediate successor of $C_i$ for $i < m\}$, since all configuration encodings are of the same length. In fact, taking product with a regular language, we can also assume that $R$ enforces $C_0$ to be an initial configuration, and $C_m$ to be a final configuration. If $R \cap \preceq_{\text{suff}}$ is nonempty, it contains a pair $(w_{C_0}, w_{C_1}, \ldots, w_{C_m}, w_{C'_1}, \ldots, w_{C'_m})$ such that $C_{i+1}$ is an immediate successor of $C_i$ for all $i < m$, i.e., iff there is an accepting computation of the LBA. This proves undecidability. The proof for $\preceq$ is very similar. 

Note that the relation $R$ constructed in the proof is definable in first-order logic, so the intersection problem for suffix and subword is undecidable even if the input relation comes from the class of star-free regular relations.

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 $\{(w, w', u, u') \mid (w, u) \in R$ and $(w', u') \in R'\}$. Let $\text{Graph}(f)$ be the graph of a function $f : \Sigma^* \rightarrow \Sigma^*$, i.e., $\{(w, f(w)) \mid w \in \Sigma^*\}$.

**Proposition V.3.** Let $R_0, R_1$ be binary relations on $\Sigma$ such that $R_0$ is recognizable and its second projection is $\Sigma^*$. Let $f$ be a strictly alphabetic morphism that is not constant. Then, for $S = R_0 \cdot \text{Graph}(f) \cdot R_1$, the problem $(\text{REG} \cap S) \not\subseteq \emptyset$ is undecidable.

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

B. 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, for relations one of whose projections is closed under taking subsequences, we also retain decidability. 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 $\text{REG}_2$, is not bounded by any multiply-recursive function. This was previously known for $R$ ranging over $\text{RAT}_2$, and was viewed as the simplest problem with non-multiply-recursive complexity [12]. 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 decidability relaxation of the Post Correspondence Problem (PCP), namely the regular Post Embedding Problem, or $\text{PEP}^{\text{reg}}$, introduced in [12], and shown to be in the level $\text{F}_{\omega_2}$ of the fast-growing hierarchy of recursive functions [27], [29]. The input to the problem consists of two morphisms $f, g : \Sigma^* \rightarrow \Gamma^*$ and a regular language $L \subseteq \Sigma^*$; it asks whether there is some $w \in L$ such that $f(w) \sqsubseteq g(w)$ (recall that in the case of the PCP the question is whether $f(w) = g(w)$ with $L = \Sigma^*$). This problem is known to be decidable, and as hard as the reachability problem for lossy channel systems [12] which cannot be bounded by any primitive-recursive function —in fact, by any multiple-recursive function [29].

The problem $\text{PEP}^{\text{reg}}$ is just a reformulation of the problem $(\text{RAT} \cap \sqsubseteq) \not\subseteq \emptyset$. 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 $\Sigma^* \rightarrow \Gamma^*$ are precisely the relations in $\text{RAT}_2$ [5], [28]). Hence, $(\text{RAT} \cap \sqsubseteq) \not\subseteq \emptyset$ is decidable, with non-multiply-recursive complexity. We show that the lower bound already applies to regular relations.

**Theorem V.4.** The problem $(\text{REG} \cap \sqsubseteq) \not\subseteq \emptyset$ is decidable, and its complexity is not bounded by any multiply-recursive function.

**Proof idea.** As already mentioned, decidability follows from [12]. To prove the lower bound, we first show that the existence of a solution to $\text{PEP}^{\text{reg}}$ is equivalent to the existence of a special solution, which we call a strict codirect solution.¹

A word $w = a_1 \cdots a_m \in \Sigma^*$ is a strict codirect solution if $f(a_1 \cdots a_m) \sqsubseteq g(a_1 \cdots a_m)$ and for every $i < m$, $f(a_1 \cdots a_i) \sqsubseteq g(a_1 \cdots a_i)$. We then show how to code

¹This is a slightly more restrictive definition than the codirect solutions used for the decidability of $\text{PEP}^{\text{reg}}$ in [12], which is essential to make possible our reduction from the $(\text{REG} \cap \sqsubseteq) \not\subseteq \emptyset$ problem.
Thus the theorem follows.

Given a rational relation \( R \subseteq \Sigma^* \times \Gamma^* \) we convert it into a length-preserving regular relation \( R' \subseteq \Sigma^*_1 \times \Gamma^*_1 \) so that \( R \) is the set of elements of \( R' \) projected onto \( \Sigma^* \times \Gamma^* \), and if \( (q, (a, b), q') \) is a transition of the NFA accepting \( R' \) so is \( (q, (\bot, \bot), q') \). If we now let \( R'' \) to be the regular relation \( R' \cap \{(e, v) \mid v \in \{\bot\}^*\} \), we obtain that:

(i) if \( w \in R'' \subseteq \Sigma^* \times \Gamma^* \), then \( w'' \in R \subseteq \Sigma^*_1 \times \Gamma^*_1 \), and \( w'' \) is the projection of \( w' \) onto \( \Sigma^*_1 \times \Gamma^*_1 \); and

(ii) for any \( w' \in R \subseteq \Sigma^* \times \Gamma^* \) there is some strict codirect solution \( w' \in R' \subseteq \Sigma^*_1 \times \Gamma^*_1 \), such that \( w'' \) is the projection of \( w' \) onto \( \Sigma^*_1 \times \Gamma^*_1 \).

Thus the theorem follows.

Whereas (i) is trivial, (ii) follows from the fact that \( w \) is a strict codirect solution. If \( w = (u, v) \in R'' \), where \( f(w) = (u)_T \), \( q(w) = (v)_T \), the complication is now that, since \( u \in \Sigma^*_1 \), it could be that \( u \not\subseteq v \) just because there is some \( \bot \) in \( u \) that does not appear in \( v \). But we build \( (u, v) \) such that whenever \( u[i] = \bot \) forces \( v[j] = \bot \) with \( j > i \) then we also have that \( u[j] = \bot \). This repeats, forcing \( v[k] = \bot \) for some \( k > j \) and so on, until we reach the tail of \( v \) that has sufficiently many \( \bot \)'s to satisfy all the accumulated demands for occurrences of \( \bot \). □

Note that one cannot solve the problem \( (\text{REG} \cap \subseteq) \) by reducing to nonemptiness of rational relations due to the following.

**Proposition V.5.** There is a binary regular relation \( R \) such that \( (R \cap \subseteq) \) is not rational.

The next question is how far we can extend the decidability of \( (\text{RAT} \cap \subseteq) \). 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

\[
R_{\subseteq} = \{(u, w) \mid u \subseteq u' \text{ and } (u', w) \in R \text{ for some } u'\}
\]

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

The following summarizes properties of subsequence-closed relations.

**Proposition V.6.**

1) SCR \( \subseteq \text{ RAT} \).

2) SCR \( \not\subseteq \text{ REG} \) and \( \text{ REG} \not\subseteq \text{ SCR} \).

3) A relation \( R \) in SCR if \( \{w \cap w' \mid (w, w') \in R\} \) is accepted by an NFA \( A = (Q, \Sigma^*_1 \times \Sigma^*_1, \delta, q_0, \delta, F) \) such that \( q, (a, b), q' \in \delta \) implies \( q, (\bot, b), q' \in \delta \) for all \( q, q' \in Q \) and \( a, b \in \Sigma^*_1 \).

When an SCR relation is given as an input to a problem, we assume that it is represented as an NFA in item 3 in the above proposition.

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

The main result about SCR relations generalizes decidability of \( (\text{RAT} \cap \subseteq) \) to \( \emptyset \).

**Theorem V.7.** The problem \( (\text{RAT} \cap \subseteq) \) is decidable.

Of course the complexity is non-multiply-recursive, since this subsumes \( (\text{REG} \cap \subseteq) \) of Theorem V.4.

**Proof idea.** Decidability is shown by reduction to a problem where solutions have a specific shape. Given a rational binary relation \( R_0 \) and a subsequence-closed relation \( R_1 \) defined by two automata \( A_0 \) and \( A_1 \) over \( \Sigma^*_1 \times \Sigma^*_1 \), we say that \( (w_0, w_1) \) is a solution if \( w_0 = u_0 \cap v_0 \in L(A_0), w_1 = u_1 \cap v_1 \in L(A_1) \) and \( (u_0)_0 = (u_1)_0, (v_0)_0 = (v_1)_0 \), where \( w_0 \) denotes projection of \( w \) onto \( \Sigma \). We say that \( (w_0, w_1) \) is a synchronized solution if it further satisfies \( v_0 = v_1 \).

The problem of finding a solution reduces to that of finding a synchronized solution. Indeed, consider the automata \( A_0', A_1' \) as the result of adding all transitions \( (q, (\bot, \bot), q) \) for every possible state \( q \) to both automata. It is clear that the relations recognized by these automata remain unchanged, and that \( A_0' \) is still a SCR automaton. It is easy to verify that there is a synchronized solution for \( (A_0', A_1') \) if, and only if, there is a solution for \( (A_0, A_1) \).

The problem of finding a synchronized solution for \( A_0, A_1 \) can be then formulated as the problem of finding words \( v, u_0, u_1 \in \Sigma^*_1 \) with \( |v| = |u_0| = |u_1| \), so that \( (u_0 \cap v, u_1 \cap v) \) is a solution. We can compute an automaton \( A \) over \( \Sigma^*_1 \) from \( A_0, A_1 \), such that \( (u_0, u_1, v) \in L(A) \) if, and only if, \( (u_0 \cap v, u_1 \cap v) \in L(A_0) \) and \( (u_0 \cap v, u_1 \cap v) \in L(A_1) \). Consider now an automaton \( A' \) over \( \Sigma^*_1 \) such that \( L(A') = \{(u_0, u_1) \mid \exists v (u_0, u_1, v) \in L(A)\} \). It corresponds to the rational automaton of the projection onto the first and second components of the ternary relation of \( A \), and it can be computed from \( A \) in polynomial time. We then deduce that there exists \( u_0 \cap u_1 \in L(A') \) so that \( (u_0 \cap u_1)_0 \subseteq (u_1)_0 \); if, and only if, there is \( v \in \Sigma^*_1 \) with \( |v| = |u_0| = |u_1| \) so that \( u_0 \cap v \in L(A_0) \) and \( u_1 \cap v \in L(A_1) \). Consider \( u_0 \cap v \in L(A_0) \), \( u_1 \cap v \in L(A_1) \), where \( (u_0)_0 \subseteq (u_1)_0 \). But this is equivalent to \( R_0 \cap R_1 \neq \emptyset \), since

- if \( ((u_1)_0, (v)_0) \in R_1 \) and \( (u_0)_0 \subseteq (u_1)_0 \), then

- if \( R_0 \cap R_1 \neq \emptyset \), then there exists a synchronized solution

We have thus reduced the problem to \( (\text{RAT} \cap \subseteq) \) which is decidable [12], as already mentioned. □

Coming back to graph logics, we obtain:
Corollary V.8. The complexity of evaluation of ECRPQ(⊆) queries is not bounded by a multiply-recursive function.

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

Corollary V.9. Let C be a class of binary relations on Σ* that is closed under intersection and contains REG. Then the nonemptiness problem for C is:

- undecidable if ⊆ or ⊆ suff is in C;
- non-multiply-recursive if ⊆ is in C.

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 ruled 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 GENINT_S(REG) for such relations S).

Even more importantly, we do not yet know what happens with the complexity of CRPQs (i.e., GENINT_S(REG) for various relations S). These questions are answered in the next section.

VI. RESTRICTED LOGICS AND 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 GENINT_S(REG), corresponding to CRPQs extended with relations S on paths.

A. Generalized intersection problem and subsequence

We know that (REG ∩ ⊆) Ʉ= ∅ is decidable, although not multiply-recursive. What about its generalized version? It turns out it remains decidable.

Theorem VI.1. The problem GENINT_S(REG) is decidable. That is, there is an algorithm that decides, for a given m-ary regular relation R and I ⊆ [m]^2, whether R ∩ I ⊆ ∅.

For checking decidability we explore the solution space tree as in the proof of Theorem V.7. However, this time the notion of saturation is different, since we need to consider a different condition for each component of the m-ary relation.

The finiteness of the computed tree follows from Higman’s Lemma this time in combination with Dickson’s Lemma [19].

Corollary VI.2. The query evaluation problem for ECRPQ(⊆) queries is decidable.

Of course the complexity is extremely high as we already know from Corollary V.8.

Note that while the intersection problem of ⊆ with rational relations is decidable, as is GENINT_S(REG), we lose the decidability of GENINT_S(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 VI.3. The problem (RAT ∩I ⊆) Ʉ= ∅ is undecidable even over ternary relations when I is one of the following: {1, 2}, {1, 2, 3}, or {1, 2}, or {1, 2, 3, 2}.

B. Generalized intersection problem for recognizable relations

We now consider the problem of answering CRPQs with rational relations S, or, equivalently, the problem GENINT_S(REG). Recall that an instance of such a problem consists of an m-ary recognizable relation R and a set I ⊆ [m]^2. The question is whether R ∩ I S ̸= ∅, i.e., whether there exists a tuple (w_1, ..., w_m) ∈ R such that (w_i, w_j) ∈ I whenever (i, j) ∈ 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 GENINT_S(REG) 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 ∈ REC is a union of relations of the form L_1 × ... × L_m, where each L_i is a regular language. Hence, as the representation of R we take the set of all such L_s involved, and as the measure of its complexity, the total size of NFAs defining the L_s.

With a set I ⊆ [m]^2 we associate an undirected graph G_I whose nodes are 1, ..., m and whose edges are {i, j} such that either (i, j) ∈ I or (j, i) ∈ I. We call an instance of (REC ∩I S) Ʉ= ∅ acyclic if G_I is an acyclic graph.

Now we can state the dichotomy result.

Theorem VI.4.

Let S be a binary rational relation. Then acyclic instances of GENINT_S(REG) are decidable in PSPACE. Moreover, there is a fixed binary relation S_0 such that the problem (REC ∩I S_0) Ʉ= ∅ is PSPACE-complete.

For every I such that G_I is not acyclic, there exists a binary rational relation S such that the problem (REC ∩I S) Ʉ= ∅ is undecidable.

Proof idea. For PSPACE-hardness we can do an easy reduction from nonemptiness of NFA intersection. Given m NFAs A_1, ..., A_m, define the (acyclic) relation I = {(i, i+1) | 1 ≤ i < m}. Then ∩_i L(A_i) is nonempty iff ∩_i L(A_i) ∩ I S_0 ̸= ∅, where S_0 = {(w, w) | w ∈ Σ^*}.
For the upper bound, we show how to construct, in exponential time, for each \( m \)-ary recognizable relation \( R \), a binary rational relation \( S \) and an acyclic \( I \subseteq [m]^2 \), an \( m \)-ary transducer \( \mathcal{A}(R, S, I) \) 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 transducer 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 \( \text{GENINT}_S(\text{REC}) \) we only need to check \( \mathcal{A}(R, S, I) \) for nonemptiness. The latter can be done in \( \text{PSPACE} \) by the standard “on-the-fly” reachability analysis.

\[ \square \]

C. CRPQs 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. Recall that CRPQ(\( S \)) queries are those of the form

\[ \varphi(\bar{x}) = \exists \bar{y} \left( \bigwedge_{i=1}^{m} (u_i, \chi_i \rightarrow u_i') \land \bigwedge_{(i, j) \in I} S(\chi_i, \chi_j) \right), \]

see (2) in Sec.IV. We call such a query acyclic if \( G_I \), the underlying undirected graph of \( I \), is acyclic.

**Theorem VI.5.** The query evaluation problem for acyclic CRPQ(\( S \)) queries is decidable for every binary rational relation \( S \). Its combined complexity is \( \text{PSPACE} \)-complete, and data complexity is \( \text{NLOGSPACE} \)-complete.

Thus, we get not only the possibility of extending CRPQs with rational relations but also a good complexity of query evaluation. The \( \text{NLOGSPACE} \)-data complexity matches that of RPQs, CRPQs, and ERCPQs \([15, 16, 3]\), and the combined complexity matches that of first-order logic, or ERCPQs 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 VI.4.

**Theorem VI.6.**

- Let \( S \) be a relation from SCR. Then \( (\text{REC} \cap \bigwedge S) \not\supseteq \emptyset \) is decidable in \( \text{NEXPTIME} \) if \( I \) is a directed acyclic graph.
- There is a relation \( I \) with a directed cycle and \( S \in \text{SCR} \) such that \( (\text{REC} \cap \bigwedge S) \not\supseteq \emptyset \) is undecidable.

In particular, if we have a CRPQ(\( S \)) query (2) where \( I \) is acyclic (as a directed graph) and \( S \in \text{SCR} \), then query evaluation has \( \text{NEXPTIME} \) combined complexity.

The proof of this result is quite different from the upper bound proof of Theorem VI.4, 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 VI.6, we get the following result:

**Proposition VI.7.** Let \( S \in \text{SCR} \) be a partial order. Then \( \text{GENINT}_S(\text{REC}) \) is decidable in \( \text{NEXPTIME} \).

**Corollary VI.8.** If \( S \in \text{SCR} \) is a partial order, then CRPQ(\( S \)) queries can be evaluated with \( \text{NEXPTIME} \) combined complexity. In particular, CRPQ(\( \subseteq \)) queries have \( \text{NEXPTIME} \) combined complexity.

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 VI.9.** The problem \( \text{GENINT}_{\subseteq \text{suff}}(\text{REC}) \) is decidable in \( \text{NEXPTIME} \). In particular, CRPQ(\( \subseteq \)) queries can be evaluated with \( \text{NEXPTIME} \) combined complexity.

VII. 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 \( \leq \), suffix \( \leq \text{suff} \), and subsequence \( \subseteq \), 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. 1. 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.

So far we dealt with the classical setting of graph data \([1, 8, 9, 15, 16]\) 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 \([23, 25]\). A natural question is to extend the positive results shown here to such a setting.

Acknowledgments: We thank Sylvain Schmitz for helpful comments and suggestions. Partial support provided by Fondecyt grant 1110171, EPSRC grant G049165, and FET-Open Project FoX, grant agreement 233599. Part of this work was done when the first author visited Edinburgh, and the third author visited Santiago.
Fig. 1. Complexity of the intersection and generalized intersection problems, and combined complexity of graph queries for subword ($\subseteq$), suffix ($\preceq_{\text{ suff}}$), and subsequence ($\subseteq$) relations.

<table>
<thead>
<tr>
<th>$R \in \text{REC}$</th>
<th>$S = \subseteq$</th>
<th>$S = \preceq_{\text{ suff}}$</th>
<th>$S = \subseteq$</th>
<th>$S$ arbitrary in $\text{RAT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R $\cap \subseteq$) $\not\subseteq$</td>
<td>$\text{PTIME}$ (cf. [5])</td>
<td>undecidable</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>(R $\cap \preceq_{\text{ suff}}$) $\not\subseteq$</td>
<td>decidable, non-multiply-recursive</td>
<td>decidable, non-multiply-recursive</td>
<td>[12]</td>
<td></td>
</tr>
<tr>
<td>(R $\cap \subseteq$) $\not\subseteq$</td>
<td>$\text{NP}$</td>
<td>$\text{undecidable}$</td>
<td>$\text{undecidable}$</td>
<td>$\text{undecidable}$</td>
</tr>
<tr>
<td>(R $\cap \preceq_{\text{ suff}}$) $\not\subseteq$</td>
<td>$\text{NP}$</td>
<td>$\text{undecidable}$</td>
<td>$\text{undecidable}$</td>
<td>$\text{undecidable}$</td>
</tr>
</tbody>
</table>

REFERENCES