Complexity of Conjunctive Regular Path Query Homomorphisms

Laurent Beaudou, Florent Foucaud, Florent Madelaine, Lhouari Nourine, Gaétan Richard

Complexity of conjunctive regular path query homomorphisms*

Laurent Beaudou[1,2], Florent Foucaud[1], Florent Madelaine[3][0000−0002−8528−7105], Lhouari Nourine[1], and Gaétan Richard[4]

[1] LIMOS, Université Clermont Auvergne, Aubière, France
[2] National Research University, Higher School of Economics, 3 Kochnovsky Proezd, Moscow, Russia
[3] LACL, Université Paris-Est Créteil, France
[4] GREYC, Université Caen Normandie, France

Abstract. A graph database is a digraph whose arcs are labelled with symbols from a fixed alphabet. A regular graph pattern (RGP) is a digraph whose edges are labelled with regular expressions over the alphabet. RGPs model navigational queries for graph databases, more precisely, conjunctive regular path queries. A match of a navigational RGP query in the database is witnessed by a special navigational homomorphism of the RGP to the database. We study the complexity of deciding the existence of a homomorphism between two RGPs. Such homomorphisms model a strong type of containment between two navigational RGP queries. We show that this problem can be solved by an EXPTIME algorithm (while general query containment in this context is EXPSPACE-complete). We also study the problem for restricted RGPs over a unary alphabet, that arise from some applications like XPath, and prove that certain interesting cases are polynomial-time solvable.

Keywords: complexity · navigational homomorphism · regular graph pattern

1 Introduction

Graphs are a fundamental way to store and organize data. Most prominently, graph database systems have been developed for three decades and are widely used; recently, such systems have seen an increased interest both in academic research and in industry [3]. A graph database can be seen as a directed graph with arc-labels (possibly also vertex-labels). Various methods are used to retrieve data in such systems, see for example the very recently developed graph query language G-CORE [2] for graph databases. Classically, matching queries

* The authors acknowledge the support received from the Agence Nationale de la Recherche of the French government throught the program "Investissements d'Avenir"(16-IDEX-0001 CAP 20-25), the IFCAM project "Applications of graph homomorphisms" (MA/IFCAM/18/39), and by the ANR project HOSIGRA (ANR-17-CE40-0022).

in graph databases can be modeled as graph homomorphisms [16]. In this setting, a query is itself a graph, and a match is modeled by a homomorphism of the query to the database, that is, a vertex-mapping that preserves the graph adjacencies and labels. Graph databases can be very large, thus it is important to study the algorithmic complexity of such queries. In modern applications, classic homomorphisms are often not powerful enough to model realistic graph data queries. In recent years, *navigational queries* have been developed [3]. Such queries are more powerful than classical queries, since they allow for non-local pattern matching, by means of arbitrary paths or walks instead of arcs. Such queries can also be modeled as a more general kind of homomorphism, called *navigational homomorphism* (*n-homomorphism* for short). The most studied type of navigational queries is the one of *regular path queries*, that is based on regular expressions [3,10,17]. A *conjunctive regular path query* is modeled by a *regular graph pattern* (RGP for short), that is, a digraph whose arcs are labelled by regular expressions, each representing a regular path query. The associated notion of navigational homomorphism is called *RGP homomorphism*. The study of the algorithmic complexity of RGP homomorphisms has been recently initiated in [19], where the authors focused on homomorphisms of RGPs to graph databases. In the present paper, we continue this study by focusing on conjunctive regular path query containment, as modeled by the existence of a RGP homomorphism between two queries.

We delay to the next section for formal definitions. We consider the following decision problem

RGPHom Input: Two RGPs P and Q. Question: Does P admit an n-homomorphism to Q?

and its non-uniform version, defined as follows for a fixed RGP Q.

RGPHom(Q) Input: A RGP P. Question: Does P admit an n-homomorphism to Q?

The latter was introduced in [19] for the restricted case where Q is a graph database (*i.e.* labels are letters rather than more complex regular expressions) – which amounts to *RGP evaluation* – and showed that this class of problems follows a dichotomy between Ptime and NP-complete. Indeed they showed it to be equivalent to the (classical) homomorphism problems a.k.a. the constraint satisfaction problems [9], whose complexity delineation follows a dichotomy based on specific algebraic properties of the template Q, as shown independently by Bulatov [5] and Zhuk [22].

In this paper, we initiate the study of these problems in full generality, that is when Q is not a graph database but any RGP. We cannot expect a Ptime/NP-complete dichotomy in the style of the result of [19], since RGPHom(Q) is in fact PSPACE-hard already for very simple cases, as it can model the problem of

deciding the inclusion between regular languages. We detail these lower bounds in Section 3.

We show in Section 4 that RGPHom is decidable by an EXPTIME algorithm. This shows that n-homomorphism-based query containment is less expensive than general query containment, which, in the case of RGP queries, is known to be EXPSPACE-complete [7,10].

In Section 5, we address the simpler case of a unary alphabet $\Sigma = \{a\}$, when all arcs are labelled either by "a" or "a^+". This includes not only all classic homomorphism problems and CSPs, but also queries over hierarchical data reminiscent of SPARQL and XPath [8,15,18]. In particular, we give a simple Ptime/NP-complete complexity dichotomy for the case of undirected (or symmetric) RGPs in the style of Hell and Nešetřil's dichotomy for H-colouring [11]. Furthermore, we show that even for arbitrary (directed) RGPs the problem follows a dichotomy by relating it to (classical) homomorphism problems. Finally, we focus on certain queries of interest. We relate the case of path templates that have only "a" labels to an interesting (Ptime) scheduling problem. We extend this result and show also that for all directed path RGP templates Q with arc labels "a" or "a^+", RGPHom(Q) is in Ptime.

2 Preliminaries

Let Σ be a fixed countable alphabet. A *graph database* $B = (D_B, E_B)$ over Σ is an arc-labelled digraph, where D_B is a finite digraph with vertex set $V(D_B)$ and arc set $A(D_B)$, and $E_B : A(D_B) \to \Sigma$ is an arc-label function. We may adapt the notion of graph homomorphism to graph databases following [16] and view the existence of a homomorphism from a graph database $Q = (D_Q, E_Q)$ – which models a *query* – to $B = (D_B, E_B)$ as the fact that *the database B matches the query Q*. A homomorphism is a mapping f of $V(D_Q)$ to $V(D_B)$ such that for every arc (x, y) in D_Q, there is an arc $(f(x), f(y))$ in D_B with $E_Q(x, y) = E_B(f(x), f(y))$. If such a homomorphism exists, we note $Q \to B$. Every homomorphic image $f(Q)$ of Q to B is a match of the query Q in B. In this classic setting, queries and graph databases coincide and the existence of a homomorphism between queries models *query containment*. Thus, the *evaluation problem* (deciding whether a query Q has a match in a database B) and the *containment problem* (deciding whether for two queries Q_1 and Q_2, for any database B, if B matches Q_1 then B matches Q_2) both amount to the following decision problem.

Hom
Input: Two arc-labelled digraphs G and H.
Question: Does G admit a homomorphism to H?

Hom is generally NP-complete, even when H is a small fixed graph (for example a symmetric triangle, in which case Hom is equivalent to the graph 3-colourability problem). To better understand the complexity of Hom, the follow-

ing version, called the *non-uniform* homomorphism problem, has been studied extensively. Here, H is a fixed arc-labelled digraph called the *template*.

HOM(H) Input: An arc-labelled digraph G. Question: Does G admit a homomorphism to H?

In a digraph D, a *directed walk* is a sequence of arcs of the digraph, such that the head of each arc is the same vertex as the tail of the next arc. A *directed path* is a directed walk where each vertex occurs in at most two arcs in the sequence.

Standard homomorphisms are not powerful enough to model all queries used in modern graph database systems. In particular, a homomorphism of a query Q to a database B can only match a subgraph of B that is no larger than the query Q itself. To the contrary, *navigational queries* are types of queries where we may allow arbitrarily large subgraphs of the database to match the query. In this setting, we still model the query Q (for database B) as an arc-labelled digraph, but the arcs are labelled with *sets of words* over the alphabet Σ, rather than letters. Now, a match of Q in B is a vertex-mapping f from $V(D_Q)$ to $V(D_B)$ such that for an arc (x, y) of Q labelled with a set $E(x, y)$ of words, there exists a directed walk (or path, depending on applications) W_{xy} in D_B from $f(x)$ to $f(y)$ such that the concatenation of labels of the arcs of W_{xy} is a word of $E(x, y)$.

Perhaps the most popular navigational queries are *conjunctive regular path queries*, studied in many contexts [3,4,8,15,18,19]. These navigational queries are based on regular languages: the labels on query arcs are regular expressions over the alphabet Σ. The advantage of considering such queries is that regular languages are a relatively simple yet powerful way of defining sets of words, that is both well-understood and sufficiently expressive for many applications. A conjunctive regular path query of this type has been called a *regular graph pattern* (RGP).

For a fixed countable alphabet Σ, we denote by $RegExp(\Sigma)$ the set of regular expressions over alphabet Σ, with the symbols $+$ (union), $*$ (Kleene star), and $\cdot$ (concatenation; sometimes this symbol is omitted). Moreover, for a regular expression X, as a notation we let $X^+ := X \cdot X^*$. For any regular expression X in $RegExp(\Sigma)$, we denote by $L(X)$ the regular language defined by X. We will use the following decision problems for regular languages.

REGULAR LANGUAGE INCLUSION Input: Two regular expressions E_1 and E_2 (over the same alphabet). Question: Is $L(E_1) \subseteq L(E_2)$?

REGULAR LANGUAGE UNIVERSALITY Input: A regular expression E over alphabet Σ. Question: Is $L(E) = \Sigma^*$?

Note that Regular Language Universality is the special case of Regular Language Inclusion where $E_1 = \Sigma^*$ and $E_2 = E$. The following are classic results.

Theorem 1 ([1,21]). Regular Language Universality *and* Regular Language Inclusion *are PSPACE-complete.*

A RGP P over an alphabet Σ is a pair (D_P, E_P), where D_P is a digraph with vertex set $V(D_P)$ and arc set $A(D_P)$ and $E_P : A(D_P) \to RegExp(\Sigma)$ is an arc-label function. Given a directed walk $W = a_{1,2} \dots a_{k-1,k}$ in a RGP P, the label $E_P(W)$ of W is the regular expression over Σ formed by the concatenation $E_P(a_{1,2}) \dots E_P(a_{k-1,k})$.

A *sub-RGP* P' of P is induced by a subset $V(D_{P'})$ of $V(D_P)$ where $A(D_{P'})$ is $A(D_P) \cap V(D_{P'}) \times V(D_{P'})$ and the arc label function $E_{P'}$ is the restriction of E_P to $A(D_{P'})$.

Given two RGPs P and Q over alphabet Σ, a *navigational homomorphism* (*n-homomorphism* for short) of P to Q is a mapping f of $V(D_P)$ to $V(D_Q)$ such that for each arc (x, y) in D_P, there is a directed walk W in Q from $f(x)$ to $f(y)$ such that the language $L(E_Q(W))$ is contained in the language $L(E_P(x, y))$. When such an n-homomorphism exists, we write $P \xrightarrow{n} Q$. It is not hard to see that $\xrightarrow{n}$ is transitive.

This definition also applies to graph databases and amounts in fact to *RGP query evaluation* when Q is a graph database and P an RGP.

A digraph D is called a *core* if it has no homomorphism to a proper subdigraph of itself; in other words, every endomorphism is an automorphism. Similarly we define the notion of a *navigational core* (*n-core* for short): a RGP P is an n-core if it has no n-homomorphism to a proper sub-RGP of itself. When studying the problem RGPHom(Q), we may always assume that Q is an n-core, since RGPHom(Q) has the same complexity as RGPHom(C_Q), where C_Q is a sub-RGP of Q that is an n-core. Unfortunately, it is coNP-complete to decide whether a graph is a core [12] (thus deciding whether a RGP is an n-core is coNP-hard, even if it is a graph database).

With respect to classic digraph homomorphisms, any digraph has (up to isomorphism) a unique minimal subgraph to which it admits a homomorphism, called *the* core. This is not the case for n-cores of RGPs and n-homomorphisms. For example, any two RGPs each consisting of a unique directed cycle with all arc labels equal to "a^+" have an n-homomorphism to each other. Thus, if we identify one vertex of two such cycles of different lengths, we obtain a RGP P with two minimal sub-RGPs of P (the two cycles) to which P has an n-homomorphism, thus these two are non-isomorphic n-cores of P.[5]

[5] There exist more complicated examples where, furthermore, the two non-isomorphic n-cores have the same size.

3 Lower Bounds

The next proposition is proved by a very simple reduction from Regular Language Inclusion to RGPHom for two RGPs each having a single arc.

Proposition 1. RGPHom *is PSPACE-hard.*

As witnessed by the simplicity of the reduction given in Proposition 1, the PSPACE-hardness of RGPHom is inherently caused by the hardness of the underlying regular language problem. This phenomenon arises also in the non-uniform case for a very simple template.

Proposition 2. *Let Σ be a fixed alphabet of size at least 2, and let D_2^Σ be the RGP of order 2 over Σ consisting of a single arc labelled Σ^*. Then,* RGPHom(D_2^Σ) *is PSPACE-complete.*

4 An EXPTIME algorithm for RGPHom

In certain models where *simple directed paths* rather than directed walks are considered, like in [17], or when the target RGP is acyclic, there is a simple PSPACE algorithm to decide RGPHom. Indeed, in those cases, the length of a walk in Q is at most $|V(D_Q)|$. Thus, we can iterate over each possible mapping f and for each mapped arc (x, y), we iterate over each possible walk W from $f(x)$ to $f(y)$, and check in polynomial space whether $L(E_Q(W)) \subseteq L(E_P(x, y))$.

However, in general, the walks may be arbitrarily long. As we will see, we can still bound their maximum length. Note that for two RGPs P and Q, if $P \xrightarrow{n} Q$ then the query Q is contained in the query P, but there are examples where the converse does not hold (see Figure 1). Thus, the problem RGPHom for two RGPs does not fully capture RGP Query Containment. Nevertheless, we will show that the former can be solved in EXPTIME, which is better than the (tight) EXPSPACE complexity of RGP Query Containment shown in [7,10].

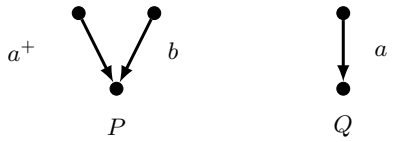

Fig. 1. Two n-core RGPs P and Q over alphabet $\{a, b\}$ which have no n-homomorphism in either direction. From P to Q because one can not map suitably the arc labelled by b, in the other direction because neither b nor a^+ is included in a. However, any database that matches the RGP P would contain a walk of arcs all labelled by a (because of the arc with label a^+ in P). The database would clearly also match Q. So Q is contained in P.

For a regular language L over alphabet Σ and a positive integer n, we denote by $L_{|n}$ the *n-truncation* of L, *i.e.* the set of words of L with length at most n.

Lemma 1. *Let A, $B_1, \dots B_k$ be a collection of regular expressions over alphabet Σ, and let n_A, n_i be the minimum number of states of an NFA recognizing $L(A)$ and $L(B_i)$, respectively. Then, we have that $L(B_1) \cdots L(B_k) \subseteq L(A)$ if, and only if, $L(B_1)_{|n_A n_1} \cdots L(B_k)_{|n_A n_k} \subseteq L(A)$ (the left hand side denotes the product of languages).*

Proof. Since $L(B_i)_{|n_A n_i} \subseteq L(B_i)$ for every i with $1 \leq i \leq k$, if $L(B_1) \cdots L(B_k) \subseteq L(A)$, then it holds also for truncations and $L(B_1)_{|n_A n_1} \cdots L(B_k)_{|n_A n_k} \subseteq L(A)$.

For the converse, we assume that $L(B_1)_{|n_A n_1} \cdots L(B_k)_{|n_A n_k} \subseteq L(A)$. That is, any word $w_1 \cdots w_k$ of $L(B_1) \cdots L(B_k)$ with $|w_i| \leq n_A n_i$ for every i with $1 \leq i \leq k$, belongs to $L(A)$. We need to prove that all words of $L(B_1) \cdots L(B_k)$ (without length restriction) belong to $L(A)$.

We proceed by induction on the vectors of subword lengths of words in $L(B_1) \cdots L(B_k)$. For such a word $w_1 \cdots w_k$, this associated vector is $(|w_1|, \dots, |w_k|)$, and these vectors are ordered lexicographically. The induction hypothesis is that all words of $L(B_1) \cdots L(B_k)$ whose associated vector is at most $(l_1, \dots, l_k)$ (where for any i with $1 \leq i \leq k$, l_i is a positive integer), belongs to A. By our assumption, the case where $l_i \leq n_A n_i$ is true.

Now, consider a word $w = w_1 \cdots w_k$ of $L(B_1) \cdots L(B_k)$, whose associated vector is $(|w_1|, \dots, |w_k|)$, and where for some $j \in \{1, \dots, k\}$, $|w_j| = l_j + 1$; whenever $i \neq j$, $|w_i| \leq l_i$. Let $\mathcal{A}$ and $\mathcal{A}_j$ be two NFAs recognizing A and B_j with smallest numbers n_A and n_j of states, respectively.

We consider the product automaton $\mathcal{A} \times \mathcal{A}_j$ of $\mathcal{A}$ and $\mathcal{A}_j$, with set of states $S \times S_j$ (where S and S_j are the sets of states of $\mathcal{A}$ and $\mathcal{A}_j$, respectively), and a transition $((s_1, s_2), a, (s'_1, s'_2))$ only if we have the transitions (s_1, a, s'_1) and (s_2, a, s'_2) in $\mathcal{A}$ and $\mathcal{A}_j$, respectively (all other transitions are "dummy transitions" to a "garbage state"). Consider the run of $\mathcal{A} \times \mathcal{A}_j$ for the word w_j. The crucial observation is that, because $|w_j| = l_j + 1 > n_A n_j$, this run necessarily visits two states of $\mathcal{A} \times \mathcal{A}_j$ twice, that is, the run contains a directed cycle. Consider the shorter run obtained by pruning this cycle. The two runs start and end at the same two states of $\mathcal{A} \times \mathcal{A}_j$. The shorter run corresponds to a word w'_j of length at most $|w_j| - 1 \leq l_j$. Since $w_j \in L(B_j)$, the end state of these runs is a pair containing an accepting state of $\mathcal{A}_j$ (thus w'_j belongs to $L(B_j)$ as well). Thus, the word w' obtained from w by replacing w_j with w'_j belongs to $L(B_1) \cdots L(B_k)$, and w' satisfies the induction hypothesis. Thus, w' belongs to $L(A)$. But now, considering the pruned cycle in $\mathcal{A} \times \mathcal{A}_j$, we can build a valid run for w_j in $\mathcal{A} \times \mathcal{A}_j$ that leads to a valid run for w in $\mathcal{A}$. This proves the inductive step and concludes the proof.

Proposition 3. *Let E be a regular expression over alphabet Σ, and $\mathcal{A}_E$ an NFA with n_E states recognizing $L(E)$. Let $Q = (D_Q, E_Q)$ be an RGP over Σ. For any two vertices u and v in Q, we can compute a walk W from u to v satisfying $L(E_Q(W)) \subseteq L(E)$ (if one exists), in time $2^{O(n_E |Q| \log(|E| + |Q|))}$. Moreover, if such a walk exists, then there exists one of length at most $2^{n_E} |Q|$.*

Proof. Since E and Q are finite, we will assume that $|\Sigma| \leq |E| + |Q|$ (if not, we simply remove the unused symbols from Σ.)

By Lemma 1, there is a walk W in Q from u to v such that $L(E_Q(W)) \subseteq L(E)$ if and only if there exists one in the RGP Q' obtained from Q by replacing each arc-label $E_Q(x,y)$ by a regular expression $E_{Q'}(x,y)$ defining the $n_E n_B$-truncation of $L(E_Q(x,y))$ (where n_B is the smallest number of states of an NFA recognizing $L(E(x,y))$). Thus, we first compute Q'. Note that $L(E_{Q'}(x,y))$ contains at most $|\Sigma|^{n_E n_B}$ words.

Next, we will construct an auxiliary digraph $G(E,Q,u,v)$. This digraph has vertex set $2^{\mathcal{S}} \times V(Q)$, where $\mathcal{S}$ is the set of states of $\mathcal{A}_E$.

Given two states s_1 and s_2 of $\mathcal{A}_E$ and a word w over Σ, we say that w *reaches* s_2 from s_1 in $\mathcal{A}_E$ if there exists a sequence of transitions of $\mathcal{A}_E$ starting at s_1 and ending at s_2 using the sequence of letters of w.

Now, for two vertices (S_1,x) and (S_2,y) of $G(E,Q,u,v)$, we create the arc $((S_1,x),(S_2,y))$ if, and only if, for each state s of S_1 and each word w of $L(E_{Q'}(x,y))$, w reaches a state of S_2 in $\mathcal{A}_E$.

Deciding whether $((S_1,x),(S_2,y))$ is an arc of $G(E,Q,u,v)$ takes time at most $|S_1|2^{n_E}|L(E_{Q'}(x,y))|$, which is at most $|\Sigma|^{O(n_E|Q|)}$. Since there are $(2^{n_E}|Q|)^2$ pairs of vertices of $G(E,Q,u,v)$, overall the construction of $G(E,Q,u,v)$ can be done in time $|\Sigma|^{O(n_E|Q|)}$.

Now, we claim that there exists a walk W from u to v with $L(E_Q(W)) \subseteq L(E)$ if and only if there is a directed path in $G(E,Q,u,v)$ from a vertex $(\{s_0\},u)$ to a vertex (S_f,v), where s_0 is the initial state of $\mathcal{A}_E$, and S_f is a subset of the accepting states of $\mathcal{A}_E$. Indeed, such a path corresponds precisely to a walk W from u to v in Q, such that all the words of $L(W)$ are accepted by $\mathcal{A}_E$.

This check can be done in linear time in the size of $G(E,Q,u,v)$ using a standard BFS search, thus we obtain an additional time complexity of $(2^{n_E}|Q|)^2$, which is also at most $|\Sigma|^{O(n_E|Q|)}$. Since $|\Sigma| \leq |E|+|Q|$ we obtain $2^{O(n_E|Q|\log(|E|+|Q|))}$.

Finally, it is clear that the length of an obtained directed path of $G(E,Q,u,v)$ is at most the number of vertices of $G(E,Q,u,v)$, which is $2^{n_E}|Q|$, as claimed. This completes the proof. □

Theorem 2. RGPHom *is in EXPTIME.*

Proof. We proceed as follows. First, we go through all possible vertex-mappings of $V(P)$ to $V(Q)$ (there are $|V(Q)|^{|V(P)|}$ such possible mappings). Consider such a vertex-mapping, f.

For each arc (x,y) in P with label $E_P(x,y)$, we proceed as follows. Let $\mathcal{A}$ be an NFA recognizing $L(E_P(x,y))$ with smallest possible number n_A of states. We apply Proposition 3 to $E = E_P(x,y)$, $\mathcal{A}$ and Q, with $u = f(x)$ and $v = f(y)$: thus we can decide in time $2^{O(n_A|Q|\log(|E_P(x,y)|+|Q|))}$ whether the mapping f satisfies the definition of an n-homomorphism for the arc (x,y). If yes, we proceed to the next arc; otherwise, we abort and try the next possible mapping. If we find a valid mapping, we return YES. Otherwise, we return NO.

Our algorithm has a time complexity of $|V(Q)|^{|V(P)|}.|P|.2^{O(|P||Q|\log(|P|+|Q|))}$. Let $n = |P| + |Q|$ be the input size. We obtain an overall running time of $2^{O(n^2 \log n)}$, which is an EXPTIME running time.

5 RGPs over a unary alphabet: the $\{a, a^+\}$ case

In this section, we consider a unary alphabet $\Sigma = \{a\}$. For unary regular languages, Regular Language Inclusion and Regular Language Universality are no longer PSPACE-complete but they are coNP-complete (see [13] and [21], respectively) and the lower bounds from Section 3 do not apply.

The case where all arc-labels of the considered RGPs are equal to "a" is equivalent to the problem of classic digraph homomorphisms, and is known to capture all CSPs [9]. When each label is either "a" or "a^+"– in which case we speak of $\{a, a^+\}$-*RGP* – we have two kinds of constraints: arcs labelled "a" must map in a classic, local, way, while arcs labelled "a^+" can be mapped to an arbitrary path in the target RGP. Thus, this setting is useful for example to model descendance relations in hierarchichal data such as XML. This setting is for example used in languages like SPARQL or XPath for XML documents, that are tree-structured [8,15,18].

For an $\{a, a^+\}$-RGP Q, let $D(Q)$ be the arc-labelled digraph with labels $\{a, t\}$ and the same vertices as Q obtained from Q as follows. The arcs labelled by a coincide. The set of arcs labelled by t in $D(Q)$ is the transitive closure of the arcs of Q (labelled by either label a or a^+).

Proposition 4. *For any $\{a, a^+\}$-RGP Q,* RGPHom(Q) *for $\{a, a^+\}$-RGP inputs is Ptime equivalent to* Hom$(D(Q))$*. Thus, the class of $\{a, a^+\}$-RGP-restricted non-uniform* RGPHom *problems follows a dichotomy between Ptime and NP-complete.*

Proof. Let D'_P be the digraph obtained from $P = (D_P, E_P)$ by replacing all arc-labels "a^+" by labels "t". For any pair of $\{a, a^+\}$-RGP P and Q, we have $P \xrightarrow{n} Q$ (n-homomorphism) if and only if $D'_P \to D(Q)$ (classic homomorphism of arc-labelled digraphs). This provides us with a Ptime reduction from RGPHom(Q) to Hom$(D(Q))$. Conversely, given a digraph D'_P with arcs labelled by a and t, let P be the $\{a, a^+\}$-RGP obtained from D'_P by replacing t labels by a^+. This is a Ptime reduction from Hom$(D(Q))$ to RGPHom(Q).

The dichotomy follows from the CSP dichotomy of [5,22]. □

Proposition 5. RGPHom *for $\{a, a^+\}$-RGPs is Ptime reducible to* Hom *for two-arc-labelled digraphs. Thus this uniform problem is in NP.*

The Ptime algorithms based on algebraic methods proposed by Bulatov [5] and Zhuk [22] for tractable CSPs are somewhat contrived and a bit overkill for the class of non-uniform RGPHom problems restricted to $\{a, a^+\}$-RGPs. This motivates us to look for simple direct combinatorial characterisations and simple algorithms for interesting $\{a, a^+\}$-RGPs that model natural queries.

5.1 Undirected $\{a, a^+\}$-RGPs

We now consider *undirected* $\{a, a^+\}$-RGPs, where arcs are pairs of vertices, called *edges* (equivalently, for each arc from x to y, we have its symmetric arc from y

to x). In an n-homomorphism between two undirected $\{a, a^+\}$-RGPs P and Q, "a"-edges must be preserved (as in a classic graph homomorphism), while the endpoints of "a^+"-edges need to be mapped to two vertices of Q that are connected by some path. Thus, this variant extends classic graph homomorphisms via additional (binary) connectivity constraints. We provide an analogue of the Hell-Nešetřil dichotomy for HOM(H) [11] in this setting.

Theorem 3. *Let Q be an undirected and connected n-core $\{a, a^+\}$-RGP. If Q has at most one edge,* RGPHOM(Q) *is in Ptime. Otherwise,* RGPHOM(Q) *is NP-complete.*

5.2 Directed path $\{a\}$-RGPs

We first consider RGPHOM(Q) when Q is a directed path whose arc labels are all "a" (Q is called an $\{a\}$-RGP) — arguably the simplest RGP directed graph example — and where inputs are $\{a, a^+\}$-RGPs. This case turns out to have an interesting connection to the following scheduling problem, which enjoys a Ptime algorithm by reduction to a shortest path problem in edge-weighted digraphs [20, Chapter 4.4, p.666].

PARALLEL JOB SCHEDULING WITH RELATIVE DEADLINES
Input: A set J of jobs, a duration function $d : J \to \mathbb{N}$, a relative deadline function $r : J \times J \to \mathbb{Z}$, and a maximum time t_{max}.
Question: Is there a feasible schedule for the jobs, that is, an assignment $t : J \to \mathbb{N}$ of start times such that every job finishes before time t_{max} and for any two jobs j_1 and j_2, j_1 starts before the time $t(j_2) + r(j_1, j_2)$?

Theorem 4. *For any directed path $\{a\}$-RGP Q,* RGPHOM(Q) *for $\{a, a^+\}$-RGP inputs is Ptime-reducible to* PARALLEL JOB SCHEDULING WITH RELATIVE DEADLINES. *Thus,* RGPHOM(Q) *is in Ptime when restricted to such inputs.*

As we will see in the next section, the second part of Theorem 4 can be generalized to all directed path $\{a, a^+\}$-RGPs using a different method.

5.3 Directed path $\{a, a^+\}$-RGPs

Our next result is more general than Theorem 4, as we use a stronger method. It also extends a result from [18], where the statement is proved for directed tree input RGPs. Here we prove it for all kinds of inputs.

For an arc-labelled digraph D and a positive integer k, the *product digraph* D^k is the digraph with vertices $V(D)^k$ and with an arc labelled ℓ from $(x_1, \ldots, x_k)$ to $(y_1, \ldots, y_k)$ iff all pairs (x_i, y_i) with $1 \leq i \leq k$ are arcs labelled ℓ in D. A homomorphism of D^k to D is called a (k-ary) *polymorphism* of D. For a set S, a function f from S^3 to S is a *majority function* if for all x, y in S, $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$.

Theorem 5 ([14]). *Let D be an arc-labelled digraph that has a ternary polymorphism that is a majority function. Then,* $\textsc{Hom}(D)$ *is in Ptime.*

It is well known that the above applies to directed paths, a result that we can lift to $\{a, a^+\}$-RGPs.

Theorem 6. *Let Q be an $\{a, a^+\}$-RGP whose underlying digraph is a directed path. Then, $D(Q)$ admits a majority polymorphism and thus* $\text{RGP}\textsc{Hom}(Q)$ *is in Ptime.*

We remark that Theorem 6 also applies to RGPs with vertex-labels (where the mapping must preserve the vertex-labels). Indeed, vertex-labels are modeled as unary relations, which trivially satisfy the properties for having a majority polymorphism. Moreover, using the same method, Theorem 6 extends to labels of the form "a^*", "a^k" or "$a^{\leq k}$" for $k \in \mathbb{N}$.

6 Conclusion

We have seen that RGPHom, which is generally PSPACE-hard (but in NP when the target RGP is a graph database), is in EXPTIME. This favorably compares to the general complexity of RGP query containment, which is EXPSPACE-complete [7,10], and motivates the use of RGP n-homomorphisms to approximate query containment. It remains to close the gap between the PSPACE lower bound and the EXPTIME upper bound.

We have also seen that the case of $\{a, a^+\}$-RGPs (a case that is also in NP, and that corresponds to XPath and SPARQL queries), we have a complete classification of the NP-complete and Ptime cases for undirected RGPs, and all RGPs whose underlying digraph is a directed path are in Ptime. It was proved in [18] that when both the input and target is a directed tree $\{a, a^+\}$-RGP, RGPHom is in Ptime. Is it true that (for general inputs) $\text{RGP}\textsc{Hom}(Q)$ is in Ptime when Q is a directed tree $\{a, a^+\}$-RGP? [6] When all arc-labels are "a", then the only n-core RGPs whose underlying digraphs are directed trees are, in fact, the directed paths. But there are many more directed tree $\{a, a^+\}$-RGP n-cores, see Figure 2 for a simple example.

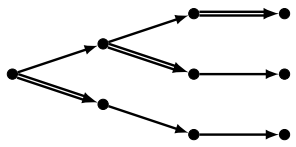

Fig. 2. An n-core directed tree $\{a, a^+\}$-RGP. Doubled edges are labelled "a^+", the others are labelled "a".

[6] This is not true for all acyclic RGPs: there are trees T such that $\textsc{Hom}(T)$ is NP-hard [6]. Thus, for the corresponding $\{a\}$-RGP $Q(T)$, $\text{RGP}\textsc{Hom}(Q(T))$ is NP-hard.

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974)
2. Angles, R., *et. al.*: G-CORE: A core for future graph query languages. SIGMOD Conference 2018. ACM (2018). https://doi.org/10.1145/3183713.3190654
3. Baeza, P.B.: Querying graph databases. PODS 2013. https://doi.org/10.1145/2463664.2465216
4. Barceló, P., Romero, M., Vardi, M.Y.: Semantic acyclicity on graph databases. SIAM J. Comput. **45**(4), 1339–1376 (2016). https://doi.org/10.1137/15M1034714
5. Bulatov, A.A.: A dichotomy theorem for nonuniform csps. FOCS 2017. https://doi.org/10.1109/FOCS.2017.37
6. Bulin, J.: On the complexity of H-coloring for special oriented trees. Eur. J. Comb. **69**, 54–75 (2018). https://doi.org/10.1016/j.ejc.2017.10.001
7. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Containment of conjunctive regular path queries with inverse. In KR 2000.
8. Czerwinski, W., Martens, W., Niewerth, M., Parys, P.: Optimizing tree patterns for querying graph- and tree-structured data. SIGMOD Record **46**(1), 15–22 (2017). https://doi.org/10.1145/3093754.3093759
9. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. SIAM J. Comput. **28**(1), 57–104 (1998). https://doi.org/10.1137/S0097539794266766
10. Florescu, D., Levy, A.Y., Suciu, D.: Query containment for conjunctive queries with regular expressions. PODS 1998. https://doi.org/10.1145/275487.275503
11. Hell, P., Nesetril, J.: On the complexity of H-coloring. J. Comb. Theory, Ser. B **48**(1), 92–110 (1990). https://doi.org/10.1016/0095-8956(90)90132-J
12. Hell, P., Nesetril, J.: The core of a graph. Discrete Mathematics **109**(1-3), 117–126 (1992). https://doi.org/10.1016/0012-365X(92)90282-K
13. Hunt, H.B., Rosenkrantz, D.J., Szymanski, T.G.: On the equivalence, containment, and covering problems for the regular and context-free languages. J. Comput. Syst. Sci. **12**, 222–268 (1976)
14. Jeavons, P., Cohen, D.A., Cooper, M.C.: Constraints, consistency and closure. Artif. Intell. **101**(1-2), 251–265 (1998). https://doi.org/10.1016/S0004-3702(98)00022-8
15. Kimelfeld, B., Sagiv, Y.: Revisiting redundancy and minimization in an xpath fragment. EDBT 2008. https://doi.org/10.1145/1353343.1353355
16. Kolaitis, P.G., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. J. Comput. Syst. Sci. **61**(2), 302–332 (2000). https://doi.org/10.1006/jcss.2000.1713
17. Mendelzon, A.O., Wood, P.T.: Finding regular simple paths in graph databases. SIAM J. Comput. **24**(6), 1235–1258 (1995). https://doi.org/10.1137/S009753979122370X
18. Miklau, G., Suciu, D.: Containment and equivalence for a fragment of xpath. J. ACM **51**(1), 2–45 (2004). https://doi.org/10.1145/962446.962448
19. Romero, M., Barceló, P., Vardi, M.Y.: The homomorphism problem for regular graph patterns. LICS 2017. https://doi.org/10.1109/LICS.2017.8005106
20. Sedgewick, R., Wayne, K.: Algorithms, 4th Edition. Addison-Wesley (2011)
21. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. STOC 1973. https://doi.org/10.1145/800125.804029
22. Zhuk, D.: A proof of CSP dichotomy conjecture. FOCS 2017. https://doi.org/10.1109/FOCS.2017.38