Data Path Queries over Embedded Graph Databases

Diego Figueira
diego.figueira@cnrs.fr
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800
Talence, France

Artur Jeż
artur.jez@uwr.edu.pl
University of Wrocław
Wrocław, Poland

Anthony W. Lin
anthony.lin@cs.uni-kl.de
Automated Reasoning Group, TU Kaiserslautern and MPI-SWS
Kaiserslautern, Germany

ABSTRACT

This paper initiates the study of data-path query languages (in particular, regular data path queries (RDPQ) and conjunctive RDPQ (CRDPQ)) in the classic setting of embedded finite model theory, wherein each graph is "embedded" into a background infinite structure (with a decidable FO theory or fragments thereof). Our goal is to address the current lack of support for typed attribute data (e.g. integer arithmetics) in existing data-path query languages, which are crucial in practice. We propose an extension of register automata by allowing powerful constraints over the theory and the database as guards, and having two types of registers: registers that can store values from the active domain, and read-only registers that can store arbitrary values. We prove NL data complexity for (C)RDPQ over the Presburger arithmetic, the real-closed field, the existential theory of automatic structures and word equations with regular constraints. All these results strictly extend the known NL data complexity of RDPQ with only equality comparisons, and provides an answer to a recent open problem posed by Libkin et al. Among others, we introduce one crucial proof technique for obtaining NL data complexity for data path queries over embedded graph databases called "Restricted Register Collapse (RRC)", inspired by the notion of Restricted Quantifier Collapse (RQC) in embedded finite model theory.

CCS CONCEPTS

• **Theory of computation → Database query languages (principles)**; **Complexity theory and logic**; **Finite Model Theory**; **Logic and databases**; *Regular languages.*

KEYWORDS

Data graphs, complexity, embedded finite models, regular path queries

ACM Reference Format:
Diego Figueira, Artur Jeż, and Anthony W. Lin. 2022. Data Path Queries over Embedded Graph Databases. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3517804.3524159

1 INTRODUCTION

The past decade witnessed a lot of practical and theoretical work in querying graph databases (e.g. [2, 3, 17, 45]), due to their rapidly growing number of applications, e.g., in social networks, semantic web, natural science, and supply chain management. On the practical side, graph databases have enjoyed a wide adoption in industry with many commercial database systems developed for them by various companies like Neo4j, Oracle, IBM, DataStax, TigerGraph, JanusGraph, and Stardog, among many others. Although such systems adopt the so-called "property graph data model", they differ in many important aspects (e.g. querying facilities), which has led to the ongoing standardization efforts of SQL/PGQ and GQL involving leading researchers and engineers in academia and industry [17].

On the theoretical side, an important research avenue has been to develop "well-behaved" graph query languages incorporating *topological* aspects of the graph, especially the existence of *paths* in a graph satisfying certain patterns and constraints. Such queries are called *path queries*, and represent an essential feature (called GPML, which stands for *Graph Pattern Matching Languages*) in the current standardization effort [17] of GQL and SQL/PGQ. Historically, Mendelzon and Wood [38] introduced in the 1990s the Regular Path Query (RPQ) language, which has been the foundation of most path query languages, e.g., Conjunctive Regular Path Queries (CRPQ) [16, 18, 23]. One of their main attractive features is that their query evaluation problem has an NL (Nondeterministic Logspace) *data complexity* (i.e. *for any fixed query*), potentially allowing traversals of large graphs in practice (e.g. see [4, 44]).

Combining data and topology. RPQ and CRPQ are unfortunately not suitable for "data querying" in graph databases, which is bread and butter for relational database query languages. More precisely, in a typical graph database each node is usually associated with some data like names and ages of the users. In relational algebra (assuming a relational encoding of graph databases), one could easily express the query that outputs two node ids for users with the same ages. In the past decade, the importance of combining data and topology in graph query languages has been brought to attention by multiple works, e.g. [6, 28, 36, 37]. There are many natural examples of such queries, e.g., find two people in the database with the same age. In particular, walk logic (WL) [28] and regular expressions with memory (REM) [37] emerged as two initial solutions for overcoming this limitation. While RPQ using REM and register automata as path constraint has NL data complexity [36, 37], WL has an non-elementary data complexity [6]. The crucial observation in [36, 37] is that paths in a graph database can be construed as a *data word*, which enables one to apply automata over data words (e.g. register automata [31], REM [37], data automata [14], etc.) as path constraints in a path-query. Other automata models over data

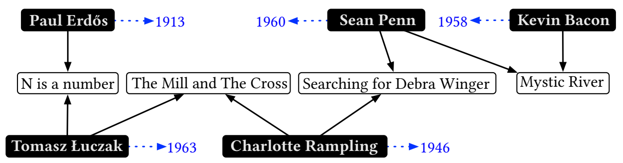

Figure 1: Part of the "Bacon Graph", where each solid (resp. dotted) edge represents the "acts in" (resp. "is born in") relation. Each actor is a black box, and each movie is a white box. Birthyears are also nodes.

Table 1: Summary of our data complexity results

$\mathbb{L}$	Data Complexity of RDPQ($\mathbb{L}, \sigma$)
$\mathrm{FO}(\mathbb{R}_{\times,+}, \sigma)$	NL
$\mathrm{FO}(\mathbb{Z}_{LA}, \sigma)$	NL
$\mathrm{EFO}^+(\mathbf{T}_{\mathrm{Aut}}, \sigma)$	NP-hard
$\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$	NL (under LogH)
$\mathbb{L}_{\mathrm{PM}}$	NL
$\mathbb{L}_{\mathrm{WE}}^+$	NL

words (e.g. data automata) increase the data complexity of RPQ from NL to NP-hard [36].

The need for reasoning about concrete data. Thus far, the solutions [6, 28, 36, 37] to combine data and topology aspects in graph queries ignore completely the *structure of the data*. For example, practical database query languages typically allow concrete data (e.g. numbers, words) and a wide variety of data operations/comparisons (e.g. substrings, inequality, difference). Consider, for instance, the "Bacon graph" (see Figure 1) and the query "*return all actors with a finite Bacon number and whose birthyear differs Bacon's birthyear by at least 30 years*," whose only answer here is Erdős. Alternatively, consider the query "*return all users who were in at least four different locations in the last month, but stayed within a 10km radius from some point in a map*," whose numbers and how they differ on a monthly basis might be insightful especially during the time of COVID-19. Finally, for a phylogenetic database, one might be interested in a path containing only nodes whose center (with respect to some edit distance measure) is not that far away from any DNA sequence along the path [15, 26]. None of the above features are expressible in the existing path-query languages. For example, while path query languages like RDPQ [36] may for instance express the existence of a path from Erdős to Bacon, sharing *the same* birth years, they only support active-domain values in their registers, as well as equality comparisons, which restrict them from expressing the three aforementioned examples.

Incorporating concrete attribute data and their operations in a first-order query language over relational databases has been intensively studied by database theorists between 1990 and early 2000s in the context of constraint databases and embedded finite model theory (e.g. see [25, 34] and [35, Chapter 13]). In particular, one considers a relational database embedded into the universe of an infinite structure with a decidable FO theory (e.g. real-closed field $\mathbb{R}_{\times,+}$ or integer linear arithmetic $\mathbb{Z}_{LA}$ [25, 34, 35]), and allows first-order queries to make use of operations in the theory. For example, over a graph embedded into $\mathbb{R}_{\times,+}$, one could ask if all edges in a graph lie on a line. The first-order query can be found in [35, Chapter 13]: $\exists u \exists v \forall x \in adom \forall y \in adom\ (E(x,y) \to y = u.x + v)$. Notice two kinds of quantifiers are in use: $\exists u$ means "there is a real number (not necessarily in the active domain)", while $\forall x \in adom$ means "for all elements in the active domain". The set of such first-order formulas over the theory $\mathbf{T}$ and relational database schema σ is denoted by $\mathrm{FO}(\mathbf{T}, \sigma)$, and its restrictions with only active-domain quantifiers by $\mathrm{FO}_{\mathrm{act}}(\mathbf{T}, \sigma)$. Despite this, the issue of querying data-paths over embedded graph databases remains hitherto unaddressed and was only left as an open problem by Libkin et al. [36].

Contributions. This paper initiates the study of data-path queries in the setting of *embedded graph databases*, i.e., finite graphs embedded in some infinite structure with a decidable first-order theory $\mathbf{T}$ or fragments thereof. Because of the different graph models in the literature we assume a general relational database schema σ and define a graph by means of *first-order views* over σ and $\mathbf{T}$. This is closer to the data model adopted by SQL/PGQ (e.g. see [17] and https://pgql-lang.org/), and can be used to encode implicitly, as well as explicitly, represented property graphs. We study the problem of query evaluation over extensions of RDPQ and CRDPQ, allowing the use of $\mathrm{FO}(\mathbf{T}, \sigma)$-formulas in the queries, which capture a range of meaningful queries involving typed attribute values like integers, reals, and words (e.g. the three aforementioned examples).

Since register automata [36] use untyped attribute data which can only be compared via equality and disequality, we first define *extended register automata (ERA)* that allow attribute data in $\mathbf{T}$ which can be manipulated via operations in $\mathrm{FO}(\mathbf{T}, \sigma)$. By analogy to the setting of embedded finite model theory, we allow two kinds of registers: active-domain registers and unrestricted registers. Each transition is associated with a guard $\varphi(curr, \mathcal{R}, \mathcal{R}', \mathcal{P}, \mathcal{P}') \in \mathrm{FO}(\mathbf{T}, \sigma)$, where $curr$ is the value in $\mathbf{T}$ representing the current node (called *node ID*), $\mathcal{R}$ (resp. $\mathcal{P}$) represents the current values of the active-domain (resp. unrestricted) registers, and $\mathcal{R}'$ (resp. $\mathcal{P}'$) represents the next values of the active-domain (resp. unrestricted) registers. The register automata model of [36] can be construed as an instance of ERA with no unrestricted registers, where $\sigma = \{E\}$, $\mathbf{T} = \langle \mathbb{N}; =, \{c_i\}_{i \in \mathbb{N}} \rangle$ with each constant c_i interpreted as the number i, and the guard uses only Boolean combinations of $\mathrm{FO}(\mathbf{T})$-atoms.

Without further restrictions, the model easily captures the universal Minsky's counter machine already with two unrestricted registers and the theory $\langle \mathbb{N}; succ \rangle$ of natural numbers with successor [41]. Hence, we impose the so-called *bounded-rewrite assumptions* to the unrestricted registers, i.e., for some constant $b \in \mathbb{N}$, each unrestricted register can be rewritten at most b times. The resulting query languages $\mathrm{RDPQ}(\mathbf{T}, \sigma)$ and $\mathrm{CRDPQ}(\mathbf{T}, \sigma)$ turn out to be sufficiently general to express many interesting examples (e.g. our three aforementioned examples). Our result on the data complexity of query evaluation is summarized in Table 1. Note that under the bounded-rewrite assumption we can assume that $b = 0$ at the expense of increasing the number of unrestricted registers, see Proposition 2.4.

First, we obtain NL data complexity for $\mathrm{RDPQ}(\mathrm{FO}, \mathbb{R}_{\times,+}, \sigma)$ and $\mathrm{RDPQ}(\mathrm{FO}, \mathbb{Z}_{LA}, \sigma)$. With these, we can express the first two out of our three examples. To prove this, we introduce the so-called *Restricted Register Collapse (RRC)* — akin to the notion of *Restricted Quantifier Collapse (RQC)* for FO over embedded finite models [35, Chapter 13] — which states that each query in RDPQ(FO, $\mathbf{T}, \sigma$) is equivalent to a query in RDPQ(FO, $\mathbf{T}, \sigma$) without unrestricted registers and so that no guards contain unrestricted quantifiers. We show that this holds for $\mathbb{R}_{\times,+}$ and $\mathbb{Z}_{LA}$.

The case of words is more involved. To permit reasoning about properties such as edit distance, one needs a sufficiently expressive word theory $\mathbf{T}$, e.g., the theory $\mathbf{T}_{\text{Aut}}$ of automatic relations [11, 13], or theory $\mathbf{T}_{\text{WE}}$ of word equations [19]. Unfortunately, FO($\mathbf{T}_{\text{Aut}}, \sigma$) and FO($\mathbf{T}_{\text{WE}}, \sigma$) have prohibitive data complexity [11] (hard for the polynomial hierarchy PH and undecidable, respectively). We provide restrictions that still allow us to reason about interesting properties like edit distance, while admitting NL data complexity. First we show that the existential fragment EFO($\mathbf{T}_{\text{Aut}}, \sigma$) admits RQC (i.e. EFO($\mathbf{T}_{\text{Aut}}, \sigma$) $\subseteq$ FO$_{\text{act}}$($\mathbf{T}, \sigma$)), implying an NC^1 data complexity. In contrast, we show that RDPQ(EFO, $\mathbf{T}_{\text{Aut}}, \sigma$) has NP-hard data complexity and does not admit RRC, unless P = NP. Fortunately, NL data complexity can be recovered for RDPQ(EFO, $\mathbf{T}_{\text{Aut}}, \sigma$) assuming the *LogH hypothesis* that the words in the database are of length logarithmic in the size of the database, which is reasonable for most graph database applications, with extremely large graphs and exponentially smaller data sizes (e.g. names, DNA sequences, etc.). Secondly, we provide two fragments of FO($\mathbf{T}_{\text{WE}}, \sigma$) (pattern-matching fragment $\mathbb{L}_{\text{PM}}$ and positive quantifier-free fragment $\mathbb{L}^+_{\text{WE}}$) that permit NL data complexity for RDPQ($\mathbf{T}_{\text{WE}}, \sigma$), even without assuming LogH. The proof is an intricate application of word equation solving.

Organization. We introduce our data model and query language in §2. We tackle data complexity of the theories over numbers (reals, integers) in §3, and over words in §4 (automatic structures) and §5 (word equations). We conclude in §6 with generalizations (e.g. conjunctive queries and property graphs), and future work.

Complexity Theory Preliminaries. We assume standard complexity classes esp. L (Logspace), NL (Nondeterministic Logspace), P, NP, PH, PSPACE. In this paper, we will also mention (the uniform versions of) circuit complexity classes like AC^0, TC^0, NC^1. It is well-known that AC^0 $\subsetneq$ TC^0 $\subseteq$ NC^1 $\subseteq$ L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PH $\subseteq$ PSPACE. All reductions mentioned in this paper can be implemented in Logspace. See [33, 35, 43] for more.

2 DATA MODEL AND QUERIES

Embedded Finite Models. Fix a relational vocabulary σ containing finitely many relation symbols R_i with arity r_i (i.e. $\sigma = \{R_1, \ldots, R_k\}$) and an infinite structure $\mathbf{T}$ with domain $\mathcal{D}$. A *finite σ-model embedded into* $\mathbf{T}$ [35, Ch. 13] is a σ-structure $\mathrm{S} := \langle U; \{R_i^{\mathrm{S}}\}_{i=1}^k \rangle$ such that $U \subseteq \mathcal{D}$ is finite and $R_i^{\mathrm{S}} \subseteq U^{r_i}$ for every i. The set U is called the *active domain* of S and hence denoted by *adom*(S). In the sequel, we shall also refer to such S as $(\mathbf{T}, \sigma)$*-structure*.

By FO($\mathbf{T}, \sigma$), we denote first-order logic that may use ($\mathbf{T} \cup \sigma$)-atomic formulas, active-domain quantification ($\exists x \in adom, \forall x \in adom$), as well as unrestricted quantification ($\exists x, \forall x$). We denote by FO$_{\text{act}}$($\mathbf{T}, \sigma$) the restriction of FO($\mathbf{T}, \sigma$) that prohibits unrestricted quantifications. For a formula $\varphi(\bar{x})$ and $\bar{a} \in \mathcal{D}^{|\bar{x}|}$, we define the notion of satisfaction $\mathrm{S} \models_{\mathbf{T}} \varphi(\bar{a})$ by interpreting each atomic formula from σ (resp. $\mathbf{T}$) with respect to S (resp. $\mathbf{T}$), each active-domain quantifier $\exists x \in adom\ \psi$ (resp. $\forall x \in adom\ \psi$) as "there exists an element b in U such that ψ" (resp. "for all elements b in U, ψ"), and each unrestricted quantifier $\exists x\ \psi$ (resp. $\forall x\ \psi$) as "there exists an element b in $\mathcal{D}$ such that ψ" (resp. "for all elements b in $\mathcal{D}$, ψ"). See the example $\exists u \exists v \forall x \in adom \forall y \in adom\ (E(x, y) \to y = u.x + v)$ in § 1 and more examples in [25, 35].

We fix now some extra notation that we will use in the sequel. We write $\models$ instead of $\models_{\mathbf{T}}$, when $\mathbf{T}$ is understood. We also write $[\![\varphi]\!]_{(\mathbf{T},\mathrm{S})}$ to denote the set of all $\bar{a} \in \mathcal{D}^{|\bar{x}|}$ such that $\mathrm{S} \models_{\mathbf{T}} \varphi(\bar{a})$. Whenever understood, we also use the same notation for a relation symbol R and the relation R^{S} that instantiates R in S. Finally, for the purpose of measuring complexity of query evaluation problems, we assume a size function $|.| : \mathcal{D} \to \mathbb{N}$ that defines the size of the representation of a datum; this will be fixed with respect to each theory under consideration.

Embedded Graph Databases. Fix a finite alphabet Σ of symbols. We define our notion of data graphs following [36]. Our results can be easily adapted to a notion of property graphs [17]; see § 6. We assume in the following a theory $\mathbf{T}$ and a finite $(\mathbf{T}, \sigma)$-structure S. Our notion of embedded graphs will be defined with respect to S via views defined through formulas over a fragment $\mathbb{L} \subseteq$ FO($\mathbf{T}, \sigma$).

Fix our view vocabulary $\nu = \{V\} \cup \{E_a\}_{a \in \Sigma}$, where V is unary (denoting the vertex view) and each E_a is binary (denoting the view of a-labeled edges). A *view definition* for a view W with arity r is an $\mathbb{L}$-formula $\varphi_W(\bar{x})$ with $r = |\bar{x}|$. Each view W will then be interpreted as $[\![\varphi_W]\!] \subseteq U^r$ containing all $\bar{a} \subseteq U^r$ such that $\mathrm{S} \models \varphi(\bar{a})$. As before, whenever understood, we will confuse each view W and their interpretation $[\![\varphi_W]\!]$. We are now ready to define our notion of (data)-graph $\mathbf{G}$ embedded into $\mathbf{T}$ (or $(\mathbf{T}, \sigma)$*-graph* or just *graph* for short): it is a pair (Θ, S), where Θ is a set of view definitions and S a finite $(\mathbf{T}, \sigma)$-structure.

In the sequel, instead of writing $(v, w) \in E_a$, we shall often write $v \to_a w$. A $\mathbf{G}$*-path* from v to w, where $v, w \in V$, is simply an alternating sequence $v = v_0 a_1 \cdots a_n v_n = w$ of vertices and edge labels of $\mathbf{G}$ (i.e. each $v_i \in V$ and each $a_j \in \Sigma$) such that $v_i \to_{a_{i+1}} v_{i+1}$ for each $i \in [0, n-1]$.

Example 2.1. Consider our Bacon graph G from Figure 1. We may assume that the database schema used to represent this is $\sigma = \{E_a, E_b, Act, Mov, Y\}$, where E_a (resp. E_b) is a binary relation representing the "$\underline{a}$cts in" (resp. "was $\underline{b}$orn in") relation, Act (resp. Mov) is a unary relation representing all the actors (resp. movies), and Y is a unary relation containing all the years in the database. Let $\mathbf{T} = \mathbb{Z}_{LA}$. To embed G into $\mathbf{T}$, we will associate each node with a unique ID, as in Table 2. For example, this embedding has tuples $E_b(1, 1913)$, $E_a(0, -4)$, etc. We may define the views as follows:

$$\begin{aligned} V(x) &:= Act(x) \\ E(x_1, x_2) &:= \exists y\ (Mov(y) \land E_a(x_1, y) \land E_a(x_2, y)) \end{aligned}$$

Note that two actors are connected by an E-edge iff they act in a common movie.

Example 2.2. Let $\mathbf{T} = \mathbb{R}_{\times,+}$ and let $\sigma = \langle T \rangle$, where T is binary. Each pair (t, n) in T indicates a time stamp $t \in \mathbb{Q}$ (with day as unit,

Table 2: Embedding of Bacon Graph into $\mathbb{Z}_{LA}$. Each year has itself as node ID.

ID	node
0	Kevin Bacon
1	Paul Erdős
2	Tomasz Łuczak
3	Sean Penn
4	Charlotte Rampling
-1	N is a number
-2	The Mill and The Cross
-3	Searching for Debra Winger
-4	Mystic River

where fractions are allowed since data are inserted as they become available) and the total number $n \in \mathbb{N}$ of active COVID cases at time stamp t. Graph views will contain only the vertex view V and a single edge view E with the following view definitions:

$$\begin{aligned} V(x) &:= \exists y \in adom\,(T(x,y)) \\ E(x_1,x_2) &:= x_1 < x_2 \wedge \exists y_1, y_2\,(T(x_1,y_1) \wedge T(x_2,y_2)) \wedge \\ &\quad \neg\exists x\,(x_1 < x < x_2 \wedge \exists y\, T(x,y)) \end{aligned}$$

In other words, we obtain a graphical representation of the number of active COVID cases whose vertices represent time stamps in the DB and edges represent the time progressions. One might be interested in a long enough stretch of time stamps in T whose number of active COVID cases exceeds some sufficiently fast growing function $p(t)$. We will mention cases which can be answered with NL data complexity.

Data Path Queries. Fix a fragment $\mathbb{L} \subseteq \text{FO}(\mathbf{T}, \sigma)$. For an alphabet Σ, let $\Sigma_\# = \Sigma \cup \{\#\}$, where $\# \notin \Sigma$ is a fixed "dummy symbol". In order to define our query language RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) (or RDPQ($\mathbb{L}, \sigma$) or just RDPQ($\mathbb{L}$) in short), we define our notion of *extended register automata (ERA)* admitting active-domain and unrestricted registers, which can take any value in the domain $\mathcal{D}$ of $\mathbf{T}$. More precisely, a $(\sigma, \mathbf{T})$*-register automaton* is a tuple $\mathcal{A} = (\mathcal{R}, \mathcal{P}, Q, Q_0, F, \Delta)$, where $\mathcal{R} = \{r_1, \ldots, r_l\}$ is a set of active-domain registers, $\mathcal{P} = \{p_1, \ldots, p_h\}$ is a set of unrestricted registers, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and Δ is a finite set of transitions[1] each of the form $(q, (a, \varphi(curr, \mathcal{P}, \mathcal{P}', \mathcal{R}, \mathcal{R}')), q')$, where $q, q' \in Q$, $a \in \Sigma_\#$, and $\varphi \in \mathbb{L}$ with $\mathcal{R}' := \{r'_1, \ldots, r'_l\}$ and $\mathcal{P}' := \{p'_1, \ldots, p'_h\}$. Intuitively, *curr* refers to current node ID, $\mathcal{P}$ (resp. $\mathcal{R}$) the current values of the unrestricted (resp. active-domain) registers, and $\mathcal{P}'$ (resp. $\mathcal{R}'$) the next values of the unrestricted (resp. active-domain) registers. More formally, given a $(\mathbf{T}, \sigma)$-graph $\mathbf{G}$, a path in it

$$\pi := v_0 \to_{a_1} \cdots \to_{a_n} v_n,$$

we write that $\mathbf{G}, \pi \models \mathcal{A}$ if there is a sequence of transitions

$$q_0 \to_{t_0} \cdots \to_{t_n} q_{n+1},$$

with $q_0 \in Q_0$, $q_{n+1} \in F$, and register values

$$\mathbf{v} = (\mu_0, \nu_0), \ldots, (\mu_n, \nu_n) \in U^{\mathcal{R}} \times \mathcal{D}^{\mathcal{P}} \qquad (*)$$

[1]After gaining familiarity with ERA, the reader is referred to Section 6 for a generalization of ERA that *directly* traverse the relational database S.

such that for every $0 < i \leq n$: (1) $t_0 = (q_0, (\#, \varphi_0), q_1)$ and $t_i = (q_i, (a_i, \varphi_i), q_{i+1})$; and (2) $\mathbf{G} \models \varphi_i(v_i, \nu_{i-1}, \nu_i, \mu_{i-1}, \mu_i)$. We write $[\![\mathcal{A}]\!]_\mathbf{G}$ to be the set of all pairs $(u, v) \in V^2$ of vertices in $\mathbf{G}$ such that there is a path π from u to v such that $\mathbf{G}, \pi \models \mathcal{A}$.

An RDPQ query is a formula F of the form $x \to_\mathcal{A} y$, such that $[\![F]\!]_\mathbf{G} = [\![\mathcal{A}]\!]_\mathbf{G}$. We are concerned with the *query evaluation problem*: given a graph $\mathbf{G}$, a pair of vertices $(s, t) \in V^2$, and an RDPQ formula $F := x \to_\mathcal{A} y$, whether $(s, t) \in [\![F]\!]_\mathbf{G}$.

Unsurprisingly, this problem is undecidable for a fixed $\mathcal{A}$, even for the decidable theory $\langle \mathbb{N}, succ \rangle$ of natural numbers with successors, since two unrestricted registers can be used to simulate universal Minsky's 2-counter machines (e.g. see [41]). For this reason, we impose the so-called *bounded-rewrite* restriction for the unrestricted registers. In the sequence of register values in $(*)$, a register p_i *undergoes b rewrites* if the number of value changes of p_i in the sequence is b. For example, if the values of p in an $\mathcal{A}$-run are 7, 7, 7, 3, 3, 10, 1, then p undergoes 3 rewrites. In the sequel, we will restrict ourselves to runs of $\mathcal{A}$ with at most b rewrites for each unrestricted register, for some constant $b \in \mathbb{N}$. We will refer to such $\mathcal{A}$ as (h, b)-rewrite register automaton, where h is the number of unrestricted registers of $\mathcal{A}$. We will tacitly assume that each RDPQ($\mathbb{L}, \mathbf{T}, \sigma$)-query uses an (h, b)-rewrite register automaton because of the following:

Proposition 2.3. *Let* $\mathbb{L} = \text{FO}(\mathbf{T}, \sigma)$ *or* $\mathbb{L} = \text{EFO}(\mathbf{T}, \sigma)$*, and assume that query evaluation for* $\mathbb{L}$ *is decidable. Then, query evaluation for* RDPQ($\mathbb{L}, \sigma$) *with* (h, b)*-rewrite register automata is decidable.*

This can be easily proven: it suffices to consider paths in the graph of length at most $|U|^l \times |Q| \times (b+1)^h$ (remember, $l = |\mathcal{R}|$) and hence we turn each such path π into a formula in $\mathbb{L}$ to check if there is a sequence of $\mathcal{A}$-configurations conforming to π.

In this paper we are concerned mostly about the issue of *data complexity*: What is the complexity of the evaluation problem for a *fixed* RDPQ($\mathbf{T}, \sigma$)-query? We count the view definitions in $\mathbf{G}$ also as part of the query, which is consistent with PGQL (see https://pgql-lang.org/). Following [36], NL data complexity is of particular interest, which is achieved for the basic RDPQ (i.e. with the base theory $\langle \mathbb{N}; =, \{c_i\}_{i \in \mathbb{N}} \rangle$, and $\sigma = \langle E_1, \ldots, E_n \rangle$). When considering data complexity, the following proposition shows that we may assume $(h, 0)$-rewrite register automata, i.e., each unrestricted register behaves like a *parameter*, a notion that appears in parametric one-counter automata [27] and parametric timed automata [1].

Proposition 2.4. RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) *with* (h, b)*-rewrite register automata is (effectively) equi-expressive with* RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) *with* $(hb, 0)$*-rewrite register automata.*

In the sequel, we assume that each unrestricted register is a *parameter* and we simplify notation accordingly (e.g. the guard in each transition is of the form $\varphi(curr, \mathcal{P}, \mathcal{R}, \mathcal{R}')$).

Example 2.5. Following Example 2.1, we are interested in finding the set of pairs (x, y) of actors whose age difference is at least 30. This can be expressed in RDPQ($\mathbb{Z}_{LA}, \sigma$) as $x \to_\mathcal{A} y$, where $\mathcal{A}$ has three states q_0, q_1, q_F (q_0 initial and q_F final) and one active-domain register r. The automaton $\mathcal{A}$ is depicted in Figure 2. Intuitively, $\mathcal{A}$ saves the birthyear of the actor x, follows the edges in E, and nondeterministically guesses an actor y whose birthyear differs by

$$\begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, "\exists z\ (E_b(curr,z) \wedge r' = z)"] & q_1 \arrow[loop above, "\top"] \arrow[r, "\exists z\ (E_b(curr,z) \wedge |r - z| \geq 30)"] & q_F \end{tikzcd}$$

Figure 2: Automaton $\mathcal{A}$ for the query in Example 2.5.

at least 30 from the birthyear of x. It can be seen that $(0, 1)$ is an answer to this query (recall that 0 and 1 refer to Erdős and Bacon).

Example 2.6. Following Example 2.2, suppose we are interested in finding a period of at least 28 days where the total number of active COVID cases grows exponentially. It is a well-known open problem (e.g. see [25, Chapter 5]) whether $\mathbb{R}_{\times,+}$ with the exponential function e^x is decidable. To circumvent this, we could use the standard trick in numerical methods to approximate $\lambda e^{\chi x}$ by the nth Taylor polynomial $p_n(x)$ for sufficiently large n, i.e., $\sum_{i=0}^{n} \lambda(\chi x)^i/i!$. (In practice, one uses small enough n, e.g., $n = 4$ is used in [5].) Let $g(\lambda, \chi, x) := p_n(x)$, with λ and χ treated also as variables. An example query could be to find any window between two time stamps $(t, t') \in E$ spanning across at least 28 days (i.e. $t' - t \geq 28$) such that there exist $\lambda \geq 1, \chi > 0.4$ such that the total number of active COVID cases at each time step s in this interval is within $B := 50$ from $g(\lambda, \chi, s) + c$ for some non-negative constant c. This can be expressed in RDPQ$(\mathbb{R}_{\times,+}, \sigma)$ as $x \rightarrow_{\mathcal{A}} y$, where $\mathcal{A}$ has three states q_0, q_1, q_2 (q_0 initial and q_2 final), one active-domain register r and three parameters λ, χ, c. The transitions are $(q_0, \psi_0(curr, \mathcal{P}, r, r'), q_1)$, $(q_1, \psi_1(curr, \mathcal{P}, r, r'), q_1)$, $(q_1, \psi_2(curr, \mathcal{P}, r, r'), q_2)$, where

$$\psi_0 := r = curr \wedge \lambda \geq 1 \wedge \chi > 0.4 \wedge c \geq 0 \wedge \theta$$

$$\psi_1 := r' = r \wedge \theta, \quad \psi_2 := \psi_1 \wedge curr - r \geq 28$$

$$\theta := \exists x \in adom(T(curr, x) \wedge |g(\lambda, \chi, curr) + c - x| \leq B).$$

3 REALS AND LINEAR ARITHMETICS

This section deals with real-closed field $\mathbb{R}_{\times,+} = \langle \mathbb{R}; +, \times, \leq, 1, 0 \rangle$ and integer linear arithmetic $\mathbb{Z}_{LA} = \langle Z; +, \leq, 1, 0, \{\equiv_k\} \rangle$, where $\equiv_k$ is the congruence modulo k relation. Both admit (effective) quantifier elimination (QE) [22, 29]: each formula $\varphi(\bar{x})$ can be effectively converted into an equivalent quantifier-free formula in the theory. Notice that each side of an (in)equation can be any general term in the theory: a multivariate polynomial with integer coefficients in the case of $\mathbb{R}_{\times,+}$, and a linear multivariate polynomial with integer coefficients in the case of $\mathbb{Z}_{LA}$. In the case of $\mathbb{R}_{\times,+}$ each element in the database is a rational number, i.e., given as a/b with two integers a, b. The size of a number is the number of bits required to represent it. Our main result is:

Theorem 3.1. *The data complexity of* RDPQ(FO, $\mathbb{R}_{\times,+}, \sigma$) *and* RDPQ(FO, $\mathbb{Z}_{LA}, \sigma$) *is NL-complete.*

The remainder of §3 presents a sketch of proof of Theorem 3.1.

Restricted Quantifier Collapse. We recall the notion of Restricted Quantifier Collapse (RQC) only for theories **T** that admit QE; see [35, Definition 13.5] for a general definition, which is not crucial for the paper, and [35, Chapter 13.4] for positive and negative examples as well as some properties of RQC. We also give a definition that applies to fragments of FO. The logic $\mathbb{L} \subseteq$ FO$(\mathbf{T}, \sigma)$ *admits RQC* if $\mathbb{L} \subseteq$ FO$_{\text{act}}(\mathbf{T}, \sigma)$, i.e., for every formula $\varphi(\bar{x}) \in \mathbb{L}$, there exists $\psi(\bar{x}) \in$ FO$_{\text{act}}(\mathbf{T}, \sigma)$ (i.e. without unrestricted quantifiers) such that $[\![\psi]\!]_\mathbf{S} = [\![\varphi]\!]_\mathbf{S}$ for each finite $(\mathbf{T}, \sigma)$-structure **S**. We write "effectively admits RQC" if the translation from φ to ψ is effective.

Proposition 3.2 ([9, 24]). *Both* FO$(\mathbb{R}_{\times,+}, \sigma)$ *and* FO$(\mathbb{Z}_{LA}, \sigma)$ *effectively admit RQC, and have data complexity* TC^0.

The proofs of RQC can also be found in the textbook [35, Theorem 13.19] and in the textbook [25, Proposition 5.32], from which data complexity immediately follows, e.g. see [9, 10] for FO$(\mathbb{R}_{\times,+}, \sigma)$. The same analysis as in [9, 10], combined with that modulo comparisons are in TC^0 (e.g. see [43]), also gives TC^0 data complexity for FO$(\mathbb{Z}_{LA}, \sigma)$. RQC provides us sometimes with a tool for coming up with an NL algorithm for RDPQ, as we shall see the case for $\mathbb{R}_{\times,+}$ and $\mathbb{Z}_{LA}$, but this is not always the case (see next section).

Restricted Register Collapse. We define an analogue of RQC for RDPQ$(\mathbb{L}, \mathbf{T}, \sigma)$. Let RDPQ$_{\text{act}}(\mathbb{L}, \mathbf{T}, \sigma)$ refer to queries $x \rightarrow_{\mathcal{A}} y$ in RDPQ$(\mathbb{L}, \mathbf{T}, \sigma)$ such that $\mathcal{A}$ has no parameters and $\mathbb{L}$ admits RQC. We say that RDPQ$(\mathbb{L}, \mathbf{T}, \sigma)$ admits *Restricted Register Collapse (RRC)* if RDPQ$(\mathbb{L}, \mathbf{T}, \sigma) \subseteq$ RDPQ$_{\text{act}}$(FO$_{\text{act}}, \mathbf{T}, \sigma$). The following lemma is crucial in obtaining NL data complexity of RDPQ(FO, $\mathbb{R}_{\times,+}, \sigma$) and RDPQ(FO, $\mathbb{Z}_{LA}, \sigma$).

Lemma 3.3. *Both* RDPQ(FO, $\mathbb{R}_{\times,+}, \sigma$) *and* RDPQ(FO, $\mathbb{Z}_{LA}, \sigma$) *admit RRC effectively.*

The proof of the lemma is quite technical, and borrows some ideas from the proofs of RQC for FO$(\mathbb{R}_{\times,+}, \sigma)$ and FO$(\mathbb{Z}_{LA}, \sigma)$. Essentially, it is shown that one can restrict the parameter valuations to a bounded number of values while preserving equivalence. Although the exact values might depend on the database **S**, they can be uniformly FO-defined through the values of active domains, i.e., in FO$_{\text{act}}(\mathbb{R}_{\times,+}, \sigma)$ or FO$_{\text{act}}(\mathbb{Z}_{LA}, \sigma)$. This allows us to replace each parameter by active-domain registers.

We now proceed the proof of RRC for $\mathbf{T} := \mathbb{R}_{\times,+}$; the proof for $\mathbb{Z}_{LA}$ is similar (see full version). Fix a vocabulary σ and a $(\mathbf{T}, \sigma)$-register automaton

$$\mathcal{A} = (\mathcal{R}, \mathcal{P}, Q, Q_0, F, \Delta)$$

with $\mathcal{P} = \{p_1, \ldots, p_k\}$. By RQC of FO$(\mathbf{T}, \sigma)$, it follows that each formula $\varphi(curr, \mathcal{P}, \mathcal{R}, \mathcal{R}')$ in Δ can be assumed to have only active-domain quantifiers. We shall construct a $(\sigma, \mathbf{T})$-register automaton

$$\mathcal{A}' = (\mathcal{R}', \mathcal{P} \setminus \{p_k\}, Q', Q'_0, F', \Delta'),$$

such that $[\![\mathcal{A}]\!]_\mathbf{G} = [\![\mathcal{A}']\!]_\mathbf{G}$ for all **T**-embedded graph **G** over σ. By recursively eliminating the parameters, we will obtain an equivalent $(\sigma, \mathbf{T})$-register automaton with no parameters. Let $Z = \mathcal{P} \cup \mathcal{R} \cup \mathcal{R}' \cup \{curr\}$ and $Z_k := Z \setminus \{p_k\}$.

We may assume (e.g. see proof of [35, Theorem 13.19]) that each $\varphi(curr, \mathcal{P}, \mathcal{R}, \mathcal{R}')$ is of the form

$$Q_1 y_1 \in adom \ldots Q_m y_m \in adom\ \alpha(\bar{y}, p_k, Z_k)$$

where each $Q_1, \ldots, Q_m$ denotes an $\forall$ or $\exists$ quantifier and α is a Boolean combination of formulas of

(1) atomic σ-formulas $R(\bar{u})$ with $\bar{u} \subseteq \bar{y}$ (e.g. $E(p_k, r)$ is replaced by $\exists z, z' \in adom(E(z, z') \wedge z = p_k \wedge z' = r)$); and

(2) expressions of the form $t(\bar{y}, Z_k, p_k) \sim 0$, where t is polynomial with integer coefficients whose variables are among $\bar{y}$, Z_k and p_k, and $\sim \in \{=, >\}$.

We will further assume that each term t that appears in some formula (of a transition) in $\mathcal{A}$ appears in *each* formula in $\mathcal{A}$, e.g., if t does not appear in a formula ψ, we may simply rewrite it with $\psi \wedge (t = 0 \vee t \neq 0)$. By the same token, we may for simplicity assume that each formula φ in $\mathcal{A}$ uses the same number m of active-domain variables $y_1, \ldots, y_m$. Letting $Y_k = \bar{y} \cup Z_k$, we may enumerate all polynomial terms t involving p_k occurring in $\mathcal{A}$ as

$$t_1(Y_k, p_k), \ldots, t_m(Y_k, p_k),$$

where t_i has degree d_i in p_k. We introduce two new active-domain registers $r^1(z)$ and $r^2(z)$ for each $z \in Y_k$. Let $r^i(Y_k)$ denote the set containing each $r^i(z)$ with $z \in Y_k$. The crux is to replace p_k by either one of the terms in the set S defined as

$$\left\{ \frac{\text{root}_i^s(r^1(Y_k)) + \text{root}_j^t(r^2(Y_k))}{2} : \begin{array}{l} i, j \in [1, m], \\ s \in [1, d_i], \ \ t \in [1, d_j] \end{array} \right\}$$
$$\cup \ \{\text{root}_i^s(r^1(Y_k)) + c : c \in \{-1, 1\} \wedge i \in [1, m], s \in [1, d_i]\}$$

where $\text{root}_i^s(Y_k)$ represents the sth root of $t_i(Y_k, p_k)$ (treated as a univariate polynomial in p_k), if exists, otherwise 0. It is easy to see that there is a $\text{FO}(\mathbb{R}_{\times,+})$ formula $\text{Root}_i^s(Y_k, x)$ which is true iff x equals $\text{root}_i^s(Y_k)$. By quantifier elimination, we may assume $\text{Root}_i^s(Y_k, x)$ is quantifier-free. At this point, we state a result from embedded finite model theory, which we will use later:

Proposition 3.4 (See Lemma 13.20 of [35]). *Let* S *be an finite structure such that* $adom(\text{S}) \neq \emptyset$ *embedded into* $\mathbf{T} = \mathbb{R}_{\times,+}$. *For each valuation* $\mu : Z_k \to \mathbb{R}$, *it is the case that* $\text{S}, \mu \models_{\mathbf{T}} \exists p_k \varphi$ *iff* $\text{S}, \mu \models_{\mathbf{T}} \bigvee_{t \in S} \varphi[t/p_k]$, *where* $\varphi[t/p_k]$ *is* φ *but with* p_k *replaced by* t.

Note that, in this proposition, we interpret an active-domain register as an active-domain variable.

We now complete the construction of the new register automaton that replaces p_k by active domain registers. Define $Q' := Q \times S$, $Q_0' = Q_0 \times S$, and $F_0' := F_0 \times S$. For each transition $(q, \varphi(Z_k, p_k), q')$ and each $t \in S$, we add to Δ' the transition

$$((q, t), \varphi'(Z_k, r^1(Y_k), r^2(Y_k), r^1(Y_k'), r^2(Y_k')), (q', t))$$

where φ' is constructed as follows depending on t:

- $t = \frac{\text{root}_i^s(r^1(Y_k)) + \text{root}_j^t(r^2(Y_k))}{2}$. Let $\varphi' := \bigwedge_{\alpha=1}^2 r^\alpha(Y_k) = r^\alpha(Y_k') \wedge$
$$\exists z, z_1, z_2 \begin{pmatrix} \varphi(Z_k, z) \wedge & Root_i^s(r^1(Y_k), z_1) \wedge \\ Root_j^t(r^2(Y_k), z_2) & \wedge \ 2z = z_1 + z_2 \end{pmatrix}$$
- $t = \text{root}_i^s(r^1(Y_k)) + c$. Let $\varphi' := \bigwedge_{\alpha=1}^2 r^\alpha(Y_k) = r^\alpha(Y_k') \wedge$
$$\exists z, z_1 (\varphi(Z_k, z) \wedge Root_i^s(r^1(Y_k), z_1) \wedge z = z_1 + c)$$

By RQC of $\text{FO}(\mathbb{R}_{\times,+}, \sigma)$, we may assume each φ' has only active-domain quantifiers.

We claim now that $[\![\mathcal{A}]\!]_{\text{G}} = [\![\mathcal{A}']\!]_{\text{G}}$ for all $(\mathbf{T}, \sigma)$-graph G. The direction $[\![\mathcal{A}]\!]_{\text{G}} \supseteq [\![\mathcal{A}']\!]_{\text{G}}$ is immediate since $\mathcal{A}'$ restricts the range of values that are permitted to p_k in $\mathcal{A}$ (i.e. to values in S, where each $r^h(Y_k)$ can be instantiated by any active-domain values). To prove the other direction $[\![\mathcal{A}]\!]_{\text{G}} \subseteq [\![\mathcal{A}']\!]_{\text{G}}$, assume an accepting computation of $\mathcal{A}$

$$\pi := (q_0, v_0) \to_{(a_1, \varphi_1)} \cdots \to_{(a_n, \varphi_n)} (q_n, v_n)$$

for a valuation $\mu : \mathcal{P} \to \mathbb{R}$, i.e., $(\mathbb{R}_{\times,+}, \text{G}), \mu \models \exists p_k \Phi(Z_k, p_k)$, where $\Phi(Z_k, p_k)$ is:

$$\exists \bar{s}, \bar{R} \in adom \bigwedge_{i=0}^{n-1} (\varphi_i(s_i, R_i, R_{i+1}, \mathcal{P}) \wedge E_{a_i}(s_i, s_{i+1})).$$

By assumption, Φ shares the same polynomial terms with any one of φ_i (up to renaming *curr*, R and R' with respectively s_i, R_i, and R_{i+1}). Using Proposition 3.4, there exist $i, j \in [1, m]$, $i \in [1, d_i]$, $j \in [1, d_j]$, and two tuples $\bar{a}, \bar{b}$ over $adom(\text{G})$, each of length $|Y_k|$, such that $(\mathbb{R}_{\times,+}, \text{G})$ with valuation μ satisfies

$$\Phi(Z_k, (\text{root}_i^s(\bar{a}) + \text{root}_j^t(\bar{b}))/2) \vee \bigvee_{c \in \{-1,1\}} \Phi(Z_k, \text{root}_i^s(\bar{a}) + c)$$

Hence, $\mathcal{A}'$ may simply instantiate $r^1(Y_k)$ (resp. $r^2(Y_k)$) with $\bar{a}$ (resp. $\bar{b}$) and keep them constant throughout the computation, as is done in the definition of $\mathcal{A}'$. Extend each v_i with this instantiation yielding v_i'. Let $\mu' := \mu_{|Z_k}$ be the restriction of μ to Z_k. The following path is then an accepting path of $\mathcal{A}'$ with valuation μ':

$$\pi' := (q_0, v_0') \to_{(a_1, \delta_0')} \cdots \to_{(a_n, \delta_n')} (q_n, v_n).$$

This completes the proof of Theorem 3.3.

NL algorithm. Theorem 3.1 is a direct consequence of the following lemma, Proposition 3.2, and Lemma 3.3.

Lemma 3.5. *We assume a sublogic* $\mathbb{L}$ *of* $\text{FO}(\mathbf{T}, \sigma)$ *with NL data complexity. Then,* $\text{RDPQ}(\mathbb{L}, \mathbf{T}, \sigma)$ *without parameters is NL-complete. It remains NL if we allow parameters that can be instantiated by data values whose sizes are logarithmically bounded.*

Proof idea. One nondeterministically simulates a run of $\mathcal{A}$ in a query $x \to_{\mathcal{A}} y$ over $\text{RDPQ}(\mathbb{L}, \mathbf{T}, \sigma)$, while using the NL algorithm for $\mathbb{L}$ as a subprocedure. References to active-domain values are saved in the working tape, hence keeping the algorithm running in NL. Full proof is in the appendix. □

4 THEORY OF AUTOMATIC RELATIONS

We consider the theory of automatic relations on finite words. Let Γ be a finite alphabet, and $\Gamma_\perp$ be $\Gamma \dot{\cup} \{\perp\}$. Given $u_1, \ldots, u_k \in \Gamma^*$, the *convolution* $u_1 \otimes \cdots \otimes u_k$ of these words is defined as the word $w \in (\Gamma_\perp^k)^*$ such that the projection onto the i-th component of w yields $u_i \cdot \perp^{|w| - |u_i|}$, for every $1 \leq i \leq k$. An *automatic relation* of arity k is a relation $R \subseteq (\Gamma^*)^k$ such that

$$R = \{(u_1, \ldots, u_k) : u_1 \otimes \cdots \otimes u_k \in L\} \tag{1}$$

for some regular language L over $(\Gamma_\perp)^k$. Let *Rel* be the set of all automatic relations over some fixed alphabet Γ. $\mathbf{T}_{\text{Aut}}$ is then the theory $\langle \Gamma^*; \{R\}_{R \in \text{Rel}} \rangle$. Some examples of automatic relations are the prefix relation $\cdot \preceq \cdot$, the equal length relation $eq\text{-}len(\cdot, \cdot)$ and the unary relation $lst_a(\cdot)$ stating that the last letter of the word is an $a \in \Gamma$. In fact, in the relational calculus, these relations coincide with the automatic relations, in the sense that the following are equivalent: (1) R is an automatic relation from Rel of arity k; (2) there is $\varphi(x_1, \ldots, x_k) \in \text{FO}(\Gamma^*, \preceq, eq\text{-}len, \{lst_a\}_{a \in \Gamma})$

such that $R = [\![\varphi]\!] = \{\bar{u} \in (\Gamma^*)^k : \Gamma^* \models \varphi[\bar{u}]\}$ [21]. This is why $\mathrm{FO}(\Gamma^*, \preceq, \mathit{eq\text{-}len}, \{\mathit{lst}_a\}_{a\in\Gamma})$ is sometimes called the "universal automatic structure" [12]. Another example of an automatic relation is the *edit-distance-k* binary relation (see [7]), for any fixed k. Observe that regular languages are exactly the class of automatic relations of arity 1.

We assume that words are represented as a list of letters, and relations as lists of lists of words. The *size* of a word is its length. We represent automatic relations inside formulas via an NFA recognizing the language L in (1).

Example 4.1. Consider a phylogenetic database, with each node defining a DNA sequence (and perhaps some other data). Using $\mathrm{FO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ we can query, whether the DNA sequence of the current element is at edit distance at most 5 from parameter p:

$$\varphi(\mathit{curr}, p) = \exists y\ \mathit{dna\text{-}seq}(\mathit{curr}, y) \land \mathit{edit\text{-}distance\text{-}5}(y, p),$$

where $\mathit{dna\text{-}seq}(\mathit{curr}, y)$ specifies that y is the DNA sequence of the current node *curr* and $\mathit{edit\text{-}distance\text{-}5}(\cdot,\cdot)$ is an automatic relation (of being at edit distance at most 5).

Unfortunately, model checking of FO over $\mathbf{T}_{\mathrm{Aut}}$-embedded structures is hard for the polynomial hierarchy (PH) [11, Prop. 4.12], implying untractability for RDPQ querying:

Proposition 4.2 ([11]). $\mathrm{RDPQ}(\mathrm{FO}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ *is hard for every level of PH in data complexity, even without parameters.*

Hence, we focus on the EFO fragment of FO-formulas in prenex form using no universal quantifiers. The set of EFO formulas using no negation is denoted by EFO^+.

Lemma 4.3 (proof in Appendix). $\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ *has effective RQC.*

Thus, model checking for EFO is tractable, since model checking for $\mathrm{FO}_{\mathrm{act}}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ is in NC^1.

Lemma 4.4. *[11, proof of Corollary 4.15] The model checking problem for* $\mathrm{FO}_{act}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ *is in* NC^1 *in data complexity.*

By Lemma 4.3, every $\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ formula is effectively equivalent to a $\mathrm{FO}_{\mathrm{act}}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ formula which in turn, by Lemma 4.4, is in NC^1, hence in NL. Thus, by Lemma 3.5 the bound transfers to RDPQ evaluation.

Corollary 4.5 (of Lemma 4.3, Lemma 4.4, and Lemma 3.5). *Evaluation of* $\mathrm{RDPQ}(\mathrm{EFO}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ *and* $\mathrm{RDPQ}(\mathrm{FO}_{act}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ *without parameters is NL-complete in data complexity.*

However, as soon as unrestricted parameters are allowed, we lose tractability, as we show next (proof in Appendix B.2).

Lemma 4.6. *Evaluation of* $\mathrm{RDPQ}(\mathrm{EFO}^+, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ *is NP-hard in data complexity, even with no registers and one parameter.*

To regain tractability we assume LogH on the $(\mathbf{T}_{\mathrm{Aut}}, \sigma)$-structure S, that is, that for some constant c, every word $w \in \mathit{adom}$ satisfies $|w| \leq c \cdot \log(|\mathrm{S}|)$. In fact, LogH implies that parameters can be taken to have a logarithmic size, and in light of Lemmas 4.3 and 3.5, we deduce that $\mathrm{RDPQ}(\mathrm{EFO}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ evaluation (under LogH) is in NL. To prove this, we use a corollary of the proof of Lemma 4.3 (in Appendix B.1):

Corollary 4.7 (of Lemma 4.3). *Every* $\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ *formula* $\psi(\bar{z})$ *is effectively equivalent to a disjunction of* $\mathrm{EFO}_{act}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ *formulas* ψ' *of the form* $A'(\bar{z}) \land \exists \bar{r} \in \mathit{adom}\ \tau(\bar{z}\bar{r})$*, where* A' *is a* $\mathbf{T}_{\mathrm{Aut}}$*-atom,* τ *is a* $\mathrm{EFO}_{act}(\sigma)$ *formula.*

Let $N = \max_{w\in \mathit{adom}} |w|$ *and let* $v : \bar{z} \to \Gamma^*$ *be a satisfying assignment for* $\psi'(\bar{z})$*, and* $z \in \bar{z}$ *such that* $|v(z)| > N$*. Suppose there is a word* $u \in \Gamma^*$ *such that* $|u| > N$ *and* $\Gamma^*, v' \models A'(\bar{z})$ *for* $v' = v[z \mapsto u]$*. Then,* v' *is also a satisfying assignment for* ψ'*.*

Theorem 4.8. *Evaluation of* $\mathrm{RDPQ}(\mathrm{EFO}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ *with parameters under LogH is NL-complete in data complexity.*

Proof. Let $\mathbf{G}$ be a $\mathbf{T}_{\mathrm{Aut}}$-embedded graph over σ, $F = \exists \mathcal{P} : x \to_{\mathcal{A},\mathcal{P}} y$ be an $\mathrm{RDPQ}(\mathrm{EFO}, \mathbf{T}_{\mathrm{Aut}}, \sigma)$ query, and $(s,t) \in V^2$ be two vertices of $\mathbf{G}$. Let N be the maximum size of a word in the active domain (of logarithmic size by LogH).

Let $s = v_0 \to_{a_1} \cdots \to_{a_n} v_n = t$ be an accepting $\mathbf{G}$-path with transitions $t_0, \ldots, t_n$, where $t_i = (q_i, (a_i, \psi_i), q_{i+1})$ and $a_0 = \#$. Then for some register valuations $\mu_0, \ldots, \mu_{n+1} \in V^{\mathcal{R}}$ the formula $\varphi(\mathcal{P}) = \bigwedge_i \psi_i(v_i, \mathcal{P}, \mu_i, \mu_{i+1})$ is satisfiable. We show that if φ is satisfiable, then it is satisfied with a valuation for $\mathcal{P}$ with log-size words.

Suppose there is a parameter valuation $\nu \in U^{\mathcal{P}}$ which satisfies φ. By Corollary 4.7, for each i, ψ_i can be written as a disjunction of formulas of the form $R_i(\bar{z}) \land \exists \bar{r} \in \mathit{adom}\ \tau(\bar{z}\bar{r})$ where R_i is a $\mathbf{T}_{\mathrm{Aut}}$-atom and τ is a $\mathrm{FO}_{\mathrm{act}}(\sigma)$ formula. Consider the DFA A_i over $\Gamma_\perp^{n_i}$ corresponding to the relation denoted by R_i, let it have n_i free variables. Let $\bar{v}_i$ be a satisfying assignment for $\bar{y}_i$. Hence, there is an accepting run of A_i over the convolution w of the words $v_i \nu \mu_i \mu_{i+1} \bar{v}_i$. These words are of logarithmic size, since they are from *adom*, with a possible exception of those from ν, which are parameters. Let w' be the N-prefix of w, and let q_i be the state of A_i after reading w' from the initial state. We define A'_i to be the same automaton as A_i but (1) setting q_i as initial state, and (2) removing any transition reading a non-$\perp$ symbol on any component which is not of the parameters. It then follows that, if w'' is any word in the language of A'_i, then $w' \cdot w''$ is in the language of A_i. Let $\pi(A'_i)$ be the automaton corresponding to the projection onto the components of the parameters of $L(A'_i)$. Consider a word $u \in \bigcap_i L(\pi(A'_i)) \subseteq (\Gamma_\perp^{|\mathcal{P}|})^*$; observe that its description is bounded by a function of the (fixed) RDPQ F, and hence u is of constant size. It then follows, by Corollary 4.7, that $u' \cdot u$ witnesses also a satisfying assignment for φ, where u' is the N-prefix of ν. This shows that we can assume that the parameter valuation for witnessing a word in the language is also of logarithmic size. Since parameters can be taken to have a logarithmic size, by Lemmas 4.3, 4.4, and 3.5 we get an NL bound. ◻

5 WORD EQUATIONS

This section deals with the theory of word equations with concatenation and regular languages, i.e., let $\mathbf{T}_{\mathrm{WE}}$ be the theory of $\langle \Gamma^*; \cdot, \{R\}_{R\in\mathrm{Reg}}\rangle$, where Γ is a finite alphabet, $\cdot$ is the concatenation, and Reg is the set of all regular languages (over Γ). The data complexity of $\mathrm{FO}(\mathbf{T}_{\mathrm{WE}}, \sigma)$ is undecidable [11], so we consider two fragments of $\mathrm{FO}(\mathbf{T}_{\mathrm{WE}}, \sigma)$: (1) $\mathbb{L}^+_{\mathrm{WE}} = \mathrm{EFO}^+_{\mathrm{act}}(\mathbf{T}_{\mathrm{WE}}, \sigma)$ of existential positive formula with active-domain quantifiers, and (2) $\mathbb{L}_{\mathrm{PM}}$ of existential positive formulas without regular relations and whose allowed $\mathbf{T}_{\mathrm{WE}}$-formulas have atoms of the form $\exists \bar{y}(\bigwedge_{i\in I} x_i = \beta_i)$ with the left-hand side x_i being a single element from $curr \cup \mathcal{R} \cup \mathcal{R}' \cup \mathcal{P}$ and the right-hand side β_i being a concatenation of elements in $\bar{y} \cup \mathcal{R} \cup \mathcal{R}' \cup \{curr\}$ and constants, and that each existentially quantified variable occurs at most once in an equation whose left-hand side is from $\mathcal{P}$. (Note that it can occur at multiple equations in which the left-hand side is from $curr \cup \mathcal{R} \cup \mathcal{R}'$.)

Example 5.1. Consider a query in a phylogenetic database, defined in Example 4.1: *is there a path, such that all DNA sequences on it are at edit distance at most k from some (unknown) DNA sequence*?

Answering such query is related to the closest string problem [15, 26]. It is known [7, Eq. (3)], that the edit distance between p, w is at most k if and on only if

$$\bigvee_{\substack{a_1,\dots,a_k \in \Gamma\cup\{\varepsilon\}\\ b_1,\dots,b_k\in\Gamma\cup\{\varepsilon\}}} \exists x_0,\dots,x_k \;\; \begin{array}{l} p = x_0 a_1 x_1 \cdots x_{k-1} a_k x_k \;\wedge \\ w = x_0 b_1 x_1 \cdots x_{k-1} b_k x_k \;\;. \end{array}$$

When p is a parameter and w in $curr$, this formula is in $\mathbb{L}_{\mathrm{PM}}$ and so this query is expressible in $\mathrm{RDPQ}(\mathbb{L}_{\mathrm{PM}}, \mathbf{T}_{\mathrm{WE}}, \sigma)$.

Theorem 5.2. *$\mathrm{RDPQ}(\mathbb{L}^+_{\mathrm{WE}}, \mathbf{T}_{\mathrm{WE}}, \sigma)$ has NL-complete data complexity, similarly $\mathrm{RDPQ}(\mathbb{L}_{\mathrm{PM}}, \mathbf{T}_{\mathrm{WE}}, \sigma)$.*

Proof idea. As in both cases the guards on a $\mathbf{G}$-path are positive formulas, the guard on any $\mathbf{G}$-path π are a positive Boolean combination of word equations over some variables and traversing such a path can be seen as constructing a system of word equations. In particular, given a valuation of the parameters and existentially quantified variables such that $\mathbf{G}, \pi \models \mathcal{A}$ we can determine, which atomic equations were true and this valuation can be seen as a solution to the system of word equations (i.e. the equations that are made true in the guards). Hence evaluating a query can be seen as solving a system of word equations, though this system is not given explicitly. The best algorithms solving word equations run in PSPACE, so in order to answer RDPQ queries in NL, we need some insight into the structure of the systems of equations that we obtain in this case. Those turn out to be different for $\mathbb{L}^+_{\mathrm{WE}}$ and $\mathbb{L}_{\mathrm{PM}}$.

For $\mathbb{L}_{\mathrm{PM}}$: each atomic equation is of the form $p = \beta$ or $\alpha = \beta$, where p is a parameter, $\alpha \in curr \cup \mathcal{R} \cup \mathcal{R}'$ and β is a sequence of (existentially quantified) variables and elements from $\Gamma \cup curr \cup \mathcal{R} \cup \mathcal{R}'$. In the second case $\alpha \in adom(\mathbf{S})$, so every (existentially quantified) variable in β is a substring of an element in $adom(\mathbf{S})$, so it can be represented in NL. For equations $p = \beta$, fix a parameter p and consider all such equations in the system created while traversing the $\mathbf{G}$-path. Every variable at the right-hand side is used once in the system and not used in any other such subsystem. Such system is satisfiable if and only if each of the equations satisfies a condition on a prefix and suffix of its sides; in particular, we do not need to consider all the equations simultaneously, they can be checked one by one, see Lemma C.1 in the appendix. The NL algorithm guesses for each parameter p an appropriate prefix and suffix, it can be shown that those can be represented as concatenations of at most $|\mathcal{A}|$ many constants and substrings of elements from $adom(\mathbf{S})$. Then in $\mathbf{G}$ it guesses consecutive edge views and evaluates the guard φ using the condition on prefixes and suffixes mentioned above, i.e. the one from Lemma C.1.

Concerning $\mathbb{L}^+_{\mathrm{WE}}$, for simplicity of presentation, we ignore regular relations; the appendix contains a brief discussion how to generalize to this case. The (existential) active domain quantifiers are instantiated by elements of $adom(\mathbf{S})$ using nondeterministic guesses. Therefore the equations occurring in guards are only in variables that are parameters. We first observe that if words substituted for parameters are polynomial-length and there is an NL access to individual letters in them, then $\mathrm{RDPQ}(\mathbb{L}^+_{\mathrm{WE}}, \mathbf{T}_{\mathrm{WE}}, \sigma)$ has NL data complexity, as desired.

Lemma 5.3. *Assume a logarithmic-size representation of valuations of parameters $\mathcal{P}$, such that each parameter is of polynomial (in $|\mathbf{G}|$) length and individual letter of each parameter and length of each parameter can be accessed in* NL. *Then we can evaluate a given* $\mathrm{RDPQ}(\mathbb{L}^+_{\mathrm{WE}}, \mathbf{T}_{\mathrm{WE}}, \sigma)$ *query for those parameters on* $\mathbf{G}$ *in* NL.

Proof. We guess consecutive vertices on a $\mathbf{G}$-path. At a given node we guess the next node in $\mathbf{G}$, the letter and transition of $\mathcal{A}$, which yields a guard φ that should be satisfied. To verify φ, we guess the existentially quantified variables, which are an element from *adom*, and then evaluate the atomic equations in the guards one by one. For a given equation $u(\bar{x}) = v(\bar{x})$ we already have the substitutions for each variable in $\bar{x}$, which is either an active-domain quantified (so was guessed) or a parameter (so we have a valuation for it). Hence we verify the equation symbol by symbol, which can be done in NL, as each substituted value is of polynomial length and individual letters can be accessed in NL. As each atom of φ is evaluated in NL, we can also evaluate φ in NL.

If we have ended in t then the traversed path π yields $\mathbf{G}, \pi \models \mathcal{A}$. On the other hand, if there is π such that $\mathbf{G}, \pi \models \mathcal{A}$ then we guess π's consecutive nodes, the transitions of $\mathcal{A}$ and substitutions for variables, for which we end up at t, as desired. ◻

We show that if there is an accepting $\mathbf{G}$-path π, and parameter valuation $\nu : \mathcal{P} \to \Gamma^*$ such that $\mathbf{G}, \pi \models \mathcal{A}$ then this also holds for a valuation ν' such that each assigned value is polynomial-size and its letters can be accessed in NL.

So assume such a path π and corresponding sequence of transitions in $\mathcal{A}$. For any guard φ take the equations made true under ν, they form a system of word equations in k variables $\mathcal{P}$; the sum of lengths of equations is polynomial in $|\mathbf{G}|$: the length $|\pi|$ is polynomial (discussion after Proposition 2.3), and each atom in it is from $\Gamma^* \cup curr \cup \mathcal{R} \cup \mathcal{R}'$, hence of linear size. Lemma C.2 in the Appendix show that such a system has a polynomial-size solution. Moreover, it is known [40] that for (some) such solutions the *length* of the substitutions together with the system of equations uniquely determines the solution and yield access to letters in the substitutions, in this case the access is in NL; see the appendix. However, the *size* of this representation is polynomial, as there are (potentially) polynomially many equations.

We exploit the form of equations in the system: fix an atomic equation $u(\bar{p}, \bar{x}) = v(\bar{p}, \bar{x})$, where $\bar{p}$ are from $\mathcal{P}$ and $\bar{x}$ are from $curr, \mathcal{R}, \mathcal{R}'$ and active domain quantified variables. For different substitutions for $\bar{x}$, the obtained equations have the same sequence of variables (from $\mathcal{P}$), separated with different words; call such equations *similar.* We show that given two similar equations we can deduce a partial valuation of variables, which makes those two equations equivalent (in appropriate sense). By considering consecutive equations we get a partial substitution that makes increasing set of equations equivalent. Moreover, this process "simplifies" the resulting equation and we can perform it $O(m)$ times, where m is the number of occurrences of variables in the equation. As a result, among a system of similar equations we can choose $O(m)$ which unify this system. This is done for each equation $u(\bar{p}, \bar{x}) = v(\bar{p}, \bar{x})$ in $\mathcal{A}$, yielding a system of $O(m|\mathcal{A}|)$ equations, which can be used to access the substitution for parameters. See appendix for details. □

6 GENERALIZATIONS AND CONCLUSIONS

We conclude by mentioning related issues and future work.

Conjunctive Queries. NL data complexity of RDPQ easily extends to CRDPQ [36], which enriches RDPQ by conjunctions. Here we enrich RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) not only by conjunctions, but also by an additional constraint in $\mathbb{L}$. Namely, queries in CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$) are conjunctions of the form

$$Q(\bar{z}) \leftarrow \bigwedge_{i=1}^{n} x_i \rightarrow_{\mathcal{A}_i} y_i \wedge \varphi(\mathcal{P}, \bar{x}, \bar{y})$$

where $\bar{z} \subseteq \bar{x} \cup \bar{y}$, $x_i \rightarrow_{\mathcal{A}_i} y_i$ is an RDPQ($\mathbb{L}, \mathbf{T}, \sigma$)-query, and $\varphi(\mathcal{P}, \bar{x}, \bar{y}) \in \mathbb{L}$. Here, $\mathcal{P}$ contains all parameters used across all register automata in Q, which may among themselves share parameters. Given a $(\mathbf{T}, \sigma)$-structure $\mathbf{S}$ with active domain U, let $[\![Q]\!]_{\mathbf{S}}$ be the set of all $\beta : \bar{z} \rightarrow U$ such that for some $\nu : \bar{x} \cup \bar{y} \rightarrow U$, with $\beta(z) = \nu(z)$ for each $z \in \bar{z}$, and $\mu : \mathcal{P} \rightarrow \mathcal{D}$, it is the case that $\mathbf{S} \models \varphi(\mu(\mathcal{P}), \nu(\bar{x}), \nu(\bar{y}))$ and $\nu(x_i) \rightarrow_{\mathcal{A}_i} \nu(y_i)$. In other words, each path constraint $x_i \rightarrow_{\mathcal{A}_i} y_i$ is satisfied where a parameter valuation μ that conforms to φ. One example of a CRDPQ query is the query that there are two actors with finite Bacon graphs whose ages differ from Bacon by an even number k, i.e., this is of the form

$$Q(y, z) \leftarrow x \rightarrow_{\mathcal{A}} y \wedge x \rightarrow_{\mathcal{A}} z \wedge k \equiv 0 \pmod 2 \wedge x = \text{Bacon},$$

where $\mathcal{A}$ uses one active-domain register r to save the birthyear y_0 of the first person in the path, and a parameter k that is nondeterministically guessed and is checked against the difference between y_0 and the birthyear of the final person in the path. We sketch in the appendix a log-space reduction from CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$) to RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) query evaluation, where the output query depends only on the input query. This extends NL data complexity to CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$).

Property Graphs. Property graphs are the data models used by most modern graph database systems [17]. A property graph associates unique IDs to both nodes and edges in a graph, i.e., they are both first-class citizens. The main difference to the standard data graph model in database theory (e.g., see [36]) is three-fold. First, some edges could be unordered and there could be multiple edges from a node to another node. second, nodes (as well as edges) may also carry labels from Σ. Finally, edges (as well as nodes) may be associated with a property value. Since we allow first-order views, we may easily view a property graph $\mathbf{G}$ as a data graph $\mathbf{G}'$ in the sense of [36] by interpreting nodes and edges in $\mathbf{G}$ as nodes in $\mathbf{G}'$. Indeed, view definitions allow a huge amount of flexibility, e.g., one can reverse the edge relation, make an edge relation symmetric, etc. This allows us to also obtain NL data complexity of RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) on the property graph data model.

Combined Complexity. The primary concern of our paper is developing data path query languages with NL data complexity. The consideration of data complexity is standard and sensible for database query evaluation since typically the size of the database is very large and the size of the query small. The combined complexity of our query languages is higher than that of the basic RDPQ from [36] (i.e. PSPACE), primarily because of the high complexity of the theories that we consider, e.g., the best known algorithm for FO($\mathbb{R}_{\times,+}$) is double-exponential time (see [8]) and FO($\mathbb{Z}_{LA}$) would require at least double exponential-time [33]. We leave it for future work to identify fragments of FO($\mathbf{T}, \sigma$) that could match the PSPACE combined complexity of the basic RDPQ.

Integrating views into the automaton. Thus far, our presentation has kept the view definitions $G = (V, \{E_a\}_{a \in \Sigma})$ of the relational database $\mathbf{S}$ rather "separate" from the automaton $\mathcal{A}$ in the query $x \rightarrow_{\mathcal{A}} y$. In effect, the automaton $\mathcal{A}$ is forced to explore the relational database $\mathbf{S}$ through the graph views as defined by G. Although this is more in line with the data model of SQL/PGQ (e.g. see [17] and https://pgql-lang.org/), a further generalization that enables $\mathcal{A}$ to directly traverse the relational structure $\mathbf{S}$ is possible, which we will discuss next.

For a start, we dispense completely with the view definitions over the database $\mathbf{S}$. How does $\mathcal{A}$ then traverse $\mathbf{S}$? The answer is simple. Each transition of $\mathcal{A}$ can be associated with a *next* variable indicating which element in $\mathbf{S}$ it can go to, i.e., it is of the form

$$(q, \varphi(\mathit{curr}, \mathit{next}, \mathcal{R}, \mathcal{R}', \mathcal{P}, \mathcal{P}'), q'),$$

where φ is a formula over the vocabulary σ of $\mathbf{S}$. Notice that Σ does not exist, and therefore an automata transition does not have any Σ entry. The semantics of a transition is the same as defined in § 2, except that by taking this transition the automaton $\mathcal{A}$ goes from *curr* to *next*. This setting is strictly more general than the setting defined in § 2 in that the graph views allowed by this extended automaton might depend on the register and parameter values. All the results in this paper generalize easily to this extended data model.

Future Work. We now mention several open questions. Firstly, in what way can our query languages be extended with aggregation (e.g., summing values along a path, taking average, etc.) without sacrificing NL data complexity? To this end, we conjecture that decidable subclasses of counter automata (e.g. [32, 42]) could prove useful. Secondly, identify the fragment of the graph pattern matching language (GPML) over property graphs [17] that can be captured in CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$). Concerning word equations, can we extend the methods used for $\mathbb{L}_{\text{WE}}^{+}$ to obtain NL data complexity for RDPQ over fragment with negation and/or with existential quantification?

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for the helpful feedback. We also thank Michael Benedikt, Leonid Libkin, Domagoj Vrgoč, and Zhilin Wu for the discussion and insightful comments. Diego Figueira is partially supported by ANR QUID, grant ANR-18-CE40-0031. Artur Jeż work is supported under National Science Centre, Poland project number 2017/26/E/ST6/00191. Anthony Lin was supported by the ERC Starting Grant 759969 (AV-SMP) and a Max-Planck Society Fellowship.

REFERENCES

[1] Étienne André. 2019. What's decidable about parametric timed automata? *Int. J. Softw. Tools Technol. Transf.* 21, 2 (2019), 203–219. https://doi.org/10.1007/s10009-017-0467-0

[2] Renzo Angles and Claudio Gutiérrez. 2008. Survey of graph database models. *ACM Comput. Surv.* 40, 1 (2008), 1:1–1:39. https://doi.org/10.1145/1322432.1322433

[3] Pablo Barceló Baeza. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, Richard Hull and Wenfei Fan (Eds.). ACM, 175–188. https://doi.org/10.1145/2463664.2465216

[4] Jorge Baier, Dietrich Daroch, Juan L. Reutter, and Domagoj Vrgoc. 2017. Evaluating Navigational RDF Queries over the Web. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media, HT 2017, Prague, Czech Republic, July 4-7, 2017*, Peter Dolog, Peter Vojtás, Francesco Bonchi, and Denis Helic (Eds.). ACM, 165–174. https://doi.org/10.1145/3078714.3078731

[5] Abebe Alemu Balcha. 2020. Curve Fitting and Least Square Analysis to Extrapolate for the Case of COVID-19 Status in Ethiopia. *Advances in Infectious Diseases* 10 (2020). Issue 3.

[6] Pablo Barceló, Gaëlle Fontaine, and Anthony Widjaja Lin. 2015. Expressive Path Queries on Graph with Data. *Log. Methods Comput. Sci.* 11, 4 (2015). https://doi.org/10.2168/LMCS-11(4:1)2015

[7] Pablo Barceló, Leonid Libkin, Anthony Widjaja Lin, and Peter T. Wood. 2012. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Trans. Database Syst.* 37, 4 (2012), 31:1–31:46. https://doi.org/10.1145/2389241.2389250

[8] Saugata Basu. 2014. Algorithms in Real Algebraic Geometry: A Survey. *CoRR* abs/1409.1534 (2014). arXiv:1409.1534 http://arxiv.org/abs/1409.1534

[9] Michael Benedikt and Leonid Libkin. 1996. On the Structure of Queries in Constraint Query Languages. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996.* IEEE Computer Society, 25–34. https://doi.org/10.1109/LICS.1996.561300

[10] Michael Benedikt and Leonid Libkin. 2000. Relational queries over interpreted structures. *J. ACM* 47, 4 (2000), 644–680. https://doi.org/10.1145/347476.347477

[11] Michael Benedikt, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. 2003. Definable relations and first-order query languages over strings. *J. ACM* 50, 5 (2003), 694–751. https://doi.org/10.1145/876638.876642

[12] Achim Blumensath and Erich Grädel. 2000. Automatic Structures. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000.* IEEE Computer Society, 51–62. https://doi.org/10.1109/LICS.2000.855755

[13] Achim Blumensath and Erich Grädel. 2004. Finite presentations of infinite structures: Automata and interpretations. *Theory of Computing Systems* 37, 6 (2004), 641–674.

[14] Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2011. Two-variable logic on data words. *TOCL* 12, 4 (2011), 27:1–27:26. https://doi.org/10.1145/1970398.1970403

[15] Christina Boucher and Bin Ma. 2011. Closest string with outliers. *BMC Bioinform.* 12, S-1 (2011), S55. https://doi.org/10.1186/1471-2105-12-S1-S55

[16] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000*, Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman (Eds.). Morgan Kaufmann, 176–185.

[17] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Hannes Voigt, Oskar van Rest, Domagoj VrgoÄŊ, Mingxi Wu, and Fred Zemke. 2021. Graph Pattern Matching in GQL and SQL/PGQ. arXiv:2112.06217 [cs.DB]

[18] Alin Deutsch and Val Tannen. 2001. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *Database Programming Languages, 8th International Workshop, DBPL 2001, Frascati, Italy, September 8-10, 2001, Revised Papers (Lecture Notes in Computer Science, Vol. 2397)*, Giorgio Ghelli and Gösta Grahne (Eds.). Springer, 21–39. https://doi.org/10.1007/3-540-46093-4_2

[19] Volker Diekert. 2002. Makanin's Algorithm. In *Algebraic Combinatorics on Words*, M. Lothaire (Ed.). Encyclopedia of Mathematics and its Applications, Vol. 90. Cambridge University Press, Chapter 12, 387–442.

[20] Volker Diekert, Artur Jeż, and Wojciech Plandowski. 2016. Finding all solutions of equations in free groups and monoids with involution. *Inf. Comput.* 251 (2016), 263–286. https://doi.org/10.1016/j.ic.2016.09.009 Conference version in Proc. CSR 2014, LNCS 8476 (2014).

[21] S Eilenberg, C.C Elgot, and J.C Shepherdson. 1969. Sets recognized by n-tape automata. *Journal of Algebra* 13, 4 (1969), 447–464. https://doi.org/10.1016/0021-8693(69)90107-0

[22] Herbert B. Enderton. 2001. *Introduction to Mathematical Logic* (2 ed.). Academic Press.

[23] Daniela Florescu, Alon Y. Levy, and Dan Suciu. 1998. Query Containment for Conjunctive Queries with Regular Expressions. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, Alberto O. Mendelzon and Jan Paredaens (Eds.). ACM Press, 139–148. https://doi.org/10.1145/275487.275503

[24] Jörg Flum and Martin Ziegler. 1999. Pseudo-Finite Homogeneity and Saturation. *J. Symb. Log.* 64, 4 (1999), 1689–1699. https://doi.org/10.2307/2586806

[25] E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein. 2007. *Finite Model Theory and Its Applications.* Springer.

[26] Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. 2003. Fixed-Parameter Algorithms for CLOSEST STRING and Related Problems. *Algorithmica* 37, 1 (2003), 25–42. https://doi.org/10.1007/s00453-003-1028-3

[27] Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. 2009. Reachability in Succinct and Parametric One-Counter Automata. In *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5710)*, Mario Bravetti and Gianluigi Zavattaro (Eds.). Springer, 369–383. https://doi.org/10.1007/978-3-642-04081-8_25

[28] Jelle Hellings, Bart Kuijpers, Jan Van den Bussche, and Xiaowang Zhang. 2013. Walk logic as a framework for path query languages on graph databases. In *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, Wang-Chiew Tan, Giovanna Guerrini, Barbara Catania, and Anastasios Gounaris (Eds.). ACM, 117–128. https://doi.org/10.1145/2448496.2448512

[29] Wilfrid Hodges. 1997. *A Shorter Model Theory.* Cambridge University Press.

[30] Artur Jeż. 2016. Recompression: a simple and powerful technique for word equations. *J. ACM* 63, 1 (Mar 2016), 4:1–4:51. https://doi.org/10.1145/2743014

[31] Michael Kaminski and Nissim Francez. 1994. Finite-Memory Automata. *Theor. Comput. Sci.* 134, 2 (1994), 329–363. https://doi.org/10.1016/0304-3975(94)90242-9

[32] Eryk Kopczynski and Anthony Widjaja To. 2010. Parikh Images of Grammars: Complexity and Applications. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom.* 80–89. https://doi.org/10.1109/LICS.2010.21

[33] Dexter C. Kozen. 2006. *Theory of Computation.* Springer.

[34] G. M. Kuper, L. Libkin, and J. Paredaens (Eds.). 2000. *Constraint Databases.* Springer.

[35] Leonid Libkin. 2004. *Elements of Finite Model Theory.* Springer.

[36] Leonid Libkin, Wim Martens, and Domagoj Vrgoc. 2016. Querying Graphs with Data. *J. ACM* 63, 2 (2016), 14:1–14:53. https://doi.org/10.1145/2850413

[37] Leonid Libkin and Domagoj Vrgoc. 2012. Regular path queries on graphs with data. In *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, Alin Deutsch (Ed.). ACM, 74–85. https://doi.org/10.1145/2274576.2274585

[38] Alberto O. Mendelzon and Peter T. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Comput.* 24, 6 (1995), 1235–1258. https://doi.org/10.1137/S009753979122370X

[39] Wojciech Plandowski. 2004. Satisfiability of word equations with constants is in PSPACE. *J. ACM* 51, 3 (2004), 483–496. https://doi.org/10.1145/990308.990312

[40] Wojciech Plandowski and Wojciech Rytter. 1998. Application of Lempel-Ziv Encodings to the Solution of Word Equations. In *ICALP (LNCS, Vol. 1443)*, Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel (Eds.). Springer, 731–742. https://doi.org/10.1007/BFb0055097

[41] Luc Segoufin and Szymon Torunczyk. 2011. Automata based verification over linearly ordered data domains. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany (LIPIcs, Vol. 9)*, Thomas Schwentick and Christoph Dürr (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 81–92. https://doi.org/10.4230/LIPIcs.STACS.2011.81

[42] Anthony Widjaja To. 2009. Model Checking FO(R) over One-Counter Processes and beyond. In *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings.* 485–499. https://doi.org/10.1007/978-3-642-04027-6_35

[43] Heribert Vollmer. 1999. *Introduction to Circuit Complexity.* Springer.

[44] Domagoj Vrgoc, Carlos Rojas, Renzo Angles, Marcelo Arenas, Diego Arroyuelo, Carlos Buil Aranda, Aidan Hogan, Gonzalo Navarro, Cristian Riveros, and Juan Romero. 2021. MillenniumDB: A Persistent, Open-Source, Graph Database. *CoRR* abs/2111.01540 (2021). arXiv:2111.01540 https://arxiv.org/abs/2111.01540

[45] Peter T. Wood. 2012. Query languages for graph databases. *SIGMOD Rec.* 41, 1 (2012), 50–60. https://doi.org/10.1145/2206869.2206879

A APPENDIX TO SECTION 3

A.1 Proof of Lemma 3.5

Fix a $(\mathbf{T}, \sigma)$-register automaton with parameters

$$\mathcal{A} = (\mathcal{R}, \mathcal{P}, Q, Q_0, F_0, \Delta),$$

and a logarithmic function f. Say $|\mathcal{R}| = k$. Given two vertices s, t of a given graph $\mathbf{G}$ with set V of nodes and universe $\mathcal{D}$ of data values, we want to know if there is some parameter assignment $\mu : \mathcal{P} \to \{d \in \mathcal{D} : |d| \leq f(|\mathbf{G}|)\}$ for which the query returns (s, t). The NL machine T first guesses μ (using logarithmic space) and then simply traverses $\mathbf{G}$, while simultaneously simulating $\mathcal{A}$ (as in standard product automata construction). More precisely, it will initially guess a state $q_0 \in Q_0$ in the simulation, and set the register values in $\mathcal{R}$ to an arbitrary active-domain values $\mathbf{r}_0 \in \mathcal{D}^k$, and start the procedure from configuration $(s, q_0, \mathbf{r})$. From any configuration $(v, q, \mathbf{r})$, the machine T guesses a node v' and a label $a \in \Sigma$ such that $(v, v') \in E_a$, the next register (active-domain) values $\mathbf{r'}$, and a transition

$$(q, (a, \varphi(curr, \mathcal{P}, \mathcal{R}, \mathcal{R}')), q')$$

such that $\mathbf{G} \models \varphi(v, \mu, \mathbf{r}, \mathbf{r'})$. The simulation proceeds to the next configuration $(v', q', \mathbf{r'})$. It will stop and accept as soon as a configuration of the form $(t, q_F, \mathbf{r})$, for some $q_F \in F$ and register values $\mathbf{r}$, is visited.

Recall that an NL machine has an input tape, and a log-space working memory tape. To ensure an NL simulation, we use the standard trick of using pointers that refer to the parts of the input tape that contain the node/datum of interest in $\mathbf{G}$. Each such a pointer reference uses space that is logarithmic in the size of the input (i.e. $\mathbf{G}$).

B APPENDIX TO SECTION 4

B.1 Proof of Lemma 4.3

We actually show that, in terms of expressive power, we have:

(1) $\mathrm{EFO}^+(\mathbf{T}_{\mathrm{Aut}}, \sigma) \subseteq \mathrm{EFO}^+_{\mathrm{act}}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ and
(2) $\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma) \subseteq \mathrm{FO}_{\mathrm{act}}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$.

To prove (1), first note that an existential-positive input formula $\varphi(\bar{x})$ is equivalent to a disjunction of formulas of the form $\varphi'(\bar{x}) = \exists \bar{y}\, \psi(\bar{z}) \wedge \bigwedge_{i \in I} H_i(\bar{z}_i)$, where variables from $\bar{z}$ and $\bar{z}_i$ range over $\bar{x}\bar{y}$, each H_i is a relation from the schema σ, and ψ is a formula of $\mathrm{EFO}^+(\mathbf{T}_{\mathrm{Aut}})$. We can further push any existential quantification of variable from $\bar{y}$ which is not in $\bigcup_{i \in I} \bar{z}_i$ inside ψ. In other words, we can assume that every variable of $\bar{y}$ is either free (i.e., in $\bar{x}$) or in some $\bar{z}_i$. Now we can consider the automatic relation $R_\psi = \{\bar{u} : \Gamma^* \models \psi[\bar{u}]\}$ and redefine φ' as the equivalent formula $\exists \bar{y} \in adom\, R_\psi(\bar{z}) \wedge \bigwedge_{i \in I} H_i(\bar{z}_i)$.

The proof for (2) is similar to the existential positive case with a twist. By a similar reasoning as before, it suffices to assume that we deal with the formula $\varphi(\bar{x})$ of the form $\exists \bar{y} \left(\bigwedge_{i \in I} (\neg) R_i(\bar{z}_i) \wedge A(\bar{z}) \right)$, where $y = y_1, \ldots, y_n$, $R_i \in \sigma$ and A is an automatic relation. Furthermore, we may assume that $\bar{y} \cap \bar{z}_i \neq \emptyset$ and $\bar{y} \cap \bar{z} \neq \emptyset$; otherwise, we pull the corresponding atom out of the scope of $\bar{y}$. Let $<_{lex}$ be the 'shortlex' lexicographic order on Σ-words (here, we implicitly assume a total ordering on Σ). That is, $v <_{lex} w$ iff (i) $|v| < |w|$, or (ii) $|v| = |w| = n$, $v = a_1 \cdots a_n$, $w = b_1 \cdots b_n$, and there is $i \in [1, n]$ such that $a_i < b_i$ and $a_1 \cdots a_{i-1} = b_1 \cdots b_{i-1}$. Observe that $<_{lex}$ is a total linear order and is also automatic. We assume below that the variables $x, x_1, x_2, \ldots$ do not appear in $\bar{z}$.

Define $\theta(x_1, x_2)$ and $\max_{adom}(x)$ as follows:

$$\begin{aligned}
\theta(x_1, x_2) &:= x_1 <_{lex} x_2 \wedge \neg \exists x \in adom(x_1 <_{lex} x <_{lex} x_2), \\
\max_{adom}(x) &:= \forall x_1 \in adom(x_1 <_{lex} x), \\
\min_{adom}(x) &:= \forall x_1 \in adom(x <_{lex} x_1).
\end{aligned}$$

Then, φ is equivalent to a big disjunction of $\psi_{\bar{C}}$, where $\bar{C} = (C_1, \ldots, C_n)$ and $C_i \in \{\min_{adom}(r_i), \theta(r_i, r_i'), \max_{adom}(r_i)\}$. The idea is that by a satisfying valuation of C_i we guess where the valuation of y_i appears w.r.t. the $<_{lex}$ order: either before the first *adom* word (case $C_i = \min_{adom}(r_i)$), or between two consecutive *adom* words (case $C_i = \theta(r_i, r_i')$), or after the last *adom* word, should there be one (case $C_i = \max_{adom}(r_i)$). Let $Y_+ \subseteq \bar{y}$ be the set of all variables z appearing in a positive literal $R_i(\bar{z}_i)$ with $z \in \bar{z}_i$. Here $\psi_{\bar{C}}$ is defined as

$$\exists r_1, r_1', \ldots, r_n, r_n' \in adom \left(\bigwedge_{i=1}^{n} C_i \wedge \bigvee_{Y \subseteq (\bar{y} \setminus Y_+)} \chi_Y \right)$$

where Y is intuitively the variables whose valuations are guessed to be not in *adom* (which of course cannot intersect Y_+). The formula χ_Y is defined as follows. Let η_Y be the variable substitution replacing each $y_i \notin Y$ by r_i, and for any formula ξ, let us write $\xi[\eta_Y]$ to denote the result of substituting the variables according to η_Y in ξ. Thus, χ_Y is a conjunction consisting of each $R(\bar{z}_i)[\eta_Y]$ appearing positive, each negative $\neg R(\bar{z}_i)[\eta_Y]$, where $\bar{z}_i \cap Y = \emptyset$, and the formula

$$\chi_Y' := \exists Y (A(\bar{z})[\eta_Y] \wedge \mu(Y))$$

where $\mu(Y) = \bigwedge_{y_i \in Y} \nu_i(y_i)$ with

$$\nu_i(y_i) := \begin{cases} y_i <_{lex} r_i & \text{if, } C_i = \min_{adom}(r_i) \\ r_i <_{lex} y_i <_{lex} r_i' & \text{if, } C_i = \theta(r_i, r_i') \text{ and,} \\ r_i <_{lex} y_i & \text{if, } C_i = \max_{adom}(r_i) \end{cases}$$

Observe that χ_Y' is automatic, and hence can be written as an atom $A'(\bar{z})$ for some $A' \in \textsc{Rel}$. The reason why $\neg R(\bar{z}_i)[\eta_Y]$, with $\bar{z}_i \cap Y \neq \emptyset$, can be removed is that the above formula already ensures that at least one of the arguments in $\bar{z}_i[\eta_Y]$ will not be in the active domain, which implies $\neg R(\bar{z}_i)[\eta_Y]$. Observe that χ_Y' is in $\mathrm{FO}(\mathbf{T}_{\mathrm{Aut}})$ and so is an automatic relation. Thus, we obtain a formula in $\mathrm{FO}_{act}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$. □

Corollary B.1 (implying Corollary 4.7). *Every* $\mathrm{EFO}(\mathbf{T}_{\mathrm{Aut}}, \sigma)$ *formula* $\psi(\bar{z})$ *is effectively equivalent to an* $\mathrm{EFO}_{act}(\mathbf{T}_{\mathrm{Aut}}, \sigma')$ *formula* ψ'*, where* σ' *is the extension of* σ *with*

$$\{\theta(\cdot,\cdot), \max_{adom}(\cdot), \min_{adom}(\cdot)\},$$

interpreted as in the proof of Lemma 4.3. Further, $\psi'(\bar{z})$ *is a disjunction of formulas of the form* $A'(\bar{z}) \wedge \exists \bar{r} \in adom\ \tau(\bar{z}\bar{r})$*, where* A' *is a* $\mathbf{T}_{\mathrm{Aut}}$*-atom,* τ *is a conjunction of* σ'*-atoms and negated* σ*-atoms.*

Let $N = \max_{w \in adom} |w|$ *and let* $v : \bar{z} \to \Gamma^*$ *be a satisfying assignment for* $\psi'(\bar{z})$*, and* $z \in \bar{z}$ *such that* $|v(z)| > N$*. Suppose there is a word* $u \in \Gamma^*$ *such that* $|u| > N$ *and* $\Gamma^*, v' \models A'(\bar{z})$ *for* $v' = v[z \mapsto u]$*. Then,* v' *is also a satisfying assignment for* ψ'*.*

B.2 Proof of Lemma 4.6

We reduce from 3-SAT. Consider a $(\sigma, \mathbf{T}_{\text{Aut}})$-register automaton with no registers and having just three states q_0, q, q_F, where q_0 is initial and q_F is final, one parameter p, and three transitions:

- $(q_0, (\#, \top), q)$,
- $(q, (a, \top), q_F)$, and
- $(q, (a, \psi(curr, p)), q)$,

with

$$\begin{aligned}\psi(curr, p) = \exists p', x', x\ & value(curr, x) \wedge x' \prec x \wedge p' \prec p \wedge \\ & eq\text{-}len(p', x') \wedge lst_1(x') \wedge \\ & ((lst_1(p') \wedge lst_1(x)) \vee (lst_0(p') \wedge lst_0(x))).\end{aligned}$$

Note that ψ says that there is a position i so that the current value x is such that: (1) the i-th letter of the parameter and the last letter of x coincide, and (2) x has a 1 at position i. The idea will be that p encodes a satisfying assignment (where the i-th variable is true iff the i-th position of p is 1), and x is of the form $0^{i-1}10 \cdots 0b$ if it encodes a literal of the i-th variable appearing either positive if $b = 1$ or negative if $b = 0$.

Given a 3-SAT formula

$$\varphi := \bigwedge_{i=1}^{m} (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$$

Consider the $\mathbf{T}_{\text{Aut}}$-embedded graph over a singleton relation signature $\{a\}$, an attribute-assigning relation $value(\cdot, \cdot)$, a vertex set $V = \{v_{i,j} : i \in [1, m], j \in \{1, 2, 3\}\} \dot\cup \{v_0, v_F\}$ and edges

$$\begin{aligned}E_a = \{&(v_0, v_{1,j}), (v_{m,j}, v_F) : j \in \{1, 2, 3\}\} \cup \\ &\{(v_{i,j}, v_{i+1,j'}) : i \in [1, n-1], j, j' \in \{1, 2, 3\}\}.\end{aligned}$$

The interpretation of *value* contains $(v_0, \varepsilon), (v_F, \varepsilon)$; and for any other $v_{i,j}$ such that $\ell_{i,j} = x_k$ (resp. $\ell_{i,j} = \neg x_k$) it contains $(v_{i,j}, u)$, where u is the word over $\{0, 1\}$ of length $m + 1$ such that:

- its k-th position has a 1,
- its last position has a 1 (resp. 0), and
- every other position has a 0.

If φ is satisfiable by setting variables $x_{i_1}, \ldots, x_{i_r}$ to true and the remaining ones to false, then there is a path from v_0 to v_F in the language with the parameter assignment $\{0, 1\}^{m+1}$ which has a 1-symbol on the i_j-th position, for every $1 \le j \le r$, and a 0-symbol otherwise. Conversely, for any path from v_0 to v_F in the language, the m-prefix of the parameter assignment witnessing membership yields a satisfying assignment for φ. □

C APPENDIX TO SECTION 5

Lemma C.1. *Consider a system of word equations of the form*

$$x = v_i(\bar{y}) \text{ for } i = 1, \ldots, h \ ,$$

where x is a fixed variable other than variables in $\bar{y}$ and each variable from $\bar{y}$ appears at most once in $v_1, \ldots, v_h$ and each v_i contains at least one variable (from $\bar{y}$).

Let w_i, s_i be the longest prefix and suffix of each v_i consisting of letters, i.e. no variables are allowed. Then the system has a solution if and only if there are w, s such that $w_i \preceq w$ and $s \succeq s_i$ for each i.

A standard proof follows by giving explicit values for the variables.

Concerning the $\mathbb{L}^+_{\text{WE}}$, as in the main part of the paper, to streamline the presentation, we present the proofs for the fragment with no regular relations on the words. The proofs generalize to the case with regular relations using a standard approach of *alphabet closure* [20]: we consider a finite monoid recognizing all used regular languages (so, roughly speaking, the Boolean transition matrices of product of automata recognizing those languages). Then for each element τ in this monoid that corresponds to some word over the input alphabet, we introduce a fresh letter a_τ. Solving word equations (with regular relations) over such extended alphabet is equivalent to solving them over the input alphabet. However, the solutions over the extended alphabet have "nicer properties" and in essence behave like solutions when no regular relations are allowed.

Size of length-minimal solutions. A solution $\bar{s}$ of a system E of word equations is *length-minimal*, when for each other solution $\bar{s}'$ it holds that

$$\sum_{e \in E} |e(\bar{s})| \le \sum_{e \in E} |e(\bar{s}')| \ .$$

It is known that the length of the length-minimal solution is doubly exponential [30, 39]. In our setting we need more refined bounds.

Lemma C.2. *Consider a system of equations E. Let $n = \sum_{e \in E} |e|$, $n_v = \max_{e \in E} |e|_{\bar{x}}$ and k be the number of variables.*

Let $\bar{s}$ be a length-minimal solution (over some alphabet). Then

$$\sum_{e \in E} |e(\bar{s})| \le p(n) \cdot g(k, n_v) \ ,$$

where p is a polynomial (with degree depending on n_v, k) and g is triply exponential function not depending on n.

The proof utilises the same approach as the standard one [30] but focuses on the bound in terms of number of occurrences of the variables in the equations. When compared to the standard proof, we try to eliminate strongly similar equations: recall that equations $u(\bar{x}) = v(\bar{x})$ and $u'(\bar{x}) = v'(\bar{x})$ are *similar*, when the sequence of variables (obtained by removing all letters) in $u(\bar{x})$ and $u'(\bar{x})$ is the same and the sequence of variables in $v(\bar{x})$ and $v'(\bar{x})$ is the same. They are *strongly similar*, when additionally the lengths of words between the corresponding variables in u, u' and v, v' are the same. Then it can be shown that if two strongly similar equations have a common solution, then either they are the same equation or can be reduced to a system of equivalent equations whose size is at most the size of one such equation. By employing this observation in the known proof [30], we limit the size of the system and obtain improved bounds; note that the bound is general, i.e. applies to any system, but the bound is triply exponential in number of occurrences of variables, this is a cost of lowering the dependence on sum of equations' lengths.

Length functions and their properties. We employ the notion of length functions and corresponding solutions introduced by Plandowski and Rytter [40]. A *length function* f assigns to each variable a length; a solution $\bar{s}$ is an f-solution, when $f(x_i) = |s_i|$ for each variable x_i. Similarly, given a solution $\bar{s}$ we call such f its length function. Given a length function, we can compute the lengths of each side of equation under (any) f-solution and therefore infer equalities of individual letters in the substitutions. For a system of equations E and length function f we define a relation $\mathcal{R}_{f,E}$ on positions of letters from the equation and positions of substitutions for a variable: Two positions are in a relation $\mathcal{R}_{f,E}$ when they represent the same position inside a variable (for different occurrences of said variable) or are the corresponding positions on the two sides of the equation, see [40] for a formal definition; note that this formalizes the intuitive way one would try to solve an equation, when the lengths of the variables are known. Let $\mathcal{R}^*_{f,E}$ be the transitive closure of the relation; clearly this is an equivalence relation. $\mathcal{R}^*_{f,E}$ can be used to test, whether there is an f-solution and if so to determine the letters of the substitution for variables.

Lemma C.3 ([40, Lemma 4]). *Given a system of word equations E and a length function f there is an f-solution of E if and only if there are no different letters in an equivalence class of $\mathcal{R}^*_{f,E}$.*

When $\bar{s}$ is length-minimal and f is its length function, there is a position of a letter from the equation in each equivalence class of $\mathcal{R}^*_{f,E}$ and all letters at positions from one equivalence class are equal.

Lemma C.3 gives an oracle with NL access to the letters of a length-minimal solution: after guessing f (for a length-minimal solution), we can verify, whether there is an f-solution: Treating positions as nodes and $\mathcal{R}_{f,E}$ as an edge relation, this is a co-reachability, which is in NL. Then using the second point we can use it for access to letters of a variable: we look for a position of a letter from the equation that is in $\mathcal{R}^*_{f,E}$ with our position of the variable. This is a reachability problem in the same setting as before, which is in NL.

However, such a representation does not fit in NL, as it requires all equations and there are polynomially many of them (and they are not written down, just obtained when traversing the $\mathbf{G}$-path π).

Choosing small subsystem. Our final step is to show that we can choose a small subsystem E' (in fact, $|E'|$ depends only on $|\mathcal{A}|$) such that $\mathcal{R}^*_{f,E'}$ also has a position of a letter in each equivalence class. Note, that we will be using the relation induced by the subsystem, i.e. E' is enough to deduce each letter of the solution, yet we will be using lengths guessed for the larger system. In particular, there is no guarantee that f-solution for E' is in fact a length-minimal solution. Formally, we show that

Lemma C.4. *Suppose that $E = \bigcup_i E_i$ is a system of word equations such that for each i the subsystem E_i consists of similar equations. Let f be the length function of a length-minimal solution for E. Let m be the maximal number of occurrences of variables in equations in E.*

Then there are subsystems $E'_i \subseteq E_i$, where $|E'_i| = O(m)$ such that the relation $\mathcal{R}^*_{f,E'}$ on positions of equations in $E' = \bigcup_i E'_i$, has a letter from equation in each equivalence class.

The main observation needed to show Lemma C.4 is that given two similar equations either they have the same set of f-solutions or we can infer a partial solution, in the sense that we can compute for some variables $\{x_i\}_{i \in I}$ prefixes $\{s_i\}_{i \in I}$ such that each f-solution assigns to x_i a word with a prefix s_i, for each i, and afterwards those two equations have exactly the same f-solutions. Moreover, such a substitution "simplifies" the equation: only $O(m)$ such partial substitutions can be made in total; intuitively, either they remove an occurrence of a variable or remove "overlap" between variables at different sides of the equation. The actual statement is rather technical and is given in the full version of the paper; in particular, we need to analyze how the substitutions affect the $\mathcal{R}$ relation of the subsystem. By considering the equations one by one we can construct the desired E'_i: this is the set of equations whose addition caused a partial substitution, so $|E'_i| = O(m)$. It can be shown that $\mathcal{R}^*_{f,E'_i}$ is the same as $\mathcal{R}^*_{f,E_i}$ and so also $\mathcal{R}^*_{f,E'}$ is the same as $\mathcal{R}^*_{f,E}$, which yields the claim of the Lemma.

In our case, all equations that are instances of a fixed equation from a guard (i.e. are obtained by substitution of *curr*, $\mathcal{R}, \mathcal{R}'$ and active domain quantified variables) are similar and in total there are at most $|\mathcal{A}|$ such classes. Hence we can choose $O(m|\mathcal{A}|) \leq O(|\mathcal{A}|^2)$ many equations from the whole system and the relation $\mathcal{R}$ they define (together with f) determines each letter of the substitution for variables. Note that those equations can be guessed beforehand, i.e. before traversing the path we guess some equations from $\mathcal{A}$ and the *curr*, $\mathcal{R}, \mathcal{R}'$ and active domain quantified variables in them and also guess the length function f. Then we check, whether the f and the guessed equation are consistent and determine each letter of a substitution. If so, we have an NL oracle. If not, we reject.

D APPENDIX FOR SECTION 6

Reduction from CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$) to RDPQ($\mathbb{L}, \mathbf{T}, \sigma$)

Assume a CRDPQ($\mathbb{L}, \mathbf{T}, \sigma$)-query:

$$Q(\bar{z}) \leftarrow \bigwedge_{i=1}^{n} x_i \rightarrow_{\mathcal{A}_i} y_i \wedge \varphi(\mathcal{P}, \bar{x}, \bar{y}).$$

Let $\bar{x} \cup \bar{y}$ has a total of N variables. Assume w.l.o.g. that each $\mathcal{A}_i$ has precisely l active-domain registers $\mathcal{R} := \{r_1, \ldots, r_l\}$. Let $\mathbf{G}$ be the graph consisting of view definitions $(V, \{E_a\}_{a \in \Sigma})$ and the underlying $\mathbf{T}$-structure $\mathbf{S}$. Finally, let μ be the given instantiation of $\bar{z}$. We construct a new $\mathcal{A}$ with $l + N$ active-domain registers $\mathcal{R}_1 := \mathcal{R} \cup \bar{x} \cup \bar{y}$. Let Q^i be the set of states of $\mathcal{A}_i$. The set Q of states of $\mathcal{A}$ is defined to be the union of $\{q_0\}$ and a disjoint union of all Q^i, say, consisting of states of the form (q, i) for each $q \in Q^i$. For each $z \in \bar{z}$, we add the new unary relation $S_z = \{\mu(z)\}$ to $\mathbf{S}$. Let $\Sigma' := \Sigma \bigcup [1, n]$ (assume $[1, n] \cap \Sigma = \emptyset$). Define $E_i(z, z') := \theta_i(z) \wedge \theta_i(z')$. Here, $\theta_i(z) := V(z)$ if $i = 1$ or $i > 1$ and $y_{i-1} \notin \bar{z}$; otherwise, $\theta_i(z) := \exists y \in adom(z = y \wedge S_{y_{i-1}}(y))$. Similarly, $\theta'_i(z) := V(z)$ if $x_i \notin \bar{z}$; otherwise, $\theta'_i(z) := \exists x \in adom(z = x \wedge S_{x_i}(x))$. The output RDPQ($\mathbb{L}, \mathbf{T}, \sigma$) query evaluation problem is $Q' := x \rightarrow_{\mathcal{A}} y$ with two arbitrary start and final nodes s, t. The automaton $\mathcal{A}$ starts with q_0, from which there is a transition $(q_0, (1, \varphi(\mathcal{P}, \bar{x}, \bar{y}), q_0^1)$, for each start state q_0^1 of $\mathcal{A}_1$. $\mathcal{A}$ will then simulate $\mathcal{A}_1, \ldots, \mathcal{A}_n$ in this order, while transitioning from $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$ using E_{i+1}. To keep the instantiations of $\bar{x}$ and $\bar{y}$ consistent, we do a check *curr* $= x_i$ in any outgoing transition from (q_0^i, i), for each start state q_0^i of $\mathcal{A}_i$, and *curr* $= y_i$ in any incoming transition to (q_F, i), for each final state q_F of $\mathcal{A}_i$. [WLOG, we assume that each start state has no incoming transitions, and each final state has no outgoing transitions.]

It is easy to see that the above reduction can be performed in logspace, and the output query Q' depends only on Q.