Navigational Queries Based on Frontier-Guarded Datalog: Preliminary Results*

Meghyn Bienvenu[1], Magdalena Ortiz[2], and Mantas Šimkus[2]

[1] LRI - CNRS & Université Paris Sud
[2] Institute of Information Systems, Vienna University of Technology

Abstract. In this paper, we introduce a navigational query language that extends binary frontier-guarded Datalog by allowing regular expressions in rule bodies and a limited use of higher-arity intensional predicates. Our query language strictly extends conjunctive two-way regular path queries (C2RPQs) and captures some of the key features advocated in recent works aimed at extending C2RPQs. We compare our language to existing proposals and establish decidability with elementary complexity of query evaluation in the presence of ontologies.

1 Introduction

The current importance of graph databases has led to renewed interest in navigational query languages that ask for nodes connected by paths that satisfy given patterns. Conjunctive two-way regular path queries (C2RPQs) and their unions (UC2RPQs), which simultaneously extend both the well-known conjunctive queries (CQs) and two-way regular path queries (RPQs), have established themselves as fundamental query languages for accessing graph databases. However, recent works have argued that such queries are lacking important features, which has motivated several extensions. For example, C2RPQs have been extended with *path variables* to support naming, comparing, and outputting paths [2]. Other extensions are motivated by the fact that C2RPQs and similar languages can only describe patterns over graphs labeled with a finite alphabet, and aim to overcome this by allowing values from possibly infinite datasets [11]. A third direction aims at increasing the *navigational power* of C2RPQs and related languages, that is, allowing queries to express more complex patterns that cannot be expressed using regular languages. This paper pursues the latter direction.

A very natural way to increase the navigational power of C2RPQs is to use *nested regular expressions (NREs)*, that can enforce that at some points along a path there exists an outgoing path, which is itself described by a (nested) regular expression. *(Conjunctive) nested 2RPQs ((C)N2RPQs)*, which extend (C)2RPQs by allowing NREs in the place of plain regular expressions, have received significant attention recently [4,3,14], due to their improved navigational capabilities. For example, consider a graph database of academic relationships between researchers that contains advisor and co-author relationships, as well as the fields of expertise of the researchers. With (conjunctive) regular path queries, one can find pairs of people that are connected by an arbitrary long chain of advisor or co-author relations, or find three researchers from different fields that are

* This work has been supported by ANR project PAGODA (ANR-12-JS02-007-01) and the Austrian Science Fund (FWF) projects T515 and P25207-N23.

pairwise connected by such a chain. However, we cannot find pairs connected by an arbitrary long chain of advisor or co-author relations, where additionally each node must, for example, have an academic ancestor that is a biologist. The latter can be easily done using nested regular expressions. However, neither C2RPQs nor CN2RPQs can express the query that requires that all people along such a chain have *the same* biologist as academic ancestor. Another query that cannot be expressed is to find pairs connected by an arbitrary chain where all nodes are connected by both the co-author and the advisor relationship. These examples illustrate two shortcomings of UCN2RPQs that have been pointed out in the literature, namely, the inability to express transitive closure over complex relations defined using UCN2RPQs and the impossibility of joining objects inside a nested expression.

Recent works have aimed at overcoming these limitations. A query language that tackles the first issue was introduced by Bourhis et al. [5] with the name of *nested positive 2RPQs*, and further studied more recently (using a different syntax) by Reutter et al. [13] under the name of *regular queries*. Essentially, this language allows full UCN2RPQs to be nested inside the regular expressions of UCN2RPQs, allowing one to define relations such as the pair of all researchers that are connected by a chain of parallel advisor and coauthor relation. This language strictly increases the expressive power of UC2NRPQs at little or no computational cost: its query evaluation problem over plain graph databases remains complete for NL in data complexity and for NP complete in combined complexity, while its query containment problem is complete for 2EXPSPACE, and even in EXPSPACE for a less succinct version of equal expressive power [13]. A way to overcome the second limitation can be found in *monadically defined queries (MODEQs)* [15], which extend monadic Datalog by using some special 'flag' constants, that are then instantiated with a given tuple of ordinary constants that behave as 'parameters'. These queries can easily express relations like pairs connected by a chain of coauthors where all of them have the same biologist academic ancestor, using one flag constant as a 'placeholder' for this common object. However, they cannot express even the simple nested patterns in CN2RPQs. This limitation is overcome by *nested MODEQs* [15]. Some variations and generalizations of MODEQs and nested MODEQs, under the generic name of *(nested) flag-and-check queries* were studied recently in [6]. Many of these languages strictly generalize regular queries and nested MODEQs, and can express all of the example queries we have mentioned. However, query evaluation becomes P-complete for data complexity, and PSPACE-complete for combined complexity, and query containment becomes non-elementary.

In this paper, we introduce a navigational query language that extends binary frontier-guarded Datalog by allowing regular expressions in rule bodies. It also allows the use of intensional predicates of higher arity, but the additional positions are designated as 'parameter positions' and subjected to special syntactic restrictions. These parameter positions can simulate the use of flag constants in flag-and-check queries and allow our query language to refer to common points inside nested subqueries as in the examples discussed above. They cannot fully simulate the power of the flag-and-check special constants in the nested versions of the query languages though, but this restriction seems to play a crucial role in avoiding the non-elementary increase in the complexity of reasoning. Our query language also allows for nesting subqueries inside regular ex-

pressions, and can in fact express all the examples discuss so far. However, in terms of nesting, our language is orthogonal to regular queries and to the nested flag-and-check queries, since they only allow for acyclic nesting, by considering only non-recursive Datalog or by allowing to nest only queries of a strictly lower nesting level. By contrast, our query language naturally supports recursive nesting: that is, an intensional predicate can occur in the C2RPQ defining it. In exchange for this recursiveness, our queries impose a 'guardedness' requirement that only allows to define complex nested relations for pairs of objects that are guaranteed to be related by an extensional relation in the input database. The resulting language seems to provide an interesting trade-off between expressiveness and complexity. As mentioned earlier, it can express all of the queries mentioned above, and the complexity of query evaluation is similar to that of nested MODEQs: P-complete for data complexity, NP-complete for combined complexity if the number of parameters is bounded, and PSPACE-hard and in EXPTIME for the full language (without arity restrictions). We do not study query containment in this preliminary note, but consider instead another challenging problem: query evaluation in the presence of ontological knowledge.

In the paradigm of *ontology-mediated query answering*, (graph) databases are enriched with an *ontology* that captures domain knowledge and defines additional relations for querying. These ontologies are usually expressed in *description logics* (DLs) [1] or in closely related formalisms based on existential rules. In general, these languages support recursion and can imply the existence of additional objects (unnamed constants or nulls). In the presence of such ontologies, the query answering problem consists in computing the *certain answers* to the queries over the set of all the potentially infinitely databases that extend the input data and satisfy all the formulas in the ontology. Hence, query answering is algorithmically much harder than the usual query evaluation over databases. In fact, for most query languages studied so far in this setting, query evaluation in the presence of ontologies formulated using *expressive DLs* is at least as hard as query containment. Query evaluation in the presence of rich ontologies has been studied for plain CN2RPQs [4], but not for any of the extensions mentioned above. Bourhis et al. provided tight non-elementary bounds for a closely related language, positive first order logic with a parametrized unary transitive closure operator (PFO+TC_1), which falls strictly between regular queries and nested MODEQs [5], thus implying non-elementary hardness of query answering for other variations of nested flag-and-check queries that generalize nested MODEQs. The language we consider here, in contrast, allows for elementary query answering even for very expressive DLs, strongly suggesting that its containment problem may also remain elementary.

2 Guarded Regular Queries

In frontier-guarded rules, all variables in a rule head must occur together in a body atom. We extend such rules by allowing regular expressions in the rule bodies. We allow the use of *intensional predicates* of arbitrary arity, but we distinguish two different kinds of positions: each predicate has one or two *main positions* and any number of *parameter positions*. Inside the regular expressions, and with respect to frontier-guardedness, we consider only the main positions and the intensional predicates behave as unary and

binary. The additional parameter positions are not subjected to guardedness, but an additional syntactic requirement that ensures they are not 'forgotten'.

Syntax Let $\mathsf{N_P}$, $\mathsf{N_V}$, and $\mathsf{N_I}$ be countably infinite sets of *predicate names*, *variable names*, and *individual names*, respectively. The set $\mathsf{N_V}$ is the disjoint union of the set $\mathsf{N_V^o}$ of *ordinary* variables and the set $\mathsf{N_V^p}$ of *parameter* variables. We assume that there is a subset $\mathsf{N_{Ans}} \subseteq \mathsf{N_P}$ that consists of a single distinguished k-ary answer predicate ans^k for every $k \geq 0$; when convenient, we will omit the arity and use ans in place of ans^k.

Each predicate $p \in \mathsf{N_P} \setminus \mathsf{N_{Ans}}$ has a *main arity* of either 1 or 2, and a *parameter arity* that can be any natural number; we will write $\mathsf{arity}(p) = (k, n)$ to indicate that p has main arity k and parameter arity n. Note that usual unary and binary predicates correspond to predicates having parameter arity 0.

We define unary and binary alphabets, Σ_1 and Σ_2, as follows:

$$\Sigma_1 = \{p_{\boldsymbol{z}} \mid p \in \mathsf{N_P} \setminus \mathsf{N_{Ans}}, \mathsf{arity}(p) = (1, n), \boldsymbol{z} \in (\mathsf{N_V^p} \cup \mathsf{N_I})^n\} \cup \{\{a\} \mid a \in \mathsf{N_I}\}$$

$$\Sigma_2 = \{p_{\boldsymbol{z}},\ p^-_{\boldsymbol{z}} \mid p \in \mathsf{N_P} \setminus \mathsf{N_{Ans}}, \mathsf{arity}(p) = (2, n), \boldsymbol{z} \in (\mathsf{N_V^p} \cup \mathsf{N_I})^n\} \cup \{U? \mid U \in \Sigma_1\}$$

We consider regular languages over Σ_2, which are defined in the usual way and represented as regular expressions or non-deterministic finite automata (NFAs).

There are three kinds of *atoms*: *unary atoms* $E_1(x)$ with $E_1 \in \Sigma_1$ and $x \in \mathsf{N_V} \cup \mathsf{N_I}$, *binary atoms* $E_2(x, y)$ with E_2 a regular language over Σ_2 and $x, y \in \mathsf{N_V} \cup \mathsf{N_I}$, and *answer atoms* $\mathsf{ans}^k(\boldsymbol{z})$ with $\boldsymbol{z} \in (\mathsf{N_V} \cup \mathsf{N_I})^k$. We use the term *base atom* to mean unary atoms, answer atoms, and binary atoms $E_2(x, y)$ with $E_2 \in \Sigma_2$. For a unary or binary atom α, its set $\mathsf{mvars}(\alpha)$ of *main variables* is defined as follows: $\mathsf{mvars}(E_1(x)) = \{x\} \cap \mathsf{N_V}$ and $\mathsf{mvars}(E_2(x, y)) = \{x, y\} \cap \mathsf{N_V}$. We also define the set $\mathsf{pvars}(\alpha)$ of *parameter variables* of α: $\mathsf{pvars}(\{a\}(x)) = \emptyset$, $\mathsf{pvars}(p_{\boldsymbol{z}}(x)) = \boldsymbol{z} \cap \mathsf{N_V}$, and $\mathsf{pvars}(E_2(x, y)) = \{z \in \mathsf{N_V} \mid p_{\boldsymbol{z}}$ appears in E_2 and $z \in \boldsymbol{z}\}$.

A *rule* ρ is an expression of the form $h \leftarrow b_1, \ldots, b_n$, where every b_i is an atom and h is a base atom. We call h the *head* of ρ and $b_1, \ldots, b_n$ the *body*. As usual, we require that every head variable also appears in the body.

A *query* is a set Q of rules such that exactly one predicate ans^k from $\mathsf{N_{Ans}}$ occurs, and this predicate only appears in rule heads. If Q contains ans^k, then the *arity* of Q is k. We will distinguish two types of predicates relative to Q: *intensional predicates* that occur in some rule head in Q, and *extensional predicates* that do not appear in any rule head. We denote by $\mathsf{int}(Q)$ (resp. $\mathsf{ext}(Q)$) the set of intensional (resp. extensional) predicates relative to Q. Observe that $\mathsf{ext}(Q) = \mathsf{N_P} \setminus \mathsf{int}(Q)$.

In this paper, we will mainly focus on *guarded regular queries*, in which the following conditions hold for every rule $h \leftarrow b_1, \ldots, b_n$ with head predicate p from $\mathsf{N_P} \setminus \mathsf{N_{Ans}}$:

1. $\mathsf{mvars}(h) \subseteq \mathsf{mvars}(b_i)$ for some basic atom b_i, $1 \leq i \leq n$,
2. $\mathsf{pvars}(b_i) \subseteq \mathsf{pvars}(h)$ for all $1 \leq i \leq n$.

Note that Condition 1 imposes frontier-guardedness w.r.t. the main variables, and Condition 2 ensures that parameter variables occurring in the body of a rule occur also its head, hence they are not 'forgotten', in a similar spirit to the stickiness property considered for existential rules [7].

Semantics The queries we have defined are a special class of Datalog programs. Indeed, an atom $p_{\boldsymbol{z}}(x, y)$ provides an alternative syntax for the standard Datalog atom

$p(x, y, \boldsymbol{z})$, and regular expressions can be naturally expressed in Datalog. Hence, the semantics of our queries is readily obtained from the semantics of Datalog, where rules are viewed as Horn clauses in first-order logic. However, for our purposes, it will prove more convenient to have a direct semantics for our queries, which we introduce next.

A *(predicate) signature* is any set Σ of predicate names. A Σ*-interpretation* $\mathcal{I}$ is a tuple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of *domain elements* and $\cdot^{\mathcal{I}}$ is a function that assigns (i) to each individual c an element $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and (ii) to each $p \in \Sigma$ a relation $p^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^{k+n}$ where $\mathsf{arity}(p) = (k, n)$. If Σ is irrelevant, we simply call $\mathcal{I}$ an *interpretation*. For $\Sigma \subseteq \Sigma'$, we say that a Σ'-interpretation $\mathcal{I}'$ *extends* a Σ-interpretation $\mathcal{I}$ if the following conditions hold: (i) $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, (ii) $c^{\mathcal{I}} = c^{\mathcal{I}'}$ for every individual name c, and (iii) $p^{\mathcal{I}} = p^{\mathcal{I}'}$ for all $p \in \Sigma$.

Consider a Σ-interpretation $\mathcal{I}$, a predicate $p \in \Sigma$, and a function $\mu : \mathsf{N}_{\mathsf{V}}^{p} \to \Delta^{\mathcal{I}}$. The interpretation of symbols from $\Sigma_1 \cup \Sigma_2$ in $\mathcal{I}$ under μ is defined as follows:

$$
\begin{aligned}
\{a\}^{\mathcal{I},\mu} &= a^{\mathcal{I}} && \text{for } \{a\} \in \Sigma_1 \\
(p_{\boldsymbol{z}})^{\mathcal{I},\mu} &= \{e \in \Delta^{\mathcal{I}} \mid (e, \mu(\boldsymbol{z}^{\mathcal{I}})) \in p^{\mathcal{I}})\} && \text{for } p_{\boldsymbol{z}} \in \Sigma_1 \\
(p_{\boldsymbol{z}})^{\mathcal{I},\mu} &= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e, e', \mu(\boldsymbol{z}^{\mathcal{I}})) \in p^{\mathcal{I}})\} && \text{for } p_{\boldsymbol{z}} \in \Sigma_2 \\
(p^{-}_{\boldsymbol{z}})^{\mathcal{I},\mu} &= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e', e) \in (p_{\boldsymbol{z}})^{\mathcal{I},\mu}\} && \text{for } p^{-}_{\boldsymbol{z}} \in \Sigma_2 \\
(U?)^{\mathcal{I},\mu} &= \{(e, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid e \in U^{\mathcal{I},\mu}\} && \text{for } U? \in \Sigma_2
\end{aligned}
$$

where $\boldsymbol{z}^{\mathcal{I}}$ denotes the result of replacing every individual c in $\boldsymbol{z}$ by $c^{\mathcal{I}}$ and $\mu(\boldsymbol{z})$ denotes the result of replacing every $v \in \mathsf{N}_{\mathsf{V}}^{p}$ in $\boldsymbol{z}^{\mathcal{I}}$ by $\mu(v)$. For a general regular expression E built over Σ_2, the binary relation $E^{\mathcal{I},\mu}$ is obtained using the composition, union and transitive closure of the base relations above.

Next suppose that in addition to $\mathcal{I}$ and μ, we have a function $\pi : \mathsf{N}_{\mathsf{V}}^{o} \to \Delta^{\mathcal{I}}$. We define the following satisfaction and entailment relations:

$$
\begin{aligned}
\mathcal{I}, \mu, \pi &\models \alpha(\boldsymbol{t}) && \text{iff} && \pi(\mu(\boldsymbol{t}^{\mathcal{I}})) \in \alpha^{\mathcal{I},\mu} \\
\mathcal{I}, \mu, \pi &\models h \leftarrow b_1, \ldots, b_n && \text{iff} && \mathcal{I}, \mu, \pi \models h \text{ or } \mathcal{I}, \mu, \pi \not\models b_i \text{ for some } i \\
\mathcal{I}, \mu &\models \rho && \text{iff} && \mathcal{I}, \mu, \pi \models \rho \text{ for all } \pi : \mathsf{N}_{\mathsf{V}}^{o} \to \Delta^{\mathcal{I}} \\
\mathcal{I} &\models \rho && \text{iff} && \mathcal{I}, \mu \models \rho \text{ for all } \mu : \mathsf{N}_{\mathsf{V}}^{p} \to \Delta^{\mathcal{I}} \\
\mathcal{I} &\models Q && \text{iff} && \mathcal{I} \models \rho \text{ for all rules } \rho \in Q
\end{aligned}
$$

Now consider a k-ary query Q, and let Σ_Q be the set of predicate names that occur in Q. We call a Σ-interpretation $\mathcal{I}$ *relevant for* Q if $\Sigma \cap \mathsf{int}(Q) = \emptyset$ and $\mathsf{ext}(Q) \cap \Sigma_Q \subseteq \Sigma$. If $\mathcal{I}$ is a Σ-interpretation that is relevant for Q, then we denote by $Q(\mathcal{I})$ the set of all k-tuples $\boldsymbol{c}$ such that $\mathcal{I}' \models \mathsf{ans}(\boldsymbol{c})$ for every $\Sigma \cup \mathsf{int}(Q)$-interpretation $\mathcal{I}'$ with $\mathcal{I}' \models Q$ which extends $\mathcal{I}$.

Examples We now illustrate the expressiveness of our query language with some examples. Consider a graph database with the following relationships between researchers: *was PhD advisor of* (ad), *is a coauthor of* (ca), *is a collaborator of* (cl), and *shares research topic* (srt). The following query Q_1 finds pairs connected by an arbitrarily long chain of parallel ad and ca relations.

$$
\mathsf{ans}(x, y) \leftarrow \ ac^{*}(x, y) \qquad\qquad ac\,(x, y) \leftarrow \ ad \cup ad^{-}(x, y), ca(x, y)
$$

This simple query cannot be expressed as a CN2RPQ; this is proven for an analogous query in [5]. The next query Q_2 finds all pairs that are connected by a chain of potential collaborators, where *potential collaborators* are people that either are collaborators, or they share a research topic and are connected by a chain of coauthors.

$$\mathsf{ans}(x,y) \leftarrow pcl^*(x,y) \qquad pcl(x,y) \leftarrow cl(x,y)$$
$$pcl(x,y) \leftarrow srt(x,y), ca^*(x,y)$$

This query is very similar to the regular query given in Example 2 of [13]. It can also be shown along the lines of [5] that this query is not expressible as a plain CN2RPQ. The next query Q_3 finds pairs of individuals who are connected by a chain of srt who are all connected to some z by a coauthorship chain.

$$\mathsf{ans}(x,y) \leftarrow cca^*_z(x,y) \qquad cca_z(x,y) \leftarrow srt(x,y), ca^*(x,z), ca^*(y,z)$$

This query is not expressible as a regular query; again, this can be shown as in [5].

Finally, we give an example query Q_4 that nests over guarded recursion. It builds a *preferred collaborator relation* that contains any pair of coauthors that have coauthored papers with a shared advisor, as well as pairs of x and y that can respectively reach researchers x' and y' via a chain of both srt and cl, where x' and y' are themselves preferred collaborators:

$$prefcl(x,y) \leftarrow ca(x,y), ca(x,z), ca(y,z), ad(x,z), ad(y,z)$$
$$prefcl(x,y) \leftarrow ca(x,y), srt^*(x,x'), cl^*(x,x'), srt^*(y,y'), cl^*(y,y'), prefcl(x',y')$$

Note that Q_4 does not conform to the syntax of regular queries, and we conjecture that it is expressible neither as a regular query, nor as a nested flag-and-check query.

3 Related Query Languages

In this section, we compare in more detail guarded regular queries with related languages that also extend the navigational capabilities of C2RPQs. First, it is easy to see that any nested CN2RPQ can be easily rewritten as a guarded regular query, by simply replacing exhaustively each nested expression $\langle E \rangle$ by $p_E?$ for a fresh intensional predicate p_E with $\mathsf{arity}(p_E) = (1,0)$, and adding a rule $p_E(x) \leftarrow E(x,y)$.

Now we compare guarded regular queries with other extensions of CN2RPQs that are closer in terms of expressiveness.

Regular queries were introduced in [13] as non-recursive binary Datalog programs that additionally allow for *transitive closure* rules of the form $P(x,y) \leftarrow R^+(x,y)$. An alternative definition, that is equivalent but exponentially less succinct, is given by taking non-recursive binary Datalog rules and allowing in the body regular expressions over extensional predicates and expressions of the form $R^+(x,y)$ (for arbitrary predicates), but restricting intensional predicates to occur only once in rule bodies. The query containment problem for regular queries is complete for 2ExpSpace, but only ExpSpace-complete (like for plain C2RPQs) if the alternative syntax is considered. The query evaluation problem is also not harder than for C2RPQs: NL-complete in data complexity and NP-complete complete in combined complexity.

Regular queries and guarded regular queries are closely related, but there are some significant differences. We have already shown above that their expressive power is orthogonal. Regular queries have no parameters and nesting is not recursive. On the other hand, their nesting is not subject to a guardedness restriction.

Nested positive 2RPQs [6] allow for positive Boolean combinations of 2RPQs, where the regular expressions may use queries of strictly lower nesting degree as binary predicates. This language is in fact equivalent to the regular queries from [13].

PFO+TC$_1$ is another extension of CN2RPQs studied by Bourhis et. al. that extends positive first-order logic with a parameterized unary transitive closure operator. This language is strictly more expressive than regular queries, and it can fully support the 'parameters' (restricted higher arities) that we allow in guarded regular queries. However, guarded regular queries and PFO+TC$_1$ are incomparable. On the one hand, PFO+TC$_1$ has NL-data complexity, while guarded regular queries are complete for P, as we show in Section 4. This implies that some guarded regular queries cannot be expressed in PFO+TC$_1$. On the other hand, PFO+TC$_1$ has been shown to have non-elementary time complexity for query containment [6] and for query answering in the presence of some DL ontologies [5]. We argue below that the latter problem is elementary for guarded regular queries,[3], which implies that not all PFO+TC$_1$ queries are expressible as guarded regular queries without a non-elementary blow-up in the formula size.

Expressive Datalog fragments having a decidable containment problem have been recently studied by Bourhis et al. [6]. They consider well-known languages like linear, monadic, and frontier-guarded Datalog and extend them with special 'flag' constants that behave analogously to the parameter positions of guarded regular queries (this had already been done for monadic Datalog [15]); they also consider *nested* versions of these languages. These *nested flag-and-check queries* are in general more expressive than regular queries and even generalize PFO+TC$_1$ [5]. However, similarly as for PFO+TC$_1$, this has a high computational cost and the containment problem becomes non-elementary [6]. Also, in contrast to guarded regular queries, all these nested queries allow only for acyclic nesting, while we allow for cyclic but guarded nesting.

One of the considered languages, GQ$^+$, is obtained by taking frontier guarded flag-and-check programs (that is, frontier guarded Datalog programs with special parameter constants), and allowing to nest into their bodies analogous programs of lower nesting level. Note that, in contrast, we have a (linear) Datalog fragment (that captures C2RPQs) at the inner query level, and use frontier-guardedness to do cyclic nesting of queries.

4 Evaluating Guarded Regular Queries

In this section, we study the complexity of query evaluation over graph databases and then outline an algorithm for query answering in the presence of DL ontologies.

Query Evaluation over Graph Databases The following proposition summarizes what we know about the complexity of evaluating guarded regular queries.

Proposition 1. *Evaluation of guarded regular queries over graph databases is:*

[3] We have not yet studied containment of guarded regular queries, but we believe it may be elementary as well.

– P-*complete for data complexity;*
– PSPACE-*hard and in* EXPTIME *for combined complexity;*
– NP-*complete for combined complexity, when the parameter arity of predicates is bounded by a fixed constant.*

Proof. For Statement 1, observe that guarded regular queries fall between binary monadic Datalog and full Datalog (cf. Section 2), and for both of these languages, the query evaluation problem is P-complete in data complexity.

Membership in EXPTIME for combined complexity is a direct consequence of the EXPTIME-completeness of query evaluation in Datalog. To obtain the PSPACE lower bound, we modify the Datalog program that was used in [10] to show PSPACE-hardness of evaluating linear sirups, in order to ensure that the resulting query satisfies the 'stickiness' requirement (Condition 2).

For Statement 3, the lower bound comes from the NP-hardness of CQ evaluation, and the upper bound is obtained by guessing a sequence of polynomial number of rule applications (along with the required homomorphisms) that leads to deriving $\mathsf{ans}(\boldsymbol{c})$.

Query Answering with DL KBs We sketch an algorithm for answering guarded regular queries over DL KBs. We do not give the details of particular DLs, but instead present the *forest model property* that many DLs have and on which our method relies.

Each DL KB $\mathcal{K}$ is defined over some set Σ of unary and binary relations with parameter arity 0. The semantics of $\mathcal{K}$ is given as a set $\mathcal{M}(\mathcal{K})$ of Σ-interpretations. Given a k-ary query Q, we let $Q(\mathcal{K})$ be the set of k-tuples $\boldsymbol{c}$ of individuals such that $\boldsymbol{c} \in Q(\mathcal{I})$ for all $\mathcal{I} \in \mathcal{M}(\mathcal{K})$. The set $Q(\mathcal{K})$ is called the *certain answer* to Q over $\mathcal{K}$.

Let $\mathbf{N}$ be the set of natural numbers and denote by $w \in \mathbf{N}^+$ the set of all non-empty words over $\mathbf{N}$. A Σ- interpretation $\mathcal{I}$ is *forest-shaped* if the following conditions hold:

(i) $\Delta^{\mathcal{I}} \subseteq \mathbf{N}^+$ is *prefix-closed*, i.e., if $w \cdot n \in \Delta^{\mathcal{I}}$ and $w \neq \epsilon$, then $w \in \Delta^{\mathcal{I}}$;
(ii) if $(e, e', \boldsymbol{d}) \in p^{\mathcal{I}}$ for some $p \in \Sigma$, then one of the following holds: (a) $e = e'$, (b) $|e| = 1$ and $|e'| = 1$, or (c) $e' = e \cdot n$ or $e = e' \cdot n$ for some $n \in \mathbf{N}$.

A KB $\mathcal{K}$ is said to possess the *forest model property* if for any $\mathcal{I} \in \mathcal{M}(\mathcal{K})$ there exists a forest-shaped interpretation $\mathcal{I}' \in \mathcal{M}(\mathcal{K})$ such that:

(i) there exists a homomorphism from $\mathcal{I}'$ to $\mathcal{I}$, and
(ii) $\Delta^{\mathcal{I}} \subseteq \{0, \ldots, r(|\mathcal{K}|)\}^*$, where r is a polynomial and $|\mathcal{K}|$ denotes the size of $\mathcal{K}$.

The forest-shaped $\mathcal{I} \in \mathcal{M}(\mathcal{K})$ that satisfy the condition $\Delta^{\mathcal{I}} \subseteq \{0, \ldots, r(|\mathcal{K}|)\}^*$ are the *forest models* of $\mathcal{K}$. It is well known that KBs written in many popular DLs have this property (cf. [8]).

Consider a KB $\mathcal{K}$ with the forest model property, a query Q of arity k, and a k-tuple $\boldsymbol{c}$ of individuals. Since our query language is monotone, it follows that for such a KB $\mathcal{K}$, $\boldsymbol{c} \notin Q(\mathcal{K})$ implies $\boldsymbol{c} \notin Q(\mathcal{I})$ for some forest model $\mathcal{I} \in \mathcal{M}(\mathcal{K})$. If $\mathcal{K}$ is in a standard DL like $\mathcal{ALCHIQ}$, one can define a tree automaton that recognizes (an encoding of) forest-shaped $\mathcal{I} \in \mathcal{M}(\mathcal{K})$. However, due to the semantics of the query language, deciding $\boldsymbol{c} \notin Q(\mathcal{I})$ involves checking the existence of an extension $\mathcal{I}'$ of $\mathcal{I}$ such that $\mathcal{I}' \models Q$ and $\mathcal{I}' \not\models \mathsf{ans}(\boldsymbol{c})$, and such $\mathcal{I}'$ need not be forest-shaped. In fact, unrestricted queries are undecidable and thus can enforce interpretations that are not tree-shaped and which cannot be coded into forest-shaped interpretations in a meaningful way.

To obtain decidability for guarded regular queries, we show that it is sufficient to consider only certain forest-shaped extensions. Given a Σ-interpretation $\mathcal{I}$, we let $\mathsf{pdom}(\mathcal{I})$ denote the domain elements that appear as parameter values in $\mathcal{I}$, i.e. those $e \in \Delta^{\mathcal{I}}$ for which there exists a predicate name $p \in \Sigma$ and a tuple $\langle e_1, \ldots, e_{k+n} \rangle \in p^{\mathcal{I}}$ with $e = e_i$ and $i > k$, where $\mathsf{arity}(p) = (k, n)$. We can prove the following:

Proposition 2. *Consider a KB $\mathcal{K}$ with the forest model property, a guarded regular query Q of arity k, and a k-tuple $\mathbf{c}$ of individuals. Let m be the maximal number of parameter variables over all rules of Q. Then $\mathbf{c} \notin Q(\mathcal{K})$ iff (†) there exists a forest-shaped $\mathcal{I} \in \mathcal{M}(\mathcal{K})$ such that for any set $g \subseteq \Delta^{\mathcal{I}}$ with $|g| \leq m$ there exists a $\Sigma \cup \mathsf{int}(Q)$-interpretation $\mathcal{I}'$ such that: (i) $\mathcal{I}'$ is forest-shaped, (ii) $\mathcal{I}'$ extends $\mathcal{I}$, (iii) $\mathcal{I}', \mu \models \rho$ for every $\rho \in Q$ and every μ with $\mathsf{ran}(\mu) \subseteq g$, (iv) $\mathcal{I}' \not\models \mathsf{ans}(\boldsymbol{c})$, and (v) $\mathsf{pdom}(\mathcal{I}') \subseteq g$.*

We next argue that condition (†) from Proposition 2 can be decided by employing tree automata. It is standard to represent forest-shaped interpretations of a DL KB as labeled trees upon which a tree automaton can operate (see e.g. [9]). In such an encoding, 'roots' of the original forest-shaped interpretation correspond to children of a dummy root in the tree representation. It is a bit more involved to obtain a tree representation of forest-shaped Σ-interpretations when Σ contains predicates with non-zero parameter arities. However, if the input interpretation $\mathcal{I}$ is such that $|\mathsf{pdom}(\mathcal{I})| \leq k$ for a finite k, then we can essentially employ the standard representation, except that we may need to increase the alphabet exponentially in the parameter arity of original relations.

The automata algorithm to check (†) can be briefly described as follows:

1. If $\mathcal{K}$ is in a standard DL like $\mathcal{ALCHIQ}$, we can build a nondeterministic tree automaton (NTA) A_1 that recognizes (the tree encoding of) tuples $(\mathcal{I}, g, \mathcal{I}')$ such that $\mathcal{I}$ is a forest-shaped model of $\mathcal{K}$, $g \subseteq \Delta^{\mathcal{I}}$ is such that $|g| \leq m$, and $\mathcal{I}'$ is a $\Sigma \cup \mathsf{int}(Q)$-interpretation that satisfies conditions (i)-(v) of Proposition 2. The naïve construction of A_1 requires a triple-exponential number of states and can be done by adapting the construction in [9].
2. We build an NTA A_2 that recognizes tuples $(\mathcal{I}, g)$ such that some triple of the form $(\mathcal{I}, g, \mathcal{I}')$ is recognized by A_1. Technically, this is done by performing a projection operation on A_1, that projects away the third component of recognized triples.
3. We complement A_2 to obtain A_3. Intuitively, A_3 accepts tuples $(\mathcal{I}, g)$ that A_2 rejects, i.e. for which there is no triple $(\mathcal{I}, g, \mathcal{I}')$ that satisfies conditions (i)-(v). This step causes an exponential blowup in the number of states.
4. By projecting away the second component, we can obtain from A_3 an NTA A_4 that accepts forest interpretations $\mathcal{I}$ for which there exists g such that we cannot find an $\mathcal{I}'$ that satisfies conditions (i)-(v).
5. By complementing A_4, we obtain an NTA that checks condition (†) in Proposition 2. This step also causes an exponential blowup in the number of states.

Due to the complexity results on testing nonemptiness of NTAs in [12], the above construction leads to an 5EXPTIME upper bound. For standard Horn DLs like Horn-$\mathcal{SHIQ}$, $\mathcal{EL}$ or *DL-Lite*, we can improve the upper bound to 4EXPTIME; these DLs have the so-called *canonical model property* and thus the last automata complementation step in the above construction is not necessary.

5 Future Work

In this paper, we proposed guarded regular queries as a new navigational query language and provided some first results regarding its relation to related proposals and the complexity and decidability of query evaluation, in particular in the presence of DL ontologies. There are number of questions that we plan to tackle in future work. First, we will complete our formal comparison of the expressive power of guarded regular queries, with the aim of showing that they are indeed incomparable to related existing formalisms. Second, we will pursue our study of the computational properties of the language by pinpointing the precise complexity of query answering in the presence of DL ontologies and by investigating the query containment problem. Finally, extending guarded regular queries by relaxing the use of parameters and frontier-guardedness is another challenging but interesting problem.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. P. Barceló, L. Libkin, A. W. Lin, and P. T. Wood. Expressive languages for path queries over graph-structured data. *ACM TODS*, 37(4):31, 2012.
3. P. Barceló, J. Pérez, and J. L. Reutter. Relative expressiveness of nested regular expressions. In *Proc. of AMW'12*, CEUR Workshop Proceedings 866, pages 180–195, 2012.
4. M. Bienvenu, D. Calvanese, M. Ortiz, and M. Šimkus. Nested regular path queries in description logics. In *Proc. of KR 2014*. AAAI Press, 2014.
5. P. Bourhis, M. Krötzsch, and S. Rudolph. How to best nest regular path queries. In *Proc. of DL'14*, volume 1193, pages 404–415. CEUR-WS.org, 2014.
6. P. Bourhis, M. Krötzsch, and S. Rudolph. Query containment for highly expressive datalog fragments. *CoRR*, abs/1406.7801, 2014.
7. A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proc. VLDB Endow.*, 3(1-2):554–565, Sept. 2010.
8. D. Calvanese, T. Eiter, and M. Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. of IJCAI 2009*, pages 714–720, 2009.
9. D. Calvanese, T. Eiter, and M. Ortiz. Answering regular path queries in expressive description logics via alternating tree-automata. *Inf. Comput.*, 237:12–55, 2014.
10. G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Information and Computation*, 183(1):104 – 122, 2003.
11. L. Libkin and D. Vrgoc. Regular path queries on graphs with data. In A. Deutsch, editor, *Proc. of ICDT'12*, pages 74–85. ACM, 2012.
12. D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(12):69 – 107, 1995.
13. J. Reutter, M. Romero, and M. Y. Vardi. Regular queries on graph databases. In M. Arenas and M. Ugarte, editors, *Proc. of ICDT'15*, 2015. To appear.
14. J. L. Reutter. Containment of nested regular expressions. CoRR Technical Report arXiv:1304.2637, arXiv.org e-Print archive, 2013.
15. S. Rudolph and M. Krötzsch. Flag & check: Data access with monadically defined queries. In *Proc. of PODS'13*, pages 151–162. ACM, 2013.