Answering Conjunctive Regular Path Queries over Guarded Existential Rules

Jean-François Baget  
Inria, France  
baget@lirmm.fr

Meghyn Bienvenu  
CNRS, France  
meghyn@lirmm.fr

Marie-Laure Mugnier  
Univ. Montpellier, France  
mugnier@lirmm.fr

Michael Thomazo  
Inria, France  
michael.thomazo@inria.fr

Abstract

Ontology-mediated query answering is concerned with the problem of answering queries over knowledge bases consisting of a database instance and an ontology. While most work in the area focuses on conjunctive queries (CQs), navigational queries are gaining increasing attention. In this paper, we investigate the complexity of answering two-way conjunctive regular path queries (CRPQs) over knowledge bases whose ontology is given by a set of guarded existential rules. We first consider the subclass of linear existential rules and show that CRPQ answering is EXPTIME-complete in combined complexity and NL-complete in data complexity, matching the recently established bounds for answering non-conjunctive RPQs. For guarded rules, we provide a non-trivial reduction to the linear case, which allows us to show that the complexity of CRPQ answering is the same as for CQs, namely 2EXPTIME-complete in combined complexity and PTIME-complete in data complexity.

1 Introduction

In ontology-mediated query answering (OMQA), a database is enriched with an ontological layer that expresses domain knowledge, and queries are answered by taking into account the information from both the data and ontology. Two main families of ontology languages are considered in this setting, namely description logics (DLs) and existential rules (see e.g. the survey chapters [Ortiz and Simkus, 2012; Kontchakov et al., 2013; Bienvenu and Ortiz, 2015] on OMQA with DLs and [Cali et al., 2009a; Mugnier and Thomazo, 2014] for existential rules).

Although most work on OMQA to date has focused on the fundamental class of conjunctive queries (CQs), navigational queries are gaining increasing attention. The simplest such queries are regular path queries (RPQs, [Florescu et al., 1998]), which ask for paths defined by a given regular language, hence allowing for a controlled form of recursion over binary predicates. Many extensions of RPQs have been considered, including conjunctive regular path queries (CRPQs) which generalize both RPQs and CQs. In particular, SPARQL 1.1, the new W3C standard for querying RDF data, extends its core queries, which roughly correspond to CQs, to CRPQs.

In recent years, such queries have been investigated for a variety of description logics, ranging from highly expressive DLs of the Z family [Calvanese et al., 2007; 2009; 2014], to Horn DLs like Horn-SROIQ [Ortiz et al., 2011] and lightweight DLs of the DL-Lite and EL families [Stefanoni et al., 2014; Bienvenu et al., 2015].

When the data are not only graphs, as in RDF and DL fact bases, but more generally hypergraphs, or, in logical terms, sets of facts with any predicate arity, navigational queries are still very relevant as binary relations remain central in data and ontological modeling. While the complexity landscape for answering navigational queries in the presence of DL ontologies is now quite clear, the combination of navigational queries and existential rules has only just recently begun to be explored. In [Bienvenu and Thomazo, 2016], the authors studied the complexity of RPQ answering over linear existential rules, a natural generalization of DL-Lite_R. It was shown that the problem complexity remains the same as for DL-Lite_R in terms of data complexity (NL-complete) and combined complexity with bounded predicate arity (PTIME-complete). However, in the unbounded arity case, the combined complexity rises to EXPTIME-complete. Interestingly, in DL-Lite_R, RPQ answering is easier than CQ answering for combined complexity (P- vs. NP-complete [Bienvenu et al., 2015]), while the opposite holds for linear rules without arity restriction (CQ answering being PSPACE-complete [Cali et al., 2009a]).

In this paper, we make a step towards a better understanding of the complexity of answering CRPQs over existential rule bases, focusing on a central class of existential rules, namely guarded rules, of which linear rules are a special case [Cali et al., 2008; 2009b]. Note that the problem is already known to be decidable (from Rudolph and Krötzsch, 2013).

Summary of the contributions. We obtain tight complexity results for both combined and data complexities over guarded rules and their linear subclass. Namely, CRPQ answering in the linear case is EXPTIME-complete in combined complexity, PSPACE-complete in combined complexity with bounded arity and NL-complete in data complexity (Thm 1); hence, it is not more difficult than RPQ answering, except for combined complexity with bounded arity (in which...
case the complexity is again the same as for DL-Lite$_R$). In
the guarded case, CRPQ answering is 2ExpTime-complete
in combined complexity, ExpTime-complete in combined
complexity with bounded arity, and PTIME-complete in data
complexity (Thm 2); hence, it is not more difficult than plain
CQ answering.

To achieve these results, we first investigate the case of
linear rules and provide a CRPQ answering algorithm that
uses RPQ answering as an oracle and runs within the men-
tioned complexity classes. The matching lower bounds come
from earlier results on RPQ and CQ answering. We next pro-
vide a non-trivial reduction of the guarded case to the linear
case. This translation involves a double exponential blow-up
of the rule base (while the instance only grows exponentially
in the predicate arity). However, a careful analysis of the al-
gorithm provided for linear rules shows that it actually runs in
2ExpTime with respect to the input guarded knowledge base
(and in ExpTime in the case of bounded-arity rules).

2 Preliminaries

We consider a vocabulary composed of a finite set of pred-
icates and an infinite set of constants. A (standard) atom $\alpha$
has the form $r(t)$ where $r$ is a predicate of arity $n$ and $t$ is a
tuple of terms (i.e., variables or constants) with $|t| = n$. For
$1 \leq i \leq |t|$, we denote by $\alpha[i]$ the term at position $i$ in $\alpha$.
We denote by $terms(\alpha)$ the set of terms in $\alpha$ and extend the
notation to a set of atoms.

A ground atom contains only constants. An instance is a
finite set of facts, which are groups of atoms. A conjunctive
query (CQ) has the form $q(x) = \exists y. B$ where $x$ and $y$ are
disjoint tuples of variables, and $B$ is a conjunction of atoms
such that $terms(B) = x \cup y$. A tuple of constants $\pi = \pi(x)$,
where $\pi$ denotes a substitution, is a certain answer to $q(x)$
over $I$ if $I \models q(\alpha)$, where $q(\alpha) = \exists y. \pi(B)$ and $I$
denotes classical logical entailment. A CQ is Boolean if $x = \emptyset$.

We often identify (the existential closure of) a conjunction of
atoms with the set of these atoms. Given two sets of atoms $A$ and $B$, a homomorphism from $B$ to $A$ is a substitution $\pi$ of
the variables in $B$ by the terms in $A$ such that $\pi(B) \subseteq A$.
It holds that $A \models B$ iff there is a homomorphism from $B$ to $A$.
Given a (possibly infinite) set of atoms $A$ and a set of terms
$T$, $A|_T$ denotes the subset of $A$ containing the atoms whose
arguments are a subset of $T$.

Existential Rules An existential rule (or simply rule) is of
the form $\forall x \forall y. [B(x, y) \rightarrow \exists z. H(y, z)]$ where $B$ and $H$
are non-empty conjunctions of atoms on variables, respec-
tively called the body and the head of the rule, and $x$ and $y$
are pairwise disjoint. The variables of $z$ are called existen-
tial variables. For brevity, quantifiers are often omitted, and
we write $B \rightarrow H$. We will make the classical assumption
that the head of a rule is restricted to a single atom. A set of
atoms $G$ is guarded if it contains an atom $\alpha$ (called a guard)
such that $terms(\alpha) = terms(G)$. Since $G$ may have several
guards, we use the notation $(G, \alpha)$ to specify that $\alpha$ is the
considered guard. A rule is guarded if its body is guarded. A
rule is linear if its body contains a single atom. A knowledge
base (KB) is of the form $K = (I, R)$, where $I$ is an instance
and $R$ a set of existential rules. Let $K = (I, R)$ be a KB
and $q(x)$ be a CQ. A certain answer to $q$ over $K$ is a tuple
of constants $a$ such that $I, R \models q(a)$.

A rule $R = B \rightarrow \alpha$ is applicable to a set of atoms $A$
if there is a homomorphism $\pi$ from $B$ to $A$. The result of the
application of $R$ on $A$ according to $\pi$ is the atom $\pi^+(\alpha)$, where
$\pi$ extends $\pi$ by assigning a distinct fresh variable (called a
null) to each existential variable. The application is said to
generate the set of atoms $A \cup \pi^+(\alpha)$, which is uniquely
defined up to the choice of nulls.

The chase is the fundamental tool on existential rules.
Starting from the instance, it performs rule applications in a
breadth-first manner. Several variants of the chase are known.
We consider here the simplest variant, known as the oblivious
chase, which at each breadth-first step performs all new rule
applications, and proceeds until no rule is applicable accord-
ing to a new homomorphism. All the chase variants compute
a so-called universal model of the KB, i.e., a possibly infinite
set of facts seen as a classical logical interpretation, which is
a model of the KB that maps by homomorphism to any other
model of the KB. The result of the chase is also called chase.
It follows that for any $K = (I, R)$ and CQ $q$, $(I, R) \models q$ iff
the chase of $K$ is a model of $q$ iff there is a homomorphism
from $q$ to the chase of $K$.

(Conjunctive) Regular Path Queries A regular language
can be represented either by a regular expression or by a non-
deterministic finite automaton (NFA). Let $\Sigma$ be a finite set of
symbols. A regular expression $E$ over $\Sigma$ is defined by the
grammar: $E \rightarrow \varepsilon | a | E \cdot E | E + E | \varepsilon^*$, where $a \in \Sigma$
and $\varepsilon$ denotes the empty word. An NFA over $\Sigma$ is a tuple
$\Delta = (S, \Sigma, \delta, s_0, F)$, where $S$ is a finite set of states, $\delta \subseteq
S \times \Sigma \times S$ is the transition relation, $s_0 \in S$ is the initial state
and $F \subseteq S$ is the set of final states.

We denote by $P_2$ the binary predicates in our vocabulary
and set $P_2^+ = P_2 \cup \{r^- | r \in P_2\}$. A path atom takes
the form $A(t, t')$, where $A$ is a regular language over $P_2^+$
and $t, t'$ are terms. Note that standard atoms with binary predicates
are a special case of path atoms. We assume w.l.o.g. that aut-
tomata associated with distinct path atoms have disjoint sets
of states.

A conjunctive two-way regular path query (CRPQ)\footnote{As we only consider the two-way variant, we will use the
abbreviation (C)RPQ instead of the more traditional (C)2RPQ.}
has the form $q(x) = \exists y. B$ and is defined as a CQ except that $B$
is a conjunction of standard and path atoms.

We extend the usual definition of an interpretation $I = (\Delta, \varepsilon)$
being a model of a conjunction of atoms by saying that,
for a regular language $\Delta$, $(d, d') \in \Delta^2$ if there are
$r_1 \ldots r_n \in \Delta$ and $(d_0 = d_0', \ldots, d_n = d_n') \in \Delta^{n+1}$ such
that $\forall 1 \leq i \leq n$, if $r_i \in P_2$ then $(d_{i-1}, d_i) \in r_i^+$, otherwise
$r_i = \varepsilon_i$ and $(d_i, d_{i-1}) \in \varepsilon_i^T$. We call $p = d_0 r_1 d_1 \ldots r_n d_n$ a
path in $I$. A match for a Boolean CRPQ $q$ in an interpretation
$I = (\Delta, \varepsilon)$ is a mapping $\pi$ from the terms in $q$ to elements
of $\Delta$ such that (i) $\pi(\alpha) = \alpha^\pi$ for each constant $\alpha$; (ii) $\pi(t) \in r(t)$
for each standard atom $r(t)$ in $q$; and (iii) $(\pi(t), \pi(t')) \in \Delta^2$
for each path atom $\Delta(t, t')$ in $q$. Note that the restriction of $\pi$
to the standard atoms of $q$ is a homomorphism. Then $I \models a$
model of \( q \) if and only if there is a match for \( q \) in \( I \). Certain answers are defined in the same way as for CQs. Since the chase of \( K = (I, R) \) is a universal model, and CRPQs are closed by homomorphism, \( I, R \models \iota \) if and only if there is a match \( \pi \) of \( q \) in the chase of \( K \). A path of terms in the chase is the natural translation of a path in an interpretation.

The CRPQ Answering problem asks, given a KB \( K = (I, R) \), a CRPQ \( q(x) \) and a tuple of constants \( a \) whether \( a \) is an answer to \( q \) w.r.t. \( K \). The CRPQ Entailment problem asks, given a KB \( K = (I, R) \) and a Boolean CRPQ \( q \), if \( I, R \models q \). Both problems are classically polynomially reducible one to another. To simplify the presentation, we will study the CRPQ Entailment problem.

3 Chase Graph of Linear Rules

With the chase is naturally associated a derivation graph, in which nodes are labelled by atoms and edges represent rule applications, i.e., there is an edge \((\nu, \nu')\) if the application of a rule to the atom that labels \( \nu \) has produced the atom that labels \( \nu' \). In the case of linear rules, this graph is a forest. Several derivation graphs can be associated with the same chase, since the same atom may be produced in different ways. However, in our proofs we need to be independent from a particular graph and to have an edge \((\nu, \nu')\) if and only if a rule application to the atom that labels \( \nu \) can produce the atom that labels \( \nu' \). Hence, we will consider a particular graph that encodes all derivations associated with a given chase. This graph is still a forest.

Definition 1 Let \( I \) be an instance and \( \mathcal{R} \) be a set of linear rules. The chase graph associated with \( (I, \mathcal{R}) \), denoted by \( CG(I, \mathcal{R}) \) (and simply \( CG \) if the context is clear), is a possibly infinite node-labeled directed forest built as follows (we use \( \ell \) for the labeling function):

- the roots are in bijection with \( I \) via \( \ell \);
- for every node \( \nu \) and rule \( R \in \mathcal{R} \) applicable to \( \ell(\nu) \), there is exactly one edge from \( \nu \) to a node \( \nu' \) labeled by the atom resulting from this rule application.

Note that an atom can label two distinct nodes, effectively encoding two different ways of generating that same atom. Given \( (I, \mathcal{R}) \), the chase graph is uniquely defined, up to the choice of nulls. We formally define the chase as the set of all atoms that label \( CG(I, \mathcal{R}) \). We denote it by \( chase(I, \mathcal{R}) \).

We can now express a fundamental property of linear rules: each atom from the instance can be ‘chased’ independently.

Property 1 Let \( \mathcal{R} \) be a set of linear rules and \( I \) be an instance. Then \( chase(I, \mathcal{R}) = \cup_{\alpha \in I} chase(\{\alpha\}, \mathcal{R}) \).

For \( \nu \) and \( \nu' \) in the same tree of \( CG \), \( glb(\nu, \nu') \) denotes their greatest lower bound (i.e., their deepest common ancestor).

An essential property of the chase graph is that, for any null \( z \), the nodes labeled by the atoms that contain \( z \) form a connected component. This is also true for the initial terms inside each tree. In the next proposition, to distinguish between undirected paths in \( CG \) and the previously introduced paths (on terms in the chase), we call the former atom-paths.

Proposition 1 (Connectivity property) Let \( \nu_1 \) and \( \nu_2 \) be two nodes of \( CG(I, \mathcal{R}) \) and \( t \in terms(\ell(\nu_1)) \cap terms(\ell(\nu_2)) \). Then:

- either \( t \) is contained in \( I \) and there are atom-paths \( P_1 \) from \( \nu_1 \) to a root \( \nu'_1 \) and \( P_2 \) from \( \nu_2 \) to a root \( \nu'_2 \) such that for every node \( \nu \in P_1 \cup P_2 \), \( \ell(\nu) \) contains \( t \);
- or there is an atom-path \( P \) from \( \nu_1 \) to \( \nu_2 \), and it is such that for every node \( \nu \in P \), \( \ell(\nu) \) contains \( t \).

The next proposition specifies the relationships between a path in the chase and an atom-path in the chase graph. It strongly relies on the connectivity property.

Proposition 2 Let \( \nu_1 \) and \( \nu_2 \) be two nodes in \( CG \) under a common root. Let \( p = t_1 \ldots t_p \) be a path in the chase such that \( t_1 \) occurs in \( \ell(\nu_1) \) and \( t_p \) occurs in \( \ell(\nu_2) \). Then for any node \( \nu \) on the atom-path from \( \nu_1 \) to \( \nu_2 \) in \( CG \), \( \ell(\nu) \) contains a term \( t_i \) from \( p \).

Example 1 Let \( \mathcal{R}_c \) contain the following rules:

\[
\begin{align*}
    b(x, y, z, t) &\rightarrow h(y, z, t, u) \\
    h(x, y, z, t) &\rightarrow b(y, z, t, u) \\
    h(x, y, z, t) &\rightarrow s(y, x) \\
    b(x, y, z, t) &\rightarrow n(z, t) \\
    h(x, y, z, t) &\rightarrow r(t, x)
\end{align*}
\]

Part of \( CG(I_c) = \{ \{a_1, a_2, a_3, a_4\}, t(a_1) \}, \mathcal{R}_c \) is represented in Fig. 1. For clarity, we represent directly the atoms that label nodes (here, the labeling is injective). The terms \( a_i \) are constants while the terms \( z_i \) are nulls. Let us consider the following path of terms in the chase: \( p = a_4 \land z_1 \land z_2 \land r \land a_3 \land s \land r \land a_1 \). We can see that the structure associated with \( p \) has no direct connection with the structure of the chase graph. However, Prop. 2 allows us to make a connection: if we take any pair of nodes \( \nu_1 \) and \( \nu_2 \) labeled by atoms that respectively contain \( a_3 \) and \( a_3 \), we can see that the atom of the same name in \( q \) is mapped to the path \( p \). The atoms of the chase graph involved in this match are underlined.

4 CRPQ Entailment over Linear Rules

In this section, we describe a method for CRPQ entailment in the presence of linear rules, which allows us to pinpoint
the worst-case complexity of this problem. Throughout this section, we let \( I \) be an instance, \( \mathcal{R} \) a set of linear rules, and \( q \) be the Boolean CRPQ we wish to answer.

As it will be important to keep track of the paths in the chase that link the terms of a pair of atoms, we define the set of transitions between two atoms.

**Definition 2** Let \( \alpha_1 \) and \( \alpha_2 \) be two atoms of \( \text{chase}(I, \mathcal{R}) \) of arities \( k_1 \) and \( k_2 \) respectively. The transitions from \( \alpha_1 \) to \( \alpha_2 \), denoted by \( T_{\alpha_1,\alpha_2}(\mathcal{R}) \), is the set of quadruples \((v_1, s_1, t_1, s_2)\) such that \( 1 \leq t_1 \leq k_1, 1 \leq i \leq k_2, \) and there exists an automaton \( \mathcal{A} \) in \( q \) and a path \( p \) in \( \text{chase}(I, \mathcal{R}) \) going from \( \alpha_1[v_1] \) to \( \alpha_2[i] \) in \( \mathcal{A} \) such that \( \ell(p) \) takes \( \mathcal{A} \) from \( s_1 \) to \( s_2 \). More generally, by transition, we will mean a quadruple \((v_1, s_1, t_1, s_2)\) where \( s_1, s_2 \) are states appearing in \( q \), and \( t_1, t_2 \) do not exceed the maximum predicate arity in \( I \cup \mathcal{R} \).

We now introduce the key technical notion underlying our approach, namely, proof schemes. Formally, a proof scheme is a directed forest whose nodes are atoms, together with transitions for different pairs of atoms occurring in the proof scheme. Intuitively, proof schemes correspond to a subset of chase atoms, with the transitions placing requirements on the paths linking these atoms.

**Definition 3** A proof scheme is a pair \( \mathcal{P} = (\mathcal{F}, \mathcal{T}) \) such that:

- \( \mathcal{F} = (V, E) \) is a directed forest;
- \( V \) is a set of atoms;
- for each null \( n \), the set of nodes in which \( n \) occurs form a connected subgraph of \( \mathcal{F} \);
- \( \mathcal{T} \) is a set of pairs \((\tau, (\alpha_1, \alpha_2))\), where \( \tau \) is a transition, \( \alpha_1 \) and \( \alpha_2 \) belong to \( V \);
- for each \((\tau, (\alpha_1, \alpha_2)) \in \mathcal{T} \), either \( \alpha_1 = \alpha_2 \), \( (\alpha_1, \alpha_2) \subseteq E \), \( (\alpha_2, \alpha_1) \subseteq E \), or both \( \alpha_1 \) and \( \alpha_2 \) are roots.

The terms of a proof scheme, denoted by \( \text{terms}(\mathcal{P}) \), is the union of all the terms of atoms in \( V \).

We next define valid proof schemes. The validity conditions enforce that the proof scheme behaves as intended: its atoms can be mapped into the chase graph in a way that respects the forest structure, and the transitions in the proof scheme are indeed realized in the chase. Note that valid proof schemes are not far, in essence, from the proof generators used in [Gottlob et al., 2015] for deciding CQ entailment in linear rule KBs; however, the presence of transitions and their interactions with linear rules make the validity check much more complex. The idea is to consider the following brute-force algorithm for CRPQ answering: enumerate all proof schemes, check whether the considered proof scheme is valid w.r.t. \( I, \mathcal{R} \) and \( q \) and whether there is a ‘match’ of \( q \) in the proof scheme. To obtain our complexity results, we will upper bound the size of the proof schemes that need to be considered and the complexity of the validity check.

To formalize the notion of a valid proof scheme, we need the classical notion of the type of an atom, as well as the notion of isomorphic atoms with respect to a bijection, which allows us to identify atoms that ‘act the same’.

**Definition 4** The type of an atom \( r(t_1, \ldots, t_k) \) is the pair \((r, \mathcal{P})\) where \( \mathcal{P} \) is the partition of \( \{1, \ldots, k\} \) induced by term equality (i.e., \( i \) and \( j \) are in the same class of \( \mathcal{P} \) iff \( t_i = t_j \)).

**Definition 5** Two atoms \( \alpha_1 \) and \( \alpha_2 \) are isomorphic with respect to a bijection \( b \) from a set of terms \( T_1 \) to a set of terms \( T_2 \) if they have the same type \((r, \mathcal{P})\) and, for any class \( C \) of \( \mathcal{P} \), if \( \alpha_1[C] \in T_1 \) then \( \alpha_2[C] = b(\alpha_1[C]) \), otherwise \( \alpha_2[C] \notin T_2 \), and \( \alpha_1[C] \) and \( \alpha_2[C] \) are both nulls.

**Definition 6** A proof scheme \( \mathcal{P} = (\mathcal{F}, \mathcal{T}) \) with \( \mathcal{F} = (V, E) \) is valid w.r.t. \((I, \mathcal{R}, q)\) if there is an injective mapping \( w \) from \( \text{terms}(\mathcal{P}) \) to \( \text{terms}(\text{chase}(I, \mathcal{R})) \) such that:

- for each root atom \( \alpha \in V \), \( w(\alpha) = \alpha \) and \( \alpha \in I \);
- for each atom \( \alpha \in V \), \( w(\alpha) \) belongs to \( \text{chase}(I, \mathcal{R}) \);
- for each atom \( \alpha \in V \), \( \alpha \) and \( w(\alpha) \) are isomorphic with respect to the identity function on \( \text{terms}(I) \);
- for each edge \((\alpha_1, \alpha_2)\), it holds that:
  - \( \alpha_2 \) and \( w(\alpha_2) \) are isomorphic with respect to the bijection \( b \) from \( \text{terms}(\alpha_1) \) to \( \text{terms}(w(\alpha_1)) \) induced by their common type;
  - \( CG(I, \mathcal{R}) \) contains nodes \( \nu_1, \nu_2 \) such that \( \nu_1 \neq \nu_2 \), \( \ell(\nu_1) = w(\alpha_1) \), \( \ell(\nu_2) = w(\alpha_2) \).

In a proof scheme, transitions label pairs of atoms \((\alpha_1, \alpha_2)\) such that \( \alpha_1 = \alpha_2 \) or both atoms are roots, or one is the parent of the other in a tree. Saturation builds all the transitions that can be obtained by composing these transitions:

**Definition 7** Let \( \mathcal{P} = (\mathcal{F}, \mathcal{T}) \) be a proof scheme. The saturation of \( \mathcal{P} \) is the smallest set \( \mathcal{T}' \) such that \( \mathcal{T} \subseteq \mathcal{T}' \) and \( \{((i_1, s_1, t_1, s_2), (\alpha_1, \alpha_2)), ((i_2, s_2, t_2, s_3), (\alpha_2, \alpha_3)), ((i_3, s_3), (\alpha_3, \alpha_3))\} \subseteq \mathcal{T}' \) implies \( (((i_1, s_1, t_1, s_2), (\alpha_1, \alpha_2)), ((i_2, s_2, t_2, s_3), (\alpha_2, \alpha_3))), ((i_3, s_3), (\alpha_3, \alpha_3)) \in \mathcal{T}' \).

After saturating the transitions of \( \mathcal{P} \), one may not keep a proof scheme as new transitions may label any pair of atoms, however if \( \mathcal{P} \) is valid, the validity conditions remain satisfied.

We next define the notion of a match of a query in a proof scheme to formalize what it means for a query to be mapped into the part of the chase represented by a valid proof scheme.

**Definition 8** Let \( \mathcal{P} = (F, T) \) be a proof scheme with \( F = (V, E) \). We say that there is a match of \( q \) in \( \mathcal{P} \) if there is a mapping \( \pi \) from \( \text{vars}(q) \) to \( \text{terms}(\mathcal{P}) \) such that:

- if \( \alpha \) is a standard atom in \( q \), then \( \pi(\alpha) \in V \);
- if \( \alpha = \Lambda(x, y) \) is a path atom in \( q \) with associated NFA \( \mathcal{A} \), then there is a pair \((i_x, s_x, f), (\alpha_1, \alpha_2)) \) in the saturation of \( T \) such that \( \pi(x) = \alpha_1[i_x] \), \( \pi(y) = \alpha_2[i_y] \) and \( s \) (resp. \( f \)) is an initial (resp. final) state of \( \mathcal{A} \).

Note that the problem of deciding if there is a match from given \( q \) in given \( \mathcal{P} \) is NP-complete. The next proposition shows that CRPQ answering reduces to the existence of a match in a valid proof scheme.

**Proposition 3** \( I, \mathcal{R} \models q \) if and only if there exists a match of \( q \) in some valid proof scheme.

**Proof Sketch:** Suppose \( I, \mathcal{R} \models q \), and let \( \pi' \) be a match of \( q \) in \( \text{chase}(I, \mathcal{R}) \). Let \( V_1 \) be the set of images of the standard atoms of \( q \), as well as the atoms in which the image of the terms of \( q \) has been created. Let \( V_1' \) contain for each \( \alpha \in V_1 \) a node \( \nu_\alpha \in CG \) such that \( \ell(\nu_\alpha) = \alpha \). Define \( V_2 = V_1' \cup \{gfb(\nu, \nu')\} \),
where the union is performed over all \((\nu, \nu') \in V_0^2 \times V_0^2\) such that \(\nu\) and \(\nu'\) have the same root in \(G\). Let \(F\) be the forest structure induced by \(G\) over \(V_0^3\). We define \(V\) as \(\{\ell(\nu) : \nu \in V_0^3\}\), and \(E\) as the pairs \((\ell(\nu), \ell(\nu'))\) for any edge \((\nu, \nu')\) in \(F\). We then define \(F\) as the transitions from \(\alpha_1\) to \(\alpha_2\) such that one is the child of the other, or both are roots, or \(\alpha_1 = \alpha_2\). One can check that \(\pi'\) is a match of \(q\) in \(P = (V, E, T)\).

The main argument is as follows: By Prop. 2, the saturation of \(P\) contains all the transitions that hold between an atom and its descendants. As the glb of any two nodes with the same root also belongs to \(F\), the saturation of \(P\) also contains all the transitions that hold between any pair of atoms of \(P\). As the mapping of standard atoms is direct, \(\pi'\) is a match of \(q\) in \(P\). The converse direction follows from the definitions.

Importantly, the size of the valid proof scheme built in the previous proof is polynomial in the size of \(q\), so it is only necessary to consider polynomial-size proof schemes. It remains to show how to test whether a given proof scheme is valid. We will make use of the following lemma, which shows how transitions between an atom in the chase graph and an ancestor atom can be computed using the transitions relating its parent atom to the same ancestor, together with the transitions that can be obtained by considering the atom in isolation.

**Lemma 1** Suppose \(\nu_3\) is a descendant of \(\nu_1\) in \(G(I, R)\), \(\nu_3\) is a child of \(\nu_2\), and \(\alpha_i = \ell(\nu_i)\) for \(1 \leq i \leq 3\). Then:

1. \((i_1, i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) if there exist \((i_1, i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) and \((i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) such that \(\alpha_2[i_2] = \alpha_3[i_4] = \alpha_3[i_3]

2. \((i_1, i_2, i_3) \in T_{q,1,R}(\alpha_1, \alpha_2)\) if there exist \((i_1, i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) and \((i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) such that \(\alpha_2[i_2] = \alpha_3[i_3]

3. \((i_1, i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) if there exist \((i_1, i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) and \((i_2, i_3, i_4) \in T_{q,1,R}(\alpha_1, \alpha_2)\) such that \(\alpha_2[i_2] = \alpha_3[i_3] \neq \alpha_3[i_4]

We now exploit this lemma and existing complexity results on RPQ answering to design a procedure for checking validity of a proof scheme, with the following complexity:

**Proposition 4** Checking the validity of a proof scheme can be done in single exponential time, in polynomial space if predicate arity is bounded, and in NL if \(q\) and \(R\) are fixed.

**Proof Sketch:** First observe that if \(\alpha, \alpha'\) are root atoms of \(P\), we can compute \(T_{q,1,R}(\alpha, \alpha')\) by calling an RPQ entailment oracle. We then work from the roots to the leaves, guessing for each edge \((\alpha_1, \alpha_2)\) in \(P\), a sequence of rule applications that take \(\alpha_1\) to \(\alpha_2\), which ensures the satisfaction of the fourth validity condition. Importantly, we can bound the length of this sequence by noticing that after exponentially many rule applications, we will have generated two atoms \(\beta_1, \beta_2\) that ‘behave the same’ (roughly: they are isomorphic w.r.t. terms(\(\alpha_1\)) and have the same transitions w.r.t. \(\alpha_1\) and w.r.t. themselves). To check the fifth validity condition, we leverage Lemma 1 to inductively construct the set of transitions between \(\alpha_1\) and \(\alpha_2\), and between \(\alpha_2\) and itself. Note that we can directly compute \(T_{q,1,R}(\alpha_2, \alpha_2)\) using RPQ entailment. The procedure we have sketched runs in non-deterministic polynomial space (constant space if \(q\) and \(R\) are fixed) with access to an RPQ entailment oracle. It then suffices to recall that RPQ entailment is in EXPTIME, in P for bounded-arity rules, and in NL if \(q\) and \(R\) are fixed.

Since all poly-size proof schemes can be enumerated in polynomial space, by Prop. 4 we obtain that the brute-force algorithm runs in EXPTIME in the general case, in PSPACE if the arity is bounded, and in NL if \(q\) and \(R\) are fixed (in which case the number of proof schemes and their size are constants, and checking the existence of a match is in constant time). Matching lower bounds are inherited from RPQ answering with linear rules [Bienvenu and Thomazo, 2016], or CRPQ answering in DL-Lite\(\pi\) [Bienvenu et al., 2015].

**Theorem 1** CRPQ entailment under linear rules is EXPTIME-complete in combined complexity, PSPACE-complete in combined complexity with bounded-predicate arity, and NL-complete in data complexity.

### 5 Extension to Guarded Rules

We reduce CRPQ answering under guarded rules to CRPQ answering under linear rules. For simplicity, we assume that all the predicates in \(I\) also occur in \(R\) (however, the reduction is easily extended to drop this restriction). We first explain the ideas that underly the reduction. Prop. 1, which states that, under linear rules, facts can be chased independently, fails for guarded rules. However, a kind of independence can be recovered if instead of single fact we consider the (maximal) subset of the chase guarded by this fact, as expressed by Prop. 5, which also applies to non-necessarily ground atoms.

**Proposition 5** For any set of atoms \(F\), it holds that \(\text{chase}(F, R)\) is equivalent to \(\bigcup_{\alpha \in F} \text{chase}(F^*, R_{\alpha})\), where \(F^* = \text{chase}(F, R)_{\text{terms}(\alpha)}\).

By recursively using Prop. 5, one can devise the following alternative chase procedure. Let \(F = I\): (i) Compute \(F^*\) the restriction of \(\text{chase}(F, R)\) to \(\text{terms}(F)\); this can be done by calling an oracle for each candidate atom on \(\text{terms}(F)\) to decide if it is entailed by \(\text{chase}(F, R)\); (ii) For each \(\alpha \in F^*\) and \(G\) the maximal subset of \(F^*\) guarded by \(\alpha\) (i.e., \(G = F_{\alpha}^*\)), for each rule \(\rho \in \mathcal{R}\), let \(\pi\) be the homomorphism from \(\text{body}(\rho)\) to \(G\) that maps the guard of \(\rho\) to \(\alpha\): (i-1) Apply \(\rho\) to \(G\) by \(\pi\) and let \(G_{\rho, G}\) be the restriction of the generated set to the subset guarded by \(\pi^*(\text{head}(\rho))\); (ii-2) Perform steps (i) and (ii) with \(F = G_{\rho, G}\); (iii) The chase is the union of all the \(F^*\) built at Step (i).

Now, let \((I, R)\) be the original KB, where \(R\) is a set of guarded rules, and \((I', R')\) be the translated KB, where \(R'\) is a set of linear rules. Roughly, the reduction allows to simulate the alternative chase outlined above, with the translation being as follows: (i) we build \(I'\) from \((I, R)\) by replacing each \(\alpha \in I\) by an atom that represents \(\text{chase}(I, R)_{\text{terms}(\alpha)}\) and, (ii) we build \(R'\) by replacing each rule \(\rho \in R\) by a set of linear rules, each representing an application of \(\rho\) to a distinct guarded subset of \(\text{chase}(I, R)\). This construction will require new predicates associated with guarded sets of atoms. We will show that \(\text{chase}(I, R)\) and \(\text{chase}(I', R')\) are homomorphically equivalent when restricted to the original predicates,
which implies that \(\text{chase}(I', \mathcal{R}')\) can be used to answer CRPQs as these are closed under homomorphism.

To obtain a finite number of guarded sets (hence a finite set of new predicates, which will ensure the finiteness of the set of rules \(\mathcal{R}'\)), we have to rename them in a canonical way. To that end, we consider a new set of canonical variables \(\mathcal{X} = \{T_1, \ldots, T_{2w}\}\), where \(w\) is the maximal predicate arity (\(2w\)) and the maximal number of terms in a rule from \(\mathcal{R}'\).

**Definition 9** Let \(\alpha\) be an atom which may contain both classical terms and canonical variables. The canonical renaming \(\Phi_{\alpha, \mathcal{Y}}\) of \(\alpha\) w.r.t. \(\mathcal{Y} \subseteq \mathcal{X}\) is a substitution from terms(\(\alpha\)) to \(\mathcal{X}\) such that \(\Phi_{\alpha, \mathcal{Y}}(t_i) = t_j\) if \(t_i \in \mathcal{X}\), otherwise \(\Phi_{\alpha, \mathcal{Y}}(t_i) = T_j\) where \(j\) is the smallest integer such that \(T_j \not\in \mathcal{Y}\). \(T_j\) does not occur in \(\alpha\), and \(T_i \neq \Phi_{\alpha, \mathcal{Y}}(t_k)\) for all \(k < i\). If \(\mathcal{Y} = \emptyset\), we simply write \(\Phi_\alpha\).

We denote by \(\mathcal{G}\) the set of guarded atoms over canonical variables. Note that \(\mathcal{G}\) is finite. For any guarded set \((G, \alpha)\), it holds that \((\Phi_{\alpha}(G), \Phi_{\alpha}(\alpha)) \in \mathcal{G}\). To each guarded set \((G, \alpha)\) in \(\text{chase}(I, \mathcal{R})\) can be assigned a complex predicate, of the form \(p_{\Phi_\alpha}(G)\) with the same arity as \(\alpha\). As \(\mathcal{G}\) is finite, so is the set of complex predicates. The canonical atom associated with a guarded set \((G, \alpha)\) is \(\text{can}(G, \alpha) = p_{\Phi_\alpha}(\text{terms}(\alpha))\).

**Definition 10** The guarded translation of \(I\) w.r.t. \(\mathcal{R}\) is \(I' = \{\text{can}(G(\alpha, I, \mathcal{R})) | \alpha \in I\}\), where \(G(\alpha, I, \mathcal{R})\) denotes the guarded set \((\text{chase}(I, \mathcal{R})|_{\text{terms}(\alpha)}), \alpha\).

**Example 2** Consider Ex. 1, where \(I_e = \{\alpha_1, \alpha_2\}\), with \(\alpha_1 = \text{b}(a_1, a_2, a_3, a_4)\) and \(\alpha_2 = \text{t}(a_1)\). In the following, we underline the guard in a guarded set. \(G(\alpha_2, I_e, \mathcal{R}_e) = \{\text{t}(a_1)\}\), hence \(\text{can}(G(\alpha_2, I_e, \mathcal{R}_e)) = p_{\Phi_\alpha}(\text{terms}(\alpha_1))\).

**Lemma 2** Let \((G, \alpha = r(t))\) be a guarded set of atoms. The following two statements hold:

- If \(r \in \mathcal{R}_c\) be applicable to \(G\) by \(\pi\), and let \(F = G \cup \pi^\circ(\text{head}(\rho))\). There exists a complex rule \(\rho'\) applicable to \(p_{\Phi_\alpha}(G(t))\) by \(\pi'\) such that:
  
  \[
  \text{expansion}(p_{\Phi_\alpha}(G(t)) \cup \pi^\circ(\text{head}(\rho'))) = F.
  \]

- If \(\rho' \in \mathcal{R}_c\) be applicable to \(G' = p_{\Phi_\alpha}(G(t))\) by \(\pi'\), and let \(F' = G' \cup \pi^\circ(\text{head}(\rho'))\). There exists \(\rho \in \mathcal{R}\) applicable to \(G\), generating \(F\), such that:
  
  \[
  \text{chase}(G, \mathcal{R}|_{\text{terms}(F)}) \models \text{expansion}(F').
  \]

**Proposition 6** Given \((I, \mathcal{R}), I'\) and \(\mathcal{R}'\) as defined above can be computed in \(2\text{EXPTIME}\) in combined complexity, \(\text{EXPTIME}\) when the arity is bounded, and in \(\text{PTIME}\) in data complexity. The number of types whose predicate appears in \(\mathcal{R}'\) is at most a double exponential in \(I\) and \(\mathcal{R}\) (exponential when the arity is bounded, and constant w.r.t. the data).

The next theorem follows from the provided reduction and our results for linear rules. Let us remark that the reduction to linear rules not being polynomial, we do not directly obtain the desired upper bounds. However, by carefully analyzing the algorithm for linear rules, we can show that it actually runs polynomially in the number of types.

**Theorem 2** CRPQ entailment under guarded rules is \(2\text{EXPTIME-complete in combined complexity, } \text{EXPTIME-complete in combined complexity with bounded-predicate arity, and } \text{PTIME-complete in data complexity.}\)

**6 Conclusion**

We have made an important step in understanding the complexity of answering navigational queries over existential rules, by determining the precise complexity of answering CRPQs under linear and guarded rules, two central subclasses of existential rules. Our results are quite positive, as we showed that answering CRPQs under guarded rules is not harder than answering the more well-known CQs, under both combined complexity (with or without arity restrictions) and data complexity. For linear rules, we proved that moving from RQP to CRPQ does not incur any additional computational cost, and for bounded-arity rules, we have the same complexity as for DL-Lite\(p\). We believe that the simulation of guarded rules by linear rules is of independent interest.

**Acknowledgements**

We thank Swan Rocher for fruitful discussions. This work was supported by the French ANR project PAGODA (ANR-12-JS02-0007).
References


