Query Rewriting for Nested Navigational Queries over Property Graphs

Bianca **Löhnert**[1], Nikolaus **Augsten**[1], Cem **Okulmus**[2] and Magdalena **Ortiz**[3]

[1]*University of Salzburg, Austria* [2]*Paderborn University, Germany* [3]*TU Wien, Austria*

Abstract

The framework of ontology-mediated data access (OBDA) has enjoyed great success in the setting of relational databases. But while querying graph data has always been seen as a key motivation of OBDA, current technologies lack support for standard graph database systems. In recent years, graph databases using the labeled property graph data model have seen increasing popularity, and two ISO standards for querying property graphs were made public in 2023: GQL and SQL-PGQ. Leveraging this momentum, we take a big step toward ontology-mediated querying of property graphs. We propose $DL\text{-}Lite_{PG}$, a DL-Lite variant tailored for property graphs that uses property values to define concepts and roles in the ontology, and present a practical rewriting algorithm for rewriting navigational queries with $DL\text{-}Lite_{PG}$ ontologies. We consider nested two-way regular path queries (N2RPQs) and a large class of conjunctive N2RPQs. Our queries can access property values within path expressions and capture a substantial portion of the GQL and SQL-PGQ standards. This is the first algorithm to fully support full 2RPQs in the presence of ontologies by leveraging nested regular paths. We present our algorithm and a proof-of-concept implementation and conclude with a set of preliminary experiments that showcase the practicality of our approach.

Keywords

ontology-based data access, graph query languages, ontology-meditated query answering, description logics

1. Introduction

A core aim of the *ontology-based data access* (OBDA) framework (additionally to the virtual integration of data sources) is to extend existing data with domain-specific knowledge without the need to materialize all ensuing facts, but still enable access to all the consequences when answering user queries. One of the key techniques to realize OBDA is via *query rewriting*: one takes as input the domain-specific knowledge in the form of an ontology, and a query specified over the extended signature (with terms from the data and the ontology) and rewrites into a new query, which uses the reduced signature that is present in the data alone, but captures all the semantic consequences that would follow in the presence of the ontology. Hence, one can evaluate the query using an existing system while still making effective use of the ontology. This is known as *ontology-mediated query answering* (OMQA) and it is one of the key techniques underlying many OBDA systems.

The OBDA framework is already seeing a number of commercial applications, albeit only in the setting of relational databases. While it is unlikely that relational databases are being replaced anytime soon, we none-the-less see new forms of storing and querying data. Graph databases are one such example, where we already have a large number of commercial systems. Instead of the relational model that underpins relational database systems, these instead focus on the labeled property graph (LPG) model. It allows for nodes and binary edges over nodes, each of which can be mapped to a number of labels and associated with data values via so-called *properties*. The increasing popularity of graph databases has recently culminated in the standardization body ISO issuing two new standard for graph query languages: GQL and SQL-PGQ. The key distinguishing feature of these query languages are their *navigational* capabilities, that allow to match arbitrarily long paths in the data. Both GQL and SQL-PGQ share the same navigational core [1], which includes the standard navigational query

languages: (two-way) regular path queries ((2)RPQs), nested 2RPQs (N2RPQs), and their conjunctive versions (C2RPQs and CN2RPQs).

The study of navigational queries in the context of OMQA is far from novel. For more than a decade we have had tight complexity results for most description logics, ranging from lightweight to highly expressive, as well as for all standard navigational languages (2RPQs, N2RPQs, C2RPQs, and CN2RPQs), e.g., [2, 3, 4]. However, these studies focused on the boundaries of decidability and computational complexity. The proposed algorithms are so unnameable to implementation that, more than a decade later, not a single practical implementation has been proposed.

Unfortunately, there were two significant roadblocks that discouraged the development of practical techniques for navigational OMQs over graph databases. First, practical languages imposed ad-hoc restrictions on navigational languages that made them inadequate. Cypher, the most widely used language for graph databases, did not support RPQs in full, and did not enable the basic homomorphism semantics that is natural in the OMQA context. Second, it has been proven that even plain RPQs in the presence of the simplest DL-Lite ontologies cannot be rewritten into C2RPQs [5]. Instead, they require nested RPQs, which existing technologies did not support. The arrival of GQL and SQL-PGQ has removed both obstacles at once. Since the standard now contains CN2RPQs and basic homomorphism semantics, practical languages are being updated quickly to support it. This has finally made it feasible to have practical OMQA for graph databases.

Seizing the opportunity, we propose the first practical query rewriting technique that covers full 2RPQs and N2RPQs in the presence of DL-Lite ontologies. We also consider conjunctive N2RPQs; however, to ensure the algorithm remains practical and useful, we impose a restriction called 'join-on-free', whereby non-answer variables cannot be shared by multiple atoms. We also consider the property data values, a central feature of the LPG model that is often disregarded in the OMQA literature. We allow property tests in queries, also along the navigational paths. On the ontology side, we define a variant of DL-Lite that can use property value tests on the left-hand-side of axioms, thus allowing to create concepts and roles based on these values.

In summary, we make the following contributions:

1. We present the first practical algorithm to rewrite join-on-free CN2RPQs into CN2RPQs, capturing the full power of navigational languages such as GQL.

2. This is the first work on OMQA on the LPG model that makes full use of properties, by including data tests over property values in both the ontology and query language.

3. We show-case the practical utility of the algorithm by providing a proof-of-concept implementation that takes as input an ontology in OWL format and the real-world query language Cypher and rewrites into Cypher.

This paper is structured as follows. In Section 2 we present the key terminology of our setting and also introduce our novel description logic to express tests over propety values. In Section 4 we present our rewriting algorithm on join-on-free CN2RPQs. In Section 5 we present our proof-of-concept implementation and an experimental evaluation that aims to show its performance. In Section 6 we summarise our results and highlight future work.

Related Work. We note that a few recent works have begun the effort of closing this gap towards the practical OMQA algorithms in the graph database setting. Already Di Martino et al. [6] leveraged the recursion in regular path queries to be able to rewrite a fragment of $\mathcal{EL}$. They use navigational queries as target language for query rewriting, but their source languages are only instance queries and conjunctive queries. It is also a theoretical work not aimed at implementation, and thus more inline with the mentioned theoretical works [2, 3] than with this paper. Aiming for practical implementation, Dragovic et al. [7] considered a restricted fragment of C2RPQs that can be rewritten into UC2RPQs, and *DL-Lite* ontologies. It has limited support for data tests. A next step came in the work from [5], where the ontology language is also a fragment of $\mathcal{EL}$ very similar to the one of Di Martino et al. [6]. There are also restrictions on the regular path expressions to ensure rewritability into C2RPQs.

Table 1
Semantics of *DL-Lite*$_{PG}$

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse role	r^-	$\{(b,a) \mid (a,b) \in r^{\mathcal{I}}\}$
exist. restriction	$\exists r.\top$	$\{a \mid (a,b) \in r^{\mathcal{I}}\}$
data test	$p \odot v\,?$	$\{a \mid (a,v') \in p^{\mathcal{I}}$ for some v' with $v' \odot v\} \cup$
		$\{(a,a') \mid ((a,a'),v') \in p^{\mathcal{I}}$ for some v' with $v' \odot v\}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
set of data tests	$\{T_1, \ldots, T_n\}$	$T_1^{\mathcal{I}} \cap \cdots \cap T_n^{\mathcal{I}}$

2. Querying Property Graphs with Navigational Queries and DL-Lite

Ontology language. We present now the ontology language that we will use in our approach, which we call *DL-Lite*$_{PG}$. It is a careful extension of the well-known DL-Lite family [8] of ontology languages, where we permit tests over properties via a concrete domain, affecting both concept and role inclusions.

We assume disjoint, countably infinite sets $\mathbf{C}$, $\mathbf{R}$, $\mathbf{N}$ and $\mathbf{K}$ of *concept names*, *role names*, *individuals* and *property keys*, respectively. The set of *roles* is defined as $\overline{\mathbf{R}} = \mathbf{R} \cup \{r^- \mid r \in \mathbf{R}\}$.

We also assume a *concrete domain* $(\mathbf{D}, \mathbf{P^D})$, with $\mathbf{D}$ a set of *values* and $\mathbf{P^D}$ a set of binary predicates over $\mathbf{D}$. For example, $(\mathbf{D}, \mathbf{P^D})$ could contain the integers with the usual $=, \leq, \geq$ predicates. We define the set of *data tests* as $\mathbf{T^D} = \{p \odot v\,? \mid p \in \mathbf{K}, \odot \in \mathbf{P^D}, v \in \mathbf{D}\}$.

Definition 1 (*DL-Lite*$_{PG}$)**.** *To define the types of inclusions, we use the following syntax:*

$$B := A \mid \exists r.\top \mid T \text{ (basic concepts and data test)} \qquad E := r \mid r^- \text{ (basic role)}$$
$$C := B \mid \neg B \text{ (general concept)} \qquad\qquad\qquad F := E \mid T \text{ (roles and data test)}$$
$$R := E \mid \neg E \text{ (general role)}$$

where $A \in \mathbf{C}$, $r \in \overline{\mathbf{R}}$ and $T \subseteq \mathbf{T^D}$. A concept inclusion *(CI) has the form $B \sqsubseteq C$ and a* role inclusion *(RI) the form $F \sqsubseteq R$. A TBox is a finite set of CIs and RIs. We use $\sqsubseteq_{\mathcal{T}}^*$ to denote the reflexive transitive closure of $\{(r,s) \mid r \sqsubseteq s \in \mathcal{T}\}$ and call r a* subrole *of s (in $\mathcal{T}$) if $r \sqsubseteq_{\mathcal{T}}^* s$.*

The semantics are given via *interpretations* of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, with $\Delta^{\mathcal{I}}$ a non-empty set called the *abstract domain*. $\cdot^{\mathcal{I}}$ is the *interpretation function*, which assigns to every $A \in \mathbf{C}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every $r \in \mathbf{R}$ a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to every $p \in \mathbf{K}$ a relation $p^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \cup (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})) \times \mathbf{D}$.

It is extended to concepts and CIs in the usual way, as seen in Table 1. Modelhood is also standard.

Example 1. *In the following, we provide a DL-Lite$_{PG}$ TBox that provides knowledge that can be applied to a social network graph, such as profile information from LinkedIn.*

$$\mathcal{T} = \{ \ \{\textsc{born} \geq \textit{1997}\,?, \textsc{born} \leq \textit{2012}\,?\} \sqsubseteq \textsf{GenZ}, \qquad \exists \textit{employs}^-.\top \sqsubseteq \textsf{Employed},$$
$$\{\textsc{time} \geq (\textit{2025-01-01})?\} \sqsubseteq \textsf{Recent}, \qquad \textsf{Employed} \sqsubseteq \neg \textsf{Unemployed},$$
$$\textsf{Opole} \sqsubseteq \textsf{Poland}, \qquad \textsf{TechCompany} \sqsubseteq \exists \textit{employs}.\textsf{Engineer},$$
$$\exists \textit{announce}.\top \sqsubseteq \textsf{Hiring}, \qquad \textit{friendsWith} \sqsubseteq \textit{friendsWith}^- \ \}$$

Data model. In this paper the data (or ABox) is given as finite *property graphs* [9, 10, 11].

Definition 2. *A* property graph *(PG) $\mathcal{A}$ has the form $(N, E, \textsf{label}, \textsf{prop})$, where:*
- *N is a non-empty set of* nodes*;*
- *E is the set of* edges*, where each edge is a triple (r, n, n') with $r \in \mathbf{R}$ and $n, n' \in N$; we may write such an edge in the form $r(n, n')$ and call it an r-edge;*

- label *is a total function* $N \to 2^{\mathbf{C}}$;
- prop *is a partial function* $(N \cup E) \times \mathbf{K} \to \mathbf{D}$ *mapping pairs* (u, p) *with* $u \in (N \cup E)$ *and* $p \in \mathbf{K}$ *to a value in* $\mathbf{D}$.

If $N \subseteq \mathbf{N}$ *and it is finite, we call it an* ABox. *We say that* $\mathcal{A}' = (N', E', \mathsf{label}', \mathsf{prop}')$ *is a* subgraph *of* $\mathcal{A}$ *if* $N' \subseteq N$, $E' \subseteq E$, $\mathsf{label}'(n) = \mathsf{label}(n)$ *for all* $n \in N'$, *and* $\mathsf{prop}'(u, p) = \mathsf{prop}(u, p)$ *for all* $u \in N' \cup E', p \in \mathbf{K}$.

Note that our definition of property graph allows only a single edge for each role between each pair of nodes. Also, the same name of property keys is used for both nodes and edges, as usually done in property graphs [10, 11].

Example 2. *Below is an example property graph, over the social network setting from Example 1.*
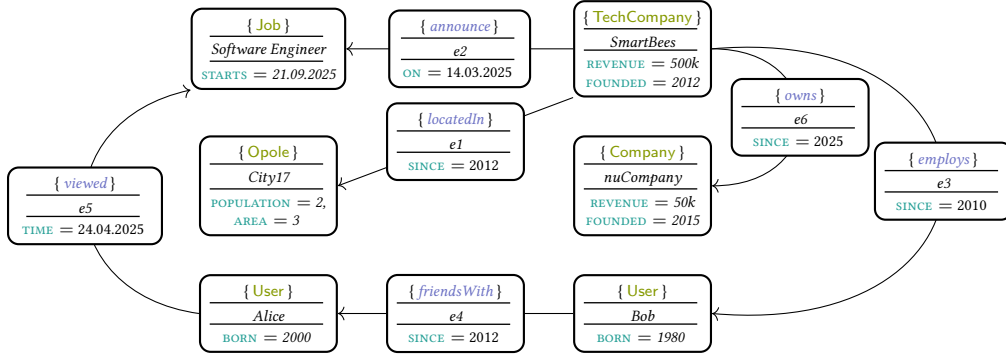
Each interpretation can be seen as a property graph, and vice-versa.

Definition 3. *For a property graph* $\mathcal{A} = (N, E, \mathsf{label}, \mathsf{prop})$, *define* $\mathcal{I}_{\mathcal{A}} = (N, \cdot^{\mathcal{I}})$ *as follows:* $C^{\mathcal{I}} = \{n \in N \mid C \in \mathsf{label}(n)\}$, $r^{\mathcal{I}} = \{(n, n') \mid r(n, n') \in E\}$ *and* $p^{\mathcal{I}} = \{(u, d) \mid d = \mathsf{prop}(u, p)\}$. *Conversely,* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *induces a (possibly infinite) property graph* $PG(\mathcal{I}) = (\Delta^{\mathcal{I}}, E, \mathsf{label}, \mathsf{prop})$, *where* $E = \{r(n, n') \mid (n, n') \in r^{\mathcal{I}}\}$, $\mathsf{label}(n) = \{C \in \mathbf{C} \mid n \in C^{\mathcal{I}}\}$ *and* $\mathsf{prop}(u, p) = \{d \in \mathbf{D} \mid (u, d) \in p^{\mathcal{I}}\}$. $\mathcal{I}$ *is a* model *of an ABox* $\mathcal{A}$, *if* $\mathcal{A}$ *is a subgraph of* $PG(\mathcal{I})$. *An ABox* $\mathcal{A}$ *is* consistent *with a TBox* $\mathcal{T}$ *iff there is a model of* $\mathcal{A}$ *and* $\mathcal{T}$.

For the problem of checking that a given ABox is consistent with an *DL-Lite$_{PG}$* TBox, we refer to the work of Artale et al. [12]; their algorithms can be easily tuned to account for key values on edges.

Query Language. We study *conjunctive nested two-way regular path queries* (CN2RPQs), the extension of C2RPQs, the navigational query language for graphs that has received most attention in OMQA. We enhance CN2RPQs by *data tests* similar as data tests for navigational conjunctive queries in [7] to query for property values, assuming that the predicates in $\mathbf{P^D}$ can be realized in GQL and Cypher. To represent CN2RPQs we rely on *nested nondeterministic finite automaton* (n-NFA), following generally the notions from [3], with some simplifications.

We will first define nested two-way regular path expressions by way of NFAs, and then give the definition of conjunctive nested two-way regular path queries based on them. We assume a given alphabet Σ, and to define nested automata, we extend Σ by allowing n-NFAs as symbols.

Definition 4. *Let* $\mathbf{A}^0$ *be the set of all NFAs over a given alphabet* Σ. *For* $k > 0$, *the set* $\mathbf{A}^k$ *of* nested NFA *(n-NFA) over* Σ *of nesting depth* k *contains all automata over* $\Sigma^k = \Sigma \cup \{\langle \alpha \rangle \mid \alpha \in \mathbf{A}^i, 1 \le i < k\}$.

We omit the nesting depth of an n-NFA when irrelevant. We are interested in n-NFAs that use the alphabet of symbols and tests that may occur in our property graphs. Note that a set of unary data tests can be simulated by multiple transitions, thus the alphabet does not include a set of data tests for nodes.

Definition 5. *A* nested two-way regular path expression *(N2RPE) is an n-NFA over the specific alphabet*

$$\mathbf{\Sigma}_{PG} = 2^{\overline{\mathbf{R}} \cup \mathbf{T^D}} \cup \mathbf{T^D} \cup \{C? \mid C \in \mathbf{C}\}.$$

This alphabet Σ_{PG} consists of three kinds of symbols: 1) sets of (possibly inverted) role labels and data tests, which allow the expression to perform a series of data tests while traversing an edge that has all the role labels in the test; 2) individual data tests, which are checked against the properties of a node and lastly, 3) concept tests, which check whether the current node is part of the required concept.

The semantics of N2RPEs that we give below is based on finding in an interpretation a path with suitable outgoing paths. But sometimes it is convenient to talk about the *word* language of N2RPEs, that is, the set $L(\alpha)$ of words over $\Sigma_{PG} \cup \bigcup_{k \in \mathbb{N}} \mathbf{A}^k$ that it accepts in the standard sense, treating the elements of $\bigcup_{k \in \mathbb{N}} \mathbf{A}^k$ as ordinary alphabet symbols. We introduce some useful definitions.

Definition 6. *We define an* inverse *function over the alphabet* $\Sigma_{PG} \cup \bigcup_{k \in \mathbb{N}} \mathbf{A}^k$ *as follows:*

$$\sigma^- = \begin{cases} \{t \mid t \in \sigma \cap \mathbf{T^D}\} \cup \{r^- \mid r \in \sigma \cap \mathbf{R}\} \cup \{r \mid r^- \in \sigma, r \in \mathbf{R}\}, & \sigma \in 2^{\overline{\mathbf{R}} \cup \mathbf{T^D}} \\ \sigma & \text{otherwise.} \end{cases}$$

The inverse w^- *of a word* $w = \sigma_1 \cdots \sigma_n$ *is* $\sigma_n^- \cdots \sigma_1^-$, *and for a language* L, *we let* $L^- = \{w^- \mid w \in L\}$.
Let $\alpha = \langle Q, \Sigma_{PG}, S, \delta, F \rangle$ *be an N2RPE with* $S \subseteq Q$ *the initial states and* $F \subseteq Q$ *are the final states. Then* α *can be* inverted *to obtain* $\bar{\alpha} = \langle Q, \Sigma_{PG}, F, \delta^-, S \rangle$, *where* $\delta^- = \{(s_j, \sigma^-, s_i) \mid (s_i, \sigma, s_j) \in \delta\}$.

Lemma 1. *For every N2RPE* α, $L(\bar{\alpha}) = L(\alpha)^-$.

We are now ready to use N2RPEs as a query language for property graphs. To this aim, we define the semantics of a 2NRPE α as the pairs of nodes that are connected via α, and where at every nested $\langle \alpha' \rangle$ we can find an outgoing path that complies with α'.

Definition 7. *For a N2RPE* α *of nesting depth* k *and an interpretation* $\mathcal{I}$, *we define* $\alpha^{\mathcal{I}}$ *as the set of pairs* $(o_0, o_n) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *such that there is a sequence of the form* $o_0 \sigma_1 o_1 \sigma_2 \ldots \sigma_n o_n$, *where each* o_i *is a node in* $\Delta^{\mathcal{I}}$, $\sigma_1, \ldots, \sigma_n$ *is a word in* $L(\alpha)$, *and for each* $i \in \{1, \ldots, n\}$:
- *If* $\sigma_i \in 2^{\overline{\mathbf{R}} \cup \mathbf{T^D}}$, *then* (i) *for every data test* $p \odot v? \in \sigma_i^{\mathcal{I}}$ *and the* v' *with* $((o_{i-1}, o_i), v') \in p^{\mathcal{I}}$, *we have* $v' \odot v$, *and* (ii) *for every* $r \in \sigma_i^{\mathcal{I}} \cap \overline{\mathbf{R}}$, *we have* $(o_{i-1}, o_i) \in r^{\mathcal{I}}$.
- *If* $\sigma_i = C?$ *for a concept* C, *then* $o_{i-1} = o_i$ *and* $o_i \in C^{\mathcal{I}}$.
- *If* $\sigma_i = p \odot v?$ *for a property* $p \in \mathbf{K}$, *then* $o_{i-1} = o_i$ *and* $v' \odot v$ *where* $(o_{i-1}, v') \in p^{\mathcal{I}}$.
- *If* $\sigma = \alpha' \in \mathbf{A}^{k-1}$, *then* $o_{i-1} = o_i$ *and there exists some* $o \in \Delta^{\mathcal{I}}$ *such that* $(o_i, o) \in \alpha'^{\mathcal{I}}$.

N2RPEs are a very rich query language in their own right, but we get even more flexible querying capabilities if we use variables to combine N2RPEs.

Definition 8. *A conjunctive N2RPQ (CN2RPQ)* $q(\vec{x})$ *is a conjunction of atoms* $\alpha_1(x_1, y_1) \wedge \cdots \alpha_i(x_i, y_i) \cdots \wedge \alpha_n(x_n, y_n)$ *with each* α_i *an N2RPE and* $\vec{x} \subseteq \bigcup_i^n \{x_i, y_i\}$ *is a tuple of* answer variables. *We call a CN2RPQ* Boolean *if* $\vec{x}$ *is empty.*
The set of join variables $Join(q(\vec{x}))$ *of a CN2RPQ* $q(\vec{x})$ *is the set of those variables that occur in two different atoms of* $q(\vec{x})$. *If* $Join(q(\vec{x})) \subseteq \vec{x}$ *for a CN2RPQ that is not Boolean, then we call* $q(\vec{x})$ *a* join-on-free *CN2RPQ.*

We sometimes abbreviate atoms of the form $\langle \alpha \rangle(x, x)$ as $\langle \alpha \rangle(x)$, and similary, $A?(x)$ as $A(x)$.
It will be convenient to assume that CN2RPQs are connected, that is, the graph whose vertices are the variables and that has an edge between two variables if they occur in the same atom is a connected graph. Disconnected queries can be answered as separate queries and then their answers intersected. Note that, under this assumption, every atom in a join-on-free CN2RPQ has at least one answer variable.

Definition 9 (Semantics of CN2RPQs)**.** *Consider a CN2RPQs* $q(\vec{x})$ *and a property graph* $\mathcal{A} = (N, E, \mathsf{label}, \mathsf{prop})$. *A match* μ *for* $q(\vec{x})$ *in* $\mathcal{A}$ *is a mapping from the variables in* q *to nodes in* N *such that* $(\mu(x), \mu(y)) \in \alpha^{\mathcal{I}_{\mathcal{A}}}$ *for every atom* $\alpha(x, y)$ *occurring as a conjunct in* $q(\vec{x})$. *The tuple* $\mu(\vec{x})$ *is then called an* answer *for* $q(\vec{x})$ *in* $\mathcal{A}$.

Example 3. *To illustrate our query language, we use the schema introduced in Example 1. The query $q_1(x, y)$ retrieves all technical companies located in Poland that are currently hiring and that have an employee who is a friend of friends of the user. In query $q_2(x, y)$ a recruiter might be interested in friends (and friends of friends) born after 2000 of an employee, who has been working for the company for over three years. In query $q_3(x, y)$, we look for companies that were founded before 2003, with a revenue of 10^5 or more, and want to find all Polish companies that they own since 2010, and also look for all companies that these companies own, and so on. In query $q_4(x, y)$, we look for all companies that employ workers that are friends with some user of GenZ, and return the pairs of company and users, when the user also viewed a job offer by the company that the friend is currently employed.*

$q_1(x, y) := friendsWith^* \cdot employs^-(x, y), \langle locatedIn \cdot \text{Poland?} \rangle \cdot \text{Hiring?} \cdot \text{TechCompany?}(y, z)$

$q_2(x, y) := \{employs^-, \text{SINCE} \leq 2023?\} \cdot friendsWith^* \cdot \text{BORN} \geq 2000?(x, y),$
$\qquad\qquad \langle \{viewed, \text{TIME} \geq 01.01.2024?\} \rangle \cdot friendsWith(z, y)$

$q_3(x, y) := \text{FOUNDED} \leq 2015? \cdot \text{REVENUE} \geq 10^5? \cdot (\{owns, \text{SINCE} \geq 2020?\} \cdot \langle locatedIn \cdot \text{Poland?} \rangle)^*(x, y)$

$q_4(x, y) := \text{Company?} \cdot employs \cdot friendsWith \cdot \text{GenZ?}(x, y), viewed \cdot \text{Job?} \cdot announce^-(y, x)$

Following the OMQA literature, we use the *homomorphism* or *walk* semantics for path queries [13], and in the presence of a TBox, adopt the *certain answer* semantics.

Definition 10. *Consider an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. A tuple $\vec{a}$ of individuals in $\mathcal{A}$ is called a* certain answer *to $q(\vec{x})$ over $(\mathcal{T}, \mathcal{A})$ if it is an answer to $q(\vec{x})$ in $PG(\mathcal{I})$ for every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$.*

3. Rewriting N2RPQs

We present a rewriting algorithm for N2RPQs with data tests. We assume in this section and the next a fixed *DL-Lite$_{PG}$* TBox $\mathcal{T}$. As usual, we can rely on the fact for every property graph $\mathcal{A}$ that is consistent with $\mathcal{T}$ there is a canonical model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ that gives exactly the certain answers to all N2RPQs. The construction of this model is standard, and given in the extended version of this paper [14]. For convenience, we may think of $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ as a set of *facts* $A(o)$ $r(o, o')$, $T(o)$ and $T(o, o')$, where o, o' are nodes, $A \in \mathbf{C}$, $r \in \mathbf{R}$ and $T \in \mathbf{T^D}$. We call a fact *explicit* if it is present in $\mathcal{A}$, and *implicit* otherwise. Note that all datatest facts are explicit, and that all implicit facts are introduced by the canonical model construction on the basis of other (explicit or implicit) facts.

We define a *skipping function* that takes an N2RPE as input and outputs a modified N2RPE that contains additional transitions which allow it to 'skip over' implicit facts, and instead traverse only the explicitly given graph. It draws inspiration from *loop computation* [15] and *clipping* [16, 17].

Definition 11 (skipping over N2RPEs). *Let $\mathcal{T}$ be a DL-Lite$_{PG}$ TBox. Given a N2RPE α with alphabet Σ_{PG}, we denote by $\text{skip}^{\mathcal{T}}(\alpha)$ the n-NFA obtained by exhaustively adding transitions as follows. In each rule, δ, s_i, s_j and s_k denote the transition function and states of one (arbitrary but fixed) automata that occurs (possibly nested) in α, and as usual, $r, s \in \overline{\mathbf{R}}$, $C, D \in \mathbf{C}$ and $T = \{t_1, \ldots, t_k\} \subseteq \mathbf{T^D}$.*

(1) if $r \sqsubseteq s \in \mathcal{T}$ and $(s_i, S, s_j) \in \delta$ with $s \in S$ (or $s^- \in S$), then add (s_i, R, s_j) to δ with $R = (S \setminus \{s\}) \cup \{r\}$ (or $R = (S \setminus \{s^-\}) \cup \{r^-\}$)

(2) if $C \sqsubseteq D \in \mathcal{T}$ and $(s_i, D?, s_j) \in \delta$, then add $(s_i, C?, s_j)$ to δ

(3) if $T \sqsubseteq D \in \mathcal{T}$ and $(s_i, D?, s_j) \in \delta$, then add, for each $t_\ell \in T$, a transition $(s'_\ell, t_\ell?, s'_{\ell+1})$ to δ, where $s_i = s'_1$, $s'_{k+1} = s_j$, and $s'_2, \ldots, s'_k$ are fresh states,

(4) if $T \sqsubseteq r \in \mathcal{T}$ and $(s_i, R, s_j) \in \delta$ with $r \in R$, then add (s_i, T, s_j) to δ

(5) if $\exists r.\top \sqsubseteq B$ and $(s_i, B?, s_j) \in \delta$, then add $(s_i, \langle \alpha_r \rangle, s_j)$ to δ for a fresh two-state NFA α_r with a single transition $\delta_r(s_r, \{r\}, s_f)$ between its only initial state s_r and its only final state s_f.

(6) for each $\exists r.\top \sqsubseteq \exists s.\top \in \mathcal{T}$ do:
*   a) if $\{(s_i, \{s\}, s_j), (s_j, \{s^-\}, s_k)\} \subseteq \delta$, then add $(s_i, \langle \alpha_r \rangle, s_k)$ to δ*
*   b) if $(s_i, \{s\}, s_k) \in \delta$ and $s_k \in F$, then add $(s_i, \langle \alpha_r \rangle, s_k)$ to δ*
*   for a fresh two-state NFA α_r with a single transition $\delta_r(s_i, \{r\}, s_k)$ between its only initial state s_i and its only final state s_k.*

*(7) for each CI $A \sqsubseteq \exists r.\top \in \mathcal{T}$ do:*
  a) if $\{(s_i, \{r\}, s_j), (s_j, \{r^-\}, s_k)\} \subseteq \delta$, then add (s_i, A, s_k) to δ
  b) if $\{(s_i, \{r\}, s_j)\} \subseteq \delta$ and $s_j \in F$, then add (s_i, A, s_j) to δ
(8) if $(s_i, \sigma, s_j), (s_j, \langle\alpha\rangle, s_k) \in \delta$ where $\sigma \in \Sigma_{PG}$ and $\sigma^- \in L(\alpha)$, then add (s_i, σ, s_k) to δ.

Intuitively, each rule application adds a transition that allows α to 'skip' an implicit fact f in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ and use instead a fact that participated in its creation. We illustrate $\mathsf{skip}^{\mathcal{T}}(\alpha)$ on a simple example.

Example 4. *Let us consider the TBox $\mathcal{T} = \{\exists r.\top \sqsubseteq A, A \sqsubseteq \exists r.B, B \sqsubseteq \exists r.C, A \sqsubseteq \exists t.A, \{p \geq 10?\} \sqsubseteq B, t \sqsubseteq s\}$ and the query $q(\vec{x}) = \alpha_1(x,y)$ with $\alpha_1 = (S_1, s_1, \delta_1, F_1)$, and the nested $\alpha_2 = (S_2, s_2, \delta_2, F_2)$ illustrated below; only the solid transitions are in the input N2RPE. The dotted transitions are those added to $\mathsf{skip}^{\mathcal{T}}(\alpha_1)$.*
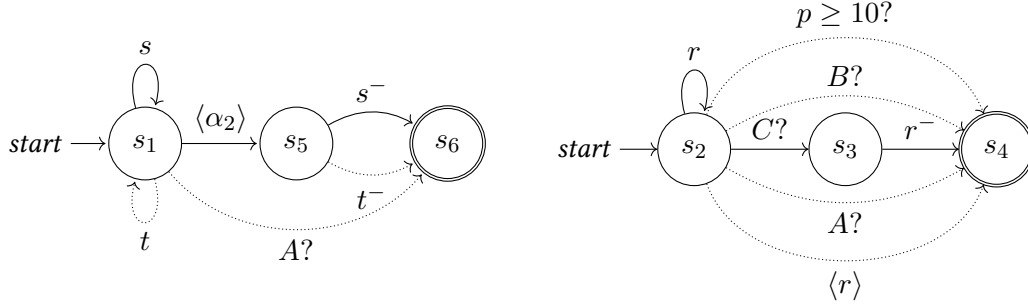
We can now show that $\mathsf{skip}^{\mathcal{T}}(\alpha)(x,y)$ is indeed a rewriting of $\alpha(x,y)$.

Lemma 2. *Let $\mathcal{T}$ be a DL-Lite$_{PG}$ TBox and α be an N2RPE. Then, for every property graph $\mathcal{A}$ and every pair of nodes o_s, o_f from $\mathcal{A}$, it holds that $(o_s, o_f) \in \alpha^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ iff $(o_s, o_f) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{A}}$.*

Proof sketch. For the *(if)* direction, we show that if an N2RPE α_{i+1} is obtained by applying a rule in Definition 11 to α_i, then $(o_s, o_f) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ implies $(o_s, o_f) \in \alpha_i^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. This is a simple rule-by-rule analysis.

For the *(only if)* direction, we rely on the fact that the canonical model construction naturally induces an ordering on the facts in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, where an implicit fact always has a strictly higher degree that the facts that participate in its creation. We then show that if $(o_s, o_f) \in \alpha_i^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and this can only be witnessed using some implicit fact f, then it is possible to apply some rule in such a way that, with the additional transition, $(o_s, o_f) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ can be witnessed using instead a fact f' of strictly lower degree than f. By applying the rules exhaustively, we eventually have that $(o_s, o_f) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ is witnessed using only facts of degree 0, that is, facts in $\mathcal{A}$, and hence $(o_s, o_f) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{A}}$. $\square$

4. Rewriting join-on-free CN2RPQs

It is now very easy to obtain an algorithm for join-on-free CN2RPQs. First we get rid af all non-answer variables using nesting.

Definition 12. *For a join-on-free CN2RPQ $q(\vec{x})$, we denote by $q^{\langle\exists\rangle}(\vec{x})$ the result of replacing each atom $\alpha(x,y)$ with $y \notin \vec{x}$ by $\langle\alpha\rangle(x)$, and each $\alpha(x,y)$ with $x \notin \vec{x}$ by $\langle\alpha^-\rangle(y)$.*

A simple inspection of the CN2RPQ semantics shows that the transformation preserves query answers.

Lemma 3. *Let $q(\vec{x})$ a join-on-free CN2RPQ. For every interpretation $\mathcal{I}$ and tuple of nodes $\vec{a}$, we have that $\vec{a}$ is an answer to $q(\vec{x})$ in $\mathcal{I}$ iff $\vec{a}$ is an answer to $q^{\langle\exists\rangle}(\vec{x})$ in $\mathcal{I}$.*

Moreover, the transformation to $q^{\langle\exists\rangle}$ removes all the non-answer variables. The rewriting of queries now straightforward, since we only need to apply $\mathsf{skip}^{\mathcal{T}}(\alpha)$ to each atom independently. We denote by $\mathsf{skip}^{\mathcal{T}}(q^{\langle\exists\rangle}(\vec{x}))$ the result of executing Algorithm 1 on $q(\vec{x})$ and $\mathcal{T}$, which applies $\mathsf{skip}^{\mathcal{T}}(\alpha)$ to each atom in $q^{\langle\exists\rangle}(\vec{x})$ and leaves all variables untouched.

Algorithm 1: `rewrite_jof_CNRPQ` – Rewriting join-on-free CN2RPQs with data tests

---

Input : CN2RPQ $q(\vec{x}), TBox\,\mathcal{T}$
Output: CN2RPQ q'

1 **function** `rewrite_jof_CNRPQ`(q):
2     $q = $`remove`$\exists$`Var`$(q)$
3   **while** $q \neq q'$ **do**
4        $q' = q$
5   **foreach** $\alpha(x,y) \in q$ **do**
6           $q' = q'[\alpha\backslash$`skipping`$(\alpha)]$
7   **return** q
8 **function** `remove`$\exists$`Var`(q):
9   **foreach** $\alpha(x,y) \in q$ **do**
10   **if** $y \notin \vec{x}$ **then**
11           $q = q[\alpha(x,y) \setminus \langle\alpha\rangle(x)]$
12   **else if** $x \notin \vec{x}$ **then**
13           $q = q[\alpha(x,y) \setminus \langle\alpha^-\rangle(y)]$
14   **return** q

Theorem 1. *Let $\mathcal{T}$ be a DL-Lite$_{PG}$ TBox and $q(\vec{x})$ be a join-on-free CN2RPQ. Then, for every ABox $\mathcal{A}$ and every tuple $\vec{a}$ of individuals from $\mathcal{A}$, it holds that $\vec{a}$ is a certain answer to $q(\vec{x})$ over $(\mathcal{T}, \mathcal{A})$ iff $\vec{a}$ is an answer to $\mathsf{skip}^{\mathcal{T}}(q^{\langle\exists\rangle}(\vec{x}))$ over $\mathcal{A}$.*

Proof. Given a *DL-Lite$_{PG}$* TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ and a join-on-free CN2RPQ $q(\vec{x})$. From For $(\Rightarrow)$ assume that $\vec{a}$ of individuals from $\mathcal{A}$ is a certain answer to $q(\vec{x})$ over $(\mathcal{T}, \mathcal{A})$. From Lemma 3 it holds that $\vec{a}$ is a certain answer to $q^{\langle\exists\rangle}(\vec{x})$ over $(\mathcal{T}, \mathcal{A})$, i.e., $\vec{a} = \mu(\vec{x})$ and $(\mu(x), \mu(y)) \in \alpha^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$ (or $(\mu(x) \in \alpha^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}})$ for each atom $\alpha(x,y)$ (or $\alpha(x)$) in $q^{\langle\exists\rangle}(\vec{x})$. By Lemma 2 we can imply that $(\mu(x), \mu(y)) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{A}}$ and $(\mu(x) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{A}}$ respectively. Thus, the claim holds that $\vec{a}$ is an answer to $\mathsf{skip}^{\mathcal{T}}(q^{\langle\exists\rangle}(\vec{x}))$ over $\mathcal{A}$. Since the claims in Lemma 2 and Lemma 3 hold in both directions, the proof for $(\Leftarrow)$ is analogous. $\square$

Example 5. *To demonstrate the full algorithm, let us consider the TBox $\mathcal{T}$ from Example 1 and the queries from Example 3. The first step of the algorithm is to remove the join-free non-answer variables of the input query, for example:*

$$q_2(x,y) := \{employs^-, \textsc{since} \leq 2023?\} \cdot friendsWith^* \cdot \textsc{born} \geq 2000?(x,y),$$
$$\langle friendsWith^- \cdot \langle\{viewed^-, \textsc{time} \leq 01.01.2023?\}\rangle\rangle(y)$$

Then, we apply the skipping function and obtain the following rewritten queries:

$$q_1(x,y) := (friendsWith^*|(friendsWith^-))^* \cdot employs^-(x,y), \langle\langle locatedIn \cdot (\mathsf{Opole}|\mathsf{Poland}?)\rangle \cdot$$
$$(\mathsf{Hiring}?|\langle announce\rangle) \cdot \mathsf{TechCompany}?\rangle(y)$$

$$q_2(x,y) := \{employs^-, \textsc{since} \leq 2023?\} \cdot (friendsWith^*|(friendsWith^-)^* \cdot \textsc{born} \geq 2000?(x,y),$$
$$\langle(friendsWith|friendsWith^-) \cdot \langle\{viewed^-, \textsc{time} \geq 01.01.2024?\}\rangle\rangle(y)$$

$$q_3(x,y) := \textsc{founded} \leq 2003? \cdot \textsc{revenue} \geq 10^5? \cdot (\{owns, \textsc{since} \geq 2010?\} \cdot$$
$$\langle locatedIn \cdot (\mathsf{Opole}?|\mathsf{Poland}?)))^*(x,y)$$

$$q_4(x,y) := \mathsf{Company}? \cdot employs \cdot (friendsWith|friendsWith^-) \cdot$$
$$(\mathsf{GenZ}?|\{\textsc{born} \geq 1997?, \textsc{born} \leq 2012?\})(x,y), viewed \cdot \mathsf{Job}? \cdot announce^-(y,x)$$

Now, the evaluation of these queries over the property graph from Example 2 gives us the following matches. For $q_1(y)$ we get two matches $\mu_1(x) = Bob$ $\mu_1(y) = SmartBees$ and $\mu_1'(x) = Alice$ $\mu_1'(y) = SmartBees$. For query q_2 there is one match $\mu_2(x) = Bob$ and $\mu_2(x) = Alice$. The third query returns two companies by the match $\mu_3(x) = SmartBees$ and $\mu_3(y) = nuCompany$. Finally, $\mu_4(x) = SmartBees$ and $\mu_4(y) = Alice$ is a valid match for query q_4.

Query	Number of Concepts	Rewriting Time [ms]	Evaluation Time [ms]
q_1	68	7237	18 084
q_2	67	3277	⏱
q_3	82	3805	6311
q_4	77	4557	5948
q_5	124	5769	11 307

Table 2
Results of experiments with data from the cognitive neuroscience domain. ⏱ indicates a timeout of 600 seconds.

5. Implementation & Experiments

As a proof-of-concept we implemented the algorithm from Section 4 in a publicly available prototype [18] and provide preliminary results in this section. The implementation makes use of the OWL API [19] to parse the input OWL ontology, ANTLR [20] for parsing the input CN2RPQ, and the Java library JgraphT [21] for the internal representation of n-NFAs. To the best of our knowledge the current version of the OWL2 standard [22] does not support to express data tests in role inclusions, i.e., axioms of the form $\{p \odot v \, ?\} \sqsubseteq r$. Therefore, the experiments do not consider these kind of axioms in the evaluation.

Ontology. As TBox we use the Cognitive Task Ontology (CogiTO) [23] and extended it by the axioms below (see [18] for this version of the ontology). There are multiple options to indicate that a participant is female; we have added axioms to cover all such cases, although we present only one representative example here. This ontology includes about 4686 concepts and 10 002 axioms, whereas 720 axioms are of the form ReadingTask $\sqsubseteq \exists has.$Read $\sqcap \exists has.$Language $-$ item. Note that CogiTO contains conjunction and qualified existentials on the left-hand side, which the prototype ignores.

$$\mathcal{T} = \Big\{ \, \{\textsc{License} = \text{"CC0" ?}\} \sqsubseteq \text{Reusable}, \qquad \{\textsc{BIDSVersion} = 1.2.0 \, ?\} \sqsubseteq \text{LatestBIDSVersion},$$
$$\{\textsc{gender} = \text{"f" ?}\} \sqsubseteq \text{Female}, \quad \{\textsc{Manufacturer} = \text{"Siemens"}\} \sqsubseteq \text{HighQuality} \qquad \Big\}$$

Data. The prototype produces a Cypher query, assuming that concepts in the ontology correspond to node labels in the database, and roles correspond to relationships (i.e., edge labels). The dataset for the experiments (see [18]) are from the domain of cognitive neuroscience [24] and is stored in a Neo4j database, consisting of 396 741 nodes and 2 870 405 relationships.

Queries. We handcrafted the following queries to provide preliminary results on the time required for rewriting and evaluating the queries.

$$q_1(x) := \text{Dataset? } \cdot \text{LatestBIDSVersion? } \cdot \text{Reusable? } \cdot has^* \cdot \text{Language} - \text{item}(x, y)$$
$$q_2(x) := \text{Dataset} \langle has^* \cdot \text{Female?} \rangle \cdot \langle has^* \cdot \text{Language} - \text{item?} \rangle (x, y)$$
$$q_3(x, y) := \text{Dataset} \cdot \langle has^* \cdot \text{HighQuality?} \rangle \langle has^* \cdot \text{Language} - \text{item?} \rangle \langle has^* \cdot \text{Read?} \rangle (x, y)$$
$$q_4(x, y) := \text{Dataset} \cdot \langle has^* \cdot \text{Female?} \rangle \cdot \langle has^* \cdot \text{HighQuality?} \rangle \cdot \langle has^* \cdot \text{Memorize?} \rangle \cdot \langle has^* \cdot$$
$$\text{Quantitative} - \text{value?} \rangle (x, y)$$
$$q_5(x, y) := \text{Dataset} \cdot \langle has^* \cdot \text{HighQuality} \rangle \cdot \langle has^* \cdot \text{Memorize?} \rangle \cdot \langle has^* \cdot \text{Language} - \text{item?} \rangle \cdot$$
$$\langle has^* \cdot \text{Read?} \rangle (x, y)$$

Setup. The experiments were executed on a virtual cluster node running Rocky Linux 8.10 with an AMD EPYC 7513 32-Core CPU @ 2.60 GHz and 400 GB RAM; Neo4j 5.18.1 runs on the same machine.

Results. In Table 2, we provide the results of our experimental evaluation. Since most of the transitions introduced by the rewriting are concept tests added due to a large concept hierarchy and existentials on the right-hand side, we give the number of concept tests as a proxy for the query size after the rewriting

(*Number of Concepts*). Additionally, we measure the time it takes to rewrite the queries (*Rewriting Time [ms]*) and to evaluate them with the Neo4j database (*Evaluation Time [ms]*), both averaged over 10 runs. The number of concepts, roughly indicating size and complexity of the rewriting, increases significantly from 3–5 concepts in the input to 67–124 in the rewritten queries. The rewriting times for all queries are within a reasonable range of a few seconds (3.3s-7.2s) suggesting that the rewriting is practically feasible. The evaluation times for the rewritten queries, with the exception of q_2 which timed out at the 600s threshold, remain under 20s. Given the size of the ontology, these numbers demonstrate an acceptable performance for the majority of our queries, although it also reveals that certain queries can result in significantly longer evaluation times. A possible reason for the outlier q_2 is a high number of Female instances in the dataset. We emphasize that the prototype is a very simple proof of concept and does not yet implement any optimizations. It has often been documented that queries obtained from rewriting algorithms tend to perform poorly: they introduce redundancy, nested disjunctions, and other complex subqueries that the engines are not optimized for. Thus, they require dedicated optimizations, which are often feasible given their somewhat predictable structure. We are confident that there is ample opportunity to optimize and improve the runtimes, and we plan to do so in the future.

6. Conclusion

In this paper, we present the first practical algorithm for rewriting N2RPQs and a significant subset of CN2RPQs, thereby capturing a substantial portion of Cypher and GQL. Our queries can access key values within path expressions. In our *DL-Lite$_{PG}$* ontology language, property value tests can be used to define concepts and roles in the ontology. The result is a highly flexible approach to ontology-mediated querying of property graphs that still allows for query rewriting into native graph database technologies. We have demonstrated the formal correctness of our approach and provided a proof-of-concept implementation that can rewrite Cypher queries to incorporate ontological knowledge and evaluate the rewritten queries using Cypher. This work brings us significantly closer to flexible, ontology-mediated querying of graph databases, and we look forward to exploring its advantages in real-world use cases.

To achieve a simple and practicable solution we focused on join-on-free CN2RPQs. This does not seem too limiting: even plain N2RPEs can already express many realistic examples that involve complex bidirectional navigation on the anonymous implicit facts; arbitrary conjunctions over the free variables make the query language even more powerful. We expect that real-world queries that require different regular paths to travel separately and join somewhere in the unnamed part of the canonical model will very rarely emerge in practice, if at all. Nevertheless, we plan to cover full CN2RPQs. We note that algorithms for full extended CN2RPQs in expressive DLs have been available for over a decade [3]. It is not difficult to adapt these algorithms to create one that is both correct and worst-case optimal for all CN2RPQs and *DL-Lite$_{PG}$* ontologies. However, its value seems limited since these techniques are highly impractical. In the extended version of this paper [14], we provide a sketch of one such technique, but it relies on automata-theoretic operations that can cause an exponential increase in size (e.g. the intersection of the word projection of N2RPEs). In contrast coming up with a practical, goal-oriented algorithm that can be used in practice seems to be a much bigger challenge, which we will address.

We plan to explore the potential of our technique for real-life OBDA systems with navigational capabilities, and to conduct experiments on systems supporting GQL as soon as they become available.

Acknowledgments

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Deutsch, N. Francis, A. Green, K. Hare, B. Li, L. Libkin, T. Lindaaker, V. Marsault, W. Martens, J. Michels, F. Murlak, S. Plantikow, P. Selmer, O. van Rest, H. Voigt, D. Vrgoc, M. Wu, F. Zemke, Graph pattern matching in GQL and SQL/PGQ, in: Z. G. Ives, A. Bonifati, A. E. Abbadi (Eds.), SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, ACM, 2022, pp. 2246–2258. doi:10.1145/3514221.3526057.

[2] M. Bienvenu, M. Ortiz, M. Simkus, Regular path queries in lightweight description logics: Complexity and algorithms, J. Artif. Intell. Res. 53 (2015) 315–374. doi:10.1613/jair.4577.

[3] M. Bienvenu, D. Calvanese, M. Ortiz, M. Simkus, Nested regular path queries in description logics, in: C. Baral, G. D. Giacomo, T. Eiter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014, AAAI Press, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/8000.

[4] D. Calvanese, T. Eiter, M. Ortiz, Regular path queries in expressive description logics with nominals, in: C. Boutilier (Ed.), IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009, 2009, pp. 714–720. URL: http://ijcai.org/Proceedings/09/Papers/124.pdf.

[5] B. Löhnert, N. Augsten, C. Okulmus, M. Ortiz, Towards practicable algorithms for rewriting graph queries beyond DL-Lite, in: The Semantic Web - 22nt International Conference, ESWC 2025, Portorož, Slovenia, June 1-5, 2025 (accepted for publication), Lecture Notes in Computer Science, Springer, 2025.

[6] M. M. Dimartino, A. Calì, A. Poulovassilis, P. T. Wood, Query rewriting under linear *EL* knowledge bases, in: M. Ortiz, S. Schlobach (Eds.), Web Reasoning and Rule Systems - 10th International Conference, RR 2016, Aberdeen, UK, September 9-11, 2016, Proceedings, volume 9898 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 61–76. URL: https://doi.org/10.1007/978-3-319-45276-0_6.

[7] N. Dragovic, C. Okulmus, M. Ortiz, Rewriting ontology-mediated navigational queries into cypher, in: O. Kutz, C. Lutz, A. Ozaki (Eds.), Proceedings of the 36th International Workshop on Description Logics (DL 2023) co-located with the 20th International Conference on Principles of Knowledge Representation and Reasoning and the 21st International Workshop on Non-Monotonic Reasoning (KR 2023 and NMR 2023)., Rhodes, Greece, September 2-4, 2023, volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3515/paper-9.pdf.

[8] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-lite family, Journal of Automated Reasoning 39 (2007) 385–429. doi:10.1007/s10817-007-9078-x.

[9] R. Angles, The property graph database model, in: D. Olteanu, B. Poblete (Eds.), Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018, volume 2100 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018. URL: https://ceur-ws.org/Vol-2100/paper26.pdf.

[10] A. Bonifati, G. H. L. Fletcher, H. Voigt, N. Yakovets, Querying Graphs, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2018. URL: https://doi.org/10.2200/S00873ED1V01Y201808DTM051. doi:10.2200/S00873ED1V01Y201808DTM051.

[11] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: G. Das, C. M. Jermaine, P. A. Bernstein (Eds.), Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018, ACM, 2018, pp. 1433–1445. URL: https://doi.org/10.1145/3183713.3190657. doi:10.1145/3183713.3190657.

[12] A. Artale, V. Ryzhikov, R. Kontchakov, Dl-lite with attributes and datatypes, in: L. D. Raedt,

C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, P. J. F. Lucas (Eds.), ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012, volume 242 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2012, pp. 61–66. URL: https://doi.org/10.3233/978-1-61499-098-7-61. doi:10.3233/978-1-61499-098-7-61.

[13] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, D. Vrgoc, Foundations of modern query languages for graph databases, ACM Comput. Surv. 50 (2017) 68:1–68:40. doi:10.1145/3104031.

[14] B. Löhnert, N. Augsten, C. Okulmus, M. Ortiz, Query rewriting for nested navigational queries over property graphs (with appendix), 2025. URL: https://gitlab.com/austrian-neurocloud/software/owl2cypher/-/releases/DL2025, accessed: 2025-07-23.

[15] M. Bienvenu, M. Ortiz, M. Simkus, Conjunctive regular path queries in lightweight description logics, in: F. Rossi (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, 2013, pp. 761–767. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6886.

[16] T. Eiter, M. Ortiz, M. Simkus, T. Tran, G. Xiao, Query rewriting for Horn-SHIQ plus rules, in: J. Hoffmann, B. Selman (Eds.), Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada, AAAI Press, 2012, pp. 726–733. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4931.

[17] B. Löhnert, N. Augsten, C. Okulmus, M. Ortiz, Towards practicable algorithms for rewriting graph queries beyond dl-lite, in: European Semantic Web Conference, Springer, 2025, pp. 342–361.

[18] B. Löhnert, N. Dragovic, Ontology-mediated querying for property graphs, 2025. URL: https://gitlab.com/austrian-neurocloud/software/owl2cypher/-/releases/DL2025, accessed: 2025-06-08.

[19] The OWL API, July 25, 2023. URL: https://owlcs.github.io/owlapi/.

[20] The OWL API, July 25, 2023. URL: https://www.antlr.org/.

[21] JGraphT, July 25, 2023. URL: https://jgrapht.org/.

[22] OWL2 Standard, July 25, 2023. URL: https://www.w3.org/TR/owl2-overview/.

[23] B. Löhnert, B. Engler, F. Hutzler, N. Augsten, Cognitive task ontology (COGITO), 2025. URL: https://gitlab.com/austrian-neurocloud/ontologies/cogito/-/releases/ESWC2025, accessed: 2025-03-27.

[24] A. Ravenschlag, M. Denissen, B. Löhnert, M. Pawlik, N. A. Himmelstoß, F. Hutzler, Effective queries for mega-analysis in cognitive neuroscience, in: G. Fletcher, V. Kantere (Eds.), Proceedings of the Workshops of the EDBT/ICDT 2023 Joint Conference, Ioannina, Greece, March, 28, 2023, volume 3379 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3379/CoMoNoS_2023_id252_Mateusz_Pawlik.pdf.

A. Proofs for Section 3 (Rewriting N2RPQs)

Definition 13 (Canonical Model). *Consider a DL-Lite$_{PG}$ TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. Let $\mathcal{I}_0$ be the initial interpretation with $\mathcal{A}$ the corresponding property graph. Then, we construct the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ by exhaustively applying the following rules to $\mathcal{I}_0$.*

(Ch1) If $A \sqsubseteq B \in \mathcal{T}$, $v \in A^{\mathcal{I}_i}$ and $v \notin B^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{v\}$.

(Ch2) If $\exists r.\top \sqsubseteq B \in \mathcal{T}$, $v \in (\exists r.\top)^{\mathcal{I}_i}$ and $v \notin B^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{v\}$.

(Ch3) If $\exists r.\top \sqsubseteq \exists s.\top \in \mathcal{T}$, $v \in (\exists r.\top)^{\mathcal{I}_i}$ and $v \notin (\exists s.\top)^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with a fresh w in $\Delta^{\mathcal{I}_{i+1}}$. In case $s \in \mathbf{R}$ then $s^{\mathcal{I}_{i+1}} = s^{\mathcal{I}_i} \cup \{(v,w)\}$, otherwise $s^{\mathcal{I}_{i+1}} = r^{\mathcal{I}_i} \cup \{(w,v)\}$.

(Ch4) If $A \sqsubseteq \exists r.\top \in \mathcal{T}$, $v \in A^{\mathcal{I}_i}$ and $v \notin (\exists r.\top)^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with a fresh w in $\Delta^{\mathcal{I}_{i+1}}$. In case $r \in \mathbf{R}$ then $r^{\mathcal{I}_{i+1}} = r^{\mathcal{I}_i} \cup \{(v,w)\}$, otherwise $r^{\mathcal{I}_{i+1}} = r^{\mathcal{I}_i} \cup \{(w,v)\}$.

(Ch5) If $r \sqsubseteq s \in \mathcal{T}$, $(v,w) \in r^{\mathcal{I}_i}$ and $(v,w) \notin s^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with $s^{\mathcal{I}_{i+1}} = s^{\mathcal{I}_i} \cup \{(v,w)\}$ in case $r \in \mathbf{R}$, otherwise $s^{\mathcal{I}_{i+1}} = s^{\mathcal{I}_i} \cup \{(w,v)\}$.

(Ch6) If $\top \sqsubseteq B \in \mathcal{T}$, $v \in \top^{\mathcal{I}_i}$ and $v \notin B^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{v\}$.

(Ch7) If $\top \sqsubseteq s \in \mathcal{T}$, $(v,w) \in \top^{\mathcal{I}_i}$ and $(v,w) \notin s^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is $\mathcal{I}_i$ with $s^{\mathcal{I}_{i+1}} = s^{\mathcal{I}_i} \cup \{(v,w)\}$.

Lemma 2. *Let $\mathcal{T}$ be a DL-Lite$_{PG}$ TBox and α be an N2RPE. Then, for every property graph $\mathcal{A}$ and every pair of nodes o_s, o_f from $\mathcal{A}$, it holds that $(o_s, o_f) \in \alpha^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ iff $(o_s, o_f) \in \mathsf{skip}^{\mathcal{T}}(\alpha)^{\mathcal{A}}$.*

Proof. For the *(if)* direction, assume that an n-NFA α_{i+1} was obtained by applying one of the rules in Definition 11 to α_i. In the following we show for each of the rules that it holds if $(o_0, o_n) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$, then also $(o_0, o_n) \in \alpha_i^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$.

- Assume that $\alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$ was obtained by applying rule (1) with $r \sqsubseteq s \in \mathcal{T}$ and $(s_i, S, s_{i+1}) \in \delta_i$ with $s \in S$. Then, $\alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$ contains the transition (s_i, R, s_{i+1}) with $R = (S \setminus \{s\}) \cup \{r\}$. By assumption it holds that $(o_i, o_{i+1}) \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and from Definition 13 it follows that $(o_i, o_{i+1}) \in s^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

- The proof for the rules (2)-(4) works equivalent to (1).

- Assume that $\alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$ was obtained by rule (5) for $\exists r.\top \sqsubseteq B$ and $(s_i, B?, s_{i+1}) \in \delta_i$. Then, we add a transition with a nested NFA α that contains a single transition with r. By Definition 7 it holds that $o_i = o_{i+1}$ and there is a pair (o_i, o''), such that $(o_i, o'') \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. From Definition 13 it follows that $o_i \in B^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

- The argumentation for (6) is equivalent to the argumentation of (5) and (7) combined.

- Assume that $\alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$ was obtained by applying rule (7) with $A \sqsubseteq \exists r.\top \in \mathcal{T}$. By Definition 7 it holds that there is a pair (o_i, o') where (o_i, o'), such that $o_i \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. From the canonical model construction in Item 4 it holds that $o_i \in (\exists r.B)^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$. For each of the cases below it holds that $(o_i, o_i) \in \alpha_i^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$.
  a) $\{(s_i, r, s_{i+1}), (s_{i+1}, r^-, s_{i+2})\} \subseteq \delta$
  b) $\{(s_i, r, s_{i+1})\} \subseteq \delta$ and $s_{i+1} \in F$

- In case that α_{i+1} was obtained by rule (8), then for each $(s_i, \sigma, s_j), (s_j, \langle\alpha\rangle, s_k) \in \delta$ where $\sigma \in \Sigma_{PG}$ and $\sigma^- \in L(\alpha)$ we add the transition (s_i, σ, s_k) to δ. Since $\sigma^- \in L(\alpha)$ and there has to be a precedent σ before α, there is a trivial matching for the NRPE α, i.e., if $(o_1, o_n) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, then also $(o_0, o_n) \in \alpha_i^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

($\Rightarrow$) In order to show completeness we first introduce the notion of degree, that comes from the canonical model construction: We denote individuals or pair of individuals in the extension of concepts and roles as *facts*, i.e., $v \in B^{\mathcal{I}_j}$ or $(v,w) \in r^{\mathcal{I}_j}$. A fact that occurs for the first time in $\mathcal{I}_j$ has degree j, and a set of facts gets the sum of the degrees of its elements. The degree of sequence $o_0 \sigma_1 o_1 \ldots \sigma_n o_n$ witnessing $(o_0, o_n) \in \alpha^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}(x,y)$ is the minimal degree of a set of facts that is sufficient to witness all conditions in Definition 7, including the nested automata. Note that there may be more than one way to witness not only the automata, but even for a fixed (top level) sequence, we may have choices, but we don't care. We set the degree to be the smallest, and then every added transition will allow us to witness the same sequence but with a strictly smaller degree.

Having the notion of degree for a pair in place we show by induction that in each rewriting step we are "moving up" the part of the canonical model that is induced by the axioms in $\mathcal{T}$. In the following we show that for each of the rules in Definition 13 there is a corresponding rule in Definition 11 producing a n-NFA α_{i+1} from α_i such that the degree of the accepted sequence is strictly smaller.

(Ch1) Consider $A \sqsubseteq B \in \mathcal{T}$ and $o_k \in A^{\mathcal{I}_j}$, $o_k \notin B^{\mathcal{I}_j}$ and $o_i \in B^{\mathcal{I}_{j+1}}$ by construction of the canonical model. Consider the sequence $(o_1 \ldots o_k B o_k \ldots o_n)$ that $(o_1, o_n) \in \alpha_i^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$. In the rewriting we apply rule applies rule (2) to obtain α_{i+1} from α_i, where we add for each transition $(s_l, B?, s_{l+1})$ a new transition $(s_l, A?, s_{l+1})$, thus $(o_1 \ldots o_k A o_k \ldots o_n)$ will be accepted by $\alpha_{i+1}^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$, for which the degree is strictly smaller then for $(o_1 \ldots o_k B o_k \ldots o_n)$.

(Ch2) Given $\exists r.\top \sqsubseteq B \in \mathcal{T}$ and by construction of the canonical model $o_k \in (\exists r.\top)^{\mathcal{I}_j}$, $o_k \notin B^{\mathcal{I}_j}$ $o_k \in B^{\mathcal{I}_{j+1}}$. Consider a sequence of the form $(o_1 \ldots o_k B o_k \ldots o_n)$ such that $(o_1, o_n) \in \alpha_i^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$. In skipping we apply rule (5) to obtain α_{i+1} and add for each $(s_i, B?, s_j) \in \delta$ the transition $(s_i, \langle \alpha_r \rangle, s_j)$ to δ, where α_r is a fresh two-state NFA with a single transition $(s_r, \{r\}, s_f)$. Since $o_k \in (\exists r.\top)^{\mathcal{I}_j}$ there has to be some sequence $(o_k r o_k')$, such that $(o_k, o_k') \in \alpha_r^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$ and $(o_1, o_n) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$.

(Ch3) For this case consider $\exists r.\top \sqsubseteq \exists t.\top \in \mathcal{T}$, then by construction of the canonical model we have $o_k \in (\exists r.\top)^{\mathcal{I}_j}$, $o_k \notin (\exists s.\top)^{\mathcal{I}_j}$ and $(o_k, o_k') \in s^{\mathcal{I}_{j+1}}$ for a fresh o_k. Further, there are two possible ways for a sequence to pass the s edge, either the sequence is of the form $(o_1 \ldots o_k s o_k' s^- o_k \ldots o_f)$ or it ends with the s edge, i.e., $(o_s \ldots o_{n-1} s o_n)$. For both cases suppose that $(o_1, o_n) \in \alpha_i^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$. First, we get rid of trivially satisfied nested expressions by rule (8). Then, we obtain α_{i+1} by applying rule (6) and add for each consecutive transitions $(s_l, s, s_{l+1}), (s_{l+1}, s^-, s_{l+2})$ in α_i a transition $(s_l, \langle \alpha_r \rangle, s_{l+2})$ to α_{i+1} and for each $\{(s_l, \{s\}, s_{l+1})\}$ such that s_{l+1} is a final state, we add $(s_l, \{s\}, s_{l+1})$ to α_{i+1}, where α_r is a fresh two-state NFA with a single transition $(s_r, \{r\}, s_f)$. Since $o_k \in (\exists r.\top)^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$ it holds that there is some o_k'' such that the sequence $(o_k r o_k'') \in \alpha_r^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$. Thus, it holds that $(o_1, o_n) \in \alpha_{i+1}^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$ and the degree of the sequence is lower. The same holds for the inverse case $s \notin \mathbf{R}$.

(Ch4) The argumentation for the case with $A \sqsubseteq \exists r.\top \in \mathcal{T}$ and rule (7) is similar to the case (Ch3), but instead of adding a nested NFA we simply add a transition with A.

(Ch5) For $r \sqsubseteq s \in \mathcal{T}$ rule 1 applies and the reasoning is similar to the case with (Ch1).

(Ch6) For $T \sqsubseteq B \in \mathcal{T}$ we apply rule (3) and the argumentation is again equivalent to (Ch3).

(Ch7) For $T \sqsubseteq s \in \mathcal{T}$ the argumentation is similar as for role inclusions by applying rule 4.

$\square$

A.1. Algorithm Sketch for full CN2RPQs

In this section we provide a sketch for an algorithm that reduces full CN2RPQs to a union of CN2RPQs. For this we first introduce the notion of a *s-join* of n-NFAs.

Definition 14. *Let $\mathbf{A}_1 \ldots \mathbf{A}_n$ be n-NFAs with the same alphabet and let $S = \{s_1 \ldots s_n\}$ be states with $s_i \in \mathbf{A_i}$ and $1 \leq i \leq n$, i.e., one state from each automaton. We define the s-join of $\mathbf{A_1} \ldots \mathbf{A_n}$ as a tuple of $n+1$ automata $\mathbf{A'}_1 \ldots \mathbf{A'}_n \mathbf{A}_{rest}$ as follows:*

- *Each $\mathbf{A'}_i$ is obtained from $\mathbf{A}_i$ by making s_i the only final state*
- *$\mathbf{A}_{rest}$ is the intersection of all $\mathbf{A}_i$ by making s_i the only initial state*

There are different ways of taking the intersection of automata, but in either case, this step is exponential. In the following we provide the instructions to get rid of existential variables that might be mapped in the anonymous part of the canonical model.

Given a full CN2RPQ $q(\vec{x})$, then we exhaustively apply the following rules:

(1) pick an existential join variable $y \notin \vec{x}$ and take all atoms $\alpha(x, y)$ or $\alpha(y, x)$ in q

(2) for atoms $\alpha(y, x)$, where y occurs in the first place, take the inverse $\alpha^-(x, y)$. The goal of this step is to have all atoms with y in the form of $\alpha_i(x_i, y)$

(3) Choose one state s_i of each automaton α_i. Let S be the selected states, and replace all the atoms by the s-join using one intermediate variable y_s, that is, replace all atoms where y occurs by
$$\mathbf{A}'_1(x_1, y_s) \wedge \cdots \wedge \mathbf{A}'_n(x_n, y_s) \wedge \mathbf{A}'_{rest}(y_s, y).$$

If we apply this rule to all non-answer variables in all possible ways, we can get rid of the original existential variables. The fresh y_s are existential, but we can assume w.l.o.g. that they are mapped to named objects, so the algorithm is complete for them.

The complete query rewriting algorithm for full CN2RPQs under a *DL-Lite$_{PG}$* TBox can then be derived by rewriting each of the queries in the output union with the Algorithm 1 for join-on-free CN2RPQs.