Rewriting Ontology-Mediated Navigational Queries into Cypher

Nikola Dragovic[1], Cem Okulmus[2] and Magdalena Ortiz[1,2]

[1]TU Wien, Austria
[2]Umeå University, Sweden

Abstract
The ontology-based data access (OBDA) paradigm has successfully grown over the last decade as a powerful means to access data from possibly diverse and incomplete sources, using a domain ontology as a mediator. The ability to query generic graph-structured data is often highlighted as an advantage of OBDA, but in practice, existing solutions do not allow to access data in popular graph database management systems (DBMS) (e.g., Neo4j) that adopt the so-called 'property graph' data model and support dedicated query languages such as Cypher.

Towards overcoming this major limitation, we propose a technique for ontology-mediated querying (OMQ) of property graphs. We tailor a suitable query language that supports path navigation in a form that can be naturally expressed in Cypher and other important graph query languages. It keeps the data complexity of query evaluation tractable even under trail semantics and is sufficient for our motivating use case in the autonomous driving domain. We address the semantic gap between the traditional path semantics adopted by most works on graph databases, and the trail semantics used in Cypher, and identify cases where both semantics coincide. To our knowledge, OMQs with trail semantics had not been addressed before. We develop a rewriting algorithm for queries mediated by DL-Lite ontologies that enables query answering using plain Cypher. The experimental evaluation of our proof-of-concept prototype on a sample set of use case queries reveals that the approach is promising, and can be a stepping stone to making OBDA applicable to data stored in graph DBMS.

Keywords
Ontology-based data access, Graph databases, Property graphs, Query rewriting

1. Introduction

The ontology-based data access (OBDA) paradigm, also known as virtual knowledge graphs (VKGs), has steadily grown over the last decade to establish itself as a powerful way to access data from possibly diverse and incomplete sources. It has been successfully exploited in various application domains [1], and state-of-the-art systems such as Ontop [2] continue to be developed and improved. In OBDA, a domain ontology provides a high-level vocabulary for formulating queries, and captures domain knowledge that can be used to infer implicit answers from incomplete data. The central problem in OBDA is answering ontology-mediated queries (OMQs), that is, computing the answers to a given query not just from the plain data, but taking into account also the facts that can be inferred using the ontology.

DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece
okulmus@cs.umu.se (C. Okulmus); magdalena.ortiz@cs.umu.se (M. Ortiz)
0000-0002-7742-0439 (C. Okulmus); 0000-0002-2344-9658 (M. Ortiz)

The ability to integrate heterogeneous sources and query generic graph data is often highlighted as an advantage of the OBDA paradigm, and while it can be successfully deployed to query RDF graphs, no solutions so far allow directly querying data stored in popular graph DBMS (GDBMS) that adopt the so-called *property graph* data model. This is a major limitation since such graph databases have gained huge popularity this century in scores of domains and they are widely deployed in practice. GDBMS support dedicated query languages for graph-structured data. For example, Neo4j is one of the most popular GDBMS nowadays, and its query language is Cypher [3]. There are dozens of GDBMS in the market, some of them supporting query languages related to Cypher. The standardisation of a query language for graphs called GQL, intended to serve as the basis and reference point for all query languages for GDBMS, is an ongoing project at the International Organisation for Standardisation (ISO), and the publication of the standard is expected soon [4]. Cypher, as the most widely adopted language for GDBMS, is included and fully supported by GQL and is likely to remain a dominant dialect of GQL for many years.

The fundamental feature of all query languages for GDBMS is the presence of *navigational features* for traversing paths flexibly [5], which allows for expressing, for example, *reachability* queries. This is not present in the query languages supported in current OBDA systems, which commonly target relational data. Even if the data is modelled as a graph, these systems cannot navigate graph structures and their evaluation is not tuned for graph data.

Typically, OBDA systems take as input a so-called *conjunctive query (CQ)*, which is essentially the most widely used *select-project-join* fragment of relational algebra; in its logical form, it is written as a positive existential conjunction of atoms that may share *join variables* [6]. The language of choice for writing the ontology is OWL 2 QL, the profile of the Web Ontology Language OWL standard tailored for efficient query answering [7], and the preferred approach for OMQ answering is *query rewriting*. Here a query mediated by an OWL 2 QL ontology is transformed using the ontology axioms into a new query in a standard query language, with no mediating ontology, that provides the same answers when evaluated over the original data sources [8]. Given a CQ and an OWL 2 QL ontology as an input, the rewriting produces a *union* (or disjunction) of CQs, that is, a UCQ. UCQs can be naturally expressed in simple fragments of SQL [6] and of SPARQL [9], the query language for RDF [10], which enables the evaluation of OMQs using off-the-shelf DBMS and SPARQL end-points. State-of-the-art OBDA systems implement this standard setting, supporting ontology-mediated CQs over data stored in relational and RDF data sources.

The main goal of this paper is to provide such a rewriting approach for OMQs with OWL 2 QL ontologies, but using a graph query language that is supported by GDBMs, such as Cypher, as both source query language, and as target for the rewriting. Navigational features have been considered in the OMQ literature, where algorithms for rewriting navigational queries in the presence of ontologies and tight complexity bounds for their evaluation are known [11, 12], but the existing theory results have never made it to practice. These works consider *conjunctive 2-way regular path queries (C2RPQ)*, the generalisation of CQs with path navigation, the language of choice for theoretical works on accessing graph databases. However, it is well-known that practical graph GDBMS do not always support C2RPQs, and sometimes adopt a different semantics. In particular, unlike C2RPQs, Cypher uses a *trail* semantics where no edge of the graph can occur twice on a path, and imposes restrictions on the bidirectional navigation of paths [3]. This gap has hindered the application of the results for C2RPQs to enable querying GDBMS in OBDA, and until now, only impracticable algorithms intended for showing theoretical results had been devised [11, 12].

The main contributions of this paper can be summarised as follows:

1. We propose a class of queries that allows for path navigation in property graphs which is sufficient for our motivating use case. Our query language is closely related to C2RPQs, but tailored to be expressible in Cypher, to support succinct query rewritings, and to have tractable data complexity even under trail semantics.
2. We consider both the traditional path semantics of C2RPQs and the trail semantics of Cypher; to our knowledge, OMQs with trail semantics had not been considered until now. We show that for our query language, query answers under both semantics coincide.
3. We present a rewriting algorithm for our query language in the presence of OWL 2 QL ontologies. We provide conditions that guarantee that the queries rewritten by the algorithm can be expressed in Cypher, and evaluated directly over GDBMS.
4. We implemented a simple prototype of the approach and evaluated it on real-world use-case data. In particular, we considered queries designed for scenario-based safety assessment for autonomous vehicles and evaluated them over an industrial dataset. We obtained promising results that suggest that our work may be a stepping stone towards practicable OBDA over GDBMS.

2. Preliminaries

In this section we recall the property graph data model. We also introduce *DL-Lite*$_R$, which is the language of OWL 2 QL defined as a description logic [13].

We consider a vocabulary consisting of countably infinite and pairwise disjoint sets $\mathsf{N_I}$ of *individual names*, $\mathsf{N_C}$ of *concept names* and $\mathsf{N_R}$ of *role names*, $\mathsf{N_E}$ of *relationship names* and K of *property keys*. Moreover, we assume a fixed set of *data type domains* $\mathbf{D}_1, \dots, \mathbf{D}_n$; each $\mathbf{D}_i$ consists of a *value domain* D_i and a fixed set of binary predicates representing binary relations over D_i. In our examples, we typically use integers and strings as data types. For integers, we use the usual comparison predicates $\leq, \geq, \neq, =$, and for the strings datatype we use the binary relations $\mathit{equal}(s_1, s_2)$, $\mathit{substring}(s_1, s_2)$, and $\mathit{prefix}(s_1, s_2)$.

Property Graphs

Our definition of property graphs follows the one given in [3]. A *labelled multigraph* is a tuple $\langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l \rangle$, where N is the set of *nodes* and E is the set of *edges*; $N \neq \emptyset$ and $N \cap E = \emptyset$. The functions $\mathsf{src}\colon E \mapsto N$ and $\mathsf{tgt}\colon E \mapsto N$ assign each edge its source and target node, respectively. We have two total labelling functions: $c_r\colon E \mapsto \mathsf{N_R}$ assigns to each edge a role name, and $c_l\colon N \mapsto 2^{\mathsf{N_C}}$ assigns to each node a set of concept names. A *property graph*

$$\mathcal{G} = \langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda \rangle$$

extends a labelled multigraph with a partial function $\lambda : (N \cup E) \times \mathsf{K} \to \bigcup_{1 \leq i \leq n} D_i$ that maps pairs (u, k) with u a node or edge of $\mathcal{G}$ and k a property key, to a value in a datatype value domain D_i.

Let $\mathcal{G} = \langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda\rangle$ and $\mathcal{G}' = \langle N', E', \mathsf{src}', \mathsf{tgt}', c_r', c_l', \lambda'\rangle$ be property graphs. A *homomorphism from $\mathcal{G}$ to $\mathcal{G}'$* is a function h that maps each $v \in N$ to some $h(v) \in N'$ and each $u \in E$ to some $h(u) \in E'$ so that

1. for every $v \in N$, $c_l(v) \subseteq c_l'(h(v))$ and if $\lambda(v,k)$ is defined for some k, then $\lambda(v,k) = \lambda'(h(v),k)$; and
2. for every $u \in E$, $c_r(u) = c_r'(h(u))$, $\mathsf{src}(u) = \mathsf{src}'(h(u))$, $\mathsf{tgt}(u) = \mathsf{tgt}'(h(u))$, and if $\lambda(u,k)$ is defined for some k, then $\lambda'(h(u),k) = \lambda(u,k)$.

We call $\mathcal{G}$ a *subgraph* of $\mathcal{G}'$ if $N \subseteq N'$, $E \subseteq E'$, and the identity on $N \cup E$ is a homomorphism from $\mathcal{G}$ to $\mathcal{G}'$.

2.1. Ontologies

An ontology is a set of axioms that encapsulates terminological knowledge of our domain. Here we write ontologies in the description logic called *DL-Lite$_R$* [8]. We define the set of roles $\mathsf{N}_\mathsf{R}^{\pm} = \mathsf{N}_\mathsf{R} \cup \{s^- \mid s \in \mathsf{N}_\mathsf{R}\}$. If $r = s^-$ for $s \in N_R^{\pm}$, then r^- denotes s. For all roles r, we call role r^- the *inverse* of r, and vice-versa. *Concepts* in *DL-Lite$_R$* take the form A or $\exists r$, where $A \in \mathsf{N}_\mathsf{C}$ and $r \in \mathsf{N}_\mathsf{R}^{\pm}$. We use possibly subindexed r and B to denote roles and concepts, respectively. *Axioms* take one of four forms:

$$B_1 \sqsubseteq B_2,\ B_1 \sqsubseteq \neg B_2,\ r_1 \sqsubseteq r_2,\ r_1 \sqsubseteq \neg r_2.$$

A set of axioms $\mathcal{T}$ is called an ontology or a *TBox*. In this paper *datasets* (called *ABoxes* in DL jargon) are given in the form of a property graph $\langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda\rangle$ where $N \subseteq \mathsf{N}_\mathsf{I}$ and $E \subseteq \mathsf{N}_\mathsf{E}$, and both sets are finite.

Paired with a TBox $\mathcal{T}$, a dataset $\mathcal{A}$ is associated to a set of *models*, which intuitively are property graphs that may extend $\mathcal{A}$ and satisfy the axioms in $\mathcal{T}$.

Formally, in this paper we define an *interpretation* $\mathcal{I}$ as a property graph

$$\langle \Delta_V^{\mathcal{I}}, \Delta_E^{\mathcal{I}}, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda\rangle$$

where $\Delta_V^{\mathcal{I}} \neq \emptyset$ and $\Delta_E^{\mathcal{I}} \neq \emptyset$. We say that $\mathcal{I}$ is a *model of dataset* $\mathcal{A}$ if it contains $\mathcal{A}$ as a subgraph. Note that we are thus making the *standard name assumption.*

The interpretation function $\cdot^{\mathcal{I}}$ for concept and role names is determined by the labelling functions c_l and c_r. Note that for a role name r, we let $r^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$, as usually done in description logic interpretations where $\Delta_E^{\mathcal{I}}$ is not present. For each concept name $A \in \mathsf{N}_\mathsf{C}$ and each role name $r \in \mathsf{N}_\mathsf{R}$, we define

$$\begin{aligned} A^{\mathcal{I}} =& \{v \in \Delta_V^{\mathcal{I}} \mid A \in c_l(v)\} \\ r^{\mathcal{I}} =& \{(v,w) \mid \exists u \in \Delta_E^{\mathcal{I}} : \mathsf{src}(u) = v, \mathsf{tgt}(u) = w, c_r(u) = r\} \end{aligned}$$

Then the interpretation of all concepts and roles is defined as usual:

$$\begin{array}{ll} (\neg B)^{\mathcal{I}} = \Delta_V^{\mathcal{I}} \setminus B^{\mathcal{I}} & (r^-)^{\mathcal{I}} = \{(v,w) \mid (w,v) \in r^{\mathcal{I}}\} \\ (\exists r)^{\mathcal{I}} = \{v \mid \exists w \in \Delta_V^{\mathcal{I}} : (v,w) \in r^{\mathcal{I}}\} & (\neg r)^{\mathcal{I}} = (\Delta_V^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus r^{\mathcal{I}} \end{array}$$

An interpretation $\mathcal{I}$ *satisfies an axiom* $\gamma \sqsubseteq \delta$ if $\gamma^{\mathcal{I}} \subseteq \delta^{\mathcal{I}}$, and we call $\mathcal{I}$ *a* model *of a TBox* $\mathcal{T}$ if $\mathcal{I}$ satisfies every axiom in $\mathcal{T}$. A dataset $\mathcal{A}$ *is consistent with* a TBox $\mathcal{T}$ if a model of $\mathcal{A}$ and $\mathcal{T}$ exists.

Canonical model When TBoxes are written in *DL-Lite$_R$*, every consistent dataset can be extended into a model in a canonical way using a technique called the *chase*; this also applies in our setting. As usual, we start form $\mathcal{A}$ and add nodes, edges and labels to satisfy all the *positive* axioms of $\mathcal{T}$, i.e., those that do not have $\neg$ on the right-hand-side.

The *canonical model* $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of a *DL-Lite$_R$* TBox $\mathcal{T}$ and a dataset $\mathcal{A}$ is built inductively as follows. First we set $\mathcal{I}_0 = \mathcal{A}$. Then the following rules are exhaustively applied:

- If $B \sqsubseteq A \in \mathcal{T}$ with $A \in \mathsf{N_C}$, $v \in B^{\mathcal{I}_i}$ and $v \notin A^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is obtained from $\mathcal{I}_i$ by setting $c_l(v) := c_l(v) \cup \{A\}$.

- If $B \sqsubseteq \exists r \in \mathcal{T}, v \in B^{\mathcal{I}_i}$ and $v \notin (\exists r)^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is obtained from $\mathcal{I}_i$ by adding a fresh w to $\Delta_V^{\mathcal{I}}$ and a fresh u to $\Delta_E^{\mathcal{I}}$. In case $r \in \mathsf{N_R}$, we set $\mathsf{src}(u) := v, \mathsf{tgt}(u) := w$ and $c_r(u) = r$ for $\mathcal{I}_{i+1}$. Otherwise, we set $\mathsf{src}(u) := w, \mathsf{tgt}(u) := v$ and $c_r(u) = r^-$.

- If $r_1 \sqsubseteq r \in \mathcal{T}$ with $r \in \mathsf{N_R^{\pm}}$, $(v, w) \in r_1^{\mathcal{I}_i}$ and $(v, w) \notin r^{\mathcal{I}_i}$, then $\mathcal{I}_{i+1}$ is obtained from $\mathcal{I}_i$ by adding a fresh element u to $\Delta_E^{\mathcal{I}}$. In case $r \in \mathsf{N_R}$, we set $\mathsf{src}(u) := v, \mathsf{tgt}(u) := w$ and $c_r(u) = r$ for $\mathcal{I}_{i+1}$. Otherwise, we set $\mathsf{src}(u) := w, \mathsf{tgt}(u) := v$ and $c_r(u) = r^-$.

We assume fairness in the application of the rules i.e., every applicable rule is eventually applied. The *canonical model* $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is defined as the limit of the sequence $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_n$. Note that in general, the canonical model can be infinite. The following result is proved in the standard way.

Lemma 1. *Let $\mathcal{T}$ be a DL-Lite$_R$ TBox and $\mathcal{A}$ a dataset consistent with $\mathcal{T}$. Then $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$, and every model of $\mathcal{A}$ and $\mathcal{T}$ can be homomorphically embedded into $\mathcal{I}_{\mathcal{T},\mathcal{A}}$.*

3. Navigational Queries for Property Graphs

We now introduce our query language for property graphs, which allows for path navigation in the style of C2RPQs. For a detailed definition C2RPQs, we refer to [12]. C2RPQs are conjunctions of unary atoms of the form $A(x)$ and binary atoms of the form $\rho(x, y)$, where A is a concept name and ρ a regular expression over the alphabet of (possibly inverse) roles. Cypher also supports regular paths, but when we restrict it to the fragment that can be written in logic as a conjunction of atoms analogous to C2RPQs, we find some important differences. In particular, Cypher imposes directionality constraints when matching paths that effectively mean that a regular path that uses the Kleene star can contain either only role names, or only inverse role names. For example, a query such as $q(x) \leftarrow [r_1 \cup r_2^-]^*(x, y)$ can not be syntactically expressed in Cypher. Also, Cypher does not allow more than one 'step' in the scope of the Kleene star, thus a query such as $q(x) \leftarrow [r_1 \cdot r_2]^*(x, y)$ can not be syntactically expressed in Cypher.

Our query language restricts regular paths so that they are naturally expressible in Cypher. We also add some useful features to CQs that are typically not present in C2RPQs, but which are easy to support in Cypher. We extend unary atoms to take a disjunction of concepts rather than a single concept. As we will see below, this allows for rewritings that are exponentially more succinct in some cases. Additionally, our query language allows querying property values by means of *test atoms*. Recall that each datatype has a fixed set of binary predicates representing binary relations over the data domain. For example, the integers may come with the usual comparison predicates $\leq, \geq, \neq, =$, and the strings datatype with binary relations `equal`, `substring`, `prefix`, etc. Property keys and data predicates are used in Boolean tests:

$$\texttt{x\_position} \geq 450 \vee (\texttt{x\_position} \geq 300 \wedge \texttt{movingDirection} = \text{`}\texttt{right}'\text{)}.$$

where we check that an object has a horizontal position of at least 450, or the position is at least 300 and the object is moving in the direction `right`.

Definition 1 (Navigational property-graph queries (NPGQs))**.** *Let v be a value in a datatype domain $\mathbf{D}$, let $\odot$ be a binary predicate in $\mathbf{D}$, and let $k \in \mathsf{K}$. The expression $k \odot v$ is called an* atomic data test*, and a Boolean combination of atomic data tests (built using $\wedge$, $\vee$, and $\neg$) is called a* data test*. We define a* navigational property-graph query (NPGQ) *as an expression of the form $q(\vec{x}) = \exists y.\phi(\vec{x}, \vec{y})$ where ϕ is a conjunction of* atoms *of the forms:*

$$T(x) \qquad T(x,y) \qquad (A_1 \cup \cdots \cup A_n)(x) \qquad (r_1 \cup \cdots \cup r_n)(x,y) \qquad (r_1 \cup \cdots \cup r_n)^*(x,y)$$

where each A_i is a concept name in $\mathsf{N_C}$, each r_i a role name in $\mathsf{N_R^\pm}$, T is a data test, and $x, y \in (\vec{x} \cup \vec{y} \cup \mathsf{N_I})$. The atoms of the first two forms are called test atoms*, and the remaining atoms are* relational atoms*. We call an atom of the last form a* star atom*, and say that it is* pure *if it contains only role names, or only inverse role names, that is, if either $\{r_1, \ldots, r_n\} \subseteq N_R$ or $\{r_1^-, \ldots, r_n^-\} \subseteq N_R$. An NPGQ q is called* pure *if all its star atoms are pure.*

3.1. Path and trail semantics

To define the semantics of NPGQs, we first define paths and trails.

Definition 2. *Consider a property graph $\mathcal{G} = \langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda \rangle$. We let $E^\pm = E \cup \{u^- \mid u \in E\}$, and for every u in $E^\pm$, we let $\mathsf{src}(u^-) = \mathsf{tgt}(u)$, $\mathsf{tgt}(u^-) = \mathsf{src}(u)$ and $c_r(u^-)$ is the inverse role of $c_r(u)$. A* path from a node v to a node w *in a property graph $\mathcal{A}$ is a sequence $u_1 u_2 \ldots u_n$ of edges $u_i \in E^\pm$, where $\mathsf{src}(u_1) = v$, $\mathsf{tgt}(u_n) = w$ and for each $1 \leq i < n$, $\mathsf{tgt}(u_i) = \mathsf{src}(u_{i+1})$. We use ϵ to denote the empty path, that is, the path with $n = 0$ from a node v to itself. A* trail from v to w *is a path $u_1 u_2 \ldots u_n$ from v to w where for each $i \neq j$, we have that $u_i \notin \{u_j, u_j^-\}$ i.e., no edge occurs more than once. For ρ of the form $(r_1 \cup \cdots \cup r_n)^*$ or $(r_1 \cup \cdots \cup r_n)$, we use $L(\rho)$ to denote the set of paths $u_1 u_2 \ldots u_n$ such that $\{s_1, \ldots, s_n\} \subseteq \{r_1 \cup \cdots \cup r_n\}$, where $s_i = c_r(u_i)$. Note that $\epsilon \in L(\rho)$ for every ρ.*

Note that every trail is a path, but not every path is a trail. Now we can define path and trail matches for NPGQs.

Definition 3. *[Path and trail matches] Consider a property graph $\mathcal{G} = \langle N, E, \mathsf{src}, \mathsf{tgt}, c_r, c_l, \lambda \rangle$. Let $o \in N \cup E$. A test $k \odot v$ is said to be* true *at o if $\lambda(o, k)$ is defined and belongs to the same data domain $\mathbf{D}_i$ as v, and the relation denoted by $\odot$ holds between $\lambda(o, k)$ and v. The truth of data tests T is interpreted as expected, and we write $o \in [\![T]\!]_\mathcal{G}$ if T is true at o in graph $\mathcal{G}$; we may omit the graph $\mathcal{G}$ if it is clear from the context.*

Let $q(\vec{x})$ be a query and π a mapping from the individuals and variables occurring in q to nodes of $\mathcal{G}$. We call π a path match *for q in $\mathcal{G}$ if N contains all individuals occurring as terms in q and:*

1. *$\pi(c) = c$ for each $c \in \mathsf{N_I}$;*
2. *for each atom $(A_1 \cup \cdots \cup A_n)(x)$ in q, $\{A_1, \ldots, A_n\} \cap c_l(\pi(x)) \neq \emptyset$;*
3. *for each atom $T(x)$ in q, $\pi(x) \in [\![T]\!]$;*
4. *for each atom $(r_1 \cup \cdots \cup r_n)(x, y)$ or $(r_1 \cup \cdots \cup r_n)^*(x, y)$, there exists a path $p \in L(\rho)$ from $\pi(x)$ to $\pi(y)$, where $\rho = r_1 \cup \cdots \cup r_n$ or $\rho = (r_1 \cup \cdots \cup r_n)^*$, respectively; and*
5. *for each atom $T(x, y)$, there exists $e \in E$ with $e \in [\![T]\!]$, $src(e) = \pi(t)$, $tgt(e) = \pi(y)$.*

If additionally the path p in item 4 is a trail, we call π a trail match. *A tuple $\vec{a} \subseteq N$ is a (trail) answer to a query $q(\vec{x}) = \exists y.\phi(x, y)$ in $\mathcal{G}$ if there exists a (trail) match π for $\vec{x}$ such that $\pi(\vec{x}) = \vec{a}$.*

Since all trails are paths, the trail answers are a subset of the path answers. For C2RPQs and many related languages, this containment is strict [4]. However, the regular paths in NGPQs are restricted in such a way that every path answer is also a trail answer, and hence trail and path answers coincide.

Proposition 1. *Let $\mathcal{G}$ be a property graph and q an NPGQ. The answers to q over $\mathcal{G}$ under trail semantics are also path answers.*

Proof (sketch). We show that every path match is also a trail match. Consider an arbitrary path match π for q in $\mathcal{G}$. To show that π is also a trail match, we argue that for every star atom $(r_1 \cup r_2 \cup \cdots \cup r_n)^*(x, y)$, there is a trail p' witnessing item 4 in Definition 3, that is, p' is a trail from $\pi(x)$ to $\pi(y)$ and $p' \in L(\rho)$, where $\rho = (r_1 \cup r_2 \cup \cdots \cup r_n)^*$. We know that such a path $p = u_1 \ldots u_n$ exists, as π is a path match, and if p is not a trail then we can drop any repeated sequences in it to obtain a trail p'. Since p' only uses edges from $\{u_1, \ldots, u_n\}$, it follows from $p = L(\rho)$ that $p' = L(\rho)$ also holds. $\square$

Answering C2RPQs under path semantics is in NL in data complexity [14], so by Proposition 1 we have:

Proposition 2. *The query answering problem for NPGQs is NL complete in data complexity, under both path and trail semantics.*

This is good news, since under trail semantics C2RPQs are intractable in general [15]. We note that Proposition 2 can also be inferred (without Proposition 1) from the trichotomy in [15].

NPGQs and Cypher Cypher cannot express some NPGQs such as $[r_1 \cup r_2^-]^*(x, y)$, but *pure* NPGQs inherit Cypher's restrictions on directional navigation and they can be easily expressed. Due to space constraints, the Cypher translation is to be found in the full version of this paper.

3.2. Semantics of Ontology-Mediated NPGQs

As is usually done for OMQs, we adopt the *certain answer semantics.*

Definition 4 (Certain answer). *Let $\mathcal{A}$ be a dataset, $\mathcal{T}$ a TBox, and $q(\vec{x})$ an NPGQ. A tuple $\vec{a} \subseteq N$ is a* (certain) answer *to $(q(\vec{x}), \mathcal{T})$ over $\mathcal{A}$ if it is an answer in each model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$.*

Note that previous works on OMQs had only considered path answers, which admit a natural certain answer semantics. We can adopt certain answers for the trail semantics thanks to Proposition 1, but this does not extend to other navigational query languages.

The certain answers coincide with the answers in the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. This follows from Lemma 1, the preservation of path matches under homomorphisms, and Proposition 1.

Lemma 2. *Let $\mathcal{A}$ be a dataset, $\mathcal{T}$ a TBox, and $q(\vec{x})$ an NPGQ. A tuple $\vec{a} \subseteq N$ is a (certain) answer to $(q(\vec{x}), \mathcal{T})$ over $\mathcal{A}$ if it is an answer in in the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$.*

4. Query Rewriting Algorithm

In this section, we a present a sound and complete rewriting algorithm for NPGQs. It transforms an NPGQ q mediated by a *DL-Lite*$_R$ ontology $\mathcal{T}$ into a plain NPGQ $q^{\mathcal{T}}$ that can be evaluated over the data alone and that gives exactly the same set of answers.

The algorithm is not very different to the classical PerfectRef [8]: it applies axioms of $\mathcal{T}$ in a right-to-left fashion to take into account every way in which a query atom could be implied. Since NPGQs allow for unions of concepts in unary atoms, we can avoid one of the most common causes for exponential growth of the rewritten queries in PerfectRef. For instance, in the presence of a TBox that contains $A_1 \sqsubseteq A_2, B_1 \sqsubseteq B_2$ a query that contains the atoms $A_2(x), B_2(y)$ can be rewritten replacing them with $(A_1 \cup A_2)(x), (B_1 \cup B_2)(y)$, rather than considering all combinations of $A_i(x), B_j(y)$.

For the treatment of the regular expressions, we exploit the fact that the query only needs to navigate in the anonymous part of the canonical model in one direction. That is, for every atom $\rho(x, y)$ in q and every match π, we can safely assume that π is a trail and no node in the canonical model is visited more than once. This 'unidirectionality' of paths spares us from needing the sophisticated techniques for treating 2-way paths that are usual for navigational queries in the presence of ontologies (e.g., the so-called *loop computation* [12]). Instead, we only introduce three novel rewriting rules, which allow us to reason about the way in which the paths witnessing star atoms may overlap, and whose application can be a prerequisite to the application of the usual rewriting rules to the star atoms.

Let α be an atom. If it is of the form $(A_1 \cup A_2 \cup \dots A_n)(x)$, we denote by $concepts(\alpha)$ the set of all concept names $\{A_1, \dots, A_n\}$ that occur in it, and if it is of the form $(s_1 \cup s_2 \cup \dots s_n)(x, y)$, then $roles(\alpha) \subseteq N_R^{\pm}$ denote the roles occurring in it. For a union of roles $\rho = r_1 \cup \dots \cup r_n$, we define $\rho^- = r_1^- \cup \dots \cup r_n^-$. We view queries as sets of atoms and, for convenience, we use $\alpha \mathrel{\overline{\in}} q$ to mean that either $\alpha \in q$, or $\alpha = \rho(x, y)$ and $\rho^-(y, x) \in q$. Recall that a query variable is *unbound* if it occurs exactly once in the query and it is not an answer variable. As usual, we denote unbound variables by '_', but each occurrence is a fresh variable, and must be denoted by a unique name if it becomes bound.

The algorithm ensures that atoms are closed under the concept and role name inclusions of $\mathcal{T}$. For a TBox $\mathcal{T}$, we call $S \subseteq N_R^{\pm}$ *$\mathcal{T}$-saturated* if $r_1 \in S$ whenever $r_2 \in S$ and either $r_1 \sqsubseteq r_2 \in \mathcal{T}$ or $r_1^- \sqsubseteq r_2^- \in \mathcal{T}$. The *$\mathcal{T}$-saturation* of S, $sat_{\mathcal{T}}(S)$, is the smallest $\mathcal{T}$-saturated set containing S. Similarly, a set M of concept names *$\mathcal{T}$-saturated* if $A \in M$ and $A' \sqsubseteq A \in \mathcal{T}$ imply $A' \in \mathcal{T}$, and the $\mathcal{T}$-saturation of M, denoted $sat_{\mathcal{T}}(M)$, is the smallest *$\mathcal{T}$-saturated* set containing M. We

Table 1
Axiom application rules to obtain $apply(q, \gamma)$ (where $i \in \{1, \ldots, , n\}$).

If $\gamma = \exists r \sqsubseteq A_i$	and there is $\alpha = (A_1 \cup \cdots \cup A_n)(x) \in q$	replace α in q by $[sat_{\mathcal{T}}(\{r\})](x, \_)$.
If $\gamma = \exists r \sqsubseteq \exists r_i$	and there is $\alpha = (r_1 \cup \cdots \cup r_n)(x, \_) \overline{\in} q$	replace α in q by $[sat_{\mathcal{T}}(\{r\})](x, \_)$.
If $\gamma = A \sqsubseteq \exists r_i$	and there is $\alpha = (r_1 \cup \cdots \cup r_n)(x, \_) \overline{\in} q$	replace α in q by $[sat_{\mathcal{T}}(\{A\})](x)$.

call an atom α *$\mathcal{T}$-saturated* if $concepts(\alpha)$ or $roles(\alpha)$ (for unary or binary α, respectively) is $\mathcal{T}$-saturated. If it is clear from the context, we may omit $\mathcal{T}$ and just talk about *saturated atoms.*

Our novel query rewriting algorithm, applies exhaustively the following transformations.

Definition 5. *Consider a DL-Lite$_R$ TBox $\mathcal{T}$ and an NPGQ q. We define the following rules:*

1. **Saturation:** *$saturate(q, \mathcal{T})$ denotes the result of replacing each atom in q by its $\mathcal{T}$-saturation.*
2. **Axiom application:** *By $apply(q, \gamma)$ we denote the result of applying an axiom γ to q using the rules in Table 1. Note that this may add an unbound variable to the query.*
3. **Reduction:** *$reduce(q, \alpha_1, \alpha_2)$ is the result of applying the most general unifier of α_1 and α_2.*
4. **Concatenation:** *Let $\beta_1 \overline{\in} q$ and $\beta_2 \overline{\in} q$ be atoms such that $roles(\beta_2) \subseteq roles(\beta_1)$, β_1 is a star atom, and β_1, β_2 share a common term. Then we obtain $concatenate(q, \beta_1, \beta_2)$ as follows: If β_1 has terms (x, y) and β_2 has terms (x, z), then the terms of β_1 in q are replaced by (z, y). This amounts to placing the atom β_2 right before β_1. If β_1 has terms (x, y) and β_2 has terms (z, y), then the terms of β_1 in q are replaced by (x, z). This amounts to placing the atom β_2 right after β_1.*
5. **Merging:** *Let $\beta_1 \overline{\in} q$ and $\beta_2 \overline{\in} q$ be atoms with $roles(\beta_1) \cap roles(\beta_2) \neq \emptyset$ and such that both atoms have terms (x, y) (in that order). Then $merge(q, \beta_1, \beta_2)$ is obtained by replacing in q both atoms by*

- *$[roles(\beta_1) \cap roles(\beta_2)^*](x, y)$ if both β_1 and β_2 are star atoms, and*
- *$[roles(\beta_1) \cap roles(\beta_2)](x, y)$ otherwise.*

6. **Dropping:** *Let $\beta \overline{\in} q$ be of the form $(r_1 \cup \cdots \cup r_n)^*(x, \_)$ or $(r_1 \cup \cdots \cup r_n)^*(\_, x)$ and such that x occurs in another atom in case it is an answer variable. Then $drop(q, \alpha) = q \setminus \alpha$.*

We use $Rewrite(q, \mathcal{T})$ to describe the set of queries that are produced by applying all of the above rules exhaustively, and accumulating all transformed new queries.

Example 1. *Suppose we are given a TBox $\mathcal{T} = \{\exists t^- \sqsubseteq \exists p\}$ and also a query q_1, defined as $q_1(x) \leftarrow (r \cup s \cup t)^*(x, y), p(z, x), s^-(y, z)$. To illustrate the query transformations, we state three example queries, produced via successive application of these transformations.*

$q_2(x) \leftarrow (r \cup s \cup t)^*(x, y), t^-(z, x), s^-(y, z)$	*from $apply(q_1, \exists t^- \sqsubseteq \exists p)$*
$q_3(x) \leftarrow (r \cup s \cup t)^*(z, y), t(x, z), s^-(y, z)$	*from $concatenate(q_2, (r \cup s \cup t)^*(x, y), t(x, z))$*
$q_4(x) \leftarrow s(z, y), t(x, z)$	*from $merge(q_3, s(z, y), (r \cup s \cup t)^*(z, y))$*

Note that in the creation of q_3 we make use of the fact that $t(x, z) \overline{\in} q_2$, which follows from $t^-(z, x) \in q_2$, and analogously we have that $s(z, y) \overline{\in} q_3$, which follows from $s^-(y, z) \in q_3$.

Note that the rewriting focuses on the atoms without data tests. Since *DL-Lite$_R$* ontologies cannot assert property key values for existentially quantified objects, atoms with data tests can only be matched in the canonical model to nodes and edges that exist already in the input dataset.

Hence these atoms remain untouched in the rewritten query, except for possible applications of the reduce step, which could *bind* variables shared with test atoms.

The rewriting algorithm we have presented is sound and complete.

Theorem 1. *Let q be an NPGQ, $\mathcal{T}$ a DL-Lite$_R$ TBox and $\mathcal{A}$ a property graph. Then $\vec{a}$ is a certain answer to q over $(\mathcal{T}, \mathcal{A})$ iff $\vec{a}$ is an answer to q' in $\mathcal{A}$ for some $q' \in \mathit{Rewrite}(q, \mathcal{T})$.*

The pseudo-code of the query rewriting algorithm will be in the full version of this paper.

Termination and Complexity The query rewriting procedure terminates. In a nutshell, all transformations make the query smaller except for the axiom application, which can introduce fresh variables, but a bound on the latter can be shown using essentially the same arguments as for the standard PerfectRef. The rewriting does not depend on the data and hence it only needs constant time for any given $\mathcal{T}$ and q. From this and Proposition 2, we get:

Theorem 2. *The query answering problem for NPGQs mediated by DL-Lite$_R$ ontologies is NL complete in data complexity, under both path and trail semantics.*

Evaluating OMQs in Cypher Even if the original input query is pure, the rewriting can add (non)-inverse roles to star atoms and result in a NPGQ that is not pure. We introduce a sufficient condition for the purity to be preserved during the rewriting, which guarantees that we can evaluate ontology-mediated NPGQs using Cypher.

Definition 6 (NPGQ compliance)**.** *Let q be an NPGQ. A TBox $\mathcal{T}$ is q-*compliant *if there is no star atom whose saturation contains both role names and inverse role names. That is, for every atom of the form $S^*(t, t') \in q$, either $sat_{\mathcal{T}}(S) \subseteq N_R$, or $\{r^- \mid r \in sat_{\mathcal{T}}(S)\} \subseteq N_R$.*

Proposition 3. *If q is a pure NPGQ and it is $\mathcal{T}$-compliant, then every $q' \in \mathit{Rewrite}(q, \mathcal{T})$ is a pure NPGQ.*

5. NPGQ Rewriting in a Use Case

This work was largely motivated by a use case from the automated driving industry, specifically *scenario-based safety assessment models* for autonomous vehicles, where driving scenarios for safety testing are retrieved from real-world data. There are two large open datasets used for such purposes: the nuScenes [16] and Lyft [17] datasets. A company in the autonomous driving industry stores both datasets in the same Neo4j graph database. The data is organized in scenes, which contain a number of samples that are temporally ordered, and the state of objects in a scene can change within an unknown time frame. Hence, we need to navigate the temporal graph of a scene to answer queries about objects that change in time. The two datasets are similar, but use different vocabularies. For example, one dataset labels instances with Bus, while the other labels them either RigidBus or BendyBus. We used a *DL-Lite$_R$* ontology to define a unified vocabulary for querying. It contains, for example, axioms such as BendyBus $\sqsubseteq$ Bus and RigidBus $\sqsubseteq$ Bus. In collaboration with a company expert, we developed by hand a test set of five representative queries. All but one of them needed navigation along paths of unbounded

Table 2
The results of our experiments on the nuScenes and Lyft dataset. To show the effectiveness of our novel 'union rewriting', we compare it with a simpler one, called 'PerfectRef*' below. All times in seconds.

Query	Union Rewr. size	Rewr. time	Output size	Exec. time for Union Rewr.	PerfectRef* rewr. size	Exec. time for PerfectRef*
Q1	1	0.003	234	0.039	8	**0.001**
Q2	33	1.112	751	**0.705**	880	111.071
Q3	33	0.922	264	**0.538**	880	109.209
Q4	11	1.152	10	**0.195**	1056	136.767
Q5	11	1.168	8	**0.222**	1056	143.516

length, but could be easily expressed as NPGQs. The navigation used only *temporal relations* in the data such as NEXT, and the compliance between queries and TBox was trivial. The queries vary from a simple query with only one atom (Q1) to queries which contain binary atoms with Kleene star (Q2-Q4). As an example, we provide one of the queries, Q5, here in full:

$$Q_5(x) \leftarrow \mathsf{pedestrian}(x), \mathsf{OF}(y,x), \mathsf{HAS}(y,z), \mathsf{pedestrian\_stationary}(z), \mathsf{NEXT}(y,w), \\ \mathsf{NEXT}^*(w,yp), \mathsf{HAS}(yp,zp), \mathsf{pedestrian\_moving}(zp)$$

We developed a prototype implementation of our rewriting technique and tested it on a sample dataset provided by the mentioned company, as a Neo4j database with metadata of the nuScenes and Lyft datasets. The source code for our implementation has been made publicly available[1]. The dataset consists of 91,891 nodes and 254,555 distinct relations. In order to test the potential feasibility of our approach, we rewrote the five queries and evaluated them over the data as Cypher queries. The results of the evaluation are seen in Table 2, on the right.

We were also curious about the impact of atoms with concept unions (such as $[A \cup B](x)$) for handling the subclass hierarchy. We compared our rewriting, which uses this union, with a naive version that just extends the classical PerfectRef with the additional rules for navigational atoms (called PerfectRef* in the table). The sizes of the rewritings produced by both approaches and their evaluation time are shown in Table 2. The very significant improvements suggest that some features of Cypher may be useful for query rewriting even for plain CQs, and they could be leveraged for more efficient OMQ evaluation with less optimisation effort.

6. Conclusion

Ontologies are useful for querying incomplete data. We have extended the ontology-mediated querying paradigm to property graphs, using as both input query language and as target of the rewritings a navigational query language which is expressible in Cypher. For answering queries, we have developed a novel rewriting algorithm based on PerfectRef. Finally, we have demonstrated the usefulness of exploiting Cypher for compact rewritings on a concrete use case. We plan to continue working on rewriting techniques for more expressive graph query languages, helping pave the way to enabling OBDA to support real graph databases.

[1]https://github.com/nikdra/omq-pg

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. It was also partially supported by the Austrian Science Fund (FWF) project P30360 and P30873.

References

[1] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: IJCAI, ijcai.org, 2018, pp. 5511–5519.

[2] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. G. Kalayci, L. Ding, J. Corman, B. Cogrel, D. Calvanese, E. Botoeva, The virtual knowledge graph system ontop, in: J. Z. Pan, V. A. M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II, volume 12507 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 259–277. URL: https://doi.org/10.1007/978-3-030-62466-8_17. doi:10.1007/978-3-030-62466-8_17.

[3] 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.

[4] N. Francis, A. Gheerbrant, P. Guagliardo, L. Libkin, V. Marsault, W. Martens, F. Murlak, L. Peterfreund, A. Rogova, D. Vrgoc, A researcher's digest of GQL (invited talk), in: ICDT, volume 255 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 1:1–1:22.

[5] 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. URL: https://doi.org/10.1145/3104031. doi:10.1145/3104031.

[6] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995. URL: http://webdam.inria.fr/Alice/.

[7] D. Calvanese, J. Carroll, G. D. Giacomo, J. Hendler, I. Herman, B. Parsia, P. F. Patel-Schneider, A. Ruttenberg, U. Sattler, M. Schneider, OWL 2 Web Ontology Language Profiles, World Wide Web Consortium (W3C), 2009. URL: https://www.w3.org/TR/owl2-profiles/.

[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, J. Autom. Reason. 39 (2007) 385–429. URL: https://doi.org/10.1007/s10817-007-9078-x. doi:10.1007/s10817-007-9078-x.

[9] World Wide Web Consortium, Sparql 1.1 query language, 2013. URL: https://www.w3.org/TR/sparql11-query/.

[10] World Wide Web Consortium, RDF 1.1 Primer, World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/rdf11-primer/.

[11] M. Bienvenu, D. Calvanese, M. Ortiz, M. Simkus, Nested regular path queries in description logics, in: KR, AAAI Press, 2014.

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

[13] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017.

[14] M. P. Consens, A. O. Mendelzon, Graphlog: a visual formalism for real life recursion, in: PODS, ACM Press, 1990, pp. 404–416.

[15] W. Martens, M. Niewerth, T. Trautner, A Trichotomy for Regular Trail Queries, in: C. Paul, M. Bläser (Eds.), 37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France, volume 154 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 7:1–7:16. URL: https://doi.org/10.4230/LIPIcs.STACS.2020.7. doi:10.4230/LIPIcs.STACS.2020.7.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuscenes: A multimodal dataset for autonomous driving, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 11618–11628. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Caesar_nuScenes_A_Multimodal_Dataset_for_Autonomous_Driving_CVPR_2020_paper.html. doi:10.1109/CVPR42600.2020.01164.

[17] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, P. Ondruska, One thousand and one hours: Self-driving motion prediction dataset, in: J. Kober, F. Ramos, C. J. Tomlin (Eds.), 4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA, volume 155 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 409–418. URL: https://proceedings.mlr.press/v155/houston21a.html.