Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

KRDB Research Centre
for Knowledge and Data

Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009

KRDB Research Centre Technical Report:

Enriching Ontology-based Data Access with Provenance (Extended Version)

Diego Calvanese[1], Davide Lanti[1], Ana Ozaki[1], Rafael Peñaloza[2], Guohui Xiao[1]

Affiliation	1: KRDB Research Centre, Free University of Bozen-Bolzano, Italy 2: University of Milano-Bicocca, Italy
Corresponding	Diego Calvanese: calvanese@inf.unibz.it Davide Lanti: davide.lanti@unibz.it Ana Ozaki: ana.ozaki@unibz.it Rafael Peñaloza: rafael.penaloza@unibz.it Guohui Xiao: xiao@inf.unibz.it
Keywords	Description Logic, Provenance Semirings, Ontology-based Data Access, Complexity, Query Answering
Number	KRDB19-02
Date	June 03, 2019
URL	http://www.inf.unibz.it/krdb/

Abstract

Ontology-based data access (OBDA) is a popular paradigm for querying heterogeneous data sources by connecting them through mappings to an ontology. In OBDA, it is often difficult to reconstruct why a tuple occurs in the answer of a query. We address this challenge by enriching OBDA with provenance semirings, taking inspiration from database theory. In particular, we investigate the problems of *(i)* deciding whether a provenance annotated OBDA instance entails a provenance annotated conjunctive query, and *(ii)* computing a polynomial representing the provenance of a query entailed by a provenance annotated OBDA instance. Differently from pure databases, in our case these polynomials may be infinite. To regain finiteness, we consider idempotent semirings, and study the complexity in the case of DL-Lite ontologies. We implement Task *(ii)* in a state-of-the-art OBDA system and show the practical feasibility of the approach through an extensive evaluation against two popular benchmarks.

Enriching Ontology-based Data Access with Provenance (Extended Version)

Diego Calvanese[1] , **Davide Lanti**[1] , **Ana Ozaki**[1] , **Rafael Peñaloza**[2] and **Guohui Xiao**[1]
[1]KRDB Research Centre, Free University of Bozen-Bolzano, Italy
[2]University of Milano-Bicocca, Italy

Abstract

Ontology-based data access (OBDA) is a popular paradigm for querying heterogeneous data sources by connecting them through mappings to an ontology. In OBDA, it is often difficult to reconstruct why a tuple occurs in the answer of a query. We address this challenge by enriching OBDA with provenance semirings, taking inspiration from database theory. In particular, we investigate the problems of *(i)* deciding whether a provenance annotated OBDA instance entails a provenance annotated conjunctive query, and *(ii)* computing a polynomial representing the provenance of a query entailed by a provenance annotated OBDA instance. Differently from pure databases, in our case these polynomials may be infinite. To regain finiteness, we consider idempotent semirings, and study the complexity in the case of DL-Lite$_\mathcal{R}$ ontologies. We implement Task *(ii)* in a state-of-the-art OBDA system and show the practical feasibility of the approach through an extensive evaluation against two popular benchmarks.

1 Introduction

Ontology-based data access (OBDA) [Xiao *et al.*, 2018] is by now a popular paradigm which has been developed in recent years to overcome the difficulties in accessing and integrating legacy data sources. In OBDA, users are provided with a high-level conceptual view of the data in the form of an ontology that encodes relevant domain knowledge. The concepts and roles of the ontology are associated via declarative mappings to SQL queries over the underlying relational data sources. Hence, user queries formulated over the ontology can be automatically rewritten, taking into account both ontology axioms and mappings, into SQL queries over the sources.

When issuing a query, in many settings it is crucial to know not only its result but also *how* it was produced, *how many* different ways there are to derive it, or how *dependent* it is on certain parts of the data [Senellart, 2017; Zimmermann *et al.*, 2012; Buneman and Kostylev, 2010]. To address these issues, which are of importance already for plain relational database management systems (RDBMSs), *provenance semirings* [Green *et al.*, 2007; Green and Tannen, 2017] were introduced as an abstract tool to record and track provenance information; that is, to keep track of the specific database tuples that are responsible for deriving an answer tuple, and of additional information associated to them. In OBDA, determining provenance is made even more challenging by the fact that answers are affected by implicit consequences derived through ontology axioms, and by the use of mappings. Such elements come indirectly into play in query rewriting, hence provenance information must be reconstructed from the rewritten queries used in the answering process [Borgida *et al.*, 2008].

In this work, we start from the semiring approach introduced for RDBMSs, and extend it to the full-fledged OBDA setting. To do so, we assume that not only database tuples are annotated with a label representing provenance information (e.g., the data source or the relation in which the tuple is stored), but also mappings and ontology axioms. Then, our task is to derive which combinations of these labels lead to the answer of a query. Such information is expressed through a *provenance polynomial*, as illustrated in the following example.

Example 1. Let $\mathsf{Mayors}[\mathsf{Person},\mathsf{City}]$ be a database relation with the tuples $(\mathsf{Renier},\mathsf{Venice})$ and $(\mathsf{Brugnaro},\mathsf{Venice})$, annotated with (sources) p and q, respectively. Assume two mappings $\mathsf{City}(Y) \leftarrow \mathsf{Mayors}(X,Y)$ and $\mathsf{headGov}(X,Y) \leftarrow \mathsf{Mayors}(X,Y)$, annotated with m and n, respectively. The mappings and the database *induce* *(i)* two times the DL assertion $\mathsf{City}(\mathsf{Venice})$, one annotated with $p \times m$ and one with $q \times m$, *(ii)* the DL assertion $\mathsf{headGov}(\mathsf{Renier},\mathsf{Venice})$, annotated with $p \times n$, and *(iii)* the assertion $\mathsf{headGov}(\mathsf{Brugnaro},\mathsf{Venice})$, annotated with $q \times n$.

Now consider the inclusion $\exists\mathsf{headGov} \sqsubseteq \mathsf{Mayor}$ annotated with s. The answer true to the Boolean conjunctive query $\exists x.(\mathsf{Mayor}(x))$ can be derived using this inclusion and any of the last two DL assertions. This information can be expressed through the provenance polynomial $((p \times n) + (q \times n)) \times s$.◁

In our OBDA setting, concept and role inclusions of the ontology affect query results, as illustrated in Example 1. By annotating the inclusions and the mappings, in addition to the tuples, we can distinguish which inclusions and mappings were involved in the derivation of a query result. This differs from the approach proposed for attributed DL-Lite$_\mathcal{R}$ [Bourgaux and Ozaki, 2019], where the inclusions are used to express constraints on the provenance information.

We investigate the problems of *(i)* deciding whether a provenance annotated OBDA instance entails a provenance annotated conjunctive query (CQ), and *(ii)* computing a provenance polynomial of a CQ entailed by a provenance annotated OBDA instance. Differently from plain databases, in our case these polynomials may be infinite. To regain finiteness, we consider idempotent semirings, and study the complexity for DL-Lite$_\mathcal{R}$ ontologies [Calvanese *et al.*, 2007]. We implement task *(ii)* in the state-of-the-art OBDA system *Ontop* [Calvanese *et al.*, 2017], and show the practical feasibility of our approach through a detailed evaluation against two popular benchmarks.

This article is an extended version of [Calvanese *et al.*, 2019], with selected proofs and additional information provided in an appendix.

2 Basic Definitions

We represent the provenance information via a *positive algebra provenance semiring* (or *provenance semiring* for short), originally introduced for databases [Green *et al.*, 2007]. Given a countably infinite set $\mathsf{N_V}$ of *variables*, the provenance semiring is the algebra $\mathbb{K} = (\mathbb{N}[\mathsf{N_V}], +, \times, 0, 1)$, where $\mathbb{N}[\mathsf{N_V}]$ denotes the space of polynomials with coefficients in $\mathbb{N}$ and variables in $\mathsf{N_V}$, the product $\times$ and the addition $+$ are two commutative and associative binary operators over $\mathbb{N}[\mathsf{N_V}]$, and $\times$ distributes over $+$. A *monomial* from $\mathbb{K}$ is a finite product of variables in $\mathsf{N_V}$. $\mathsf{N_M}$ and $\mathsf{N_P}$ denote the sets of all monomials from $\mathbb{K}$, and of all finite sums of monomials in $\mathsf{N_M}$, respectively; i.e., $\mathsf{N_P}$ contains only polynomials of the form $\sum_{1\leq i\leq n}\prod_{1\leq j_i\leq m_i} a_{i,j_i}$, with $a_{i,j_i} \in \mathsf{N_V}$, and $n, m_i > 0$. Since all coefficients are in $\mathbb{N}$, they disappear in this *expanded form*; e.g., $2a$ is $a + a$. A polynomial in expanded form is a finite sum of monomials, each formed by a finite product of variables. By distributivity, every polynomial can be equivalently rewritten in expanded form; however, the expanded form of a polynomial may become exponentially larger. By our definitions, $\mathsf{N_V} \subseteq \mathsf{N_M} \subseteq \mathsf{N_P}$.

Annotated OBDA. The provenance information of each axiom in an ontology, each mapping, and each tuple in a data source, is stored as an annotation. For this paper, we consider the standard OBDA setting with ontologies written in DL-Lite$_\mathcal{R}$ [Calvanese *et al.*, 2007], standard relational databases as data sources, and mappings given by GAV rules. Consider three mutually disjoint countable sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$. Assume that these sets are also disjoint from $\mathsf{N_V}$. DL-Lite$_\mathcal{R}$ *role* and *concept inclusions* are expressions of the form $S \sqsubseteq T$ and $B \sqsubseteq C$, respectively, where S, T are role expressions and B, C are concept expressions built through the grammar rules

$$S ::= R \mid R^-,\ T ::= S \mid \neg S,\ B ::= A \mid \exists S,\ C ::= B \mid \neg B,$$

with $R \in \mathsf{N_R}$ and $A \in \mathsf{N_C}$. A DL-Lite$_\mathcal{R}$ *axiom* is a DL-Lite$_\mathcal{R}$ role or concept inclusion. An *annotated* DL-Lite$_\mathcal{R}$ *ontology* is a finite set of *annotated axioms* of the form (α, p), where α is a DL-Lite$_\mathcal{R}$ axiom and $p \in \mathsf{N_M}$.

A *schema* $\mathcal{S}$ is a finite set of predicate symbols disjoint from $\mathsf{N_C} \cup \mathsf{N_R}$ with $\mathsf{ar}(P)$ the arity of $P \in \mathcal{S}$. An *annotated data instance* $\mathcal{D}$ over $\mathcal{S}$ maps every $P \in \mathcal{S}$ to a finite subset $P^\mathcal{D}$ of $\mathsf{N_I}^{\mathsf{ar}(P)} \times \mathsf{N_V}$. An *annotated mapping* is a finite set of *annotated rules* (ρ, p), where ρ is a (GAV) rule and $p \in \mathsf{N_V}$. A rule ρ is of the form $E(\vec{x}) \leftarrow \varphi(\vec{x}, \vec{y}, \vec{z})$, with $E \in \mathsf{N_C} \cup \mathsf{N_R}$ and $\varphi(\vec{x}, \vec{y}, \vec{z})$ a conjunction of atoms $P(\vec{t}, t)$, with $P \in \mathcal{S}$, $\vec{t}$ an $\mathsf{ar}(P)$-tuple of terms in $\vec{x} \cup \vec{y}$, and $t \in \vec{z}$. We restrict φ to a conjunction of atoms for simplicity of our theoretical development, also in line with the idea that semirings capture the provenance of positive queries [Green *et al.*, 2007]. See Sec. 5 for handling arbitrary OBDA mappings in our implementation.

An *annotated OBDA specification* $\mathcal{P}$ is a triple $(\mathcal{O}, \mathcal{M}, \mathcal{S})$, where $\mathcal{O}$ is an ontology with annotated axioms, $\mathcal{S}$ is a data source schema whose signature is disjoint from the signature of $\mathcal{O}$, and $\mathcal{M}$ is a set of annotated mappings, connecting $\mathcal{S}$ to $\mathcal{O}$ [Xiao *et al.*, 2018]. The pair $(\mathcal{P}, \mathcal{D})$ of an annotated OBDA specification $\mathcal{P}$ and an annotated data instance $\mathcal{D}$ is an *annotated OBDA instance*. In OBDA, data sources and mappings induce virtual assertions. In annotated OBDA, virtual assertions are annotated with the provenance information of the mapping and of matching tuples in the data instance. Formally, an *annotated assertion* $(E(\vec{a}), p)$ is an expression of the form $(A(a), p)$ or $(R(a, b), p)$, with $A \in \mathsf{N_C}$, $R \in \mathsf{N_R}$, $a, b \in \mathsf{N_I}$, and $p \in \mathsf{N_M}$. We write $\varphi(\mu(\vec{x}, \vec{y}, \vec{z})) \subseteq \mathcal{D}$ if μ is a function mapping $\vec{x}, \vec{y}$ to $\mathsf{N_I}$, $\vec{z}$ to $\mathsf{N_V}$, and $(\mu(\vec{t}, t)) \in P^\mathcal{D}$, for every atom $P(\vec{t}, t)$ in $\varphi(\vec{x}, \vec{y}, \vec{z})$. Given an annotated mapping $\mathcal{M}$ and data instance $\mathcal{D}$, the set $\mathcal{M}(\mathcal{D})$ of annotated assertions

$$(E(\mu(\vec{x})),\ p \times \textstyle\prod_{z\in\vec{z}}\mu(z)),\ \text{satisfying}$$

$(E(\vec{x}) \leftarrow \varphi(\vec{x}, \vec{y}, \vec{z}),\ p) \in \mathcal{M}$ and $\varphi(\mu(\vec{x}, \vec{y}, \vec{z})) \subseteq \mathcal{D}$ is the set of *virtual annotated assertions* for $\mathcal{M}$ over $\mathcal{D}$.

The semantics of annotated OBDA instances is based on interpretations over the signature of the ontology, extending classical DL-Lite$_\mathcal{R}$ interpretations to track provenance, when relevant. An *annotated interpretation* is a triple $\mathcal{I} = (\Delta^\mathcal{I}, \Delta^\mathcal{I}_\mathsf{m}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ and $\Delta^\mathcal{I}_\mathsf{m}$ are non-empty disjoint sets (called the *domain* of $\mathcal{I}$ and the *domain of monomials* of $\mathcal{I}$, respectively), and $\cdot^\mathcal{I}$ is the *annotated interpretation function* mapping

- every $a \in \mathsf{N_I}$ to some $a^\mathcal{I} \in \Delta^\mathcal{I}$;
- every $p, q \in \mathsf{N_M}$ to some $p^\mathcal{I}, q^\mathcal{I} \in \Delta^\mathcal{I}_\mathsf{m}$ s.t. $p^\mathcal{I} = q^\mathcal{I}$ iff the monomials p and q are mathematically equal (modulo associativity and commutativity, e.g., $(p\times q)^\mathcal{I} = (q\times p)^\mathcal{I}$ by commutativity);
- every $A \in \mathsf{N_C}$ to some $A^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}$; and
- every $R \in \mathsf{N_R}$ to some $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}$.

We extend $\cdot^\mathcal{I}$ to further DL-Lite$_\mathcal{R}$ expressions as natural:

$$\begin{array}{rcl}
(R^-)^\mathcal{I} & = & \{(e, d, p^\mathcal{I}) \mid (d, e, p^\mathcal{I}) \in R^\mathcal{I}\}\ , \\
(\neg S)^\mathcal{I} & = & (\Delta^\mathcal{I} \times \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}) \setminus S^\mathcal{I}, \\
(\exists S)^\mathcal{I} & = & \{(d, p^\mathcal{I}) \mid \exists e \in \Delta^\mathcal{I} : (d, e, p^\mathcal{I}) \in S^\mathcal{I}\}\ , \text{and} \\
(\neg B)^\mathcal{I} & = & (\Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}) \setminus B^\mathcal{I}.
\end{array}$$

The annotated interpretation $\mathcal{I}$ *satisfies*:

$$\begin{array}{ll}
(A(a), p), & \text{if } (a^\mathcal{I}, p^\mathcal{I}) \in A^\mathcal{I}; \\
(R(a, b), p), & \text{if } (a^\mathcal{I}, b^\mathcal{I}, p^\mathcal{I}) \in R^\mathcal{I}; \\
(B \sqsubseteq C, p), & \text{if, for all } q \in \mathsf{N_M}, (d, q^\mathcal{I}) \in B^\mathcal{I} \\
 & \qquad \text{implies that } (d, (q \times p)^\mathcal{I}) \in C^\mathcal{I};\ \text{and} \\
(S \sqsubseteq T, p), & \text{if, for all } q \in \mathsf{N_M}, (d, e, q^\mathcal{I}) \in S^\mathcal{I} \\
 & \qquad \text{implies that } (d, e, (q \times p)^\mathcal{I}) \in T^\mathcal{I}.
\end{array}$$

$\mathcal{I}$ satisfies an annotated ontology $\mathcal{O}$, in symbols $\mathcal{I} \models \mathcal{O}$, if it satisfies all annotated axioms in $\mathcal{O}$. $\mathcal{I}$ satisfies an annotated OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$ if $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{M}(\mathcal{D})$.

Example 2. Consider the OBDA instance of Example 1 and an annotated interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{\mathsf{Renier}, \mathsf{Venice}, \mathsf{Brugnaro}\}$, $\Delta^{\mathcal{I}}_{\mathsf{m}}$ containing $p \times n$, $q \times n$, $p \times m$, $q \times m$, $p \times n \times s$, $q \times n \times s$, with such individuals and monomials interpreted by themselves, and

$$\begin{aligned} \mathsf{headGov}^{\mathcal{I}} &= \{(\mathsf{R}, \mathsf{V}, p \times n), (\mathsf{B}, \mathsf{V}, q \times n)\}, \\ \mathsf{Mayor}^{\mathcal{I}} &= \{(\mathsf{R}, p \times n \times s), (\mathsf{B}, q \times n \times s)\}, \\ \mathsf{City}^{\mathcal{I}} &= \{(\mathsf{V}, p \times m), (\mathsf{V}, q \times m)\}. \end{aligned}$$

$\mathcal{I}$ is a model of such OBDA instance, where R, V, and B stand for Renier, Venice, and Brugnaro, respectively. ◁

Following the database approach [Green *et al.*, 2007; Green and Tannen, 2017], we annotate facts in interpretations with provenance information. However, in Green *et al.*'s setting, the database "is" the (only) interpretation, while in our case we adopt the open world assumption (as in OBDA), so the semantics is based on multiple interpretations. Our semantics ensures that, if we have a tuple $(d, p^{\mathcal{I}}) \in C^{\mathcal{I}}$ and $(C \sqsubseteq D)$ is annotated with n, then $(d, (p \times n)^{\mathcal{I}}) \in D^{\mathcal{I}}$. So derivations are also represented in interpretations, and thus can be entailed. Each derivation is independent of the others.

Regarding the semantics of negation, we point out that, at the level of an interpretation, the lack of provenance information is a support for the negation of a fact. This apparent counterintuitive behaviour does not hold in all interpretations, hence it does not manifest in the entailments. In fact, our focus in this paper is *query* entailment (defined next), negations are only defined to comply with the usual syntax and semantics of DL-Lite$_{\mathcal{R}}$. They do not affect query results, as in DL-Lite$_{\mathcal{R}}$.

Annotated Queries. We extend the notion of conjunctive queries in DLs by allowing binary and ternary predicates, where the last term of a tuple may contain provenance information represented as a monomial (by definition of the semantics of annotated OBDA instances, the last element of a tuple can only contain monomials, not sums). More specifically, a *Boolean conjunctive query (BCQ)* q is a sentence $\exists \vec{x}.\varphi(\vec{x}, \vec{a}, \vec{p})$, where φ is a conjunction of (non-repeating) atoms of the form $A(t_1, t)$, $R(t_1, t_2, t)$, and t_i is either an individual name from $\vec{a}$, or a variable from $\vec{x}$, and t (the last term of each tuple) is either an element of $\mathsf{N_M}$ in the list $\vec{p}$ or a variable from $\vec{x}$. We often write $P(\vec{t}, t)$ to refer to an atom which can be either $A(t_1, t)$ or $R(t_1, t_2, t)$ and $P(\vec{t}, t) \in q$ if $P(\vec{t}, t)$ is an atom occurring in q.

A *match* of the BCQ $q = \exists \vec{x}.\varphi(\vec{x}, \vec{a}, \vec{p})$ in the annotated interpretation $\mathcal{I}$ is a function $\pi : \vec{x} \cup \vec{a} \cup \vec{p} \to \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{I}}_{\mathsf{m}}$, such that $\pi(b) = b^{\mathcal{I}}$, for all $b \in \vec{a} \cup \vec{p}$, and $\pi(\vec{t}, t) \in P^{\mathcal{I}}$, for every $P(\vec{t}, t) \in q$. $\mathcal{I}$ satisfies the BCQ q, written $\mathcal{I} \models q$, if there is a match of q in $\mathcal{I}$. A BCQ is *entailed by* an annotated OBDA instance if it is satisfied by every model of it. For a BCQ q and an interpretation $\mathcal{I}$, $\nu_{\mathcal{I}}(q)$ denotes the set of all matches of q in $\mathcal{I}$. The *provenance* of q on $\mathcal{I}$, denoted $\mathsf{prov}_{\mathcal{I}}(q)$, is the (potentially infinite) expression:

$$\textstyle\sum_{\pi \in \nu_{\mathcal{I}}(q)} \prod_{P(\vec{t}, t) \in q} \pi^{-}(t)$$

where $\pi(t)$ is the last element of the tuple $\pi(\vec{t}, t) \in P^{\mathcal{I}}$; and $\pi^{-}(t)$ is any $v \in \mathsf{N_M}$ s.t. $v^{\mathcal{I}} = \pi(t)$. For $p \in \mathsf{N_P}$, we write $p \subseteq \mathsf{prov}_{\mathcal{I}}(q)$ if p is a sum of monomials and for each occurrence of a monomial in p we find an occurrence of it in $\mathsf{prov}_{\mathcal{I}}(q)$. $\mathcal{I}$ *satisfies* q with provenance $p \in \mathsf{N_P}$, written $\mathcal{I} \models (q, p)$, if $\mathcal{I} \models q$ and $p \subseteq \mathsf{prov}_{\mathcal{I}}(q)$. The annotated OBDA instance $(\mathcal{P}, \mathcal{D})$ *entails* q, $(\mathcal{P}, \mathcal{D}) \models q$, if for all annotated interpretations $\mathcal{I}$, if $\mathcal{I} \models (\mathcal{P}, \mathcal{D})$ then $\mathcal{I} \models q$; and $(\mathcal{P}, \mathcal{D}) \models (q, p)$, if $(\mathcal{P}, \mathcal{D}) \models q$ and $p \subseteq \mathsf{prov}_{\mathcal{I}}(q)$, for all $\mathcal{I}$ satisfying $(\mathcal{P}, \mathcal{D})$.

In our syntax, the atoms of the queries contain an additional parameter which may either be a variable or a monomial. As a result, one can filter query results based on provenance information by specifying constraints in the last parameter of the atoms, which was not possible in the original approach by Green *et al.* [Green *et al.*, 2007; Green and Tannen, 2017]. For example, $\exists xy.A(x, p) \wedge B(x, y)$ can be used to specify that we are only interested in matches of the query where the first atom is associated with a particular provenance. Variables can also be repeated, e.g. $\exists xy.A(x, y) \wedge B(x, y)$. One can fall back to the original setting from databases, where no constraints are imposed, by simply associating the last term of each atom with a fresh variable (see standard queries in Section 4).

The *size* $|X|$ of an annotated OBDA instance, a polynomial or a BCQ X is the length of the string that represents X. We assume a binary encoding of elements of $\mathsf{N_C}$,$\mathsf{N_R}$,$\mathsf{N_I}$ and $\mathsf{N_P}$ occurring in X. We may omit 'annotated' in front of terms such as 'OBDA,' 'queries,' 'inclusions,' 'assertions,' and others, whenever this is clear from the context.

Reasoning Problems. Annotating OBDA instances with provenance information does not impact consistency checking. That is, an annotated OBDA instance is satisfiable precisely when the OBDA instance that results from removing the annotations is satiafiable. We thus focus on the problem of *query entailment* w.r.t. a provenance polynomial: given an (annotated) OBDA instance $(\mathcal{P}, \mathcal{D})$, a query q and a polynomial $p \in \mathsf{N_P}$ decide if $(\mathcal{P}, \mathcal{D}) \models (q, p)$. Another important and related problem is to compute the provenance of a query: given an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query q, compute the set of all $p \in \mathsf{N_P}$ such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$. In our formalism, the latter problem depends on whether there is a finite set of polynomials which we can compute. As shown next, in DL-Lite$_{\mathcal{R}}$ the set of provenance polynomials may be infinite.

Example 3. Consider an OBDA instance $(\mathcal{P}, \mathcal{D})$ as in Ex. 1, but where now $\mathcal{O}$ of $\mathcal{P}$ contains also $(\mathsf{Mayor} \sqsubseteq \exists \mathsf{headGov}, t)$. For all $i \in \mathbb{N}$, $(\mathcal{P}, \mathcal{D}) \models (\mathsf{Mayor}(\mathsf{Renier}), p \times n \times s^{i+1} \times t^{i})$. Indeed, for any model $\mathcal{I}$ of $(\mathcal{P}, \mathcal{D})$, $(\mathsf{Renier}, (p \times n \times s)^{\mathcal{I}}) \in \mathsf{Mayor}^{\mathcal{I}}$ implies $(a, (p \times n \times s \times t)^{\mathcal{I}}) \in (\exists \mathsf{headGov})^{\mathcal{I}}$, which implies $(\mathsf{Renier}, (p \times n \times s^2 \times t)^{\mathcal{I}}) \in \mathsf{Mayor}^{\mathcal{I}}$, and so on. ◁

In Section 3 we consider the problem of query entailment w.r.t. a provenance polynomial. Note that in Example 3, if the semiring is multiplicatively idempotent (i.e., $s \times s = s$), the set of provenance polynomials is finite: the only polynomial is $p \times n \times s \times t$. This is not a coincidence; under multiplicative-idempotency, the set of provenance polynomials is always finite. The following proposition states that multiplicative-idempotency is indeed sufficient to guarantee a finite set of polynomials.

Proposition 1. *Under multiplicative idempotency, for any satisfiable OBDA instance* $(\mathcal{P}, \mathcal{D})$ *and BCQ* q*, the set of polynomials* $p \in \mathsf{N_P}$ *such that* $(\mathcal{P}, \mathcal{D}) \models (q, p)$ *is finite.*

In Section 4 we study idempotent semirings and consider the problem of computing the provenance of a query.

3 Provenance Annotated Query Entailment

We establish complexity results for the problem of deciding whether an OBDA instance entails a (provenance annotated) query. For clarity of presentation, we split our proof in two parts. We first show that for an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query (q, p), there is an OBDA instance $(\mathcal{P}_m, \mathcal{D}_m)$ and a set $\mathsf{Tr}(q_m, p_m)$ of (non-annotated) queries such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$ iff $(\mathcal{P}_m, \mathcal{D}_m)$ entails some $q' \in \mathsf{Tr}(q_m, p_m)$. Moreover, the sizes of $(\mathcal{P}_m, \mathcal{D}_m)$ and q' are polynomial in the sizes of $(\mathcal{P}, \mathcal{D})$ and (q, p). Then, we adapt the query rewriting algorithm $\mathsf{PerfectRef}$ [Calvanese *et al.*, 2007] to decide whether $(\mathcal{P}_m, \mathcal{D}_m) \models q'$.

Part 1: Characterization. Lemma 1 states that, given an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query (q, p), there is an OBDA instance $(\mathcal{P}_m, \mathcal{D}_m)$ and a query (q_m, p_m) that can be used to decide $(\mathcal{P}, \mathcal{D}) \models (q, p)$ and, moreover, all monomials in p_m are mathematically distinct (modulo associativity, commutativity, and distributivity).

Lemma 1. *Given a satisfiable OBDA instance* $(\mathcal{P}, \mathcal{D})$ *and a query* (q, p)*, there are* $(\mathcal{P}_m, \mathcal{D}_m)$ *and* (q_m, p_m) *such that*

- *any two monomials* p_1*,* p_2 *appearing in* p_m *are mathematically distinct;*
- $(\mathcal{P}, \mathcal{D}) \models (q, p)$ *iff* $(\mathcal{P}_m, \mathcal{D}_m) \models (q_m, p_m)$*; and*
- $|(\mathcal{P}_m, \mathcal{D}_m)| + |(q_m, p_m)|$ *is polynomially bounded by* $|(\mathcal{P}, \mathcal{D})| + |(q, p)|$.

We show that, given $(\mathcal{P}_m, \mathcal{D}_m)$ and (q_m, p_m) as in Lemma 1, (q_m, p_m) can be translated into a set of queries such that $(\mathcal{P}_m, \mathcal{D}_m)$ entails (q_m, p_m) iff it entails at least one of these queries. We first define the translation of a BCQ where all terms are variables (no individual names and no polynomials), and then adapt the translation for the general case. Given the BCQ $q_m = \exists \vec{x}.\ \varphi(\vec{x})$ with k atoms and $p_m \in \mathsf{N_P}$ with n monomials, define $\mathsf{Tr}(q_m, p_m)$ as the set of all BCQs:

$$\exists \vec{y}.\ \textstyle\bigwedge_{1 \leq i \leq n} \varphi_i(\vec{x_i}), \tag{1}$$

where $\vec{y} = \vec{x_1}, \ldots, \vec{x_n}$ and each $q_i = \exists \vec{x_i}.\ \varphi_i(\vec{x_i})$ is a 'copy' of q in which we replace each variable $x \in \vec{x}$ by a fresh variable $x_i \in \vec{x_i}$. We check whether we can find the monomials of the polynomial in these matches by replacing the last variable in each j-th atom of q_i by a monomial $p_{i,j} \in \mathsf{N_M}$ built from symbols occurring in p_m such that $\prod_{1 \leq j \leq k} p_{i,j} = p_i$ for some $p_i \in \mathsf{N_P}$, with $1 \leq i \leq n$; and $\sum_{1 \leq i \leq n} p_i = p$.

The translation of a BCQ with individual names is similar, except that we must add such individual names in each copy of the query; that is, we would replace the corresponding variable in the translation with the individual name occurring in the query. Theorem 1 formalises the correctness of our translation, where we write $(\mathcal{P}, \mathcal{D}) \models \mathsf{Tr}(q, p)$ to express that there is $q' \in \mathsf{Tr}(q, p)$ such that $(\mathcal{P}, \mathcal{D}) \models q'$.

Example 4. Consider the query

$$q = \exists xyzw.(\mathsf{headGov}(x, y, z) \wedge \mathsf{City}(y, w))$$

and the polynomial $p = (s \times t) + (s \times r)$. Then,

$$\exists x_1 y_1 x_2 y_2.(\mathsf{headGov}(x_1, y_1, s) \wedge \mathsf{City}(y_1, t) \wedge \\ \mathsf{headGov}(x_2, y_2, s) \wedge \mathsf{City}(y_2, r))$$

is in $\mathsf{Tr}(q, p)$. ◁

Theorem 1. *Let* $(\mathcal{P}, \mathcal{D})$ *be an OBDA instance,* q *a BCQ and* $p \in \mathsf{N_P}$ *a polynomial formed of mathematically distinct monomials.* $(\mathcal{P}, \mathcal{D}) \models (q, p)$ *iff* $(\mathcal{P}, \mathcal{D}) \models \mathsf{Tr}(q, p)$.

Without assuming that $p \in \mathsf{N_P}$ is formed of mathematically distinct monomials, we would need to add inequalities to the queries in $\mathsf{Tr}(q, p)$ (there is no way to distinguish $\mathsf{Tr}(q, p + p)$ from $\mathsf{Tr}(q, p)$). By Lemma 1, given the OBDA instance $(\mathcal{P}, \mathcal{D})$ and query (q, p), there are $(\mathcal{P}_m, \mathcal{D}_m)$ and (q_m, p_m), satisfying the assumption of Theorem 1, which we can use to decide whether $(\mathcal{P}, \mathcal{D}) \models (q, p)$. This is crucial for query entailment since entailment of conjunctive queries with inequalities in DL-Lite$_\mathcal{R}$ is undecidable [Gutiérrez-Basulto *et al.*, 2015].

Part 2: Query Rewriting. We adapt the classical query rewriting algorithm $\mathsf{PerfectRef}$ [Calvanese *et al.*, 2007] to decide whether $(\mathcal{P}, \mathcal{D}) \models q'$, for $q' \in \mathsf{Tr}(q, p)$, where $(\mathcal{P}, \mathcal{D})$ and (q, p) are as in Theorem 1. When possible, we use the definitions and terminology from [Calvanese *et al.*, 2007, Sec. 5.1], adapting some of them to our setting if needed.

For simplicity, for each role R^- occurring in an OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$, we add to $\mathcal{O}$ the annotated role inclusions $(R^- \sqsubseteq \overline{R}, p_R)$ and $(\overline{R} \sqsubseteq R^-, p'_R)$, where $\overline{R}$ is a fresh role name and p_R, p'_R are fresh variables of a provenance semiring. We assume w.l.o.g. that inverse roles only occur in such role inclusions by replacing other occurrences of R^- with $\overline{R}$. The symbol "$\_$" denotes non-distinguished non-shared variables. A positive inclusion I is a provenance annotated role or concept inclusion without negations. I is *applicable* to $A(x, p)$ if I is annotated with v occurring in p and it has A in its right-hand side. A positive inclusion I is applicable to $R(x, y, p)$ if *(i)* $x =\_$, I is annotated with v occurring in p, and the right-hand side of I is $\exists R$, or *(ii)* I is a role inclusion annotated with v occurring in p and its right-hand side is R or R^-. Given $p \in \mathsf{N_M}$ and $v \in \mathsf{N_V}$ occurring in p, we denote by $p_{|v}$ the result of removing one occurrence of v from p.

Definition 1. Let g be an atom and I a positive inclusion applicable to g. The atom obtained from g by applying I, denoted by $gr(g, I)$, is defined as follows:

- $gr(A(x, p), (A_1 \sqsubseteq A, v)) = A_1(x, p_{|v})$;
- $gr(A(x, p), (\exists R \sqsubseteq A, v)) = R(x, \_, p_{|v})$;
- $gr(R(x, \_, p), (A \sqsubseteq \exists R, v)) = A(x, p_{|v})$;
- $gr(R(x, \_, p), (\exists R_1 \sqsubseteq \exists R, v)) = R_1(x, \_, p_{|v})$;
- $gr(R(x, y, p), (R_1 \sqsubseteq R, v)) = R_1(x, y, p_{|v})$;
- $gr(g, I) = R_1(y, x, p_{|v})$, if $g = R(x, y, p)$ and either $I = (R_1 \sqsubseteq R^-, v)$ or $I = (R_1^- \sqsubseteq R, v)$. ◁

We use $\mathsf{PerfectRef}$ (Algorithm 1) originally presented in [Calvanese *et al.*, 2007], except that the applicability of a positive inclusion I to an atom g is as previously described and

Algorithm 1 PerfectRef

```
Input: a BCQ $q$, a set of positive inclusions $\mathcal{O}_\mathcal{T}$
Output: a set of BCQs $PR$
 1: $PR := \{q\}$
 2: repeat
 3:     $PR' := PR$
 4:     for all $q \in PR'$, all $g, g_1, g_2 \in q$ and all $I \in \mathcal{O}_\mathcal{T}$ do
 5:         if $\{q[g/gr(g, I)]\} \notin PR$ and $I \in \mathcal{O}_\mathcal{T}$ is applicable
            to $g \in q$ then
 6:             $PR := PR \cup \{q[g/gr(g, I)]\}$
 7:         if there are $g_1, g_2 \in q$ such that $g_1$ and $g_2$ unify then
 8:             $PR := PR \cup \{\tau(\mathsf{reduce}(q, g_1, g_2))\}$
 9: until $PR' = PR$
10: return $PR$
```

$gr(g, I)$ follows Definition 1. Let $q[g/g']$ denote the BCQ obtained from q by replacing the atom g with a new atom g'; let τ be a function that takes as input a BCQ q and returns a new BCQ obtained by replacing each occurrence of an unbound variable in q with the symbol '$\_$'; and let reduce be a function that takes as input a BCQ q and two atoms g_1, g_2 and returns the result of applying to q the most general unifier of g_1 and g_2 (unifying mathematically equal terms). $\mathsf{PerfectRef}(q, \mathcal{O}_\mathcal{T})$ is the output of the algorithm PerfectRef over q (with a monomial in $\mathsf{N_M}$ in the last parameter of each atom) and a set $\mathcal{O}_\mathcal{T}$ of positive inclusions of an OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$.

Example 5. Consider an OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$ as in Ex. 1. We call Algorithm 1 with $\mathcal{O}_\mathcal{T}$ and the query $q = \exists x.\mathsf{Mayor}(x, p \times n \times s)$ as input. Since I is applicable to $g = \mathsf{Mayor}(x, p \times n \times s)$, in Line 6, Alg. 1 adds to PR the result of replacing g by $gr(g, I) = \mathsf{headGov}(x, \_\,, p \times n)$ in q. Hence, $q^\ddagger = \exists x, y.\ \mathsf{headGov}(x, y, p \times n) \in \mathsf{PerfectRef}(q, \mathcal{O}_\mathcal{T})$. Indeed $q^\ddagger$ is a rewriting of q. $\lhd$

Our next theorem states the correctness of Algorithm 1.

Theorem 2. *Let q be a BCQ and $\mathcal{O}_\mathcal{T}$ the set of positive inclusions of an OBDA specification $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$. Given q and $\mathcal{O}_\mathcal{T}$ as input, Algorithm 1 terminates and outputs a set of BCQs PR such that, for all data instances $\mathcal{D}$ where $(\mathcal{P}, \mathcal{D})$ is satisfiable, $(\mathcal{P}, \mathcal{D}) \models q$ iff there is $q^\ddagger \in PR$ such that $((\emptyset, \mathcal{M}, \mathcal{S}), \mathcal{D}) \models q^\ddagger$.*

Termination of our modified version of PerfectRef is analogous to [Calvanese *et al.*, 2007, Lemma 34], except that now the number of terms is exponential in the size of monomials occurring in the query, and thus in the size of the query. This is due to Definition 1, where we 'break' the monomial into a smaller one. Our modification does not change the upper bounds obtained with the algorithm, since for data complexity the query is not part of the input and the upper bound for combined complexity, which we establish in Theorem 3, is obtained by a non-deterministic version of the algorithm.

Theorem 3. *Answering provenance annotated queries w.r.t. OBDA instances is* NP*-complete (combined complexity).*

4 Computing the Provenance of a Query

We now consider the problem of computing the provenance of a query. To avoid the case of an infinite provenance, we focus on the special case where the provenance semiring is fully idempotent, which is a sufficient condition for finite provenance (Proposition 1). The semiring is *fully idempotent* if for every polynomial $p \in \mathsf{N_P}$, $p \times p = p$ and $p + p = p$. This is the case, e.g., if the provenance refers to the name of the source of the knowledge; having several times the same name does not affect the result. Alternatively, one can model access rights and observe whether certain pieces of knowledge are needed for the entailment of a query w.r.t. an OBDA instance.

For fully idempotent semirings, the task corresponds to computing relevant monomials. More precisely, in this special case we want to compute all monomials p such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$. The *provenance of the query w.r.t. the OBDA instance* is the addition of all these monomials. This definition is equivalent to the general one since the semiring is idempotent: repetitions of a monomial do not affect the result, and repetitions of a variable within a monomial can be removed. If the semiring is only multiplicatively idempotent, then computing monomials does not suffice, as some of them may appear several times. However, the problem is still simplified to find the (finite) number of repeated monomials to be observed. In general, the query polynomial may be composed of exponentially many monomials, even if the query is a simple one of the form $\exists x.A(a, x)$, with $A \in \mathsf{N_C}$.

Proposition 2. *There exists an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a simple query q such that the provenance polynomial of q w.r.t. $(\mathcal{P}, \mathcal{D})$ is formed of exponentially many monomials.*

For some queries, provenance cannot be expressed by a provenance polynomial of polynomial length in the size of the ontology, even if an expanded form is not required. This follows from known results in monotone complexity [Karchmer and Wigderson, 1990]: there is no monotone Boolean formula (i.e., propositional formula using only the connectives $\wedge$ and $\vee$) of polynomial length expressing all the simple paths between two nodes in a graph. This holds already for *complete* graphs. Graphs can be described in DL-Lite$_\mathcal{R}$ (and simpler logics) using basic inclusion axioms, and monotone Boolean formulas are provenance polynomials over an idempotent semiring, where the $\wedge$ and $\vee$ serve as product and addition. Hence we have the following result [Peñaloza, 2009].

Proposition 3. *There exist an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query q such that the provenance of q w.r.t. $(\mathcal{P}, \mathcal{D})$ cannot be represented in polynomial space. This holds even for idempotent semirings, and if every axiom has a unique label.*

On the other hand, if every axiom is labeled with a unique variable, then the provenance polynomial for *instance queries* can be computed efficiently, whenever its length does not increase greatly; that is, it can be computed in polynomial time in the size of the input *and the output*. The proof of this claim follows the same ideas from [Peñaloza and Sertkaya, 2017], based on the fact that all the relevant monomials from the provenance are enumerable with polynomial delay.

Lemma 2. *The provenance p of an instance query w.r.t. an OBDA instance $(\mathcal{P}, \mathcal{D})$ can be computed in polynomial time in the size of $(\mathcal{P}, \mathcal{D})$* and of the polynomial *p.*

We give an algorithm for computing the provenance of a BCQ w.r.t. an OBDA instance. We focus on BCQs that do

Algorithm 2 ComputeProv

```
Input: a BCQ $q_0$, an OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$
Output: the provenance $p$ of $q$ w.r.t. $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$
1: $PR := \mathsf{PerfectRef}^{\star}(q_0^{\star}, \mathcal{O}_{\mathcal{T}})$,
2: for all $q \in PR$ do
3:     for all matches $\pi$ of $q_{\vec{y}}$ in $\mathcal{I}_{\mathcal{M}(\mathcal{D})}$ do
4:         $PR := PR \cup \{q^{-\star}_{\vec{y},\pi}\}$
5:     $PR := PR \setminus \{q\}$
6: return $p := \sum_{q \in PR} \prod_{P(\vec{t},t) \in q} t$
```

not have monomials in the last term of the atom. A BCQ $q = \exists \vec{x}.\varphi(\vec{x}, \vec{a})$ is *standard* if, for all $P(\vec{t}, t) \in q$, t is a fresh variable in $\vec{x}$. Algorithm 2 computes the provenance of a standard BCQ w.r.t. an OBDA instance. We adopt the same notation used for describing PerfectRef [Calvanese *et al.*, 2007] (also used in Section 3). PerfectRef* is a variant of PerfectRef (Algorithm 1), where the notions of applicability of an inclusion I w.r.t. an atom g and the definition of $gr(g, I)$ are as follows. I is applicable to an atom $A(x, p)$ if I has A in its right-hand side. A positive inclusion I is applicable to an atom $R(x, y, p)$ if *(i)* $x = \_$, and the right-hand side of I is $\exists R$, or *(ii)* the right-hand side of I is either R or R^-. Given $p \in \mathsf{N_M}$ and $v \in \mathsf{N_V}$, we define p^v as $p \times v$ if v does not occur in p, and we define p^v as p, otherwise. E.g., $vw^v = vw$.

Definition 2. Let g be an atom and I a positive inclusion applicable to g. The atom obtained from g by applying I, denoted by $gr(g, I)$, is defined as follows:

- $gr(A(x,p), (A_1 \sqsubseteq A, v)) = A_1(x, p^v)$;
- $gr(A(x,p), (\exists R \sqsubseteq A, v)) = R(x, \_, p^v)$;
- $gr(R(x, \_, p), (A \sqsubseteq \exists R, v)) = A(x, p^v)$;
- $gr(R(x, \_, p), (\exists R_1 \sqsubseteq \exists R, v)) = R_1(x, \_, p^v)$;
- $gr(R(x,y,p), (R_1 \sqsubseteq R, v)) = R_1(x, y, p^v)$;
- $gr(g, I) = R_1(y, x, p^v)$, if $g = R(x, y, p)$ and either $I = (R_1 \sqsubseteq R^-, v)$ or $I = (R_1^- \sqsubseteq R, v)$. ◁

For standard BCQs, Algorithm 2 is sound and complete. Termination of Algorithm 2 is an easy consequence of termination of PerfectRef. The main difference between Algorithm 2 and Algorithm 1 (Section 3) is that here we assume that a standard BCQ is given (without any provenance information) and we aim at computing its provenance. Instead of removing variables of the semiring while applying positive inclusions (Definition 1), we add the variables of the semiring whenever the associated positive inclusion is applied (Definition 2). In Line 1, we write $q^{\star}$ to denote the result of replacing each t in $P(\vec{t}, t) \in q$ by $\star$, where $\star$ is a fresh symbol from $\mathsf{N_V}$. This transformation ensures that in Definition 2 the last term is always an element of $\mathsf{N_M}$. In Line 3, we denote by $q_{\vec{y}}$ the result of replacing, for each $P(\vec{t}, t) \in q$, the last term t by a fresh variable from $\vec{y}$ (i.e., $q_{\vec{y}}$ is a standard BCQ). We perform another transformation in Line 4, denoted by $q^{-\star}_{\vec{y},\pi}$, which is the result of replacing, for each $P(\vec{t}, t) \in q$, the symbol $\star$ in t by $u \in \mathsf{N_M}$ such that $u^{\mathcal{I}} = \pi(y)$ (if there are multiple mathematically equal such u, we simply choose u arbitrarily), where y is the last term of the corresponding atom in $q_{\vec{y}}$ (that is, $P(\vec{t}, y) \in q_{\vec{y}}$). Observe that π is a match of $q_{\vec{y}}$ in $\mathcal{I}_{\mathcal{M}(\mathcal{D})}$.

Example 6. Assume Algorithm 2 receives as input the standard query $q_0 = \exists xz.\mathsf{Mayor}(x, z)$ and an OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$ with $\mathcal{O} = \{(\exists \mathsf{headGov} \sqsubseteq \mathsf{Mayor}, s)\}$ and

$$\mathcal{M}(\mathcal{D}) = \{(\mathsf{headGov}(\mathsf{Renier}, \mathsf{Venice}), u), \\ (\mathsf{headGov}(\mathsf{Brugnaro}, \mathsf{Venice}), v)\}.$$

In Line 1, Algorithm 2 calls PerfectRef*, defined as a variant of PerfectRef (Algorithm 1), where the notions of applicability of an inclusion I w.r.t. an atom g and the definition of $gr(g, I)$ are as in Section 4. The return of PerfectRef* is $PR = \{\exists x.\mathsf{Mayor}(x, \star), \exists xz.\mathsf{headGov}(x, z, \star \times s)\}$. Then, for all $q \in PR$ and all matches π of $q_{\vec{y}}$ in $\mathcal{I}_{\mathcal{M}(\mathcal{D})}$ (if they exist) the algorithm adds $q^{-\star}_{\vec{y},\pi}$ to PR. In this example, assume $q = \exists xz.\mathsf{headGov}(x, z, \star \times s)$. We have two matches of $q_{\vec{y}} = \exists xzy.\mathsf{headGov}(x, z, y)$ in $\mathcal{M}(\mathcal{D})$, one mapping y to u (call this match π) and the other mapping y to v (call it π'). So, $q^{-\star}_{\vec{y},\pi} = \exists xz.\mathsf{headGov}(x, z, u \times s)$ and $q^{-\star}_{\vec{y},\pi'} = \exists xz.\mathsf{headGov}(x, z, v \times s)$. In Line 5, Algorithm 2 removes $q_0^{\star}$ from PR. Finally, in Line 6, it returns the polynomial $u \times s + v \times s$. ◁

Theorem 4. *Let q be a standard BCQ and $(\mathcal{P}, \mathcal{D})$ an OBDA instance. Given q and $(\mathcal{P}, \mathcal{D})$ as input to Algorithm 2, it outputs the provenance of q w.r.t. $(\mathcal{P}, \mathcal{D})$.*

The upper bounds from the previous section for the general case obviously apply in the restricted idempotent case as well.

5 Evaluation

To evaluate the feasibility of our approach, we implemented a prototype system (*OntoProv*) that extends the state-of-the-art OBDA system *Ontop* [Calvanese *et al.*, 2017] with the support for provenance. *Ontop* supports SPARQL query answering over ontologies in OWL 2 QL, the W3C standard corresponding to DL-Lite$_{\mathcal{R}}$ [Motik *et al.*, 2012]. The algorithm of *Ontop* has two stages, an offline stage, which classifies the ontology and saturates the input set of mappings, and an online stage, which rewrites the input queries according to the saturated set of mappings. *OntoProv* enriches these steps by taking into account provenance information, and relies on ProvSQL [Senellart *et al.*, 2018] to handle provenance from the database and queries in the mappings that go beyond the CQ fragment. We compare *Ontop* v3.0.0-beta-3 and *OntoProv* over the BSBM [Bizer and Schultz, 2009] and the NPD [Lanti *et al.*, 2015] benchmarks. Experiments were run on a server with 2 Intel Xeon X5690 Processors (24 logical cores at 3.47 GHz), 106 GB of RAM and five 1 TB 15K RPM HDs. As RDBMS we have used PostgreSQL 11.2.

Evaluation with the BSBM Benchmark. The BSBM benchmark is designed to test the different features of SPARQL. It provides a baseline for our tests, since it comes with an empty ontology and therefore it does not require ontological reasoning. In this experiment we restrict to a set of parametric queries (called here *query mix*) in the benchmark (9 in total) that are supported by our theoretical framework.

Table 1 compares the average time (over three test runs) to evaluate the query mix with both *Ontop* and *OntoProv*, on two datasets containing 10k and 1M products (resp., *bsbm10k* and

Dataset	*mixTime Ontop*	*mixTime OntoProv*
bsbm10k	2.0s	3.2s
bsbm1M	326s	364s

Table 1: BSBM Experiment

	Ontop			*OntoProv*					
Q	*#unf*	*t*	*t10*	*#unf*	*#uinst*	*tinst*	*tinst10*	*#prov*	*#prov10*
1	0	29.5	172.4	16	16	5.2	1.7	1524	49
2	0	.5	3.1	32	32	.3	.4	28	60
3	24	5.0	51.1	16	16	248.0	296.5	84	153
4	0	3.2	24.3	16	16	437.6	297.3	90	53
5	0	.1	.2	0	0	.1	1.5	1	1
6	13	107.5	804.3	369	369	1439.3	tout	426	tout
7	0	.4	.2	0	0	.1	.3	1	1
9	15	6.8	53.4	64	55	.3	.9	5	5
10	1	.4	25.4	4	4	.8	11.3	1	7
11	6	53.6	760.8	184	184	1342.6	tout	474	tout
12	8	69.1	1215.5	185	185	1248.1	tout	476	tout
31	21	60.3	633.0	248	239	1.9	5.3	120	60

Table 2: NPD Experiment (times in seconds)

bsbm1M). Evaluation times for both systems are very close. Hence, without a complex ontology or complex mappings, the overhead for computing provenance is rather small.

Evaluation with the NPD Benchmark. As opposed to BSBM, the NPD Benchmark is specifically tailored to OBDA systems; it comes with a complex ontology, complex mappings, and queries of various kinds. We restrict to 12 user queries that are supported by our framework. We use the dataset *NPD*, containing real-world data about the oil extraction domain, and the dataset *NPD10*, which is 10 times the size of *NPD* and is generated by a *data scaler* [Lanti *et al.*, 2019]. Differently from the BSBM benchmark, in NPD we observed many timeouts (set to 40 minutes) when running the benchmark queries with *OntoProv*. This is due to the fact that, in NPD, the optimizations performed by *Ontop* over the query unfoldings are crucial for getting reasonably compact SQL queries. Such optimizations, however, need to be disabled in *OntoProv* to guarantee completeness. In fact, we are interested in *all* the possible ways to derive a result, and cannot identify and discard redundant derivations. For a broader discussion about these aspects, please refer to the additional material.

We assume that a user of *OntoProv* is more interested in understanding the reason for a *specific* answer tuple, rather than getting in bulk all possible explanations for all possible answer tuples. To simulate such user interaction, in our tests we have instantiated the NPD queries with answer tuples, and have run the obtained *instantiated queries* (which are, in fact, BCQs) over *OntoProv*. Table 2 contains the aggregate results of our runs. For each of our tests, we performed 5 test runs.

The columns *#unf* and *#uinst* denote the number of times a `UNION` operator appears in the unfolding of an NPD query and an instantiated query, respectively. This measure gives an idea on the complexity of the unfolding, and we can observe that the unfoldings produced by *OntoProv* are much more complex than those produced by *Ontop*. As argued above, this is because *OntoProv* disallows some optimizations. Columns *t* and *t10* denote the average execution times of the queries over the datasets *NPD* and *NPD10*, respectively, and for instantiated queries these values are respectively denoted by *tinst* and *tinst10*. The execution times for *OntoProv* are generally much higher than for *Ontop*. We attribute this to the increased complexity of the unfoldings. Columns *#prov* and *#prov10* denote the number of results for the instantiated queries, respectively over *NPD* and *NPD10*. These numbers can be interpreted as the number of possible ways an answer tuple can be derived, and give an indication on the complexity of the benchmark itself. For instance, for query *1* over the *NPD* dataset there are on average 1524 explanations for a single answer tuple.

This test shows that the approach is feasible even with complex ontologies and mappings, but also that more work is needed in order to devise optimization techniques dedicated to a setting with provenance.

6 Conclusions and Discussion

We investigated the problem of dealing with provenance within OBDA, based on the provenance semiring approach introduced for databases. In our case, every element of an OBDA instance is annotated with provenance information. We showed that query rewriting techniques can be applied to deal with provenance as well. An evaluation based on a prototypical implementation shows that our methods are feasible in practice.

A key difference between the problem of provenance computation (or its decision version) and that of axiom pinpointing [Schlobach and Cornet, 2003; Kalyanpur *et al.*, 2007; Baader *et al.*, 2007] and query explanation [Calvanese *et al.*, 2013; Croce and Lenzerini, 2018; Bienvenu *et al.*, 2019] is that axiom pinpointing and query explanation focus on tracing the *minimal* causes of a consequence (or the lack of it). In contrast, all possible derivations are relevant for provenance, independently of whether a cause is minimal or not.

As future work, we plan to investigate provenance with the monus operator. We will also study the provenance of SPARQL query answering [Geerts *et al.*, 2013] in OBDA. Our implementation computes the provenance of a query assuming that the semiring is multiplicatively idempotent. While this assumption is useful to identify which parts of the knowledge base contribute to the query result, it restricts the applicability of our approach to other settings, in particular, to the numerical ones. For capturing probabilities, it is important to distinguish repetitions, so (multiplicative) idempotency is not suitable. In our setting, dropping the idempotency condition leads to cases where the polynomial can be infinite. It would be interesting to investigate whether the polynomial can be finitely represented, so that its computation could be applied in a numerical setting.

References

[Artale *et al.*, 2009] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. of Artificial Intelligence Research*, 36(1):1–69, 2009.

[Baader *et al.*, 2007] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th Annual German Conf. on Artificial Intelligence (KI)*, pages 52–67, 2007.

[Bienvenu *et al.*, 2019] Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Computing and explaining query answers over inconsistent DL-Lite knowledge bases. *J. of Artificial Intelligence Research*, 64:563–644, 2019.

[Bizer and Schultz, 2009] Christian Bizer and Andreas Schultz. The Berlin SPARQL benchmark. *Int. J. on Semantic Web and Information Systems*, 5(2):1–24, 2009.

[Borgida *et al.*, 2008] Alexander Borgida, Diego Calvanese, and Mariano Rodriguez-Muro. Explanation in the *DL-Lite* family of description logics. In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 5332 of *LNCS*, pages 1440–1457. Springer, 2008.

[Bourgaux and Ozaki, 2019] Camille Bourgaux and Ana Ozaki. Querying attributed DL-Lite ontologies using provenance semirings. In *Proc. of the 33rd AAAI Conf. on Artificial Intelligence (AAAI)*, 2019.

[Buneman and Kostylev, 2010] Peter Buneman and Egor V. Kostylev. Annotation algebras for RDFS data. In *Proc. of the 2nd Int. Workshop on the Role of Semantic Web in Provenance Management (SWPM@ISWC)*, 2010.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2013] Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni. Reasoning about explanations for negative query answers in DL-Lite. *J. of Artificial Intelligence Research*, 48:635–669, 2013.

[Calvanese *et al.*, 2017] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web J.*, 8(3):471–487, 2017.

[Calvanese *et al.*, 2019] Diego Calvanese, Davide Lanti, Ana Ozaki, Rafael Peñaloza, and Guohui Xiao. Enriching ontology-based data access with provenance. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2019.

[Croce and Lenzerini, 2018] Federico Croce and Maurizio Lenzerini. A framework for explaining query answers in DL-Lite. In *Proc. of the 21st Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW)*, pages 83–97, 2018.

[Geerts *et al.*, 2013] Floris Geerts, Grigoris Karvounarakis, Vassilis Christophides, and Irini Fundulaki. Algebraic structures for capturing the provenance of SPARQL queries. In *Proc. of the 16th Int. Conf. on Database Theory (ICDT)*, pages 153–164. ACM, 2013.

[Glimm and Ogbuji, 2013] Birte Glimm and Chimezie Ogbuji. SPARQL 1.1 entailment regimes. W3C Recommendation, World Wide Web Consortium, March 2013. Available at http://www.w3.org/TR/sparql11-entailment/.

[Green and Tannen, 2017] Todd J. Green and Val Tannen. The semiring framework for database provenance. In *Proc. of the 36th ACM Symp. on Principles of Database Systems (PODS)*, pages 93–99. ACM, 2017.

[Green *et al.*, 2007] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *Proc. of the 26th ACM Symp. on Principles of Database Systems (PODS)*, pages 31–40, 2007.

[Gutiérrez-Basulto *et al.*, 2015] Víctor Gutiérrez-Basulto, Yazmín Ibáñez-García, Roman Kontchakov, and Egor V. Kostylev. Queries with negation and inequalities over lightweight ontologies. *J. of Web Semantics*, 35(P4):184–202, 2015.

[Harris and Seaborne, 2013] Steve Harris and Andy Seaborne. SPARQL 1.1 query language. W3C Recommendation, World Wide Web Consortium, March 2013. Available at http://www.w3.org/TR/sparql11-query.

[Kalyanpur *et al.*, 2007] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. In *Proc. of the 6th Int. Semantic Web Conf. (ISWC)*, volume 4825 of *LNCS*, pages 267–280. Springer, 2007.

[Karchmer and Wigderson, 1988] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc. of the 20th ACM Symp. on Theory of Computing (STOC)*, pages 539–550, 1988.

[Karchmer and Wigderson, 1990] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. on Discrete Mathematics*, 3(2):255–265, 1990.

[Kontchakov *et al.*, 2014] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC)*, volume 8796 of *LNCS*, pages 552–567. Springer, 2014.

[Lanti *et al.*, 2015] Davide Lanti, Martin Rezk, Guohui Xiao, and Diego Calvanese. The NPD benchmark: Reality check for OBDA systems. In *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT)*, pages 617–628. OpenProceedings.org, 2015.

[Lanti *et al.*, 2019] Davide Lanti, Guohui Xiao, and Diego Calvanese. VIG: Data scaling for OBDA benchmarks. *Semantic Web J.*, 10(2):413–433, 2019.

[Motik *et al.*, 2012] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language profiles. W3C Recommendation, World Wide Web Consortium, December 2012. Available at http://www.w3.org/TR/owl2-profiles/.

[Peñaloza and Sertkaya, 2017] Rafael Peñaloza and Barış Sertkaya. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artificial Intelligence*, 250:80–104, 2017.

[Peñaloza, 2009] Rafael Peñaloza. *Axiom Pinpointing in Description Logics and Beyond*. PhD thesis, Dresden University of Technology, 2009.

[Schlobach and Cornet, 2003] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003.

[Senellart *et al.*, 2018] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. ProvSQL: Provenance and probability management in PostgreSQL. *Proc. of the VLDB Endowment*, 11(12):2034–2037, 2018.

[Senellart, 2017] Pierre Senellart. Provenance and probabilities in relational databases. *SIGMOD Record*, 46(4):5–15, 2017.

[Xiao *et al.*, 2018] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: A survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5511–5519, 2018.

[Zimmermann *et al.*, 2012] Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general framework for representing, reasoning and querying with annotated Semantic Web data. *J. of Web Semantics*, 11:72–95, 2012.

A Proofs for Section 2

Proposition 1. *Under multiplicative idempotency, for any satisfiable OBDA instance $(\mathcal{P}, \mathcal{D})$ and BCQ q, the set of polynomials $p \in \mathsf{N_P}$ such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$ is finite.*

PROOF (SKETCH). Under multiplicative-idempotency, for any OBDA instance $(\mathcal{P}, \mathcal{D})$ and BCQ q, the number of possible monomials occurring $p \in \mathsf{N_P}$ such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$ is finite. Thus, the only possibility for this set to be infinite is if the monomials repeat an unlimited number of times.

To entail such polynomials, arbitrarily many repetitions should happen in all models of $(\mathcal{P}, \mathcal{D})$. However, under multiplicative-idempotency, any OBDA instance (based on an annotated DL-Lite$_\mathcal{R}$ ontology) enjoys the finite domain property [Artale *et al.*, 2009]. That is, if an OBDA instance has a model then it has a model $\mathcal{I}$ with $\Delta^\mathcal{I}$ finite. □

B Proofs for Section 3

To show Lemma 1 we use the following notation. An OBDA instance $(\mathcal{P}_m, \mathcal{D}_m)$ is *marked for an OBDA instance* $(\mathcal{P}, \mathcal{D})$ if it is the result of:

1. replacing each $(R(a, b), p)$ by $(R_{a,b}(a, b), p)$, where $R_{a,b}$ is a fresh role name;
2. for all $a, b \in \mathsf{N_I}$ and all $R \in \mathsf{N_R}$ occurring in $(\mathcal{P}, \mathcal{D})$, adding a concept inclusion $(\exists R^{(-)}_{a,b} \sqsubseteq C, v)$ for each $(\exists R^{(-)} \sqsubseteq C, u)$ occurring in it, where $v \in \mathsf{N_V}$;
3. for all $a, b \in \mathsf{N_I}$ and all $R, S \in \mathsf{N_R}$ occurring in $(\mathcal{P}, \mathcal{D})$, adding a role inclusion $(R^{(-)}_{a,b} \sqsubseteq S^{(-)}_{a,b}, v)$ for each $(R^{(-)} \sqsubseteq S^{(-)}, u)$ occurring in it, where $v \in \mathsf{N_V}$;
4. replacing the annotation of each axiom, mapping and tuple in a relation of $\mathcal{D}$ by fresh $v \in \mathsf{N_V}$ (including the axioms of items above), so that each v is unique.

Intuitively, we want to ensure that there is a model of $(\mathcal{P}_m, \mathcal{D}_m)$ where elements in the anonymous part (i.e., not in the image of $\mathsf{N_I}$) connected (via roles) to the image of an individual are associated with monomials containing at least one variable of the semiring, which is not shared by anonymous elements connected to the image of another individual. In other words, we want to 'mark' monomials associated to elements derived from assertions of named individuals.

Conditions 1–4 are necessary and sufficient to ensure that we cannot find two monomials which are mathematically equal in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$. Clearly, if Condition 4 does not hold we may find monomials mathematically equal in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$. As we show in the following example, this may also happen if 4 holds but not 1–3.

Example 7. Let $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$ be an OBDA instance with

$$\begin{array}{ll} (\exists R \sqsubseteq \exists S, u) & (\exists R^- \sqsubseteq \exists S^-, r) \\ (\exists S \sqsubseteq \exists R^-, s) & (\exists S^- \sqsubseteq \exists R, t) \end{array}$$

in $\mathcal{O}$ and $(R(a, b), p) \in \mathcal{M}(\mathcal{D})$. Then there are two tuples in $R^{\mathcal{I}_{(\mathcal{P},\mathcal{D})}}$ with the annotation $p \times r \times s \times u \times t$. ◁

To show Lemma 1 we use the classical notion of a *canonical model* of an OBDA instance. As in Section 3, for each role R^- in the OBDA instance $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$, we extend $\mathcal{O}$ with the role inclusions $(R^- \sqsubseteq \overline{R}, p_R)$ and $(\overline{R} \sqsubseteq R^-, p'_R)$, where $\overline{R}$ is a fresh role name and p_R, p'_R are fresh variables of a provenance semiring. Assume w.l.o.g. that inverse roles only occur in such equivalences. Let $\mathsf{N_{M_{min}}}$ be a *min*imal subset of $\mathsf{N_M}$ such that for all $p \in \mathsf{N_M}$ there is $q \in \mathsf{N_{M_{min}}}$ where p and q are mathematically equal. We define the canonical model $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ of a marked OBDA instance $(\mathcal{P}, \mathcal{D})$, with $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$, as the union of $\mathcal{I}_0, \mathcal{I}_1, \ldots$, where the $\mathcal{I}_n$s are inductively defined as follows. For $n = 0$, $\mathcal{I}_0$ is defined by:

$$\begin{array}{rcl} \Delta^{\mathcal{I}_0} & = & \mathsf{N_I}, \\ \Delta^{\mathcal{I}_0}_{\mathsf{m}} & = & \mathsf{N_{M_{min}}}, \\ A^{\mathcal{I}_0} & = & \{(a, p) \mid (A(a), p) \in \mathcal{M}(\mathcal{D})\}, \\ R^{\mathcal{I}_0} & = & \{(a, b, p) \mid (R(a, b), p) \in \mathcal{M}(\mathcal{D})\}, \end{array}$$

for all $A \in \mathsf{N_C}$ and all $R \in \mathsf{N_R}$, $a^{\mathcal{I}_0} = a$ for every $a \in \mathsf{N_I}$ and $p^{\mathcal{I}_0} = q \in \mathsf{N_{M_{min}}}$, with p and q mathematically equal, for every $p \in \mathsf{N_M}$. Assume now that $\mathcal{I}_n$ is defined. We define $\mathcal{I}_{n+1}$ by choosing a positive inclusion $I \in \mathcal{O}$ and applying one of the following rules,

- if $I = (A_1 \sqsubseteq A, p)$, $(a, v^{\mathcal{I}_n}) \in A_1^{\mathcal{I}_n}$, and $\vec{t} = (a, (p \times v)^{\mathcal{I}_n}) \notin A^{\mathcal{I}_n}$, then add $\vec{t}$ to $A^{\mathcal{I}_n}$,
- if $I = (R_1 \sqsubseteq R, p)$, $(a, b, v^{\mathcal{I}_n}) \in R_1^{\mathcal{I}_n}$, and $\vec{t} = (a, b, (p \times v)^{\mathcal{I}_n}) \notin R^{\mathcal{I}_n}$, then add $\vec{t}$ to $R^{\mathcal{I}_n}$,
- if $I = (R_1 \sqsubseteq R^-, p)$ or $I = (R_1^- \sqsubseteq R, p)$, $(a, b, v^{\mathcal{I}_n}) \in R_1^{\mathcal{I}_n}$, and $\vec{t} = (b, a, (p \times v)^{\mathcal{I}_n}) \notin R^{\mathcal{I}_n}$, then add $\vec{t}$ to $R^{\mathcal{I}_n}$,
- if $I = (\exists R \sqsubseteq A, p)$, there is b such that $(a, b, v^{\mathcal{I}_n}) \in R^{\mathcal{I}_n}$, and $\vec{t} = (a, (p \times v)^{\mathcal{I}_n}) \notin A^{\mathcal{I}_n}$, then add $\vec{t}$ to $A^{\mathcal{I}_n}$,
- if $I = (A \sqsubseteq \exists R, p)$, $(a, v^{\mathcal{I}_n}) \in A^{\mathcal{I}_n}$, and there is no b such that $\vec{t} = (a, b, (p \times v)^{\mathcal{I}_n}) \in R^{\mathcal{I}_n}$ then add a fresh element b to $\Delta^{\mathcal{I}_n}$ and add $\vec{t}$ to $R^{\mathcal{I}_n}$,
- if $I = (\exists R_1 \sqsubseteq \exists R, p)$, there is b such that $(a, b, v^{\mathcal{I}_n}) \in R_1^{\mathcal{I}_n}$, and there is no c such that $\vec{t} = (a, c, (p \times v)^{\mathcal{I}_n}) \in R^{\mathcal{I}_n}$ then add a fresh element c to $\Delta^{\mathcal{I}_n}$, add $\vec{t}$ to $R^{\mathcal{I}_n}$.

We assume that rule application is fair, i.e., if a rule is applicable at a certain place, it will eventually be applied there. $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is defined as the union of all such $\mathcal{I}_n$, where $x^{\mathcal{I}_{(\mathcal{P},\mathcal{D})}} = x^{\mathcal{I}_0}$ for every $x \in \mathsf{N_I} \cup \mathsf{N_M}$.

Importantly, the last element of each tuple in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is mathematically distinct from the others. This holds for $\mathcal{I}_0$ since each axiom of $\mathcal{O}$ is associated with a variable of a semiring appearing in at most one axiom. For $n > 0$, we have to consider tuples created using fresh anonymous elements. The interpretation of each individual from $\mathsf{N_I}$ occurring in $\mathcal{O}$ can be connected via roles to an anonymous part of $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$. We propagate the annotations (which are unique) associated to an individual to this anonymous part. Clearly, if these individuals are not connected, annotations associated to them form disjoint sets and the monomials are mathematically distinct. For connected individuals, we use the assumption that $\mathcal{O}$ is marked. Our assertions are of the form $(R_{a,b}(a, b), p)$ and since we add only concept and role inclusions of the form $(\exists R^{(-)}_{a,b} \sqsubseteq C, v)$ and $(R^{(-)}_{a,b} \sqsubseteq S^{(-)}_{a,b}, v)$, the extension of the fresh roles $R_{a,b}$ can only have elements in the image of named individuals and the extension of R can only have tuples where at least one element is anonymous. So $R_{a,b}^{\mathcal{I}_{(\mathcal{P},\mathcal{D})}} \cap R^{\mathcal{I}_{(\mathcal{P},\mathcal{D})}} = \emptyset$.

With this separation, if an anonymous element is connected to a after applying, e.g., an inclusion $(\exists R_{a,b} \sqsubseteq \exists S, p)$ in the construction of $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$, then we know that p must occur in the monomials associated to tuples containing this anonymous individual. On the other hand, if another anonymous element is connected to b after applying an inclusion $(\exists R^-_{a,b} \sqsubseteq \exists S, v)$ in the construction of $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$, then we know that now v must occur in the monomials associated to tuples containing this other anonymous individual. We never re-apply these inclusions containing fresh roles. So p and v mark the anonymous part of $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ connected to a and b, respectively. By definition of $(\mathcal{P},\mathcal{D})$, v and u are distinct variables of a semiring, this means that the last element of each tuple in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is mathematically distinct from the others.

Let $\mathcal{I}, \mathcal{J}$ be annotated interpretations. A *homomorphism* is a function $h : \mathcal{I} \to \mathcal{J}$ from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{J}}$ such that:

- $h(a^{\mathcal{I}}) = a^{\mathcal{J}}$ for all $a \in \mathsf{N_I} \cup \mathsf{N_M}$; and
- $(\vec{a}, p^{\mathcal{I}}) \in E^{\mathcal{I}}$ implies $(h(\vec{a}), h(p^{\mathcal{J}})) \in E^{\mathcal{J}}$, for all $E \in \mathsf{N_C} \cup \mathsf{N_R}$ and $p \in \mathsf{N_M}$;

where $\vec{a} = a^{\mathcal{I}}$, if $E \in \mathsf{N_C}$, and $\vec{a} = (a^{\mathcal{I}}, a'^{\mathcal{I}})$, with $h(\vec{a}) = (h(a^{\mathcal{I}}), h(a'^{\mathcal{I}}))$, if $E \in \mathsf{N_R}$, for $a, a' \in \mathsf{N_I}$. We write $\mathcal{I} \to \mathcal{J}$ if there is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$.

Lemma 3. *Let $(\mathcal{P},\mathcal{D})$ be a satisfiable marked OBDA instance (for some OBDA instance), q a BCQ and $p \in \mathsf{N_P}$ a polynomial. $(\mathcal{P},\mathcal{D}) \models (q,p)$ if, and only if, $\mathcal{I}_{(\mathcal{P},\mathcal{D})} \models (q,p)$.*

PROOF. By assumption, $(\mathcal{P},\mathcal{D})$ is satisfiable and so, by construction, $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is a model of $(\mathcal{P},\mathcal{D})$. So $(\mathcal{P},\mathcal{D}) \models (q,p)$ implies $\mathcal{I}_{(\mathcal{P},\mathcal{D})} \models (q,p)$. The converse direction follows from the standard notion that if an interpretation $\mathcal{J}$ (here annotated) satisfies $(\mathcal{P},\mathcal{D})$ then there is a homomorphism $\mathcal{I}_{(\mathcal{P},\mathcal{D})} \to \mathcal{J}$. Since the last element of each tuple in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is mathematically distinct from the others $\mathcal{I}_{(\mathcal{P},\mathcal{D})} \to \mathcal{J}$ can only be injective. So not only $\mathcal{J} \models q$ but there is also a 1 to 1 correspondence between the matches of q in $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$, with the respective annotations, and the matches of q in $\mathcal{J}$. Thus, if $\mathcal{I}_{(\mathcal{P},\mathcal{D})} \models (q,p)$ then $\mathcal{J} \models (q,p)$. As $\mathcal{J}$ is an arbitrary annotated intepretation satisfying $(\mathcal{P},\mathcal{D})$, it follows that $(\mathcal{P},\mathcal{D}) \models (q,p)$. □

We are now ready to prove Lemma 1.

Lemma 1. *Given a satisfiable OBDA instance $(\mathcal{P},\mathcal{D})$ and a query (q,p), there are $(\mathcal{P}_m,\mathcal{D}_m)$ and (q_m,p_m) such that*

- *any two monomials p_1, p_2 appearing in p_m are mathematically distinct;*
- *$(\mathcal{P},\mathcal{D}) \models (q,p)$ iff $(\mathcal{P}_m,\mathcal{D}_m) \models (q_m,p_m)$; and*
- *$|(\mathcal{P}_m,\mathcal{D}_m)| + |(q_m,p_m)|$ is polynomially bounded by $|(\mathcal{P},\mathcal{D})| + |(q,p)|$.*

PROOF. We first argue that if $(\mathcal{P}_m,\mathcal{D}_m)$ is marked for some OBDA instance $(\mathcal{P},\mathcal{D})$ then $(\mathcal{P}_m,\mathcal{D}_m)$ only entails annotated queries (q_m,p_m) such that any two monomials p_1, p_2 appearing in p_m are mathematically distinct. By assumption $(\mathcal{P},\mathcal{D})$ is satisfiable, so $(\mathcal{P}_m,\mathcal{D}_m)$ is satisfiable and, moreover, $\mathcal{I}_{(\mathcal{P},\mathcal{D})}$ is a model of $(\mathcal{P},\mathcal{D})$. Assume that $p \in \mathsf{N_P}$ contains two monomials which are mathematically equal. Then, for any BCQ q, $(\mathcal{P},\mathcal{D}) \models (q,p)$ iff $(\mathcal{P},\mathcal{D}) \models q$ and for every model $\mathcal{I}$ of $(\mathcal{P},\mathcal{D})$, $p \subseteq \mathsf{prov}_{\mathcal{I}}(q)$. This means that for each occurrence of a monomial in p we find an occurrence of it in $\mathsf{prov}_{\mathcal{I}}(q)$. However, for $\mathcal{I} = \mathcal{I}_{(\mathcal{P},\mathcal{D})}$, we know that we cannot find two monomials which are mathematically equal in $\mathsf{prov}_{\mathcal{I}}(q)$.

Now we argue about the second point of this lemma. Let $(\mathcal{P}_m,\mathcal{D}_m)$ be a marked OBDA instance for $(\mathcal{P},\mathcal{D})$ and let $\dagger : \mathsf{N_V} \to \mathsf{N_V}$ be the function that maps $v \in \mathsf{N_V}$ occurring in $(\mathcal{P}_m,\mathcal{D}_m)$ to $\dagger(v)$ in $(\mathcal{P},\mathcal{D})$. Given $p_m \in \mathsf{N_P}$ and a function $\dagger : \mathsf{N_V} \to \mathsf{N_V}$, $p_m^{\dagger}$ is the result of simultaneously replacing each occurrence of $v \in \mathsf{N_V}$ in p_m by $\dagger(v)$. Similarly, given a query q_m and $q_m^{\dagger}$ is the result of simultaneously replacing each occurrence of $v \in \mathsf{N_V}$ in q_m by $\dagger(v)$ and, in addition, we replace each $R_{a,b}$ by R, where $R \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. We use $\dagger$ to define a mapping between interpretations. Given an annotated interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{I}}_{\mathsf{m}}, \cdot^{\mathcal{I}})$, we define $\mathcal{I}^{\dagger}$ as $(\Delta^{\mathcal{I}}, \Delta^{\mathcal{I}}_{\mathsf{m}}, \cdot^{\mathcal{I}^{\dagger}})$ with $\cdot^{\mathcal{I}^{\dagger}}$ satisfying:

- $a^{\mathcal{I}_{\dagger}} = a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$;
- $p^{\mathcal{I}_{\dagger}} = (p^{\dagger})^{\mathcal{I}} \in \Delta^{\mathcal{I}}_{\mathsf{m}}$;
- $A^{\mathcal{I}^{\dagger}} = \{(d, (p^{\dagger})^{\mathcal{I}}) \mid (d, p^{\mathcal{I}}) \in A^{\mathcal{I}}\}$;
- $R^{\mathcal{I}^{\dagger}} = \{(d, e, (p^{\dagger})^{\mathcal{I}}) \mid (d, e, p^{\mathcal{I}}) \in R^{\mathcal{I}}\} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}, (p^{\dagger})^{\mathcal{I}}) \mid (a^{\mathcal{I}}, b^{\mathcal{I}}, p^{\mathcal{I}}) \in R^{\mathcal{I}}_{a,b}\}$,

for every $a, b \in \mathsf{N_I}$, $p \in \mathsf{N_M}$, $A \in \mathsf{N_C}$ and $R \in \mathsf{N_R}$. The following claim can be proved by structural induction.

Claim 1. *Let $(\mathcal{P}_m,\mathcal{D}_m)$ be a marked OBDA instance for $(\mathcal{P},\mathcal{D})$ and let $\dagger : \mathsf{N_V} \to \mathsf{N_V}$ be the function that maps $v \in \mathsf{N_V}$ occurring in $(\mathcal{P}_m,\mathcal{D}_m)$ to $\dagger(v)$ in $(\mathcal{P},\mathcal{D})$. For every annotated interpretation $\mathcal{I}$, the following holds:*

- *if $\mathcal{I} \models (\mathcal{P}_m,\mathcal{D}_m)$ then $\mathcal{I}^{\dagger} \models (\mathcal{P},\mathcal{D})$; and*
- *if $\mathcal{I} \models (\mathcal{P},\mathcal{D})$ then there is $\mathcal{J}$ such that $\mathcal{I} = \mathcal{J}^{\dagger}$ and $\mathcal{J} \models (\mathcal{P}_m,\mathcal{D}_m)$.*

We first show that if $(\mathcal{P}_m,\mathcal{D}_m) \models (q_m,p_m)$ and $(q_m^{\dagger}, p_m^{\dagger}) = (q,p)$ hold, then $(\mathcal{P},\mathcal{D}) \models (q,p)$. Assume that $(\mathcal{P}_m,\mathcal{D}_m) \models (q_m,p_m)$ and $(q_m^{\dagger}, p_m^{\dagger}) = (q,p)$. If $\mathcal{I} \models (\mathcal{P},\mathcal{D})$ then, by Claim 1, there is $\mathcal{J}$ such that $\mathcal{I} = \mathcal{J}^{\dagger}$ and $\mathcal{J} \models (\mathcal{P}_m,\mathcal{D}_m)$. As $(\mathcal{P}_m,\mathcal{D}_m) \models (q_m,p_m)$, we have that $\mathcal{I}_{(\mathcal{P}_m,\mathcal{D}_m)} \models (q_m,p_m)$ and there is a homomorphism from $\mathcal{I}_{(\mathcal{P}_m,\mathcal{D}_m)}$ to $\mathcal{J}$. By definition of $(\mathcal{P}_m,\mathcal{D}_m)$, a tuple (in a relation) can only be associated with multiple annotations in $\mathcal{I}_{(\mathcal{P}_m,\mathcal{D}_m)}$ if this tuple is in $\mathcal{I}_0$, meaning that it is in the image of an assertion in $\mathcal{M}_m(\mathcal{D}_m)$, with $\mathcal{P}_m = (\mathcal{O}_m, \mathcal{M}_m, \mathcal{S}_m)$. Moreover, $(\mathcal{P}_m,\mathcal{D}_m)$ has the same number of assertions as in $(\mathcal{P},\mathcal{D})$, the only difference is the renaming of roles in Point 1, and the renaming of annotations in Point 4. Thus, if a tuple (in a relation) is associated with k annotations in $\mathcal{I}_0$ then the corresponding assertion is also associated with k annotations in $(\mathcal{P},\mathcal{D})$, where the mapping between annotations is given by $\dagger$. So $\mathcal{I}^{\dagger}_{(\mathcal{P}_m,\mathcal{D}_m)}$ respects the multiplicity of assertions in $\mathcal{I}_{(\mathcal{P}_m,\mathcal{D}_m)}$. Thus, $\mathcal{I}_{(\mathcal{P}_m,\mathcal{D}_m)} \models (q_m,p_m)$ implies $\mathcal{I}^{\dagger}_{(\mathcal{P}_m,\mathcal{D}_m)} \models (q_m^{\dagger}, p_m^{\dagger})$. By definition of $\mathcal{I}^{\dagger}_{(\mathcal{P}_m,\mathcal{D}_m)}$, there is a homomorphism from $\mathcal{I}^{\dagger}_{(\mathcal{P}_m,\mathcal{D}_m)}$ to $\mathcal{J}^{\dagger}$. Since $(q_m^{\dagger}, p_m^{\dagger}) = (q,p)$ and $\mathcal{J}^{\dagger} = \mathcal{I}$ we have that $\mathcal{I} \models (q,p)$. As $\mathcal{I}$ is an arbitrary annotated interpretation satisfying $(\mathcal{P},\mathcal{D})$, we conclude that $(\mathcal{P},\mathcal{D}) \models (q,p)$.

We now show that if $(\mathcal{P}, \mathcal{D}) \models (q, p)$ holds, then $(\mathcal{P}_m, \mathcal{D}_m) \models (q_m, p_m)$, for some query (q_m, p_m) and function $\dagger : \mathsf{N_V} \to \mathsf{N_V}$ mapping $v \in \mathsf{N_V}$ in $(\mathcal{P}_m, \mathcal{D}_m)$ to $\dagger(v)$ in $(\mathcal{P}, \mathcal{D})$ and such that $(q, p) = (q_m^\dagger, p_m^\dagger)$. Assume $(\mathcal{P}, \mathcal{D}) \models (q, p)$. In this lemma we assume that $(\mathcal{P}, \mathcal{D})$ is satisfiable, and so, for any marked $(\mathcal{P}_m, \mathcal{D}_m)$ for $(\mathcal{P}, \mathcal{D})$, it follows from the definition that $(\mathcal{P}_m, \mathcal{D}_m)$ is satisfiable. This means that a canonical model $\mathcal{I}_{(\mathcal{P}_m, \mathcal{D}_m)}$ of $(\mathcal{P}_m, \mathcal{D}_m)$ satisfies $(\mathcal{P}_m, \mathcal{D}_m)$. Then, by Claim 1, $\mathcal{I}^\dagger_{(\mathcal{P}_m, \mathcal{D}_m)} \models (\mathcal{P}, \mathcal{D})$. As $(\mathcal{P}, \mathcal{D}) \models (q, p)$, we have that $\mathcal{I}^\dagger_{(\mathcal{P}_m, \mathcal{D}_m)} \models (q, p)$. The fact that $\mathcal{I}_{(\mathcal{P}_m, \mathcal{D}_m)} \models (q_m, p_m)$ for some (q_m, p_m) such that $(q, p) = (q_m^\dagger, p_m^\dagger)$ follows from the construction of $\mathcal{I}^\dagger_{(\mathcal{P}_m, \mathcal{D}_m)}$. Since $\mathcal{I}_{(\mathcal{P}_m, \mathcal{D}_m)}$ is a canonical model of $(\mathcal{P}_m, \mathcal{D}_m)$, by Lemma 3, $(\mathcal{P}_m, \mathcal{D}_m) \models (q_m, p_m)$.

Finally, as $(\mathcal{P}_m, \mathcal{D}_m)$ is marked for $(\mathcal{P}, \mathcal{D})$, it is easy to see that $|(\mathcal{P}_m, \mathcal{D}_m)|$ is polynomial in $|(\mathcal{P}, \mathcal{D})|$, and therefore, also in $|(\mathcal{P}, \mathcal{D})| + |(q, p)|$. Regarding the size of (q_m, p_m), one can easily see that it depends polynomially on the size of (q, p), since (q, p) is the result of replacing role names and variables from $\mathsf{N_V}$ in (q_m, p_m). Dependence on $|(\mathcal{P}, \mathcal{D})|$ is due to the function $\dagger$ and the requirement that annotations are unique in $(\mathcal{P}_m, \mathcal{D}_m)$, which affects the number of bits necessary to encode the annotations occurring in (q_m, p_m). □

With the help of this lemma, we can state the main result of this section; namely, that the translation for marked ontologies is correct.

Theorem 1. *Let $(\mathcal{P}, \mathcal{D})$ be an OBDA instance, q a BCQ and $p \in \mathsf{N_P}$ a polynomial formed of mathematically distinct monomials. $(\mathcal{P}, \mathcal{D}) \models (q, p)$ iff $(\mathcal{P}, \mathcal{D}) \models \mathsf{Tr}(q, p)$.*

PROOF. By definition of entailment from an ontology, it suffices to show that, for every annotated interpretation $\mathcal{I}$,

$$\mathcal{I} \models (q, p) \text{ iff } \mathcal{I} \models \mathsf{Tr}(q, p). \qquad (*)$$

Indeed, if $(*)$ holds, then $(\mathcal{P}, \mathcal{D}) \models (q, p)$ iff for every model $\mathcal{I}$ of $(\mathcal{P}, \mathcal{D})$ we have $\mathcal{I} \models (q, p)$ iff for every model $\mathcal{I}$ of $(\mathcal{P}, \mathcal{D})$ we have $\mathcal{I} \models \mathsf{Tr}(q, p)$ iff $(\mathcal{P}, \mathcal{D}) \models \mathsf{Tr}(q, p)$.

We now show the claim $(*)$. Let $\mathcal{I}$ be an arbitrary annotated interpretation.

Assume first that $\mathcal{I} \models (q, p)$, and let n be the number of monomials in p. Then, $\mathcal{I} \models q$ and $p \subseteq \mathsf{prov}_\mathcal{I}(q)$. By definition of $\mathsf{prov}_\mathcal{I}(q)$, there is a set $\chi_\mathcal{I}(q)$ consisting of n matches of q in $\mathcal{I}$ such that:

$$p = \sum_{\pi \in \chi_\mathcal{I}(q)} \prod_{P(\vec{t}, t) \in q} \pi^-(t). \qquad (2)$$

Consider first the case that all terms in q are variables. By definition of $\mathsf{Tr}(q, p)$, there is a function ϵ mapping $\chi_\mathcal{I}(q)$ to $\{1, \ldots, n\}$ and a BCQ q' of the form presented in Equation 1 such that the last term t of atom $P(\vec{t}, t)$ in $q_{\epsilon(\pi)}$ is any monomial $v \in \mathsf{N_M}$ such that $v^\mathcal{I} = \pi^-(t)$. Let $\pi'_{\epsilon(\pi)}$ be the result of replacing each $x \in \vec{x}$ in the domain of $\pi_{\epsilon(\pi)}$ by $x_{\epsilon(\pi)} \in \vec{x}_{\epsilon(\pi)}$. By definition of $q_{\epsilon(\pi)}$, each $\pi'_{\epsilon(\pi)}$ is a match of $q_{\epsilon(\pi)}$ in $\mathcal{I}$. Then, $\bigcup_{1 \leq \epsilon(\pi) \leq n} \pi'_{\epsilon(\pi)}$ is a match of q' in $\mathcal{I}$. So, $\mathcal{I} \models \mathsf{Tr}(q, p)$.

Conversely, assume that $\mathcal{I} \models \mathsf{Tr}(q, p)$. Then there is a match $\pi = \bigcup_{1 \leq i \leq n} \pi_i$ of some $q' \in \mathsf{Tr}(q, p)$ in $\mathcal{I}$, where each π_i is a match if the 'copy' q_i of q in $\mathcal{I}$, $1 \leq i \leq n$. Let π'_i be the result of replacing each variable $x_i \in \vec{x}_i$ by $x \in \vec{x}$, $1 \leq i \leq n$. By definition of q_i and q, each π'_i is a match of q in $\mathcal{I}$. Moreover, since any two monomials in p are not mathematically equal, $\pi'_1, \ldots, \pi'_n$ are all distinct. By definition of q_i, the product p_i of the polynomials in the domain of π'_i is a monomial in p, and the sum of the monomials $p_1, \ldots, p_n$ is equal to p. We then obtain the same equality of Equation 2. Thus, $p \subseteq \mathsf{prov}_\mathcal{I}(q)$. The proof for BCQs with individual names is similar, except that 'copies' of the query contain individual names in the corresponding positions. □

Theorem 2 establishes the main properties Algorithm 1. As already mentioned, termination can be proved using a similar argument as the one used for PerfectRef [Calvanese *et al.*, 2007, Lemma 34]. The important point is about the correctness of the rewritings, which we argue next, as part of our proof for Theorem 3.

Theorem 3. *Answering provenance annotated queries w.r.t. OBDA instances is* NP*-complete (combined complexity).*

PROOF. The lower bound follows from NP-hardness of conjunctive query answering in relational databases. We argue that the problem is in NP. Let $(\mathcal{P}, \mathcal{D})$ be an OBDA instance and let (q, p) be a query. By Lemma 1, there is $(\mathcal{P}_m, \mathcal{D}_m)$ and (q_m, p_m) such that

- $(\mathcal{P}, \mathcal{D}) \models (q, p)$ iff $(\mathcal{P}_m, \mathcal{D}_m) \models (q_m, p_m)$;
- for any two monomials p_1, p_2 appearing in p_m, it holds that p_1 and p_2 are mathematically distinct; and
- $|(\mathcal{P}_m, \mathcal{D}_m)| + |(q_m, p_m)|$ is polynomially bounded by $|(\mathcal{P}, \mathcal{D})| + |(q, p)|$.

By Theorem 1, for such polynomials $p_m \in \mathsf{N_P}$, $(\mathcal{P}_m, \mathcal{D}_m) \models (q_m, p_m)$ iff there is $q' \in \mathsf{Tr}(q_m, p_m)$ s.t. $(\mathcal{P}_m, \mathcal{D}_m) \models q'$. Then, to establish our upper bound we proceed as follows.

Given an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query (q, p), we first check in LOGSPACE $\subseteq$ PTIME satisfiability of $(\mathcal{P}, \mathcal{D})$ [Artale *et al.*, 2009]. If $(\mathcal{P}, \mathcal{D})$ is unsatisfiable then for all queries (q, p), we have that $(\mathcal{P}, \mathcal{D}) \models (q, p)$ holds trivially. Then, assume $(\mathcal{P}, \mathcal{D})$ is satisfiable. We guess an OBDA instance $(\mathcal{P}_m, \mathcal{D}_m)$ marked for $(\mathcal{P}, \mathcal{D})$, a query (q_m, p_m), and $q' \in \mathsf{Tr}(q_m, p_m)$ such that $(\mathcal{P}, \mathcal{D}) \models (q, p)$ iff $(\mathcal{P}_m, \mathcal{D}_m) \models q'$. By construction of q', $|q'|$ is polynomial in $|(q_m, p_m)|$, which in turn is polynomial in $|(\mathcal{P}, \mathcal{D})| + |(q, p)|$ (Lemma 1).

We now adapt the query rewriting algorithm PerfectRef [Calvanese *et al.*, 2007] to decide whether $(\mathcal{P}_m, \mathcal{D}_m) \models q'$. We guess a rewriting $q^\ddagger$ of q', and guess a sequence of pairs (I, n) where I is a positive inclusion and n is an identifier for an atom position in a query. Each (I, n) represents a transformation of PerfectRef on a query. There is a non-deterministic version of PerfectRef which would follow this sequence of transformations and return $q^\ddagger$, with q' as input. The sequence of transformations is of polynomial size, since every query returned by PerfectRef can only be generated after a polynomial number of transformations of the initial query. By definition of PerfectRef, each $q^\ddagger \in \mathsf{PerfectRef}(q', \mathcal{O}_\mathcal{T})$ is polynomial in $|q'|$, where $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$ and $\mathcal{O}_\mathcal{T}$ is the set of positive inclusions in $\mathcal{O}$.

Membership in NP follows from the fact that we can check in polynomial time that:

1. $q' \in \mathsf{Tr}(q_m, p_m)$;
2. $q^\ddagger \in \mathsf{PerfectRef}(q', \mathcal{O}_\mathcal{T})$ (using the sequence of transformations);
3. p is the result of replacing each occurrence of $v \in \mathsf{N_V}$ in p_m by $\dagger(v)$, where $\dagger$ is a function that maps $v \in \mathsf{N_V}$ occurring in $(\mathcal{P}_m, \mathcal{D}_m)$ to $\dagger(v)$ in $(\mathcal{P}, \mathcal{D})$;
4. q is the result of each occurrence of $v \in \mathsf{N_V}$ in q_m by $\dagger(v)$, in addition to replacing roles $R_{a,b}$ by R in q_m, where $\dagger$ is as in Point (3); and

our modified algorithm is correct. Correctness is shown as in [Calvanese *et al.*, 2007], but here we change the notion of applicability of a positive inclusion I to an atom g and the definition of $gr(g, I)$ (Definition 1) to ensure that each transformation respects the semantics of annotated DL-Lite$_\mathcal{R}$. □

C Proofs for Section 4

Proposition 2. *There exists an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a simple query q such that the provenance polynomial of q w.r.t. $(\mathcal{P}, \mathcal{D})$ is formed of exponentially many monomials.*

PROOF. Consider the OBDA instance with the ontology $\mathcal{O}$ containing the axioms,

$$(A \sqsubseteq B_1, x), (A \sqsubseteq C_1, x), (B_n \sqsubseteq D, x),$$
$$(B_i \sqsubseteq B_{i+1}, x_i), (B_i \sqsubseteq C_{i+1}, y_i), (C_i \sqsubseteq B_{i+1}, x_i),$$
$$(C_i \sqsubseteq C_{i+1}, y_i), (C_n, \sqsubseteq D, x),$$

for $1 \leq i < n$, and $\mathcal{M}(\mathcal{D}) = \{(A(a), p)\}$. Consider the simple query $q = D(a)$. Every monomial $p = x \times \prod_{i=1}^{n} z_i$, where each $z_i \in \{x_i, y_i\}$, is such that $\mathcal{O} \models (q, p)$ (and none other). Hence, the polynomial of the query q is formed by the sum of 2^n different monomials; that is, it is exponential on the size of the ontology. □

Proposition 3. *There exist an OBDA instance $(\mathcal{P}, \mathcal{D})$ and a query q such that the provenance of q w.r.t. $(\mathcal{P}, \mathcal{D})$ cannot be represented in polynomial space. This holds even for idempotent semirings, and if every axiom has a unique label.*

PROOF. Let N be a set with n concept names such that $A, B \in N$. Let $((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$ be an OBDA instance with

$$\mathcal{O} = \{(C \sqsubseteq D, x_{CD}) \mid C, D \in N\},\ \mathcal{M}(\mathcal{D}) = \{A(a), x_A\}.$$

Consider the query $q = B(a)$. We have that $\mathcal{O}$ simulates a complete graph over the nodes in N. Every derivation of $B(a)$ represents a path from A to B in this graph. Hence, if the provenance of q could be expressed with polynomial space, there would be a monotone Boolean formula representing all the paths from A to B in the complete graph with n nodes, contradicting the results from [Karchmer and Wigderson, 1988; Karchmer and Wigderson, 1990]. □

Theorem 4. *Let q be a standard BCQ and $(\mathcal{P}, \mathcal{D})$ an OBDA instance. Given q and $(\mathcal{P}, \mathcal{D})$ as input to Algorithm 2, it outputs the provenance of q w.r.t. $(\mathcal{P}, \mathcal{D})$.*

PROOF. Recall that in Section 4 we consider idempotent semirings. Assume a standard BCQ q and an OBDA instance $(\mathcal{P}, \mathcal{D})$ is given as input to Algorithm 2 and it returns $p \in \mathsf{N_P}$. Then, it suffices to show that every monomial $m \in \mathsf{N_M}$ in p is such that $(\mathcal{P}, \mathcal{D}) \models (q, m)$ and, conversely, if there is $m' \in \mathsf{N_M}$ such that $(\mathcal{P}, \mathcal{D}) \models (q, m')$ then there is a monomial in p that is mathematically equal to m' (by 'monomial in a polynomial p' we mean that the monomial is one of the elements of the sum of monomials, given that p is in expanded form).

By definition of Algorihm 2, if $m \in \mathsf{N_M}$ is a monomial in p then there is a match of $q'_{\vec{y}}$ where q' is a rewriting of $q^\star$ in $\mathcal{I}_{\mathcal{M}(\mathcal{D})}$. By Definition 2, the last term of each atom of the rewriting q' is associated with the product of $\star$ and the annotations of the inclusions used to derive the atom. We then replace $\star$ in each atom by the corresponding annotation of the tuple in the match. By soundness of the algorithm PerfectRef [Calvanese *et al.*, 2007] and the semantics of annotated interpretations, $(\mathcal{P}, \mathcal{D}) \models (q, m)$.

Conversely, if $(\mathcal{P}, \mathcal{D}) \models (q, m)$ then, by the semantics of annotated interpretations, m corresponds to a derivation of q using axioms of $(\mathcal{P}, \mathcal{D})$. By completeness of the algorithm PerfectRef [Calvanese *et al.*, 2007], there is a query rewriting q' of $q^\star$ in PR, following Definition 2, and a match of $q'_{\vec{y}}$ in $\mathcal{I}_{\mathcal{M}(\mathcal{D})}$ such that m is the product of the annotations resulting from replacing $\star$ in the last term of each atom of q' by the corresponding annotation of the tuple in the match. By definition of Algorihm 2, m is added to the polynomial returned by the algorithm. □

D Algorithm for Section 5

We first briefly recall the query answering algorithm used in *Ontop*, and introduce the ProvSQL extension of PostgreSQL. Then we explain how *OntoProv* extends *Ontop* with the support for provenance.

D.1 ProvSQL

The ProvSQL project is a PostgreSQL extension to add support for (m-)semiring provenance. It supports semiring provenance, with or without monus (m-semiring). Note that the monus operator is beyond the semiring framework considered here. ProvSQL adds an extra *provsql* column to each table in the database. This column associates to each tuple a *universally unique identifier* (UUID) as a provenance token. Likewise, each answer is also associated to a UUID. ProSQL providdes several functions to intepret these tokens into several kinds of provenance information. The next example shows how ProvSQL works.

Example 8. Consider the following two tables

EMP

EMPNO	ENAME	JOB	DEPTNO	*provsql*
7367	SMITH	CLERK	10	t11
9527	JOHN	HR	10	t12
4839	MARY	PROGR	10	t13
4839	RALPH	SYSADM	10	t14

Algorithm 3 Ontop

Input: a query q, an OBDA instance $I = ((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$
Output: the answer to q w.r.t. I

```
// offline
$\mathcal{O}' = \mathit{classify}(\mathcal{O})$
$\mathcal{M}^{sat} = \mathit{saturate}(\mathcal{M}, \mathcal{O}')$
$\mathcal{M}^{sat} = \mathit{optimizeM}(\mathcal{M}^{sat})$
// online
$q' = \mathit{unfold}(q, \mathcal{M}^{sat})$
$Q = \mathit{optimizeQ}(q', S)$
return $\mathit{eval}(Q, \mathcal{D})$
```

DEPT

DEPTNO	DNAME	LOC	provsql
10	APPSERVER	NEW YORK	t21

By calling the `add_provenance` functions, ProvSQL adds a column `provsql` to both tables, add generates a unique provenance token for each tuple.

Consider the SQL query:

```
SELECT e.ENAME, d.DNAME
FROM EMP e, DPT d
WHERE e.DEPTNO = d.DEPTNO
   AND JOB='PROGR'
```

To this query ProvSQL returns the following result

ENAME	DNAME	provsql
Mary	APPSERVER	t31

The additional column *provsql* is added automatically by ProvSQL, and it contains a fresh token which encodes the provenance information for the tuple in the result. Such token can be then fed to ProvSQL functions, such as the `where_provenance` function, to decode the desired provenance information. In our example, by using `where_provenance('t31')` we would get

```
{[EMP:t13:2],[DEPT:t21:2]}
```

saying that the result tuple has been derived from the second attributes of the two tuples t13 and t21 in the tables EMP and DEPT. ◁

D.2 Algorithm of *Ontop*

In *Ontop*, tree-witness rewriting is switched off by default. This because it is recognized that queries which trigger tree-witness rewriting are mostly of theoretical relevance, and extremely rare in practice (actually we are not aware of any real-world scenario in which such queries are utilized). A reason for this is that whereas in CQs it is natural to write queries with existentially quantified variables (non-answer variables are always existentially quantified), in SPARQL (under the OWL 2 QL entailment regime) doing so is more difficult.

Ontop is an OBDA system that operates over OWL 2 QL ontologies [Motik *et al.*, 2012] and SPARQL [Harris and Seaborne, 2013] queries. The standard semantics for such setting is the OWL 2 QL entailment regime [Glimm and Ogbuji, 2013], which slightly diverges from the one adopted in the DL context [Kontchakov *et al.*, 2014]. For performance reasons, *Ontop* does not rely on PerfectRef, but instead adopts a mixed strategy based on *mapping saturation* and *tree-witness rewriting* [Calvanese *et al.*, 2017]. Mapping saturation simulates the saturation of the ABox with respect to basic concept and role inclusions, whereas tree-witness rewriting rewrites the query so as to take into account those axioms with an existential on the right-hand side.

In *Ontop*, tree-witness rewriting is switched off by default. Indeed, for compliance with the OWL 2 QL entailment regime, also non-answer variables in a SPARQL query have to match known individuals (and not existentially implied ones). Therefore, SPARQL queries for which applying tree-witness rewriting may actually result in additional answers are rather unnatural (they are obtained by constructing a complex concept expression that makes use of value restriction within the query itself), and it is commonly recognized that they are rare in practical scenarios (actually we are not aware of any real-world scenario in which such queries are utilized). Note that this differs from the semantics of CQs, where every non-answer variable can match also existentially implied individuals. Therefore it is more natural to write CQs for which applying tree-witness rewriting has a concrete impact on query answering.

We have adapted the mapping saturation process to our setting with provenance, so as to support query answering with provenance in relevant practical scenarios, and be able to measure the performance in them. A full implementation that also adapts the tree-witness rewriting is nevertheless important and will be part of future work, but it is not relevant for the (real-world) tests in this work. The *Ontop* algorithm, limited to the mapping saturation approach, is outlined in Algorithm 3. The algorithm takes as inputs an OBDA instance I and a SPARQL query q, and returns the answers to q w.r.t. I. The workflow can be divided into an offline stage and an online stage. During the offline stage (i.e., system start-up), *Ontop* first classifies the ontology $\mathcal{O}$. The result of the classification is a complete hierarchy of classes and properties, which is stored in-memory as a directed acyclic graph. Then it compiles the classified ontology into the input mapping $\mathcal{M}$, thus obtaining the saturated mapping $\mathcal{M}^{sat}$, also known as the T-mapping.

During the online stage (i.e., query execution), *Ontop* transforms the input SPARQL query q into an SQL query q' by exploiting the saturated-mapping $\mathcal{M}^{sat}$, and then produces an optimized SQL query Q by exploiting the database integrity constraints $\mathcal{S}$. Finally, Q is evaluated over the database instance $\mathcal{D}$.

Next example shows the steps from Algorithm 3.

Example 9. Consider the setting from Example 8. Consider the following ontology stating that programmers are employees.

```
ax1 SubClassOf(:Programmer, :Employee).
```

Consider the following mapping assertions:

```
MapID:  m1
Target: triple(:emp/{EMPNO},rdf:type,:Employee).
        triple(:emp/{EMPNO},ex:name,{ENAME}).
        triple(:emp/{EMPNO},ex:dept,:dept/{DEPTNO}).
Source: SELECT * FROM EMP

MapID:  m2
```

```
Target: triple(:dept/{DEPTNO},rdf:type,:Department).
        triple(:dept/{DEPTNO},ex:name,{DNAME}).
        triple(:dept/{DEPTNO},ex:loc, {LOC}).
Source: SELECT * FROM DEPT

MapID: m3
Target: triple(:emp/{EMPNO},rdf:type,:Programmer).
Source: SELECT * FROM EMP WHERE JOB='PROGR'
```

After ontology classification and mapping saturation, the saturated set of mappings will contain the original mappings plus the following mapping:

```
MapID: m3_ax1
Target: triple(:emp/{EMPNO},rdf:type,:Employee).
Source: SELECT * FROM EMP WHERE JOB='PROGR'
```

This mapping is derived from mapping `m3` and axiom `ax1`. Since the SQL query in `m3_ax1` is contained in the SQL query in `m1`, the mapping is in fact redundant and gets removed in the `optimizeM` step. Therefore, the final saturated set of mappings will coincide with the original set of mappings.

Consider the following SPARQL query relating employees to departments they work in:

```
SELECT ?eName ?dName WHERE {
   ?e rdf:type :Employee;
      ex:name  ?eName;
      ex:dept  ?d .
   ?d ex:name dName.
}
```

Such query has the following algebra tree:

```
SELECT ?eName ?dName
   JOIN
      triple(?e, rdf:type, :Employee).
      triple(?e, ex:name, ?eName).
      triple(?e, ex:dept, ?d).
      triple(?d, ex:name, ?dName).
```

In the unfold step, each `triple` in the algebra tree is replaced by the corresponding source part in the mapping. The query q' look like:

```
SELECT e1.ENAME as eName, d.DNAME as dName,
FROM EMP e1, EMP e2, EMP e3, DPT d
WHERE e1.EMPNO=e2.EMPNO AND
      e1.EMPNO=e3.EMPNO AND
      e1.DEPTNO = d.DEPTNO AND
      e1.JOB='PROGR'
```

In the `optimizeQ` step, the redundant self-joins are removed and the final query Q will be:

```
SELECT e.ENAME as eName, d.DNAME as dName,
FROM EMP e, DPT d
WHERE e.DEPTNO = d.DEPTNO
   AND JOB='PROGR'
```

D.3 The *OntoProv* System

OntoProv accepts a BGP query, possibly with a FILTER condition, and returns the answer of the query together with the provenance for each answer tuple.

Algorithm 4 outlines the *OntoProv* approach. The main workflow is the same, but now each step has to deal also with the provenance information.

We explain the algorithm by means of an example.

Algorithm 4 OntoProv

```
Input: a query $q$, an OBDA instance $I = ((\mathcal{O}, \mathcal{M}, \mathcal{S}), \mathcal{D})$
Output: the answer to $q$ w.r.t. $I$
// offline
$\mathcal{O}' = \mathit{classifyAndStorePaths}(\mathcal{O})$
$\mathcal{M}_{prov} = addMappingsProvenance(\mathcal{M})$
$\mathcal{M}' = \mathcal{M} \cup \mathcal{M}_{prov}$
$\mathcal{M}^{sat} = \mathit{saturate}(\mathcal{M}', \mathcal{O}')$
$\mathcal{M}^{sat} = \mathit{optimizeM}(\mathcal{M}^{sat})$
// online
$q^{\mathcal{M}^{sat}} = \mathit{unfold}(q, \mathcal{M}^{sat})$
$Q = \mathit{optimizeQ}(q^{\mathcal{M}^{sat}}, S)$
return $\mathit{eval}(Q, \mathcal{D})$
```

Example 10. Consider the scenario from Example 9. The function `classifyAndStorePaths` classifies the ontology and stores, for each (basic) concept and role in the ontology, the paths to their descendants. For our example, it stores the path:

```
p[:Employee,:Engineer]
```

In `addMappingsProvenance`, the mappings are used to generate new mappings encoding the provenance information (i.e., mappings IDs and provenance returned by ProvSQL) in their target parts (which are now quadruples). For our example, these mappings are:

```
MapID:  m1quad
Target: quad(:emp/{EMPNO},rdf:type,:Employee,
               :prov/mkey-m1/dkey-{provsql}).
        quad(:emp/{EMPNO},ex:name,{ENAME},
               :prov/mkey-m1/dkey-{provsql}).
        quad(:emp/{EMPNO},ex:dept,:dept/{DEPTNO},
               :prov/mkey-m1/dkey-{provsql}).
Source: SELECT * FROM EMP

MapID:  m2quad
Target: quad(:dept/{DEPTNO},rdf:type,:Department,
                :prov/mkey-m2/dkey-{provsql}).
        quad(:dept/{DEPTNO},ex:name,{DNAME},
                :prov/mkey-m2/dkey-{provsql}).
        quad(:dept/{DEPTNO},ex:loc, {LOC},
                :prov/mkey-m2/dkey-{provsql}).
Source: SELECT * FROM DEPT

MapID: m3quad
Target: quad(:emp/{EMPNO},rdf:type,:Programmer,
                 :prov/mkey-m3/dkey-{provsql}).
Source: SELECT * FROM EMP WHERE JOB='PROGR'
```

The saturated set of mappings is derived as in Example 9. Hence, it will contain the following mapping assertion:

```
MapID: m3_ax1quad
Target: quad(:emp/{EMPNO},rdf:type,:Employee,
               :prov/mkey-m3/okey-p[:Programmer,:Employee
                    ]/dkey-{provsql}).
Source: SELECT * FROM EMP WHERE JOB='PROGR'
```

Such mapping encodes also the ontology axioms that have been used to derive it. In particular, it encodes the path

`okey-p[:Programmer,:Employee]`

Observe that, although the SQL query in `m3_ax1quad` is contained in the SQL query in `m1quad`, the target parts of such mappings do not coincide. Therefore, the added mapping is *not* redundant, and it will not be removed by the function optimizeM.

The input query gets rewritten into an equivalent query containing *named graphs patterns*:

```
SELECT ?eName ?dName ?p1 ?p2 ?p3 ?p4 WHERE {
   GRAPH ?p1 {?e rdf:type :Employee.}
   GRAPH ?p2 {?e ex:name  ?eName.}
   GRAPH ?p3 {?e ex:dept  ?d .}
   GRAPH ?p4 {?d ex:name dName.}
}
```

The algebra tree for such query is:

```
  SELECT ?eName ?dName ?p1 ?p2 ?p3 ?p4
   JOIN
      quad(?e, rdf:type, :Employee, ?p1).
      quad(?e, ex:name, ?eName, ?p2).
      quad(?e, ex:dept, ?d, ?p3).
      quad(?d, ex:name, ?dName, ?p4).
```

The unfolding and optimizations proceed in the same way as for Example 9. Due to the presence of the mapping m3_ax1quad, the final SQL query will contain a union:

```
SELECT e.ENAME as eName, d.DNAME as dName,
       ’:prov/mkey-m1/dkey-’ || e.provsql as p1,
       ’:prov/mkey-m2/dkey-’ || e.provsql as p2
FROM EMP e, DPT d
WHERE e.DEPTNO = d.DEPTNO
UNION
SELECT e.ENAME as eName, d.DNAME as dName,
       ’:prov/mkey-m3/okey-p[:Programmer,:Employee]/dkey-’
             || e.provsql as p1,
       ’:prov/mkey-m2/dkey-’ || e.provsql as p2
FROM EMP e, DPT d
WHERE e.DEPTNO = d.DEPTNO
   AND JOB=’PROGR’
```

Such union keeps track of the fact that employees can either be derived by the mapping $m1$ alone, or by exploiting $m3$ together with the ontology axiom. ◁