A framework for explaining query answers in *DL-Lite*

Federico Croce and Maurizio Lenzerini

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti"
Sapienza Università di Roma, Rome, Italy.
{croce, lenzerini}@diag.uniroma1.it

Abstract. An Ontology-based Data Access system is constituted by an ontology, namely a description of the concepts and the relations in a domain of interest, a database storing facts about the domain, and a mapping between the data and the ontology. In this paper, we consider ontologies expressed in the popular *DL-Lite* family of Description Logic, and we address the problem of computing explanations for answers to queries in an OBDA system, where queries are either positive, in particular conjunctive queries, or negative, i.e., negation of conjunctive queries. We provide the following contributions: (i) we propose a formal, comprehensive framework of explaining query answers in OBDA systems based on *DL-Lite*; (ii) we present an algorithm that, given a tuple returned as an answer to a positive query, and given a weighting function, examines all the explanations of the answer, and chooses the best explanation according to such function; (iii) we do the same for the answers to negative queries. Notably, on the way to get the latter result, we present what appears to be the first algorithm that computes the answers to negative queries in *DL-Lite*.

1 Introduction

Ontology-based Data Access [10, 11] (OBDA) is a relatively new paradigm for accessing data by posing queries over a formal conceptualization of the domain of interest. The intrinsically declarative nature of this approach has several benefits and peculiar features that characterizes both the functionality of the system managing an OBDA application, and the services exposed by it. Indeed, in its most elementary form, the one we refer to in this paper, an OBDA system can be seen as constituted by two components, called the TBox and the ABox, respectively. The TBox is a description of the concepts and the relations that are relevant in the domain of the information system under consideration, and the ABox is the representation of the data about the domain, i.e., the data regarding the instances of the concepts and the relations.

In a traditional database system, the ABox would represent a complete representation of the data (Closed World Assumption), while in the case of an OBDA system, the ABox is a set of data that is valid for the domain, but is not complete (Open World Assumption): more data can be derived by using the axioms in the

TBox. In what follows, we refer to the pair ⟨TBox, ABox⟩ collectively as "the ontology", and, as usual, we assume that such ontology is expressed in terms of a Description Logic of the *DL-Lite* family [4]. Most of the research carried out in the last years on OBDA has concentrated on query answering [5, 4, 7, 13], i.e., on the design of algorithms for computing the answers to queries posed to an ontology, for the case of basic forms of queries, in particular conjunctive queries. Note, however, that even for such simple queries, this problem is more challenging than the classical query answering problem, precisely because of the presence of the TBox axioms. Consequently, the task of explaining why a certain tuple is an answer to a query is far from being trivial. For instance, if `b` is a student in the ABox, and the TBox sanctions that every student is a person, then explaining the answer "`b` is a person" involves exhibiting both the fact that `b` is a student, and the TBox axiom `Student` is-a `Person`.

The usefulness of query explanation spans from helping out knowledge engineers in debugging ontologies to clarifying the automated reasoning of the system to an end user. Moreover, having the query answers explained is valuable for several related applications. First, it helps in improving Data Quality, because explained answers facilitate the understanding of the underlying database. Secondly, explaining query answers is tightly related to tagging the tuples returned by the system with semantic meaningful context, which might be crucial in Open Data publishing. Furthermore, many researches explored other interesting fields in which this topic could be relevant. For example, [12, 8] propose a justification-based explanation mechanism, i.e. a form of explanation for inclusion axioms rather than for answers to a query.

The problem of providing explanations for answers in *DL-Lite* has been addressed in two seminal papers [2, 6], where a specific technique for this problem is proposed, based on the fundamental assumption that explanations are strictly related to deductions. The main goal of this paper is to present a general framework for explaining conjunctive query answers in an OBDA system, where queries are either positive, or negative, i.e., negation of conjunctive queries. In particular, we provide the following contributions.

- We describe a formal, comprehensive framework for explaining query answers in OBDA systems based on *DL-Lite*. Our framework is inspired by the one described in [2, 6], but has important differences. First, our framework aims at defining all possible explanations of a query answer, whereas the approach followed by [2, 6] is based on choosing some explanations, and ignoring others. For example, in their approach, an explanation that is longer than another one will never be returned by the proposed algorithm. On the contrary, in our framework, one can decide that an explanation exposing a few facts in the ABox is preferred to an explanation disclosing more ABox axioms, even if the latter is shorter. Secondly, the framework envisions the use of a function that is able to associate to each explanation a weight, so as to compare different explanations of the same answer. Thirdly, we base our notion of explanation on a concept of variant of ABoxes, in such a way that ABoxes that are variant of each other give the same explanations.

– We present a generic algorithm that, given a tuple returned as an answer to a positive query, explores all the explanations of the answer according to the ontology, and is able to choose the best explanation in accordance with a predefined weighting function.
– We do the same for the answers to negative queries. Notably, on the way to get the latter result, we present what appears to be the first algorithm that computes the answers to negative queries in *DL-Lite*.

We believe that the proposed framework brings several advantages both in terms of quality (i.e., completeness) of the derived explanations, and in terms of flexibility with respect to the most preferred explanations. Indeed, the task of choosing the best explanation for the answers of a query is intrinsically subjective, so that the end result should be user-centered and customizable.

The paper is organized as follows. In section 2 we illustrate the background knowledge that we consider essential for a smooth reading of the paper. In section 3 we present the framework for explaining query answers in *DL-Lite*. In section 4 we describe a complete procedure for computing explanations to positive queries, whereas in section 5 we do the same for negative queries. Finally, we conclude the paper in section 6.

2 Background

In this section we provide the background notions and techniques that we refer to in the rest of the paper. In particular, we introduce the *DL-Lite* family [4] of Description Logics which is a well-known set of logic-based ontology languages aiming at an optimal trade-off between expressiveness and computational complexity of the reasoning services. Then, we recall some basic notions about conjunctive queries in the context of OBDA, we introduce the notion of negative conjunctive query, and finally, we discuss the algorithm described in [4], that deals with rewriting a conjunctive query in order to compute its answers.

The* DL-Lite *family of Description Logics. Description Logics (DLs) [1, 9, 14] represents a domain of interest in terms of *concepts*, denoting sets of objects, *roles*, denoting binary relations between (instances of) concepts, and *attributes*, denoting binary relations between concepts and value sets. In this paper, we refer to the logic *DL-Lite*$_{\mathcal{A}}$ [4][1], but our approach is valid for all logics of the *DL-Lite* family.

Concepts and roles in *DL-Lite*$_{\mathcal{A}}$ are formed according to the following syntax:

$$B \longrightarrow A \mid \exists R \qquad R \longrightarrow P \mid P^{-}$$

$$C \longrightarrow B \mid \neg B \qquad E \longrightarrow R \mid \neg R$$

where A, B, and C denote an *atomic concept*, a *basic concept*, and a *general concept*, respectively, whilst P denotes an *atomic role*, and R a *general role*.

[1] For the sake of simplicity, we do not deal with attributes in this paper, but they can be added without any problem.

Intuitively, P^- represents the *inverse* of the role P, $\exists P$ (resp. $\exists P^-$) denotes the projection on the first (resp. second) component of the role P (resp. P^-), and $\neg B$ (resp. $\neg R$) denotes the complement of B (resp. of R).

A *DL-Lite*$_\mathcal{A}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a formal description of a domain of interest expressed in *DL-Lite*$_\mathcal{A}$ constituted of the two sets of axioms $\mathcal{T}$, and $\mathcal{A}$. The TBox $\mathcal{T}$ represents the intensional knowledge regarding the domain, i.e., universally quantified statements about the concepts and the roles used in $\mathcal{O}$, and the ABox $\mathcal{A}$ represents the extensional knowledge, i.e., ground statements about individuals. More precisely, each TBox axiom has one of the following forms: $B \sqsubseteq C$ (concept inclusion), $R \sqsubseteq E$ (role inclusion), and (`funct` R) (global functionality of the role R). More details about the axioms allowed in *DL-Lite*$_\mathcal{A}$, and the semantics of the language can be found in [4].

Conjunctive Queries. A *conjunctive query* (CQ) over a *DL-Lite*$_\mathcal{A}$ ontology has the form:

$$Q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\ conj(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are variable vectors, $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms, each one of the form $A(z)$ or $P(z_1, z_2)$, with A and P atomic concepts and roles of the ontology, respectively, and z,z_1,z_2 are terms, i.e., either individual constants in the ontology or variables in $\boldsymbol{x}$ or $\boldsymbol{y}$. The variables appearing in $\boldsymbol{x}$ are called *distinguished* and represents the output of the query, while those appearing in y are called *non-distinguished*, and are existentially quantified. The cardinality of $\boldsymbol{x}$ is the arity of the query, and, for the sake of simplicity, in what follows we assume it is greater than 0, although all results of the paper still holds for queries of arity 0. When it is irrelevant to indicate which are the non-distinguished variables, we simply write $Q(\boldsymbol{x})$ to refer to the query $Q(\boldsymbol{x})$ instead of $Q(\boldsymbol{x}, \boldsymbol{y})$. An atom with only constants is called *ground*.

The basic reasoning service we are dealing with in this paper is *conjunctive query answering*: given an ontology $\mathcal{O}$ and a conjunctive query $Q(\boldsymbol{x})$ over $\mathcal{O}$, compute the *certain answers* to Q over $\mathcal{O}$, the tuples $\boldsymbol{c}$ of individuals in $\mathcal{O}$ such that $\mathcal{O} \models Q(\boldsymbol{c})$, i.e., $Q(\boldsymbol{c})$ is true in **every** model of $\mathcal{O}$. Here, $Q(\boldsymbol{c})$ denotes the formula obtained from $Q(\boldsymbol{x})$ by substituting every $x_i \in \boldsymbol{x}$ with $c_i \in \boldsymbol{c}$.

In this paper, we will also consider negative conjunctive queries, written in the form $\neg Q(\boldsymbol{x}, \boldsymbol{y})$, where $Q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\ conj(\boldsymbol{x}, \boldsymbol{y})$ is a CQ. The certain answers to $\neg Q(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{O}$ is the set of tuples $\boldsymbol{c}$ of individuals in $\mathcal{O}$ such that $\mathcal{O} \models \neg Q(\boldsymbol{c})$, i.e., $\mathcal{O} \models \neg \exists \boldsymbol{y}\ conj(\boldsymbol{c}, \boldsymbol{y})$, which is equivalent to state that $\exists \boldsymbol{y}\ conj(\boldsymbol{c}, \boldsymbol{y})$ is false in **every** model of $\mathcal{O}$. Note, due to the Open World Assumption, the certain answers to $\neg Q(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{O}$ are in general different from the complement of the certain answers to $Q(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{O}$. Note also that, if $Q(\boldsymbol{x}, \boldsymbol{y})$ is empty w.r.t. $\mathcal{O}$ (i.e., it returns the empty answer in all the models of $\mathcal{O}$), then all tuples of individuals in $\mathcal{O}$ will be a certain answer of $\neg Q(\boldsymbol{x}, \boldsymbol{y})$. For this reason, in the following we will assume to deal with negative queries $\neg Q(\boldsymbol{x}, \boldsymbol{y})$ such that $Q(\boldsymbol{x}, \boldsymbol{y})$ is non-empty.

***Computing certain answers in* DL-Lite$_\mathcal{A}$.** We refer to [4] for the method we use for computing the certain answers to conjunctive queries in the *DL-Lite* family. The method is based on first rewriting the original query into a set of alternate queries, then by evaluating these queries over the ABox of the ontology treated as a closed database, and finally by returning the union of the results. The rewriting of the query with respect to the ontology is carried out by a combination of applications of the following two fundamental steps:

- *Replacement*: a *replacement step* s w.r.t. a query Q and a *TBox* $\mathcal{T}$ can be applied to an atom α of the query when the corresponding predicate appears on the right hand side of an inclusion axiom in $\mathcal{T}$. It returns a new query (that we say is produced by s from Q) with atom α replaced by another one, using the predicate appearing on the left hand side of the inclusion axiom in $\mathcal{T}$.
- *Unification*: a *unification step* s w.r.t. a query Q and a *TBox* $\mathcal{T}$ produces a query obtained by merging two atoms that have the same predicate and such that the corresponding arguments can be unified, and applying the unification to all atoms of the query.

For the purpose of this paper, we will usually refer to either the *replacement* or the *unification* step by the more general term of *reformulation step*. Once a query is rewritten according to the above rules, all queries resulting from the process are inserted into a set, that constitute the rewriting. All the queries in the rewriting are then evaluated over the ABox, by resorting to the well-known notion of *homomorphisms*.

As for negative queries, we are not aware of any paper illutrating an algorithms for computing the certain answers of such queries. The result presented in Section 5 will implicitely provide what appears to be the first algorithm that computes the answers to negative queries in *DL-Lite*.

3 Framework

In this section we describe our framework for explaining query answers in the *DL-Lite* family of Description Logics. We start with a set of definitions that are meant to provide the notions we will use for presenting our proposal.

Definition 1. *Given a* TBox $\mathcal{T}$*, and a conjunctive query* Q*, a* $(Q, \mathcal{T})$-deductive path *is a sequence* $\langle Q_0, s_1, Q_1, s_2, \ldots, Q_n \rangle$*, where* $n \geq 0$*,* $Q_0 = Q$*, and for each* $i = 1, \ldots, n$*,* s_i *is a reformulation step that produces* Q_i *from* Q_{i-1}*.*

Intuitively, a $(Q, \mathcal{T})$-deductive path $\langle Q_0, s_1, Q_1, s_2, \ldots, Q_n \rangle$ encodes the chain of reasoning justifying the fact that in order to prove that $\boldsymbol{t}$ is a certain answer of Q_0 with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ is sufficient to prove that $\boldsymbol{t}$ is a certain answer of Q_n with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$.

With the goal of illustrating an example of a deductive path, consider the following ontology.

$\mathcal{T} = \{$ $\texttt{ForeignStud} \sqsubseteq \texttt{Student}$, $\texttt{AttendedBy} \sqsubseteq \texttt{Attends}^-$,
$\texttt{Attends}^- \sqsubseteq \texttt{AttendedBy}$, $\exists\texttt{HasCSTopic} \sqsubseteq \exists\texttt{HasTopic}$,
$\texttt{Student} \sqsubseteq \neg\texttt{Professor}$, $\texttt{AssociateProf} \sqsubseteq \texttt{Professor}$ $\}$
$\mathcal{A} = \{$ $\texttt{ForeignStud(Ann)}$, $\texttt{AttendedBy(DB,Ann)}$,
$\texttt{HasCSTopic(DB,SQL)}$, $\texttt{HasCSTopic(DB,ER)}$ $\}$

Now, consider the query:

$Q(x, y) \leftarrow \texttt{Student}(x), \texttt{Attends}(x, y), \texttt{HasTopic}(y, z)$

It is easy to see that, for example, $\langle Q, s_1, Q_1, s_2, Q_2, s_3, Q_3\rangle$ is a $(Q, \mathcal{T})$-deductive path, where

- s_1 is the reformulation step based on $\texttt{ForeignStud} \sqsubseteq \texttt{Student}$ that produces $Q_1(x, y) \leftarrow \texttt{ForeignStud}(x), \texttt{Attends}(x, y), \texttt{HasTopic}(y, z)$ from Q,
- s_2 is the reformulation step based on $\exists\texttt{HasCSTopic} \sqsubseteq \exists\texttt{HasTopic}$ that produces $Q_2(x, y) \leftarrow \texttt{ForeignStud}(x), \texttt{Attends}(x, y), \texttt{HasCSTopic}(y, z)$ from Q_1
- s_3 is the reformulation step based on $\texttt{AttendedBy} \sqsubseteq \texttt{Attends}^-$ that produces $Q_3(x, y) \leftarrow \texttt{ForeignStud}(x), \texttt{AttendedB}(y, x), \texttt{HasCSTopic}(y, z)$ from Q_2.

3.1 Explanations for positive queries

The most obvious way to define an explanation for $Q(\boldsymbol{t})$ with respect to the ontology $\langle \mathcal{T}, \mathcal{A}\rangle$ is to see it as a $(Q, \mathcal{T})$-deductive path $\langle Q, s_1, Q_1, s_2, \ldots, Q_n\rangle$ associated to a homomorphism from $Q_n(\boldsymbol{t})$ to $\mathcal{A}$. Indeed, the presence of the homomorphism proves that $\mathcal{A} \models Q_n(\boldsymbol{t})$, and the $(Q, \mathcal{T})$-deductive path is a chain of deduction explaining how to conclude $\langle \mathcal{T}, \mathcal{A}\rangle \models Q(\boldsymbol{t})$ from $\mathcal{A} \models Q_n(\boldsymbol{t})$.

If we adopt this approach in the above example, one can verify that the $(Q, \mathcal{T})$-deductive path $\langle Q, s_1, Q_1, s_2, Q_2, s_2, Q_3\rangle$, associated for example to the homomorphism $\{x \mapsto \texttt{Ann}, y \mapsto \texttt{DB}, z \mapsto \texttt{SQL}\}$ from Q_3 to $\mathcal{A}$ is actually an explanation for $Q(\texttt{Ann}, \texttt{DB})$.

However, our notion of explanation is more articulated. First, we will rely only on deductive paths that are not redundant, i.e., that do not contain identical subpaths. More precisely, a $(Q, \mathcal{T})$-deductive path $\langle Q, s_1, Q_1, s_2, \ldots, Q_n\rangle$ is said to be non-redundant if for all $i \neq j \in \{1, \ldots, n\}$, we have that $Q_i \neq Q$, and $Q_i \neq Q_j$. Second, in order to explain $Q(\boldsymbol{t})$ with respect to $\langle \mathcal{T}, \mathcal{A}\rangle$, our approach relies not only on $(Q, \mathcal{T})$-deductive paths $\langle Q, s_1, Q_1, s_2, \ldots, Q_n\rangle$ such that $\mathcal{A} \models Q_n(\boldsymbol{t})$, but also on $(Q, \mathcal{T})$-deductive paths $\langle Q, s_1, Q_1, s_2, \ldots, Q_n\rangle$ such that $\mathcal{A}' \models Q_n(\boldsymbol{t})$, where $\mathcal{A}$ and $\mathcal{A}'$ are in a certain mutual relationship, i.e., $\mathcal{A}'$ is a variant of $\mathcal{A}$ with respect to $\mathcal{T}$. Intuitively, by stating that $\mathcal{A}'$ is a variant of $\mathcal{A}$ with respect to $\mathcal{T}$, we sanction that they are indistinguishable from the point of view of a user posing queries to the ontology. The consequence is that, in order to explain a certain answer, we can use $\mathcal{A}$ and $\mathcal{A}'$ interchangeably. For a specific definition of variant, tailored for *DL-Lite* ontologies, we refer the reader to the last part of this section. Here, we want to notice that the notion of variant ABoxes will allow us to consider explanations that are shorter than in the usual approaches. We are now ready to present the definition of $\langle \mathcal{T}, \mathcal{A}\rangle$-explanation for $Q(\boldsymbol{t})$ in our approach.

Definition 2. *A* $(\mathcal{T}, \mathcal{A})$-explanation *for* $Q(\boldsymbol{t})$ *is a pair* $\langle \Pi, \Gamma \rangle$*, where*

- Π *is a non-redundant* $(Q, \mathcal{T})$*-deductive path of the form* $\langle Q(\boldsymbol{x}), s_1, Q_2(\boldsymbol{x}), s_2, \ldots, Q_n(\boldsymbol{x}) \rangle$*, and*
- Γ *is the image* $Q_n(\boldsymbol{t}, \boldsymbol{t'})$ *of a homomorphism from* $Q_n(\boldsymbol{x}, \boldsymbol{y})$ *to any ABox that is a* $\mathcal{T}$*-variant of* $\mathcal{A}$*.*

In a $(\mathcal{T}, \mathcal{A})$*-explanation* $\langle \Pi, \Gamma \rangle$*, the ground formula constituted by the atoms in* Γ *is called an* explanatory seed *for* Π*.*

Intuitively, the pattern represented by Γ provides the reason why $\boldsymbol{t}$ satisfies $Q_n(\boldsymbol{x})$ w.r.t. the ABox $\mathcal{A}$, and the $(Q, \mathcal{T})$-deductive path provides explanation why from this pattern we can conclude that $\boldsymbol{t}$ is a certain answer to the query Q. Referring to the example above, it is immediate to verify that both the pair $\langle \Pi_1, \Gamma_1 \rangle$, and the pair $\langle \Pi_1, \Gamma_2 \rangle$ are $(\mathcal{T}, \mathcal{A})$-explanations for $Q(\texttt{Ann}, \texttt{DB})$, where

- $\Pi_1 = \langle Q, s_1, Q_1, s_2, Q_2, s_3, Q_3 \rangle$,
- $\Gamma_1 = \{$ `ForeignStud(Ann)`, `AttendedBy(DB,Ann)`, `HasCSTopic(DB,SQL)` $\}$,
- $\Gamma_2 = \{$ `ForeignStud(Ann)`, `AttendedBy(DB,Ann)`, `HasCSTopic(DB,ER)` $\}$.

Indeed, Γ_1 and Γ_2 are the images of two homomorphisms from Q_3 to $\mathcal{A}$. Note, however, that, if the ABox

$\mathcal{A}' = \{$ `ForeignStud(Ann)`, `Attends(Ann,DB)`,
`HasCSTopic(DB,SQL)`, `HasCSTopic(DB,ER)` $\}$

is a $\mathcal{T}$-variant of $\mathcal{A}$, then another $(\mathcal{T}, \mathcal{A})$-explanation is, for example, $\langle \Pi_2, \Gamma_3 \rangle$, where

- $\Pi_2 = \langle Q, s_1, Q_1, s_2, Q_2 \rangle$,
- $\Gamma_3 = \{$ `ForeignStud(Ann)`, `Attends(Ann,DB)`, `HasCSTopic(DB,SQL)` $\}$

since Γ_3 is the image of a homomorphism from Q_2 to $\mathcal{A}'$.

3.2 Explanations for negative queries

In order to define the notion of explanation for negative queries in our approach, we need to introduce a few concepts.

The first one is the concept of disjointness step as a new deduction step when reasoning about a query. Intuitively, a *disjointness step* is applied to a query Q and a disjoint axiom of the form $\beta_1 \sqsubseteq \neg\beta_2$ in $\mathcal{T}$, when β_i $(i \in \{1, 2\})$ unifies with an atom in Q by means of the unification ϕ. The application of such a disjointness step produces the query composed of the atom γ, obtained by applying the unification ϕ to β_j ($j \in \{1, 2\}$, and $j \neq i$). The second notion is the one of subtuple: a tuple $\boldsymbol{t'}$ is called a subtuple of $\boldsymbol{t}$ if every element of $\boldsymbol{t'}$ appear also in $\boldsymbol{t}$. Finally, the third notion is the one of reverse $\langle Q, \mathcal{T} \rangle$-deductive path, defined as follows.

Definition 3. *Given a* TBox $\mathcal{T}$*, and a conjunctive query* Q*, a* reverse $(Q, \mathcal{T})$-deductive path *is a sequence* $\langle \neg Q_n, s_n, \neg Q_{n-1}, \ldots, s_1, \neg Q_0 \rangle$*, where* $n \geq 0$*,* $Q_n = Q$*, and* $\langle Q_0, s_1, Q_1, s_2, \ldots, Q_n \rangle$ *is a* $(Q_0, \mathcal{T})$*-deductive path.*

Analogously to $(Q, \mathcal{T})$-deductive paths, a reverse $(Q, \mathcal{T})$-deductive path $\langle \neg Q_n, s_n, \neg Q_{n-1}, \ldots, s_1, \neg Q_0 \rangle$ encodes the chain of reasoning justifying the fact that in order to prove that $\boldsymbol{t}$ is a certain answer of $\neg Q_n$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ is sufficient to prove that $\boldsymbol{t}$ is a certain answer of $\neg Q_0$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$.

Definition 4. *A $(\mathcal{T}, \mathcal{A})$-explanation for $\neg Q(\boldsymbol{t})$ is a pair $\langle \Sigma, \Gamma \rangle$, where Σ is a sequence of the form $\langle \neg Q, s_1, \neg Q_2, s_2, \ldots, \neg Q_m, s_m, Q_{m+1}, s_{m+2}, \ldots Q_{m+n} \rangle$ such that*

- *$m, n \geq 0$,*
- *$\langle \neg Q, s_1, \neg Q_2, s_2, \ldots, \neg Q_m \rangle$ is a reverse $(Q, \mathcal{T})$-deductive path,*
- *s_m is a disjointness step that produces Q_{m+1} from $\neg Q_m$, and*
- *$\langle \langle Q_{m+1}, s_{m+2}, \ldots, Q_{m+n} \rangle, \Gamma \rangle$ is a $(\mathcal{T}, \mathcal{A})$-explanation for $Q_{m+1}(\boldsymbol{t'})$, where $\boldsymbol{t'}$ is a subtuple of $\boldsymbol{t}$.*

Intuitively, $\langle \langle Q_{m+1}, s_{m+2}, \ldots, Q_{m+n} \rangle, \Gamma \rangle$ explains why $\boldsymbol{t'}$ is a certain answer of Q_{m+1}, s_m is the disjointness step proving that $\boldsymbol{t}$ is a certain answer of $\neg Q_m$ by exploiting the fact that $\boldsymbol{t'}$ is a certain answer of Q_{m+1}, and the reverse $(Q, \mathcal{T})$-deductive path $\langle \neg Q, s_1, \neg Q_2, s_2, \ldots, \neg Q_m \rangle$ proves that $\boldsymbol{t}$ is a certain answer of $\neg Q$. Coming back to the example, we can easily verify that, given the query $Q(x) \leftarrow \texttt{AssociateProf}(x), \texttt{Teaches}(x, y)$, a $(\mathcal{T}, \mathcal{A})$-explanation for $\neg Q(\texttt{Ann})$ is $\langle \langle \neg Q, s_1, \neg Q_1, s_0, Q_2, s_2, Q_3 \rangle, \{\texttt{ForeignStud(Ann)}\} \rangle$, where Q_1, Q_2 and Q_3 are defined as follows

- $Q_1 \leftarrow \texttt{Professor}(x), \texttt{Teaches}(x, y)$
- $Q_2(x) \leftarrow \texttt{Student}(x)$,
- $Q_3(x) \leftarrow \texttt{ForeignStud}(x)$,

s_1 is the reformulation step that produces Q_1 from Q, s_0 is the disjointness step based on the axiom $\texttt{Student} \sqsubseteq \neg\texttt{Professor}$ to produce Q_2 from $\neg Q_1$, s_2 is the reformulation step that produces Q_3 from Q_2.

3.3 The notion of variant in *DL-Lite*

While we left the notion of variant generic in the above considerations, we provide here a specific formalization of the notion of $\mathcal{T}$-variant for the case of *DL-Lite* ontologies. In what follows, we denote with $\mathcal{E}_\mathcal{T}$ the set of *DL-Lite* assertions of the form $E_1 \equiv E_2$ (where E_1, E_2 are either both concepts or both roles) that are logically implied by $\mathcal{T}$. Also, we say that two ground atoms α, β are $\mathcal{T}$-equivalent if $\mathcal{T} \models \alpha \equiv \beta$.

Definition 5. *If $\mathcal{T}$ is a TBox, and $\mathcal{A}, \mathcal{A}'$ are two ABoxes, then $\mathcal{A}$ is a $\mathcal{T}$-variant of $\mathcal{A}'$ if $\mathcal{A}'$ can be obtained from $\mathcal{A}$ by a set of substitutions of atoms with $\mathcal{T}$-equivalent atoms.*

In other words, $\mathcal{A}$ and $\mathcal{A}'$ are $\mathcal{T}$-variant when their logical equivalence can be proved by using only pairwise $\mathcal{T}$-equivalences of atoms.

Coming back to the example, by using the notion of variant just presented, we can verify that another explanation for $Q(\texttt{Ann}, \texttt{DB})$ in our approach

is based on the $(Q, \mathcal{T})$-deductive path $\langle Q, s_1, Q_2, s_2, Q_3 \rangle$, because, although $\mathcal{A} \not\models Q_3(\texttt{Ann}, \texttt{DB})$, the following ABox

$\mathcal{A}' = \{$ `ForeignStud(Ann)`, `Attends(Ann,DB)`, `HasCSTopic(DB,SQL)` $\}$

is a $\mathcal{T}$-variant of $\mathcal{A}$, and is such that $\{x \mapsto \texttt{Ann}, y \mapsto \texttt{DB}, z \mapsto \texttt{SQL}\}$ is a homomorphism from Q_3 to $\mathcal{A}'$, thus proving that $\mathcal{A}' \models Q_3(\texttt{Ann}, \texttt{DB})$. Notice that ABoxes can be obviously seen as conjunctive queries, in particular, ground conjunctive queries, and therefore the notion of deductive path can be applied to ABoxes as well. This property is exploited in the following theorem, that will be used in the technical development presented in the rest of the paper.

Theorem 1. *The ABox $\mathcal{A}$ is a $\mathcal{T}$-variant of the ABox $\mathcal{A}'$ if and only if there is an* $(\mathcal{A}, \mathcal{E}_\mathcal{T})$-deductive path *of the form* $\langle \mathcal{A}, s_1, \mathcal{A}_1, s_2, \ldots, \mathcal{A}' \rangle$.

Proof. If-part. Suppose that there is an $(\mathcal{A}, \mathcal{E}_\mathcal{T})$-*deductive path* that has the form $\langle \mathcal{A}_0, s_1, \mathcal{A}_1, s_2, \ldots, \mathcal{A}_n \rangle$ where $\mathcal{A} = \mathcal{A}_0$, and $\mathcal{A}_n = \mathcal{A}'$. We show by induction on the length n of such path that $\mathcal{A}$ is a $\mathcal{T}$-variant of $\mathcal{A}'$. If n is 0, then the thesis trivially holds. If n is greater than 0, then $\langle \mathcal{A}_1, s_2, \ldots, \mathcal{A}' \rangle$ is an $(\mathcal{A}_1, \mathcal{E}_\mathcal{T})$-*deductive path* whose length is $n - 1$. By induction hypothesis, we have that $\mathcal{A}_1$ is a $\mathcal{T}$-variant of $\mathcal{A}'$, i.e., $\mathcal{A}'$ can be obtained from $\mathcal{A}_1$ by a set of substitutions of equivalent atoms. It remains to show that $\mathcal{A} = \mathcal{A}_0$ is a $\mathcal{T}$-variant of $\mathcal{A}_1$, thus showing that $\mathcal{A}'$ can be obtained from $\mathcal{A}$ by a set of substitutions of equivalent atoms. By the definition of $(\mathcal{A}, \mathcal{E}_\mathcal{T})$-*deductive path*, we have that $\mathcal{A}_1$ is obtained from $\mathcal{A}$ by means of a reformulation step that substitutes an atom α in $\mathcal{A}$ with an atom β using an axiom of the form $\alpha \equiv \beta$ in $\mathcal{E}_\mathcal{T}$. This obviously implies that $\mathcal{A}$ is a $\mathcal{T}$-variant of $\mathcal{A}_1$.

Only-if-part. Suppose that $\mathcal{A}$ is a $\mathcal{T}$-variant of the ABox $\mathcal{A}'$. We proceed by induction on the number n of atoms in $\mathcal{A}$ that we have to substitute in order to obtain $\mathcal{A}'$. If $n = 0$, then the thesis trivially holds. If n is greater than 0, then let $\alpha \in \mathcal{A}$ be one of the atoms to be substituted with $\beta \in \mathcal{A}'$ such that $\mathcal{T} \models \alpha \equiv \beta$, and let $\mathcal{A}_1$ be the ABox obtained from $\mathcal{A}$ by means of such substitution. By induction hypothesis, there is an $(\mathcal{A}, \mathcal{E}_\mathcal{T})$-*deductive path* of the form $\langle \mathcal{A}_1, s_2, \mathcal{A}_2, s_3, \ldots, \mathcal{A}' \rangle$, and it is immediate to verify that from $\mathcal{T} \models \alpha \equiv \beta$ we can derive a reformulation step that produces $\mathcal{A}_1$ from $\mathcal{A}$, thus proving that there is an $(\mathcal{A}, \mathcal{E}_\mathcal{T})$-*deductive path* of the form $\langle \mathcal{A}, s_1, \mathcal{A}_1, s_2, \ldots, \mathcal{A}' \rangle$. □

3.4 Weighting explanations

As we said in the introduction, the framework presented in this paper intentionally leaves unspecified the strategy for evaluating the multiple explanations that are computed for the same tuple, query, and ontology. This is reflected by the fact that the framework envisions the existence of a function that is able to associate to each explanation a weight, so as to compare different explanations of the same answer. Although in existing approaches the weighting function is based essentially on the length of the deduction corresponding to the explanation, we argue that such function should reflect the idea that choosing the best explanation for the answers of a query is intrinsically subjective, and can be

characterized by different properties in different contexts. For this reason, in the next section we will keep the weighting function completely generic.

4 Computing explanations for positive queries

Given a query Q, a tuple $\boldsymbol{t}$, and an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, the algorithm Explain essentially builds a tree τ whose nodes are in one-to-one correspondence with the non-redundant $(Q, \mathcal{T})$-deductive paths, and computes all $(\mathcal{T}, \mathcal{A})$-explanations of $Q(\boldsymbol{t})$ based on such paths. For each such explanation, it also computes the associated weight, and the final result derives from the one with the maximum weight. The algorithm makes use of the following notions.

- For each node n of the tree τ:
  - $\mathsf{father}(n)$ denotes the father of node n in the tree τ; $\mathsf{father}(n)$ is assumed to be *null* if n is the root.
  - $\mathsf{equivEdge}(n, m)$ is true if n and m are nodes different from *null*, n is the father of m in τ, and the edge from n to m is labeled with a reformulation step based on $\alpha \sqsubseteq \beta$ such that $\mathcal{T} \models \beta \sqsubseteq \alpha$; it is false otherwise.
  - $\mathsf{query}(n)$ is set by the algorithm in such a way to denote the query associated to n;
  - $\mathsf{images}(n)$ is set by the algorithm in such a way to denote the set of images of all homomorphisms from $\mathsf{query}(n)$ that are relevant for computing the explanations; note that every image is a set of ground facts, and that $\mathsf{images}(\mathit{null})$ is assumed to be empty;
- best is a record managed by the algorithm in such a way that it stores information about the best explanation currently found. The record contains three items: best.node stores the node n corresponding to the deductive path representing the explanation; best.image stores the set of facts constituting the image of the homomorphism from $\mathsf{query}(n)$ to $\mathcal{A}$ which gives the best explanation, and best.weight is the value of the weight of such explanation.
- $\mathsf{transferImages}(n, m)$ denotes the process of transferring the sets of facts stored in $\mathsf{images}(n)$ to $\mathsf{images}(m)$, in the case where there is an edge e connecting n and m such that $\mathsf{equivEdge}(n, m)$ is true. If $\mathsf{equivEdge}(n, m)$ is false, then $\mathsf{transferImages}(n, m)$ has no effect. Let s be the reformulation step that is the label of e, and let s be based on $\alpha \equiv \beta$, where α is the predicate of an atom in n, and β is the predicate of an atom in m. Then, for each $\gamma_1 \in \mathsf{images}(n)$, insert $\gamma_2 \in \mathsf{images}(m)$, where γ_2 is obtained from γ_1 by substituting the α-atom with the corresponding β-atom. For example, if $\mathsf{images}(n) = \{ \{ \texttt{ForeignStud(Ann)}, \texttt{AttendedBy(DB,Ann)}, \texttt{HasCSTopic(DB,SQL)} \}, \{ \texttt{ForeignStud(Ann)}, \texttt{AttendedBy(DB,Ann)}, \texttt{HasCSTopic(DB,ER)} \} \}$, and there is an edge e connecting n and m such that $\mathsf{equivEdge}(n, m)$ is true, and the reformulation step that is the label of e is based on $\texttt{AttendedBy} \sqsubseteq \texttt{Attends}^-$ such that $\mathcal{T} \models \texttt{AttendedBy} \equiv \texttt{Attends}^-$, then we have that the execution of $\mathsf{transferImages}(n, m)$ results into $\mathsf{images}(m) =$

{ { ForeignStud(Ann), Attends(Ann,DB), HasCSTopic(DB,SQL) },
{ ForeignStud(Ann), Attends(Ann,DB), HasCSTopic(DB,ER) } }.

- If n is a node of τ, and $\gamma \in \mathsf{images}(n)$, then $\mathsf{ComputeWeight}(n, \gamma)$ computes the weight associated to the explanation represented by n and γ, and stores it as $\mathsf{weight}(\gamma)$.

We are now ready to present the algorithm, whose main function is Explain. Given query Q, tuple $\boldsymbol{t}$, and ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $\langle \mathcal{T}, \mathcal{A} \rangle \models Q(\boldsymbol{t})$, such function returns the record best storing all relevant data used to recontruct the best $(\mathcal{T}, \mathcal{A})$-explanation of $Q(\boldsymbol{t})$. As we said before, Explain defines the tree τ initially constituted only by the root r, and then complete the construction of the tree by means of the function BuildTree called on r. The goal of the latter function is indeed to build the tree in such a way that each node n of τ have the associated data described above, namely, $\mathsf{query}(n)$, and $\mathsf{images}(n)$. On each node, the BuildTree also calls the function EvaluateAndPropagate, whose goal is to compute the weight of each explanation associated to the nodes that it visits, to update the record best, if needed, and to propagate the set $\mathsf{images}(n)$ to other nodes of the tree, if needed. In order to compute the weight of the various explanations, it makes use of the function ComputeWeight, that we leave generic: any strategy that associates a positive value to an explanation is valid in our approach. The final result computed by Explain derives from the explanations with the maximum weight.

The following theorems are crucial for proving the correctness of the algorithm.

Theorem 2. *Let τ be the tree built by the execution of $\mathsf{Explain}(Q, \boldsymbol{t}, \mathcal{T}, \mathcal{A})$, and let r be the root of τ. From each path from r to a node of τ, it is possible to derive a non redundant $(Q, \mathcal{T})$-deductive path, and, conversely, from each non redundant $(Q, \mathcal{T})$-deductive path, it is possible to derive a path from r to a node of τ.*

Proof. (Sketch) First part. Let n be a node in τ. The proof is based on induction on the length of the path from r to n. If the length is 0, then the thesis trivially holds. If the length is greater than 0, then there is a node m in τ that is the parent of n, and by the induction hypothesis, we can derive a non redundant $(Q, \mathcal{T})$-deductive path Π_i using the path π_1 from r to m. It is easy to see that we can add a subsequence to Π_i in order to obtain a non redundant $(Q, \mathcal{T})$-deductive path associated to n. *Second part.* Let Π a non redundant $(Q, \mathcal{T})$-deductive path. The proof is based on induction on the length of Π. If the length is 0, then it is immediate to verify that the corresponding path in τ is simply constituted by the root r. If the length is greater than 0, then Π can be seen as a $(Q, \mathcal{T})$-deductive path Π', plus an element corresponding to a reformulation step s. By the induction hypothesis there is a node m for which we can single out a path π from r to m corresponding to the $(Q, \mathcal{T})$-deductive path Π'. Now, using s it is easy to see that we can add en edge to π', and obtain a new path π from r to a node n corresponding to Π. □

Algorithm 1: The algorithm `Explain` for positive query answers.

```
1  Function Explain(Q, t, T, A) : record
2      define τ as a tree with root r
3      query(r) ← Q(t)
4      best ← ⟨null, ∅, 0⟩
5      BuildTree(r)
6      return best

1  Function EvaluateAndPropagate(n)
2      foreach γ ∈ images(n) do
3          ComputeWeight(n, γ)
4          if weight(γ) ≥ best.weight then update best
5      if images(father(n)) = ∅ and equivEdge(father(n), n) then
6          transferImages(n, father(n))
7          EvaluateAndPropagate(father(n))
8      foreach child m of n do
9          if images(m) = ∅ and equivEdge(n, m) then
10             transferImages(n, m)
11             EvaluateAndPropagate(m)

1  Function BuildTree(n)
2      if equivEdge(father(n), n) and images(father(n)) ≠ ∅ then
3          images(n) ← transferImages(father(n), n)
4      else images(n) ← images of homomorphisms from query(n) to A
5      EvaluateAndPropagate(n)
6      foreach reformulation step s that produces Q' from query(n) and such
         that query(n_i) ≠ Q' for every node n_i in the path from n to r do
7          create child m of n and label the edge from n to m with s
8          BuildTree(m)
```

Theorem 3. *Let τ be the tree built by the execution of $\mathsf{Explain}(Q, \boldsymbol{t}, \mathcal{T}, \mathcal{A})$. Then, for each node n of τ such that $\Gamma \in \mathsf{images}(n)$, there is a $(\mathcal{T}, \mathcal{A})$-explanation of $Q(\boldsymbol{t})$ of the form $\langle \Pi, \Gamma \rangle$, where Π is the $(Q, \mathcal{T})$-deductive path corresponding to the path from the root of τ to n.*

Theorem 4. *Let τ be the tree built by the execution of $\mathsf{Explain}(Q, \boldsymbol{t}, \mathcal{T}, \mathcal{A})$. Then, for each $(\mathcal{T}, \mathcal{A})$-explanation of $Q(\boldsymbol{t})$ of the form $\langle \Pi, \Gamma \rangle$, there is a node n of τ such that $\Gamma \in \mathsf{images}(n)$, and the path from the root of τ and n is the one corresponding to the $(Q, \mathcal{T})$-deductive path Π.*

Finally, the following theorem shows the correctness of the function Explain with respect to the goal of computing the best explanation for $Q(\boldsymbol{t})$. The proof proceeds by showing that $\mathsf{Explain}(Q, \boldsymbol{t}, \mathcal{T}, \mathcal{A})$ explores all possible $(\mathcal{T}, \mathcal{A})$-explanations for $Q(\boldsymbol{t})$, computes the corresponding weight, and then returns the one (or anyone) with the maximum weight.

Algorithm 2: The algorithm Explain for negative query answers.

```
1  Function ExplainNegative(Q, 𝒯, 𝒜, t) : ⟨𝒯,𝒜⟩-explanation for ¬Q(t)
2      Initialize NEP as an empty pair ⟨sequence,image⟩
3      foreach disjoint axiom Δ ∈ 𝒯 do
4          Let (g1, g2) be the two atoms of the violating query associated with Δ
5          foreach atom α in Q do
6              Let t' be a subtuple of t
7              Let γ be the predicate associated with α
8              Let η1, η2 be the predicates associated with g1, g2
9              if 𝒯 ⊨ γ ⊑ η1 then
10                 if ⟨𝒯,𝒜⟩ ⊨ g2(t') then
11                     g' = g1, g'' = g2
12             else
13                 if 𝒯 ⊨ γ ⊑ η2 then
14                     if ⟨𝒯,𝒜⟩ ⊨ g1(t') then
15                         g' = g2, g'' = g1
16             if g' and g'' are defined then
17                 ⟨⟨g'', sn, Qn+1, ..., Qm⟩, Γ⟩= Explain(g'', 𝒯, 𝒜, t')
18                 foreach reverse (𝒬,𝒯)-deductive path
                    ⟨¬Q, s1, ¬Q2, s2, ..., ¬g'⟩ do
19                     Let Π =
                        ⟨⟨¬Q, s1, ¬Q2, s2, ..., ¬g', Δ, g'', sn, Qn+1, ..., Qm⟩, Γ⟩ if
                        ComputeWeight(Π) is better than NEP then
20                         Update NEP with Π
21     return NEP
```

Theorem 5. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, Q be a query, and $\boldsymbol{t}$ be a tuple such that $\langle \mathcal{T}, \mathcal{A} \rangle \models Q(\boldsymbol{t})$. Then* Explain$(Q, \boldsymbol{t}, \mathcal{T}, \mathcal{A})$ *computes the best $(\mathcal{T}, \mathcal{A})$-explanation for $Q(\boldsymbol{t})$, according to the strategy represented by the function* ComputeWeight.

5 Computing explanations for negative queries

In this section wepresent the Algorithm 2 that deals with building the negative explanation for a tuple $\boldsymbol{t}$, with respect to a satisfiable ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, and a non-empty query Q. For each disjoint axiom in the input ontology, the procedure checks whether the corresponding violating query has a non-empty evaluation over $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{Q}(\boldsymbol{t}) \rangle$ (in *DL-Lite*, if $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg Q(\boldsymbol{t})$ this has to be true for some violating query). If this is the case, the algorithm builds a negative explanation, and evaluates it with a predetermined evaluation function that plays the same role as the function `ComputeWeight` in the case of positive explanations. The output will be the explanation with the highest evaluation. Specifically, let ω be an arbitrary violating query, g_1, g_2 be their atoms, and η_1, η_2 their corresponding

predicates. For each atom of Q, the algorithm verifies whether its predicate γ is a subset of either η_1 or η_2 in $\mathcal{T}$. Let for instance $\mathcal{T} \models \gamma \sqsubseteq \eta_1$, the algorithm searches for a $\langle \mathcal{T}, \mathcal{A} \rangle$-explanation for the query $g_2(\boldsymbol{t'})$, where $\boldsymbol{t'}$ is a subtuple of $\boldsymbol{t}$. Then, for all the possible reverse $(\mathcal{Q}, \mathcal{T})$-deductive paths that leads to g_1, the algorithm builds and evaluates a negative explanation by connecting the aforementioned deductive and explanation paths. The correctness of the algorithm is sanctioned by the following theorem.

Theorem 6. *Given a conjunctive query Q, a TBox $\mathcal{T}$, an ABox $\mathcal{A}$ and a tuple $\boldsymbol{t}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, Q is not empty over $\langle \mathcal{T}, \mathcal{A} \rangle$, and $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg Q(\boldsymbol{t})$, then Algorithm 2 evaluates all and only the $\langle \mathcal{T}, \mathcal{A} \rangle$-explanations for $\neg Q(\boldsymbol{t})$.*

Proof (sketch). According to a well-known property of the *DL-Lite* language, $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg Q(\boldsymbol{t})$ if and only if $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{Q}(\boldsymbol{t}) \rangle$ is unsatisfiable. For each disjoint axiom of $\mathcal{T}$, consider the corresponding two atoms violating query. A *DL-Lite* ontology is unsatisfiable if and only if there exists a violating query for which the evaluation over the ontology is not empty. Since by hypothesis $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, and the input query Q is not empty over $\langle \mathcal{T}, \mathcal{A} \rangle$, for each violating query ω, it has to be that $\langle \mathcal{T}, \mathcal{A} \rangle \not\models \omega$ and $\langle \mathcal{T}, \mathcal{Q}(\boldsymbol{t}) \rangle \not\models \omega$. As a consequence, let ρ_1, ρ_2 be the two atoms of any violating query, and $\boldsymbol{t'}$ a subtuple of $\boldsymbol{t}$, a negative explanation for $\mathrm{Q}(\boldsymbol{t})$ exists if and only if either $\langle \mathcal{T}, \mathcal{A} \rangle \models \rho_1(\boldsymbol{t'})$ and $\langle \mathcal{T}, \mathcal{Q}(\boldsymbol{t}) \rangle \models \rho_2(\boldsymbol{t'})$, or $\langle \mathcal{T}, \mathcal{A} \rangle \models \rho_2(\boldsymbol{t'})$ and $\langle \mathcal{T}, \mathcal{Q}(\boldsymbol{t}) \rangle \models \rho_1(\boldsymbol{t'})$. The algorithm considers both the above cases and provides explanations for the holding entailments. □

6 Conclusions

In this paper, we addressed the problem of providing explanations both for positive and negative answers to queries over an ontology. In section 3 we introduced a general framework to deal with these issues, and in seciton 4 and 5 we illustrated techniques for inspecting all possible explanations, both for positive and for negative conjunctive queries.

The issue of multiple explanations is addressed by conceiving the use of a weighting function that assigns a weight to every possible explanation, so that they can be compared according to a set of predefined criteria. We have implemented the procedures described in this paper in the Java tool for Ontology Based Data Access MASTRO [3]. Future works involve dealing with the problem of deriving an effective method for visualizing, and exposing to the users the explanations produced with our techniques, as well as analyzing the effect, on the notions and methodologies introduced in this paper, of extending the ontology to languages that are more expressive than *DL-Lite*.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York, NY, USA, 2003.

2. A. Borgida, D. Calvanese, and M. Rodriguez-Muro. *Explanation in the DL-Lite Family of Description Logics*, pages 1440–1457. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The mastro system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, Oct 2007.
5. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artificial Intelligence*, 195:335 – 360, 2013.
6. D. Calvanese, M. Ortiz, M. Simkus, and G. Stefanoni. Reasoning about explanations for negative query answers in dl-lite. *J. Artif. Intell. Res.*, 48:635–669, 2013.
7. T. Eiter, C. Lutz, M. Ortiz, and M. Simkus. Answering in description logics with transitive roles. In *IJCAI*, 2009.
8. M. Horridge. *Justification based explanation in ontologies*. PhD thesis, University of Manchester, UK, 2011.
9. H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78 – 93, 02 1987.
10. M. Lenzerini. Ontology-based data management. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 5–6, New York, NY, USA, 2011. ACM.
11. M. Lenzerini. Managing data through the lens of an ontology. *AI Magazine*, 39(2):65–74, 2018.
12. R. Penaloza and B. Sertkaya. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artificial Intelligence*, 250(Supplement C):80 – 104, 2017.
13. U. Straccia. Towards top-k query answering in description logics: The case of dl-lite. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Logics in Artificial Intelligence*, pages 439–451, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
14. F. van Harmelen, V. Lifschitz, and B. Porter. *Handbook of Knowledge Representation*. Elsevier Science, San Diego, USA, 2007.