Expressive Recursive Answers for Ontological Knowledge Bases

Luca Andolfi, Gianluca Cima, Marco Console, Maurizio Lenzerini

Sapienza University of Rome
{andolfi, cima, console, lenzerini}@diag.uniroma1.it

Abstract

A fundamental use of knowledge bases (KBs) is query answering, i.e., retrieving the information entailed by the KB in response to a user query. When both the KB and the query are specified as logical formulae, the standard form of answer provided to users is certain answers (CAs): tuples of constants that satisfy the query in every model of the KB. Despite their wide adoption, CAs are known to be just a lossy representation of the information that a KB and a query provide. However, while several alternative forms of answers have been proposed, no general consensus has emerged on the most suitable approach to answer queries over ontological KBs.

In this paper, we introduce Regularly Recurrent Answers (RRAs), a novel form of answer for queries over ontological KBs. RRAs support the representation of infinite sets of tuples via a generation mechanism based on regular expressions. Such a simple mechanism allows RRAs to represent a fundamental fragment of the certain information entailed by Unions of Conjunctive Queries over DL-Lite KBs. The contribution of this paper includes the formal definition of RRAs, a formal characterization of their informativeness, and a study of their computational characteristics.

1 Introduction

A *knowledge base (KB)* is a symbolic representation of a domain of interest, formulated in a formal language and equipped with *semantics* (i.e., a set of mathematical objects representing its meaning). One of the most important tasks performed with a KB is, arguably, *query answering* (Calvanese et al. 2007; Bienvenu and Ortiz 2015). At the most general level, query answering amounts to extracting from a KB the information requested by a user (*a query*) and constructing a suitable syntactic object (*an answer*) that represents that information. In fact, one can view query answers simply as representations of the information required by a query and, as such, they come with their own syntax and semantics (i.e., an *answer language*). Depending on the answer language adopted, a *query answering system* can provide more or less accurate answers to a query.

Hereinafter, we adopt the logical approach to KBs (Reiter 1980; Levesque and Lakemeyer 2001) in which information is represented by a *logical theory*, and the *models of this theory* constitute the semantics of the KB. In this approach, a query is defined by a *logical formula*, and answers consist of (a representation of) the *individuals that satisfy this formula* in the models of the KB. When the KB has only a single finite model (e.g., in the case of relational databases (Abiteboul, Hull, and Vianu 1995)), it is customary to answer a query by returning *the set of all tuples of constants* that satisfy it in that model. Such a simple answer language (i.e., finite sets of tuples of constants) usually suffices to convey all the information requested by the queries.

In many application scenarios, however, KBs have multiple (and sometimes infinite) models. This is the case, e.g., for incomplete databases (Abiteboul, Hull, and Vianu 1995), data integration and exchange (Lenzerini 2002), and ontology-based data access and ontology-mediated query answering (Calvanese et al. 2007; Calì, Gottlob, and Kifer 2008; Bienvenu and Ortiz 2015).

In all these settings, the literature often focuses on Boolean queries and, essentially, reduces query answering to logical entailment. In concrete applications, however, users are often interested in more than just yes/no answers. To solve these cases, a prominent approach is to use the same tuple-based answer language and return *certain answers* (CAs), i.e., sets of tuples of constants that satisfy the query in *every* model of the KB. Introduced in the late 1970s (Lipski 1979), CAs have become the de facto standard for query answering over KBs across different domains.

Example 1. *Consider the Description Logic KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$*, where* $\mathcal{A} = \{\mathsf{hS}(\mathit{Ava}, \mathit{Bea})\}$ *and* $\mathcal{T}$ *is defined as*

$$(1)\ \mathsf{E} \sqsubseteq \exists \mathsf{hS} \quad (2)\ \exists \mathsf{hS}^- \sqsubseteq \mathsf{S} \quad (3)\ \mathsf{S} \sqsubseteq \mathsf{E}$$

Informally, $\mathcal{K}$ *states that each employee is assigned to a supervisor* (1)*, every supervisor is an employee* $(2) - (3)$*, and Bea is the supervisor of Ava (*$\mathcal{A}$*).*

To retrieve the information about employees and their supervisors, we could issue the query $q = \{(x, y) \mid \mathsf{hS}(x, y)\}$*. The CAs for* q *over* $\mathcal{K}$ *is the set* $CA(q, \mathcal{K}) = \{(\mathit{Ava}, \mathit{Bea})\}$*.*

Despite their wide adoption, CAs are far from ideal for KBs with multiple models (Libkin 2016). Since the answer language used is too simple, CAs can represent only a small portion of the information that a query actually requires. To see this, consider again Example 1 and observe that the information that *someone* supervises Bea is certain in $\mathcal{K}$ (i.e.,

true in all its models). Clearly, one would expect such information in the answer for q, but there is no mention of it in $CA(q, \mathcal{K})$. Upon inspecting such an answer, users could be even tempted to conclude that Bea has no supervisor, effectively subverting the meaning of the theory and the query.

Limitations of CAs are well known in the literature and have been studied extensively in the context of databases (Console et al. 2020), but they have been left largely unexplored for more complex KBs with only a few notable exceptions. The work in (Borgida, Toman, and Weddell 2016) introduces *referring expressions* (RefExp): an answer language that uses Description Logic (DL) constructs to describe unnamed individuals. While RefExp provide more intuitive (and therefore usable) answers to users, issues arise when one wants to use them to capture more information coming from the query (see Section 3 for additional details). Specifically, to obtain a full answer, one may need to return infinitely many and often redundant such descriptors (Toman and Weddell 2019). While this issue can be partially mitigated via *regular expressions*, the case of *Unions of Conjunctive Queries* (UCQs) and DL languages with *role inclusions* has not been treated in full.

Inspired by the standard approach of *answer-tuples with nulls* for incomplete databases (Imielinski and Lipski 1984), the works in (Lutz and Przybylko 2022, 2023) introduce *minimal partial answers with multi-wildcards* (MPAW), a tuple-based answer language that allows both constants and special symbols (wildcards) to represent *unnamed* individuals. While such symbols can partially mitigate the information loss in CAs, their expressive power remains limited due to necessary syntactic restrictions.

A more systematic study of the informativeness of answer languages for ontological KBs is presented in (Andolfi et al. 2024) where a formal framework for this purpose is introduced. This framework is based on the simple observation that, in general, the behavior of a query q over a KB $\mathcal{K}$ is fully determined by the collection $q(\mathcal{K})$ of sets of tuples of individuals obtained by evaluating q over all the models of $\mathcal{K}$ (see, e.g., (Lipski 1979; Imielinski and Lipski 1984) for an early use of this idea). Using $q(\mathcal{K})$, one can define the *certain knowledge* C entailed by q and $\mathcal{K}$ as the set of all formulae (in a given language) that are satisfied by every element of $q(\mathcal{K})$. A good answer for q over $\mathcal{K}$, in terms of certain knowledge, is any symbolic representation that satisfies (i.e., *preserves*) all the formulae in C. The paper also shows that both CAs and MPAW fail to preserve even a very simple language and propose *certain n-answers* (CnA, with $n \in \mathbb{N}$), a novel answer language that mitigates this issue.

Example 2. *The set of all MPAW for q over $\mathcal{K}$ is $\{(Ava, Bea), (Bea, \star)\}$. Similarly, the set of all C2A to q over $\mathcal{K}$ is $\{(Ava, Bea), (Bea, \star_1), (\star_1, \star_2)\}$. The semantics of both answers is straightforward: the $\star$-symbols represent tuples of individuals (named or unnamed) that certainly satisfy q. The set of MPAW preserves the information that Bea is supervised by some individual $\star$ but not that $\star$ is themselves supervised. This additional information is preserved by the set of C2A, where, however, the information that the supervisor $\star_2$ of $\star_1$ is themselves supervised is lost.*

As one may easily deduce from Example 2, none of the previously discussed languages are able to preserve the certain knowledge entailed by q and $\mathcal{K}$ that can be expressed in the *existential positive fragment of first-order logic* ($\exists Pos$). This is a consequence of a more general result from (Andolfi et al. 2024), which shows that no answer language based on finite sets of tuples can preserve such information. This limitation holds even when considering CQs and *DL-Lite*$_\mathcal{R}$ KBs.

In light of these observations, we propose a novel answer language that we call *regularly recurrent answers (RRAs)*. The **goal of RRAs** is to provide a simple yet effective way to represent the infinite sets of tuples required to properly answer UCQs over ontological KBs. To achieve this goal, RRAs employ a mechanism based on *regular expressions*. We chose regular expressions as the basis for RRAs because they provide a simple, intuitive, and widely accepted mechanism for generating new tuples whose computational properties are well understood. More specifically, each RRA consists of a finite set of *regularly recurrent tuples* (RRTs): pairs of the form $\langle \bar{a}, \rho \rangle$, where $\bar{a}$ is a tuple of terms and ρ is a (set of) regular expression(s) that specifies how to generate other tuples of terms starting from $\bar{a}$.

Example 3. *Consider again Example 1. An RRA R for q over $\mathcal{K}$ consists of RRTs $t_1 = \langle (Ava, Bea), \emptyset \rangle$ and $t_2 = \langle (x, s(x)), \{[x, \langle s^*, Bea \rangle]\} \rangle$. The semantics of t_1 is the set $\{(Ava, Bea)\}$ while that of t_2 is the set of tuples $\{(Bea, s(Bea)), (s(Bea), s(s(Bea))), ...\}$. The semantics of R is simply the union of the semantics of t_1 and t_2 and it is easy to observe that R preserves the certain knowledge entailed by q and $\mathcal{K}$ that can be expressed in $\exists Pos$.*

The main contribution of this paper is the presentation of an answer language of the kind defined in Example 3 and the study of its computational characteristics. Specifically, our contribution is the following: we introduce RRAs, our novel answer language; we show that, for every UCQ q and *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$, there always exists an RRA, that we call *canonical*, whose semantics can be isomorphically mapped into the answers that q obtains over a canonical model of $\mathcal{K}$; we show that canonical RRAs preserve the certain knowledge entailed by q and $\mathcal{K}$ that can be expressed in $\exists Pos$ (as defined in (Andolfi et al. 2024)). Finally, we show that computing canonical RRAs is in PTIME in data complexity in the aforementioned scenario.

It is important to emphasize that RRAs are intended as a foundational theoretical contribution to the study of expressive answer languages. We are aware that, in their current form, RRAs are not yet suited for direct interaction with end users: raw regular expressions are unlikely to align with the expectations and needs of non-experts. Nonetheless, establishing the expressive power required to represent answers provides a solid theoretical basis for further research in this area. In particular, future studies on usability can build on this foundation and draw from the extensive literature on the visual presentation of regular languages (Beck et al. 2017).

Organization Section 2 introduces preliminary notions and provides a concise overview of the framework for informativeness presented in (Andolfi et al. 2024). RRAs are presented in Section 3 where we also compare them to other

languages from the literature. Then Section 4 is devoted to proving informativeness of RRAs for UCQs over *DL-Lite*$_\mathcal{R}$ KBs while Section 5 studies the computational properties of relevant problems. In Section 6 we conclude.

2 Preliminaries

We fix three countably infinite and pairwise disjoint sets of symbols $\Sigma_\mathcal{C}$, $\Sigma_\mathcal{V}$, and $\Sigma_\mathcal{F}$, for *constants*, *variables*, and *unary functions*, respectively. The set $\Sigma_\mathcal{T}$ of *terms* is defined inductively as follows: (*i*) every $t \in \Sigma_\mathcal{C} \cup \Sigma_\mathcal{V}$ is in $\Sigma_\mathcal{T}$; (*ii*) for every $t \in \Sigma_\mathcal{T}$ and $f \in \Sigma_\mathcal{F}$, $f(t) \in \Sigma_\mathcal{F}$.

We will often write a term $f_1(f_2(\dots(f_k(c))\dots))$ simply as $f_1 f_2 \dots f_k(c)$ (dropping parenthesis); abbreviate a sequence of function symbols $f_1 f_2 \dots f_k$ as $\bar{f}$; and use $\bar{g}\bar{f}(t)$ for the term defined by the composition of the sequences of function symbols $\bar{g}$ and $\bar{f}$ applied to the term t. Additionally, given a term $t = f_1 f_2 \dots f_k(k)$, with $k \in \Sigma_\mathcal{C} \cup \Sigma_\mathcal{V}$, we call f_1 the *leading function symbol of* t, the sequence $f_1 f_2 \dots f_k$ of function symbols of t the *word of* t, and c the *base of* t. Finally, given a function $\mu : \Sigma_\mathcal{V} \to \Sigma_\mathcal{V} \cup \Sigma_\mathcal{C}$ and a term $t = \bar{g}(k)$, with $k \in \Sigma_\mathcal{V} \cup \Sigma_\mathcal{C}$, $\mu(t)$ is the term $\bar{g}(\mu(k))$, if k is a variable, and t itself, if k is a constant.

Regular Expressions. A *regular expression* χ *over* $\Sigma_\mathcal{F}$ (regex) is defined by the syntax: $\chi ::= \emptyset \mid f \mid \chi_1\chi_2 \mid (\chi_1 + \chi_2) \mid \chi^*$, where $f \in \Sigma_\mathcal{F}$. The *language* $\mathcal{L}(\chi)$ *defined by a regex* χ is the subset of $\Sigma_\mathcal{F}^*$ defined as customary.

First-order Logic A *predicate symbol* P is a symbol not in $\Sigma_\mathcal{C} \cup \Sigma_\mathcal{V} \cup \Sigma_\mathcal{F}$ with an associated integer $\mathsf{ar}(P) \geq 0$ called its *arity*. For a set of predicate symbols $\mathcal{S}$, $\mathbf{FO}(\mathcal{S})$ denotes the set of all function-free *first-order formulae* built using symbols of $\mathcal{S}$, $\Sigma_\mathcal{C}$, and $\Sigma_\mathcal{V}$. We will use $\exists\mathsf{Pos}(\mathcal{S})$ for the sub-language of $\mathbf{FO}(\mathcal{S})$ without free variables (*sentences*) of the form $\bigvee_i \exists\bar{x}_i.\varphi_i(\bar{x}_i)$, where each $\varphi_i(\bar{x}_i)$ is a conjunction of atomic formulae using constants from $\Sigma_\mathcal{C}$ and variables from $\bar{x}_i$. A *first-order interpretation for a set of predicate symbols* $\mathcal{S}$ (simply, interpretation for $\mathcal{S}$ or interpretation) is a pair $\mathcal{I} = \langle \Sigma_\mathcal{T}, \cdot^\mathcal{I} \rangle$ where $\cdot^\mathcal{I}$ is a function s.t. $c^\mathcal{I} = c$, for each $c \in \Sigma_\mathcal{C}$, and $P^\mathcal{I} \subseteq \Sigma_\mathcal{T}^n$, for each $P \in \mathcal{S}$ with $ar(P) = n$. Interpretations do not consider functions in $\Sigma_\mathcal{F}$ and enforce the standard names assumption over constants (Levesque and Lakemeyer 2001). An interpretation $\mathcal{I} = \langle \Sigma_\mathcal{T}, \cdot^\mathcal{I} \rangle$ for $\mathcal{S}$ and a mapping $\mu : \Sigma_\mathcal{V} \to \Sigma_\mathcal{T}$ satisfy $\varphi \in \mathbf{FO}(\mathcal{S})$ (written $\mathcal{I}, \mu \models \varphi$) in the usual sense. If φ is a sentence, $\mathcal{I} \models \varphi$ denotes $\mathcal{I}, \mu \models \varphi$, for each $\mu : \Sigma_\mathcal{V} \to \Sigma_\mathcal{T}$.

Morphisms Given interpretations $\mathcal{I}, \mathcal{J}$ for $\mathcal{S}$, a *homomorphism* from $\mathcal{I}$ to $\mathcal{J}$ is a mapping $h : \Sigma_\mathcal{T} \to \Sigma_\mathcal{T}$ s.t., $h(\bar{a}) \in R^\mathcal{J}$, for each $\bar{a} \in R^\mathcal{I}$ and $R \in \mathcal{S}$. A homomorphism h is *constant-preserving* if it is the identity over $\Sigma_\mathcal{C}$ and an *isomorphism* if it is bijective and h^{-1} is a homomorphism.

Queries A n*-ary query* q for a set of predicates $\mathcal{S}$ is an expression of the form $q(\bar{x}) = \{\bar{x} \mid \varphi(\bar{x})\}$, where $\bar{x} \in \Sigma_\mathcal{V}^n$ and $\varphi(\bar{x}) \in \mathbf{FO}(\mathcal{S})$ is a formula whose free variables occur in $\bar{x}$. We use $\Sigma_\mathcal{V}(q)$ for the set of all variables occuring in $\varphi(\bar{x})$. Given an interpretation $\mathcal{I}$ for $\mathcal{S}$, an *answer tuple* for q over $\mathcal{I}$ is a tuple $\mu(\bar{x})$ such that $\mathcal{I}, \mu \models \varphi(\bar{x})$. A *query language* $\mathbb{Q}$ is a collection of queries, and *conjunctive queries* (CQs) and their *unions* (UCQs) are languages defined as customary in the literature. Additionally, we will use *connected CQs* (CCQs), i.e, CQs for which the Gaifman graph of the defining formula is connected. It is well known that $\bar{a}$ is an answer tuple for a UCQ $q(\bar{x})$ over an interpretation $\mathcal{I}$ if there exists a homomorphism from (the canonical interpretation of) of a disjunct of $q(\bar{x})$ into $\mathcal{I}$ that maps $\bar{x}$ into $\bar{a}$. We call such homomorphisms the *supports of* $\bar{a}$ *and* q *over* $\mathcal{I}$.

Knowledge Bases (KBs) We assume the reader is familiar with Description Logics KBs and refer to (Calvanese et al. 2007) for additional details on *DL-Lite*$_\mathcal{R}$. Let $\Sigma_\mathcal{P}$ be a countably infinite set of predicate symbols partitioned into the disjoint sets Σ_A of *atomic concepts* (unary predicates) and Σ_R of *atomic roles* (binary predicates). A *KB* is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ (TBox) is a finite set of TBox axioms using predicates in $\Sigma_\mathcal{P}$, and $\mathcal{A}$ (ABox) is a finite set of ABox axioms using predicates in $\Sigma_\mathcal{P}$ and constants in $\Sigma_\mathcal{C}$. Additionally, we define the query languages CCQ, CQ, and UCQ, respectively, as those CCQs, CQs, and UCQs using only predicates in $\Sigma_\mathcal{P}$. Semantics for KBs is given as customary using interpretations: we define a *model of a KB* $\mathcal{K}$ as an interpretation $\mathcal{I}$ for $\Sigma_\mathcal{P}$ that satisfies all the axioms of $\mathcal{K}$ (written $\mathcal{I} \models \mathcal{K}$), and use $\mathsf{mod}(\mathcal{K})$ for the set of all models of $\mathcal{K}$.

2.1 A Framework for Informativeness

We now briefly recall the framework presented in (Andolfi et al. 2024) for analyzing the informativeness of query answers. At a high level, the framework is grounded on a simple intuition: given a query q and a KB $\mathcal{K}$, the collection $q(\mathcal{K})$ of all sets of answers that q yields over the models of $\mathcal{K}$ (Imielinski and Lipski 1984) describes all the information that q can possibly retrieve from $\mathcal{K}$. Given a Query Answering System (i.e., a formal model of a concrete query answering mechanism), we ask what portion of the information in $q(\mathcal{K})$ is actually returned by it.

Next, we define the collection $q(\mathcal{K})$ formally. To this end, we define semantics for logical queries using interpretations. Let Σ_ans be the countably infinite set of predicates $\bigcup_{i=0}^{\infty}\{\mathsf{ans}_i\}$ where each ans_i has arity i and does not occur in $\Sigma_\mathcal{P}$. Given an n-ary query q and a KB interpretation $\mathcal{I}$, the *complete answer for* q *over* $\mathcal{I}$ is the interpretation for Σ_ans $q(\mathcal{I}) = \langle \Sigma_\mathcal{T}, \cdot^{q(\mathcal{I})} \rangle$ such that: (*i*) $\mathsf{ans}_n^{q(\mathcal{I})}$ is the set of all the answer tuples for q over $\mathcal{I}$; and (*ii*) $\mathsf{ans}_j^{q(\mathcal{I})} = \emptyset$, for each $j \neq n$. Given a KB $\mathcal{K}$, the *complete answer for* q *over* $\mathcal{K}$ is defined as $q(\mathcal{K}) = \{q(\mathcal{I}) \mid \mathcal{I} \in \mathsf{mod}(\mathcal{K})\}$. Intuitively, $q(\mathcal{K})$ contains all the information that q can retrieve form $\mathcal{K}$.

In principle, to be maximally informative, a query answering mechanism should return the whole $q(\mathcal{K})$. This is, however, impossible since such a collection may easily become infinite. In these cases, query answering mechanisms return a *representation of* $q(\mathcal{K})$ *in some language*. Next, we provide a general model of such representations. An *answer domain* is a pair $\mathbb{D} = \langle \mathcal{D}, \models_\mathcal{D} \rangle$ such that $\mathcal{D}$ is a set of *syntactic objects* (abstract answers) and $\models_\mathcal{D} \subseteq \mathcal{D} \times \mathbf{FO}(\Sigma_\mathsf{ans})$ is a *satisfaction relation* over $\mathcal{D}$. Intuitively, $\mathcal{D}$ contains all the answers that a query answering mechanism using $\mathbb{D}$ may provide, i.e., the answer language, while $\models_\mathcal{D}$ provides semantics for such objects in terms of the alphabet used for

$q(\mathcal{K})$. The next definition provides a formal model of query answering mechanisms.

Definition 1. *Let $\mathbb{Q}$ be a query language, $\mathbb{K}$ a KB language, and $\mathbb{D} = \langle \mathcal{D}, \models_{\mathcal{D}} \rangle$ an answer domain. A* query answering system *(*QAS*) for $\mathbb{Q}$ over $\mathbb{K}$ with answers in $\mathbb{D}$ is a tuple $\langle \mathbb{Q}, \mathbb{K}, \mathbb{D}, eval \rangle$ where $eval$ is a function from $\mathbb{Q} \times \mathbb{K}$ to $\mathcal{D}$.*

Intuitively, $eval$ (evaluation function) characterizes the answers that the query answering mechanism modeled by a QAS provides for every pair of query and KB.

Definition 2. *Let $\mathbb{S}$ be the QAS $\langle \mathbb{Q}, \mathbb{K}, \langle \mathcal{D}, \models_{\mathcal{D}} \rangle, eval \rangle$ and let $\mathcal{F} \subseteq \boldsymbol{FO}(\Sigma_{\mathsf{ans}})$. We say that $\mathbb{S}$ preserves the certain knowledge definable in $\mathcal{F}$ if, for each $q \in \mathbb{Q}$, $\mathcal{K} \in \mathbb{K}$, and $\varphi \in \mathcal{F}$, the following holds: $eval(q, \mathcal{K}) \models_{\mathcal{D}} \varphi$ if and only if $A \models \varphi$, for each $A \in q(\mathcal{K})$.*

We can now formally state our main goal. Specifically, in what follows we present a QAS for UCQ over *DL-Lite*$_{\mathcal{R}}$ that preserves the certain knowledge of $\exists Pos(\Sigma_{\mathsf{ans}})$.

3 Regularly Recurrent Answers

We now introduce our novel answer language and use it to define a family of QAS for UCQ over *DL-Lite*$_{\mathcal{R}}$ KBs that preserve the certain knowledge definable in $\exists Pos(\Sigma_{\mathsf{ans}})$. Firstly, we define suitable objects for the answer domain.

Definition 3. *A term expression τ is an expression of the form $\langle \chi, c \rangle$ where $c \in \Sigma_{\mathcal{C}}$ and χ is a regular expression over $\Sigma_{\mathcal{F}}$. The semantics $[\![\tau]\!]$ of τ is the set of all terms $t \in \Sigma_{\mathcal{T}}$ such that the base of t is c and the word of t belongs to $\mathcal{L}(\rho)$.*

Example 4. *Let $\tau = \langle (c + s)^*, Ava \rangle$ with $c, s \in \Sigma_{\mathcal{F}}$ and $Ava \in \Sigma_{\mathcal{C}}$. The semantics of τ is the set $[\![\tau]\!]$ of all terms $f_1(\dots(f_k(Ava))\dots)$ s.t. $f_i \in \{c, s\}$, for each $i \in [k]$.*

Our goal is to use term expressions to represent *sets of tuples* whose terms are defined by their semantics. To formalize this intuition, we need to introduce additional notation.

A *substitution rule* ρ is an expression of the form $\rho = [x, \tau]$ where $x \in \Sigma_{\mathcal{V}}$ and τ is a term expression, and a *substitution set* is a finite set of substitution rules with distinct variables. Given a substitution set S, a *substitution function for S* is a function $f : \Sigma_{\mathcal{V}} \to \Sigma_{\mathcal{T}}$ such that $f(x) \in [\![\tau]\!]$, for each $x \in \Sigma_{\mathcal{V}}$ for which $[x, \tau] \in S$. The following definition introduces the basic building block of our answer language.

Definition 4. *A* regularly recurrent tuple (RRT) *is an expression of the form $\langle \bar{a}, S \rangle$ where $\bar{a}$ is a tuple of terms and S is a substitution set such that, for every variable x in $\bar{a}$, there is a substitution rule $[x, \tau]$ in S.*

Additionally, we define the *arity of an RRT* $r = \langle \bar{a}, S \rangle$ as the arity of $\bar{a}$. Intuitively, r acts as a placeholder for the tuples of terms that can be obtained by applying S over $\bar{a}$.

Definition 5. *Let $r = \langle (t_1, \dots, t_n), S \rangle$ be an RRT. The semantics $[\![r]\!]$ of r is the following set of tuples:*
$\{\big(f(t_1), \dots, f(t_n)\big) \mid f \text{ is a substitution function for } S\}$.

It should be clear that $[\![\langle \bar{a}, S \rangle]\!]$ contains no variables since each variable in $\bar{a}$ has an associated substitution rule in S.

Example 5. *Let $r = \langle \bar{a}, \{\rho\} \rangle$, where $\bar{a} = (x, s(x), d(x))$ and $\rho = [x, \tau]$ with τ as in Example 4. The semantics $[\![r]\!]$ of r is the set of all the tuples (t_1, t_2, t_3) such that, for some $n \in \mathbb{N}$ and $f_1, \dots, f_n \in \{c, s\}$ we have: $t_1 = f_1(\dots(f_n(Ava))\dots)$, $t_2 = s(t_1)$, $t_3 = d(t_1)$.*

We will call a finite set of RRTs of the same arity a *regularly recurrent answer (RRA)*, and define the arity $\mathsf{ar}(\mathsf{R})$ of an RRA R as the arity of its elements. The following definition provides a formal semantics to RRAs.

Definition 6. *The semantics $[\![\mathsf{R}]\!]$ of an RRA R is the interpretation $\langle \Sigma_{\mathcal{T}}, \cdot^{\mathsf{R}} \rangle$ for Σ_{ans} such that $ans_i^{\mathsf{R}} = \bigcup_{r \in \mathsf{R}} [\![r]\!]$, for $i = ar(\mathsf{R})$, and $ans_i^{\mathsf{R}} = \emptyset$, for every $i \neq ar(\mathsf{R})$.*

RRAs will be the answer domain of the family of QASs we are constructing. To this end, we need a suitable satisfaction relation: an RRA R *satisfies a formula* $\varphi \in \mathbf{FO}(\Sigma_{\mathsf{ans}})$ if $[\![\mathsf{R}]\!] \models \varphi$. We use $\models_{\mathsf{RRA}}$ for the satisfaction relation we just introduced, and we define the *answer domain* $\mathsf{RRA} = \langle \mathcal{R}, \models_{\mathsf{RRA}} \rangle$, where $\mathcal{R}$ is the family of all RRAs.

Example 6. *Consider the RRA $R = \{r\}$, where r is defined as in Example 5. Then, $R \not\models_{\mathsf{RRA}} \varphi_1$ and $R \models_{\mathsf{RRA}} \varphi_2$ where $\varphi_1 = \exists x_1, x_2.Ans_3(x_1, x_2, x_2)$ and $\varphi_2 = \exists x_1, x_2, x_3, x_4.Ans_3(Ava, x_1, x_2) \wedge Ans_3(x_1, x_3, x_4)$.*

With our answer domain in place, we are now ready to define a query evaluation function for our QAS. To this end, we define a special family of RRAs that we call canonical.

Definition 7. *Let q be a query and $\mathcal{K}$ a KB. An* RRA R is canonical for q over $\mathcal{K}$ *if the following conditions hold:*

1. *for every $\mathcal{I} \in \mathsf{mod}(\mathcal{K})$, there is a constant-preserving homomorphism from $[\![\mathsf{R}]\!]$ to $q(\mathcal{I})$; and*
2. *there exists $\mathcal{I} \in \mathsf{mod}(\mathcal{K})$ such that $[\![\mathsf{R}]\!] = q(\mathcal{I})$.*

Example 7. *Consider the ABox $\mathcal{A}' = \{E(Ava)\}$ and the TBox $\mathcal{T}'$ that extends $\mathcal{T}$ from Example 1 with the following:*

$$(4)\ \mathsf{E} \sqsubseteq \exists \mathsf{hC} \quad (5)\ \exists \mathsf{hC}^- \sqsubseteq \mathsf{E} \quad (6) \mathsf{E} \sqsubseteq \exists \mathsf{hD}$$

The additional axioms state that every employee has as coach (4) that is an employee (5), and every employee belongs to a department (6). The following query asks for all the employees with their supervisors and departments:

$$q(x, y, z) = \{(x, y, z) \mid \mathsf{E}(x) \wedge \mathsf{hS}(x, y) \wedge \mathsf{hD}(x, z)\}$$

One can show that R from Example 7 is a canonical RRA for q over $\langle \mathcal{T}', \mathcal{A}' \rangle$. Intuitively, this is because, in every model of the KB, there is an employee (Ava), their supervisor ($s(Ava)$), their department ($d(Ava)$), and all the other pairs induced by the axioms in $\mathcal{T}$.

If we add $\mathsf{E}(Bea)$ to $\mathcal{A}'$, R is not canonical anymore. This is because, in every model $\mathcal{I}$ of the new KB, $q(\mathcal{I})$ will contain also the information relative to Bea. To obtain a canonical RRA for the new KB, one could add to R the RRT $\langle (y, s(y), d(y)), \langle y, \tau \rangle \rangle$ with $\tau = \langle (c + s)^*, Bea \rangle$.

In general, there is no guarantee that a canonical RRA for a given pair of query and KB exists due to the finiteness requirements we impose. A QAS $\langle \mathbb{Q}, \mathbb{K}, \mathsf{RRA}, eval \rangle$ is called *RRA-canonical* if $eval(q, \mathcal{K})$ is a canonical RRA for q over $\mathcal{K}$, for every pair of $q \in \mathbb{Q}$ and $\mathcal{K} \in \mathbb{K}$. The following claim follows from the definition of canonical RRAs.

Theorem 1. *Every RRA-canonical QAS preserves the certain knowledge definable in $\exists \mathsf{Pos}(\Sigma_{\mathsf{ans}})$.*

The goal of Section 4 will be to show that an RRA-canonical QAS for UCQ over satisfiable *DL-Lite*$_{\mathcal{R}}$ exists.

3.1 RRAs and Other Answer Languages

Before concluding this section, we compare RRAs with other answer languages from the literature. As we pointed out in the introduction, results in (Andolfi et al. 2024) show that a QAS for UCQ over *DL-Lite*$_\mathcal{R}$ whose answers can be represented as finite interpretations for Σ_{ans} cannot preserve the certain information definable in $\exists Pos(\Sigma_{\mathsf{ans}})$. In turn, Theorem 1 implies that an RRA-canonical QAS is strictly more expressive than those based on certain n-answers or minimal partial answers with multi-wildcards.

An answer language that is close in spirit to RRAs are Referring Expressions (RefExpr) as defined in (Toman and Weddell 2019). Indeed, following the construction from that work, one can obtain a QAS for *Conjunctive Queries* (CQs) over $\mathcal{ALC}$ ontologies which preserves the certain information definable in $\exists Pos(\Sigma_{\mathsf{ans}})$. However, RefExprs present issues that make them unsuitable for our context. Firstly, the *singularity* requirement of Definition 5 from (Toman and Weddell 2019) is easily violated if *role inclusion assertions* are allowed in the TBox. As a result, in these cases one may easily end up with no RefExprs for unnamed individuals in the answers. Secondly, the same unnamed individual may satisfy more than one RefExpr. To avoid redundant information, one needs to compute ABox closures, which is usually expensive in terms of space and time.

Finally, the information provided by an RRA cannot, in general, be reconstructed by computing the CAs of finitely many Boolean queries to the KB. For simplicity, consider the KB in Example 1. To discover that Ava's supervisor has a supervisor, one could issue the Boolean query $\exists x, y.hS(Ava, x) \wedge hS(x, y)$ and, upon receiving a non-empty answers, conclude that such an individual certainly exists. However, if the user is interested in ascending the chain of supervisors further, they would need to issue additional queries. *If the ontology is unknown to them*, there seems to be no straightforward way to conclude that such a chain is infinite. On the contrary, it is easy to observe that a canonical RRA $\{\langle(x, s(x)), \{[x, \langle s^*, Ava\rangle]\}\rangle\}$ for the query $\{(x, y) \mid hS(x, y)\}$ conveys such information.

4 RRA-Canonical QAS for UCQ in *DL-Lite*$_\mathcal{R}$

This section is devoted to the proof of the following theorem.

Theorem 2. *There exists an RRA-canonical QAS for* UCQ *over the language of satisfiable DL-Lite*$_\mathcal{R}$ *KBs.*

Theorem 2 and Theorem 1 together prove our main result: there exists a QAS for UCQ over *DL-Lite*$_\mathcal{R}$ KBs that preserves the certain knowledge definable in $\exists Pos(\Sigma_{\mathsf{ans}})$. To construct such a QAS, we rely on the notion of canonical model of a *DL-Lite*$_\mathcal{R}$ KB. In this work, a *canonical model* for a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$ is an interpretation $\mathcal{C} = \langle\Sigma_\mathcal{T}, \cdot^\mathcal{C}\rangle$ for $\Sigma_\mathcal{P}$ s.t. $\mathcal{C} \in \mathsf{mod}(\mathcal{K})$, and, for every $\mathcal{I} \in \mathsf{mod}(\mathcal{K})$, there exists a constant-preserving homomorphism from $\mathcal{C}$ to $\mathcal{I}$.

Lemma 1. *Let $\mathcal{K}$ be a DL-Lite*$_\mathcal{R}$ *KB, $\mathcal{C}$ a canonical model for $\mathcal{K}$, and $q \in \mathsf{UCQ}$. If an RRA R is such that $[\![\mathsf{R}]\!] = q(\mathcal{C})$, then R is canonical for q over $\mathcal{K}$.*

To prove Theorem 2, we rely on Lemma 1 in the following way. Given a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$, we construct a specific type of forest-like canonical model $\mathcal{C}$ for $\mathcal{K}$ and show that $\mathcal{C}$ admits a finite representation highlighting its recurrent patterns. Then, given $q \in \mathsf{UCQ}$, we construct an RRA R that represents all those recurrent patterns of $\mathcal{C}$ that satisfy q and show that indeed $[\![\mathsf{R}]\!] = q(\mathcal{C})$.

The remainder of this section is split in two parts: Section 4.1 presents our construction that yields a canonical model. While the construction is rather standard, we provide the main details that are needed to understand the rest of the proof. Then, in Section 4.2, we construct the desired RRA and present its properties.

4.1 Forest-Like Canonical Models

Our construction makes use of the set $\Sigma^+_\mathcal{P}$ of *extended predicates* consisting of the following: all symbols in $\Sigma_\mathcal{P}$; all symbols R^-, where R is an atomic role; and all symbols $\exists P$, where P is an atomic role or its inverse. Intuitively, the set of extended predicates contains all the *DL-Lite*$_\mathcal{R}$ symbols that can be generated using $\Sigma_\mathcal{P}$.

Next, we introduce *graphs of facts* (GoF), a convenient family of graphs that we use as a scaffolding for our canonical models. In what follows, a GoF $G = \{V, E, l\}$ is simply an edge-labeled directed graph whose nodes are terms in $\Sigma_\mathcal{T}$ and whose edges are labeled by sets of symbols in $\Sigma^+_\mathcal{P}$. Sometimes we will treat l as a set of pairs and write $l' = l \cup \{(e, t)\}$, with $e \in E$ and $t \in \Sigma^+_\mathcal{P}$, for the labeling function such that $l'(e) = l(e) \cup \{(e, t)\}$, and $l'(e') = l(e')$, for every $e \neq e' \in E$. Given a graph of facts G, one can construct an equivalent interpretation $\mathsf{I}(G)$ as follows.

Definition 8. *The* interpretation $\mathsf{I}(G) = \langle\Sigma_\mathcal{T}, \cdot^{\mathsf{I}(G)}\rangle$ associated to a GoF $G = (V, E, l)$ *is an interpretation for $\Sigma_\mathcal{P}$ s.t $C^{\mathsf{I}(G)} = \{t \mid e = (t, t) \in E$ and $C \in l(e)\}$, for each atomic concept $C \in \Sigma_\mathcal{P}$, and $R^{\mathsf{I}(G)} = \{(t_1, t_2) \mid e = (t_1, t_2) \in E$ and $R \in l(e)\} \cup \{(t_2, t_1) \mid e = (t_1, t_2) \in E$ and $R^- \in l(e)\}$, for each atomic role $R \in \Sigma_\mathcal{P}$.*

Similarly, given a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$, one can construct a GoF $G(\mathcal{A})$ that represents $\mathcal{A}$ as follows. The nodes of $G(\mathcal{A})$ are all the constants occurring in $\mathcal{A}$; for each constant c occurring in $\mathcal{A}$, $G(\mathcal{A})$ has an edge (c, c) whose label consists of all $P \in \Sigma_\mathcal{P}$ such that $P(c) \in \mathcal{A}$, all symbols $R \in \Sigma_\mathcal{P}$ such that $R(c, c) \in \mathcal{A}$, all symbols $\exists R \in \Sigma^+_\mathcal{P}$, such that $R(c, d) \in \mathcal{A}$, and all symbols $\exists R^- \in \Sigma^+_\mathcal{P}$, such that $R(d, c) \in \mathcal{A}$; for each pair of distinct constants c, b occurring in $\mathcal{A}$, $G(\mathcal{A})$ has an edge (c, b) whose label consists of all symbols $P \in \Sigma_\mathcal{P}$ such that $P(c, b) \in \mathcal{A}$.

Given a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$, our goal is now to construct a GoF $G_\mathcal{K}$ from $G(\mathcal{A})$ such that $\mathsf{I}(G_\mathcal{K})$ is a canonical model of $\mathcal{K}$. To this end, we use the notion of *unraveling*, a specialization of the *Skolem chase* (Calvanese et al. 2007; Marnette 2009) for graphs of facts.

Let $\mathcal{T}$ be a *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$. To each axiom $\gamma \in \mathcal{T}$, we associate a distinct function symbol $\mathfrak{s}_\gamma \in \Sigma_\mathcal{F}$ and an *unraveling rule* ρ_γ. An unraveling rule is an expression of the form $\alpha \rightsquigarrow \beta$ where both α and β are of the form $X[t_1, t_2]$, X is a symbol from $\Sigma^+_\mathcal{P}$, t_1, t_2 are (not necessarily distinct) terms in $\Sigma_\mathcal{T}$, and all variables in β also occur in α. Unraveling

γ	ρ_γ
$A \sqsubseteq B$	$A[x,x] \rightsquigarrow B[x,x]$
$A \sqsubseteq \exists R$	$A[x,x] \rightsquigarrow R[x,\mathfrak{s}_\gamma(x)]$
$A \sqsubseteq \exists R^-$	$A[x,x] \rightsquigarrow R^-[x,\mathfrak{s}_\gamma(x)]$
$\exists R \sqsubseteq A$	$\exists R[x,x] \rightsquigarrow A[x,x]$
$\exists R \sqsubseteq \exists S$	$\exists R[x,x] \rightsquigarrow S[x,\mathfrak{s}_\gamma(x)]$
$\exists R \sqsubseteq \exists S^-$	$\exists R[x,x] \rightsquigarrow S^-[x,\mathfrak{s}_\gamma(x)]$
$\exists R^- \sqsubseteq A$	$\exists R^-[x,x] \rightsquigarrow A[x,x]$
$\exists R^- \sqsubseteq \exists S$	$\exists R^-[x,x] \rightsquigarrow S[x,\mathfrak{s}_\gamma(x)]$
$\exists R^- \sqsubseteq \exists S^-$	$\exists R^-[x,x] \rightsquigarrow S^-[x,\mathfrak{s}_\gamma(x)]$
$R \sqsubseteq S$	$R[x,y] \rightsquigarrow S[x,y]$
$R \sqsubseteq S^-$	$R[x,y] \rightsquigarrow S^-[x,y]$
$R^- \sqsubseteq S$	$R^-[x,y] \rightsquigarrow S[x,y]$
$R^- \sqsubseteq S^-$	$R^-[x,y] \rightsquigarrow S^-[x,y]$

Table 1: Unraveling Rules

rules for *DL-Lite*$_\mathcal{R}$ are given in Table 1, where $x, y \in \Sigma_\mathcal{V}$, A, B are atomic concepts, and R, S are atomic roles.

Next, we define how unraveling rules are applied. For $X \in \Sigma^+_\mathcal{P}$, the *inverse of* X (denoted by $\mathsf{inv}(X)$) is defined as follows: $\mathsf{inv}(X) = X$, if X is an atomic concept; $\mathsf{inv}(X) = X^-$, if X is an *atomic role*; $\mathsf{inv}(X) = \exists R^-$, if $X = \exists R$ and R is an *atomic role*; $\mathsf{inv}(X) = R$, if $X = R^-$; and $\mathsf{inv}(X) = \exists R$, if $X = \exists R^-$. Similarly, the *existential qualification of* X ($\mathsf{ex}(X)$) is defined as $\mathsf{ex}(X) = \exists X$, if X is an atomic role or its inverse, and $\mathsf{ex}(X) = P$, otherwise.

Let now $\rho = P[t_1, t_2] \rightsquigarrow Q[t_3, t_4]$ be an unraveling rule with t_1, t_2, t_3, t_4 not necessarily distinct. A *trigger* τ *for* ρ *in a GoF* $G = \langle V, E, l\rangle$ is a mapping $\tau\colon \Sigma_\mathcal{V} \to V$ s.t. either (*i*) $(\tau(t_1), \tau(t_2)) \in E$ and $P \in l((\tau(t_1), \tau(t_2)))$ or (*ii*) $(\tau(t_2), \tau(t_1)) \in E$ and $\mathsf{inv}(P) \in l((\tau(t_2), \tau(t_1)))$.

Definition 9. *Let* τ *be a trigger for the unraveling rule* $\rho = P[t_1, t_2] \rightsquigarrow Q[t_3, t_4]$ *in a GoF* $G = \langle V, E, l\rangle$. *The* application of τ to G *(written* $\tau(G)$*) is the GoF* $G' = \langle V', E', l'\rangle$ *defined as follows:*

- $V' = \{\tau(t_3), \tau(t_4)\} \cup V$;
- $E' = \{(\tau(t_3), \tau(t_4)), (\tau(t_3), \tau(t_3)), (\tau(t_4), \tau(t_4))\} \cup E$;
- $l' = \{\langle(\tau(t_3), \tau(t_4)), Q\rangle\} \cup \{\langle(\tau(t_3), \tau(t_3)), \mathsf{ex}(Q)\rangle\} \cup \{\langle(\tau(t_4), \tau(t_4)), \mathsf{ex}(\mathsf{inv}(Q))\rangle\} \cup l$.

We can extend the notion of application of a trigger to a whole *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$ in the natural way: given a GoF G, $\mathcal{T}(G)$ is the graph obtained by applying all the triggers for $\mathcal{T}$ in G. Clearly, $\mathcal{T}(G)$ defines a monotone operator and, thus, we can naturally talk about the least fix point $\mathsf{unravel}(\mathcal{T}, G)$ of $\mathcal{T}$ over a GoF G. Additionally, given a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$, we use $\mathsf{unravel}(\mathcal{K})$ for the GoF $\mathsf{unravel}(\mathcal{T}, G(\mathcal{A}))$. One can show that $\mathsf{unravel}(\mathcal{K})$ yields a canonical model of $\mathcal{K}$ as the following lemma shows.

Lemma 2. *Let* $\mathcal{K}$ *be a satisfiable DL-Lite*$_\mathcal{R}$ *KB. Then,* $\mathsf{I}(\mathsf{unravel}(\mathcal{K}))$ *is a canonical model for* $\mathcal{K}$.

We now argue about the topology of $\mathsf{unravel}(\mathcal{K})$. A TBox axiom is *non-generative* if its unraveling rule contains no function symbols. The $\mathcal{T}$*-closure of* $\mathcal{A}$ is $\mathcal{A}^\mathcal{T} = \mathsf{unravel}(\mathcal{T}_0, \mathcal{A})$, where $\mathcal{T}_0$ denotes the set of non-generative axioms in $\mathcal{T}$. The *roots of* $\mathcal{K}$ ($\mathfrak{R}(\mathcal{K})$) is the collection of all sub-graphs of $\mathcal{A}^\mathcal{T}$ induced by one of the constants in $\mathcal{A}$.

Lemma 3. *Let* $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ *be a DL-Lite*$_\mathcal{R}$ *KB. Then* $\mathsf{unravel}(\mathcal{K}) = \mathcal{A}^\mathcal{T} \cup \bigcup_{g'' \in \mathfrak{R}(\mathcal{K})} \mathsf{unravel}(\mathcal{T}, g'')$ *and, for every* $g, g' \in \mathfrak{R}(\mathcal{K})$ *we have:*

1. $\mathsf{unravel}(\mathcal{T}, g)$ *is an arborescence (removing self-loops);*
2. $\mathsf{unravel}(\mathcal{T}, g)$ *and* $\mathsf{unravel}(\mathcal{T}, g')$ *share no vertexes;*
3. *For every* $t, t' \in \Sigma_\mathcal{T}$ *s.t. there is a path in* $\mathsf{unravel}(\mathcal{T}, g)$ *from* t *to* t'*, there is a sequence of function symbols* $\bar{g}$ *in* $\Sigma_\mathcal{F}$ *for which* $t' = \bar{g}(t)$.

4.2 Answering Queries with Unravelling

We are now ready to provide a constructive proof of Theorem 2. Specifically, we show that, for a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$ and $q \in \mathsf{UCQ}$, there exists an RRA R such that $[\![\mathsf{R}]\!] = q(\mathsf{I}(\mathsf{unravel}(\mathcal{K})))$. The proof is based on the fact that $\mathsf{unravel}(\mathcal{K})$ consists of a collection of isomorphic sub-trees. To identify these sub-trees, we use the notion of replica.

Definition 10. *Let* $\mathcal{K}$ *be a DL-Lite*$_\mathcal{R}$ *KB and let* $t, t' \notin \Sigma_\mathcal{C}$ *be nodes in* $\mathsf{unravel}(\mathcal{K})$. *Then,* t' *is a* replica of t in $\mathcal{K}$ *if (*i*) there exists a path from* t *to* t' *in* $\mathsf{unravel}(\mathcal{K})$*; (*ii*) the leading function symbol of* t *and* t' *coincide; (*iii*) and* $t \neq t'$.

One can easily show that, if t', t'' are both replicas of t, then the sub-trees of $\mathsf{unravel}(\mathcal{K})$ rooted in the three terms are all isomorphic. This is due to the fact that t, t', and t'' are all generated by an application of the *same unraveling rule* defined by $\mathcal{T}$ and, thus, one can show inductively that the *same unraveling steps* are applied to all of them.

The next step of our construction is to define a replica-aware version of $\mathsf{unravel}(\mathcal{K})$. The *root of a term* t *of* $\mathsf{unravel}(\mathcal{K})$ (denoted by $\mathfrak{R}(t)$) is the unique $g \in \mathfrak{R}(\mathcal{K})$ such that t occurs in $\mathsf{unravel}(\mathcal{T}, g)$. We use $Anc(t)$ for the set of vertexes in the *unique loop-free* path in $G = \mathsf{unravel}(\mathcal{T}, \mathfrak{R}(t))$ from the root of G to t. The *replica-level of a term* t *in* $\mathcal{K}$ *(denoted by* $lv_\mathcal{K}(t)$*)* is defined as:

$$max_{t' \in Anc(t)} |\{t'' \in Anc(t') \mid t'' \text{ is a replica of } t' \text{ in } \mathcal{K}\}|$$

Definition 11. *Let* $\mathcal{K}$ *be a DL-Lite*$_\mathcal{R}$ *KB and* $n \in \mathbb{N}$. *The* n-th replica-level of $\mathsf{unravel}(\mathcal{K}) = \langle V, E, l\rangle$ ($\mathsf{unravel}_{r=n}(\mathcal{K})$) *is the sub-graph induced by* $\{v \in V \mid lv_\mathcal{K}(v) \leq n\}$.

It is easy to observe that, for every $g \in \mathfrak{R}(\mathcal{K})$ and $n \in \mathbb{N}$, $\mathsf{unravel}_{r=n}(\mathcal{T}, g)$ is a finite arborescence when self-loops are removed. Next, we proceed to identify two distinct types of answers for UCQs over $\mathsf{I}(\mathsf{unravel}(\mathcal{K}))$. Given an n-ary $q \in \mathsf{UCQ}$ and $\bar{a} \in \Sigma^n_\mathcal{T}$, a *near support* of $\bar{a}$ and q is a support μ of $\bar{a}$ and q over $\mathsf{I}(\mathsf{unravel}(\mathcal{K}))$ such that $\mu(x)$ occurs in $\mathsf{unravel}_{r=0}(\mathcal{K})$, for some $x \in \Sigma_\mathcal{V}(q)$. We use near supports to partition the set of answers for a UCQ in two collections.

Definition 12. *Let* $\mathcal{K}$ *be a DL-Lite*$_\mathcal{R}$ *KB and let* q *be an* n*-ary query in* UCQ. *A tuple* $\bar{a} \in \Sigma^n_\mathcal{T}$ *is a* near answer to q over $\mathcal{K}$ *if there exists a near support for* $\bar{a}$ *and* q *over* $\mathsf{I}(\mathsf{unravel}(\mathcal{K}))$. *Otherwise,* $\bar{a}$ *is a* far answer to q over $\mathcal{K}$.

It is clear that every $\bar{a} \in q(\mathsf{I}(\mathsf{unravel}(\mathcal{K})))$ is either near or far, depending on the existence of an associated near support. We now show that there exists an RRA whose semantics consists of both. In what follows, we *focus on CCQs* and discuss the general UCQs at the end of the section.

Lemma 4. *Let* $\mathcal{K}$ *be a satisfiable DL-Lite*$_\mathcal{R}$ *KB, and let* $q \in \mathsf{CCQ}$ *with* $|\Sigma_\mathcal{V}(q)| \leq n$. *Then,* $\bar{a}$ *is a near answer to* q *over*

$\mathcal{K}$ if and only if there exists a near support μ for $\bar{a}$ and q over $\mathsf{I}(\mathsf{unravel}_{r=n}(\mathcal{T},\mathcal{A}))$.

A *near RRT for q over $\mathcal{K}$* is an RRT $r = \langle \bar{a}, \emptyset \rangle$ such that there exists a near support for $\bar{a}$ and q over $\mathsf{unravel}_{r=n}(\mathcal{K})$. Next, we turn our attention to far answers.

Definition 13. *A generator for a DL-Lite$_\mathcal{R}$ KB $\mathcal{K}$ is a term $t \in \Sigma_\mathcal{T}$ for which there exists a term $t' \in \Sigma_\mathcal{T}$ such that (*i*) t, t' appear in* $\mathsf{unravel}_{r=1}(\mathcal{T},\mathcal{A})$*, and (*ii*) t' is a replica of t.*

From what we said above, a generator marks the beginning of a repeating sub-tree of $\mathsf{unravel}(\mathcal{K})$. Characterizing these repetitions is crucial for our proofs. Recall that, if t' is a replica of t, then $t' = \bar{g}(t)$, for some sequence of functions symbols $\bar{g}$ in $\Sigma_\mathcal{F}$ (Item 3 of Lemma 3).

Definition 14. *Let $t = f\bar{g}(c)$ with $c \in \Sigma_\mathcal{C}$ be a generator in* $\mathsf{unravel}(\mathcal{K})$ *and let $f\bar{g}_1 f\bar{g}(c), \ldots, f\bar{g}_k f\bar{g}(c)$ be its replicas in* $\mathsf{unravel}_{r=1}(\mathcal{T},\mathcal{A})$*. The* replica expression of t *is the regular expression* $\mathsf{e}(t) = (f\bar{g}_1 + \ldots + f\bar{g}_n)$.

Intuitively, the regular expression $\mathsf{e}(t)f\bar{g}$ generates the word of every replica of $t = f\bar{g}(c)$ and, thus, it captures some of the terms of $\mathsf{unravel}(\mathcal{K})$ that are generated by the axiom that generated t. However, other "copies" of t could be generated by the generators preceding it. To obtain a regular expression for all such terms, we need the following.

Let t be a generator in $\mathsf{unravel}(\mathcal{K})$. The *generator sequence of* t is the sequence $\langle g_1, \ldots, g_n \rangle$ of generators in the unique path from the root of $\mathsf{unravel}(\mathfrak{R}(t))$ to t *in reverse order w.r.t. how they occur in that path* and $g_1 = t$. Observe now that, for each $i < j$, there is a path from g_j to g_i in $\mathsf{unravel}(\mathcal{K})$. Due to Item 3 of Lemma 3 then, the word of g_i is equal to $s'_i s_j$, where s_j is the word of g_j and s'_i is a sub-expression of the word of g_i. The *word decomposition of* t is the sequence of strings $\langle s_1, \ldots, s_n \rangle$ such that, for each $i \in [n]$, $s_i s_{i+1} \ldots s_n$ is the word g_i, i.e., the i-th term in the generator sequence of t. We are finally ready to provide a regular expression that defines the roots of all sub-trees of $\mathsf{unravel}(\mathcal{K})$ generated by the axioms that generated t.

Definition 15. *Let t be a generator whose base is $c \in \Sigma_\mathcal{C}$. The* generation rule $\rho(t)$ of t *is the term expression $\langle \mathsf{e}, c \rangle$, where* e *is the regex* $\mathsf{e}(\bar{g}_1)^*(s_1)\mathsf{e}(\bar{g}_2)^*(s_2)\ldots\mathsf{e}(\bar{g}_n)^*(s_n)$ *and s_i and g_i are, respectively, the i-th elements of the word decomposition and the generator sequence of t.*

We are finally ready to construct the desired RRA. Let t be a generator for $\mathcal{K}$, the *generator graph* G_t of t is the sub-graph of $\mathsf{unravel}_{r=1}(\mathcal{T},\mathcal{A})$ induced by t. Let now q be a CCQ with $|\Sigma_\mathcal{V}(q)| = n$. A *generator RRT for q over $\mathcal{K}$* is an RRT $\langle \bar{a}, \{\sigma\} \rangle$ for which there exists a generator t for $\mathcal{K}$ and $\bar{c} \in q(\mathsf{I}(\mathsf{unravel}_{r=n}(\mathcal{T}, G_t)))$ such that $\sigma = \langle x, \rho(t) \rangle$ and $\bar{c}$ is obtained from $\bar{a}$ bt substituting x with t.

Lemma 5. *Let $\mathcal{K}$ be a satisfiable DL-Lite$_\mathcal{R}$ KB and $q(\bar{x})$ a CCQ. If a tuple $\bar{c}$ is a far answer to q over $\mathcal{K}$, then there is a generator RRT r for q over $\mathcal{K}$ such that $\bar{c} \in [\![r]\!]$.*

Let now $eval_\mathsf{R}$ be the function that maps every pair of CCQ q and satisfiable *DL-Lite$_\mathcal{R}$* KB $\mathcal{K}$ into the finite RRA consisting of all the *near RRTs* and all the (unique up to variable renaming) *generator RRTs* for q over $\mathcal{K}$. Using Lemma 4 and 5, one can show the following.

Lemma 6. *For every satisfiable DL-Lite$_\mathcal{R}$ KB $\mathcal{K}$ and $q \in$ UCQ, $[\![eval_\mathsf{R}(q,\mathcal{K})]\!] = q(\mathsf{I}(\mathsf{unravel}(q,\mathcal{K}))$*

To prove the existence of a canonical RRA in the case of general CQs, we observe the following. Given a $q \in \mathsf{CQ}$, one can split it into its connected components, apply the construction above to obtain canonical RRAs for each of these components, and then return the Cartesian product of the results. To handle UCQs, one can simply return the union of the results of each of its disjuncts. These observations allow one to easily obtain Theorem 2.

5 Computational Aspects

We now study two computational problems related to the QAS defined in Section 4. The first problem is query answering, i.e., computing $eval_\mathsf{R}(q,\mathcal{K})$, for a given satisfiable *DL-Lite$_\mathcal{R}$* KB $\mathcal{K}$ and $q \in \mathsf{UCQ}$. We are interested in the so called *data complexity* of the problem, i.e., the complexity that depends on the size of the input ABox alone. By inspecting the proof in Section 4, it is easy to observe that one can compute $\mathsf{unravel}_{r=n}(\mathcal{K})$ in PTIME w.r.t. the size of $\mathcal{A}$. This automatically yields a polynomial upper bound for the aforementioned problem. Interestingly, one can show that it is possible to obtain an even tighter upper bound by constructing a suitable FOL formula (independent from $\mathcal{A}$) to be evaluated over $\mathcal{A}$. The latter yields the following result.

Theorem 3. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite$_\mathcal{R}$ KB and let $q \in$ UCQ. Computing $eval_\mathsf{R}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ can be done with logarithmic space overhead w.r.t. the size of $\mathcal{A}$.*

The results stated in Theorem 3 are in line with classical query answering using CAs and, thus, provide a strong motivation for using RRAs in practice.

The second problem we study is checking whether a formula $\varphi \in \exists Pos(\Sigma_\mathsf{ans})$ is satisfied by an RRA R. This problem refers to the possibility of using RRAs effectively as views over the original KB. Here we are interested in the complexity coming only from the input RRA, and the complexity of the whole problem.

Theorem 4. *For a given RRA* R *and $\varphi \in \exists Pos(\Sigma_\mathsf{ans})$, checking whether $\mathsf{R} \models_\mathsf{R} \varphi$ is PSPACE-Complete, and can be done in PTIME w.r.t. the size of* R.

It is still open whether the aforementioned problem is PTIME-complete w.r.t. the size of the RRA R.

6 Conclusions

In this paper, we presented a novel answer language for KBs (RRAs) and used it to define a query answering mechanism for UCQs over *DL-Lite$_\mathcal{R}$* KBs that is provably more informative than others in the literature. Additionally, we studied the complexity of related decision problems.

Future work includes using RRAs with different query and KB languages, possibly beyond the DL family. Moreover, one could study QAS that preserve languages more expressive than $\exists Pos(\Sigma_\mathsf{ans})$. Finally, we would like to use RRAs as the theoretical foundation of concrete software systems to visualize query answering results.

Acknowledgments

This work has been supported by MUR under the PNRR project FAIR (PE0000013).
Additionally, the authors would like to thank the anonymous referees for their insightful comments.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison Wesley Publishing Company.

Andolfi, L.; Cima, G.; Console, M.; and Lenzerini, M. 2024. What Does a Query Answer Tell You? Informativeness of Query Answers for Knowledge Bases. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 24)*, 10442–10449.

Beck, F.; Burch, M.; Diehl, S.; and Weiskopf, D. 2017. A taxonomy and survey of dynamic graph visualization. In *Computer graphics forum*, 1, 133–159. Wiley Online Library.

Bienvenu, M.; and Ortiz, M. 2015. Ontology-Mediated Query Answering with Data-Tractable Description Logics. In *Reasoning Web. Semantic Technologies for Intelligent Data Access – Eleventh International Summer School Tutorial Lectures (RW 2015)*, volume 9203 of *Lecture Notes in Computer Science*, 218–307.

Borgida, A.; Toman, D.; and Weddell, G. E. 2016. On Referring Expressions in Query Answering over First Order Knowledge Bases. In *Proceedings of the Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2016)*, 319–328.

Calì, A.; Gottlob, G.; and Kifer, M. 2008. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008)*, 70–80.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *Journal of Automated Reasoning*, 39(3): 385–429.

Console, M.; Guagliardo, P.; Libkin, L.; and Toussaint, E. 2020. Coping with Incomplete Data: Recent Advances. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, 33–47. ACM.

Imielinski, T.; and Lipski, W., Jr. 1984. Incomplete Information in Relational Databases. *Journal of the ACM*, 31(4): 761–791.

Lenzerini, M. 2002. Data Integration: A Theoretical Perspective. In *Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, 233–246.

Levesque, H. J.; and Lakemeyer, G. 2001. *The Logic of Knowledge Bases*. The MIT Press.

Libkin, L. 2016. Certain answers as objects and knowledge. *Artificial Intelligence*, 232: 1–19.

Lipski, W. 1979. On Semantic Issues Connected with Incomplete Information Databases. *ACM Trans. Database Syst.*, 4(3): 262–296.

Lutz, C.; and Przybylko, M. 2022. Efficiently Enumerating Answers to Ontology-Mediated Queries. In *Proceedings of the Forty-First ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2022)*, 277–289.

Lutz, C.; and Przybylko, M. 2023. Efficient Answer Enumeration in Description Logics with Functional Roles. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023)*, 6483–6490.

Marnette, B. 2009. Generalized Schema-Mappings: from Termination to Tractability. In *Proceedings of the Twentyeighth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2009)*, 13–22.

Reiter, R. 1980. Data Bases: A Logical Perspective. In *Proc. of the Workshop on Data Abstraction, Databases and Conceptual Modelling*, 174–176.

Toman, D.; and Weddell, G. E. 2019. Finding ALL Answers to OBDA Queries Using Referring Expressions. In Liu, J.; and Bailey, J., eds., *AI 2019: Advances in Artificial Intelligence - 32nd Australasian Joint Conference, Adelaide, SA, Australia, December 2-5, 2019, Proceedings*, volume 11919 of *Lecture Notes in Computer Science*, 117–129. Springer.